Mengenal dan Menggunakan Bahasa Pemrograman Java®

Dr. Noprianto

Termasuk pembahasan interpreter bahasa pemrograman Singkong

Penerbit:
PT. Stabil Standar Sinergi
Download gratis buku ini di:
https://nopri.github.io

ISBN 978-602-52770-0-9

9 786025 277009

Mengenal dan Menggunakan Bahasa Pemrograman Java®

Dr. Noprianto

Termasuk pembahasan interpreter bahasa pemrograman Singkong

Penerbit:
PT. Stabil Standar Sinergi
Download gratis buku ini di:
https://nopri.github.io

ISBN 978-602-52770-0-9

9 786025 277009

Mengenal dan Menggunakan Bahasa Pemrograman Java®

Penulis:
Dr. Noprianto

ISBN: 978-602-52770-0-9

Editor:
Dr. Noprianto

Penerbit:
PT. Stabil Standar Sinergi

Redaksi:
Email: info@stabilstandar.com

Cetakan pertama: September, 2018
Cetakan kedua: November, 2019
Cetakan ketiga: Januari, 2020
Cetakan keempat: April, 2020
Cetakan kelima: Mei, 2020

Tentang penulis:

Dr. Noprianto menyukai pemrograman, memiliki sertifikasi Java (OCP), dan telah menulis beberapa buku cetak: Python (2002), Debian GNU/Linux (2006), OpenOffice.org (2006), Java (2018), dan Singkong (2020). Beliau juga menulis beberapa buku elektronik gratis: wxWidgets (2006), Python (2009), SQLiteBoy (2014), dan OpenERP (rekan penulis, 2014).

Noprianto lulus dari jurusan teknik informatika (2003), manajemen sistem informasi (2015), dan doktor ilmu komputer (2019) dari Universitas Bina Nusantara.

Noprianto mengembangkan bahasa pemrograman Singkong dan interpreternya sejak 2019.

Buku dan softwarenya dapat didownload dari https://nopri.github.io

Kata Pengantar

Java® adalah bahasa pemrograman *general-purpose* (dapat digunakan untuk membangun aplikasi berbagai domain), mendukung beberapa paradigma pemrograman (termasuk *object-oriented* berbasis *class*), dengan sistem tipe *strong* dan *static*, dan termasuk bahasa pemrograman *high-level*.

Manual pendamping training ini difokuskan pada dasar-dasar bahasa pemrograman Java untuk digunakan di desktop (termasuk bekerja dengan GUI dan sistem database relasional). Materi dirancang agar siap diterapkan dan setiap bagian pembahasan dapat diakhiri dengan contoh soal.

Mempersiapkan sebuah manual dasar-dasar bahasa pemrograman Java adalah pengalaman yang menyenangkan, sekaligus menyita perhatian. Java adalah bahasa yang besar, disertai pustaka bawaan yang besar dan kompleks, dengan sejumlah aturan baku. Tidaklah mudah memilih topik yang akan dibahas dan seberapa topik tersebut perlu dibahas (sesuai kemampuan penulis), untuk menjaga manual ini tetap ringkas dan sederhana. Penulis banyak belajar kembali ketika mempersiapkan manual ini.

Terima kasih.

Jakarta, September 2018 / April 2020
Noprianto

Daftar Isi

Daftar Isi

Daftar Isi

Daftar Isi

Daftar Isi

Daftar Isi

Revisi

Disediakan untuk revisi, sengaja dikosongkan

Revisi

Disediakan untuk revisi, sengaja dikosongkan

Revisi

Disediakan untuk revisi, sengaja dikosongkan

Merek Dagang

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Sebelum Memulai

- Java adalah bahasa pemrograman yang besar (dengan file spesifikasi versi 7 berisikan 670 halaman keseluruhan), dengan pustaka bawaan yang besar dan kompleks, dan sejumlah aturan yang baku.
- Fokus manual ini adalah penerapan secara praktis. Aturan yang berlaku – apabila relevan – akan langsung dituliskan. Aturan lanjutan mungkin dibahas secara terpisah (apabila diperlukan).
- Compiler dan runtime Java yang digunakan dalam manual ini adalah versi 8, namun sebagian kode program dibuat agar dapat dikompilasi di Java versi 7 (tidak menggunakan fitur yang disediakan di Java versi 8, kecuali disebutkan lain). Beberapa catatan mungkin akan diberikan, apabila pembahasan tentang kompatibilitas diperlukan.
- Manual ini disarankan untuk dibaca dari awal, secara berurut, tidak langsung pindah pada topik tertentu.
- Untuk menjaga manual ini seringkas dan sesederhana mungkin, beberapa topik pembahasan mungkin ditempatkan berdekatan dengan topik yang membutuhkan. Di lain sisi, beberapa topik mungkin dibahas sekilas terlebih dahulu dan akan dibahas lebih lanjut, apabila dibutuhkan.
- Untuk memastikan setiap bagian dapat diakhiri dengan contoh soal, topik tertentu – walau tidak terkait langsung – akan disertakan dalam bagian pembahasan tersebut.
- Manual ini dirancang sebagai pendamping training, dan oleh karenanya, langkah-langkah instalasi program tidak dibahas.
- Manual ini tidak dimaksudkan untuk menggantikan berbagai buku pemrograman Java.

Bagian 1

Memulai Java

Pengantar

- Java merupakan bahasa pemrograman general-purpose
  - Digunakan membangun aplikasi dalam berbagai domain
- Mendukung beberapa paradigma pemrograman
  - Termasuk: object-oriented berbasis class
- Sistem tipe: strong dan static
  - Static: setiap variabel dan ekspresi memiliki tipe dan diketahui saat kompilasi
  - Strong: tipe membatasi nilai yang dapat ditampung oleh variabel atau yang dapat dihasilkan oleh ekspresi, tipe membatasi operasi yang didukung, dan menentukan arti operasi

    (Java Language Specification, Bab 4)
- Bahasa pemrograman high-level
  - Representasi mesin tidak tersedia pada bahasa

Pengantar

- Umumnya: source code (*.java) dikompilasi menjadi format bytecode (*.class)
- File *.class tersebut dapat dijalankan pada sistem yang mendukung Java, tanpa perlu dilakukan kompilasi ulang (Write Once, Run Anywhere)
  - Mendukung: Java Virtual Machine tersedia untuk sistem tersebut

Pengantar

- Versi 1.0 dirilis pada tahun 1996
- Versi 8 merupakan rilis dengan Long-Term Support

Pengantar

- Platform:
  - High-performance computing
  - Server
  - Web
  - Desktop
  - Mobile
  - Embedded
  - IoT
  - (dan lainnya)
- Dalam berbagai industri

Memulai Java

- Untuk pengembangan program:
  - Java Development Kit (JDK)
  - Dapat didownload dari http://www.oracle.com/technetwork/java/
    - Atau lewat package management sistem yang Anda gunakan
  - Program untuk mengetikkan source code: editor teks atau Integrated Development Tool
- Untuk menjalankan program:
  - Java Virtual Machine / Java Runtime Environment (JRE)
  - Dapat didownload dari http://www.java.com
    - Atau lewat package management sistem yang Anda gunakan

Memulai Java

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          System.out.println("Hello, World");
4      }
5  }
```

```
javac Test.java
```

```
java Test
Hello, World
```

Catatan:
- Hampir semua statement diakhiri dengan ;
- Java membedakan huruf besar dan kecil (case-sensitive)
- Komentar dituliskan diantara /* dan */ atau diawali // sampai akhir baris

Memulai Java

Test.class

```
javap Test.class
Compiled from "Test.java"
class Test {
  Test();
  public static void main(java.lang.String[]);
}
```

Editor

- Editor teks biasa atau Integrated Development Environment?
  - Editor teks biasa: mengetik semuanya sendiri, dengan panduan dari dokumentasi
    - Cocok untuk belajar (misal: persiapan ujian)
  - IDE: umumnya dilengkapi fasilitas auto-complete, GUI builder, database manager, dan lainnya
    - Cocok untuk pekerjaan sehari-hari

Editor

HelloFrame.java

```
1  import java.awt.Dimension;
2  import javax.swing.JFrame;
3  import javax.swing.JLabel;
4  import javax.swing.SwingUtilities;
5
6  public class HelloFrame {
7      public static void main(String[] args) {
8          SwingUtilities.invokeLater(
9              new Runnable() {
10                 public void run() {
11                     JFrame frame = new JFrame("Hello");
12                     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13
14                     JLabel label = new JLabel("Hello, World");
15                     label.setPreferredSize(new Dimension(300, 100));
16                     frame.getContentPane().add(label);
17
18                     frame.pack();
19                     frame.setVisible(true);
20                 }
21             }
22         );
23     }
24 }
```

Editor

- Program serupa dengan IDE NetBeans

  (tanpa mengetik sebaris kode pun)
  - Buat proyek baru (Java Application)
  - Jangan buat Main Class
  - Pada bagian Projects → <Nama Proyek> → Source packages → <default package>
    - Klik kanan
    - Pilih menu New → JFrame Form…
    - Isikan nama class
    - Drag Label dari Palette, klik ganda pada label untuk ubah teks
  - Pilih menu Run → Run Project <Nama Proyek>
    - Pilih class JFrame tersebut ketika diminta menentukan main class

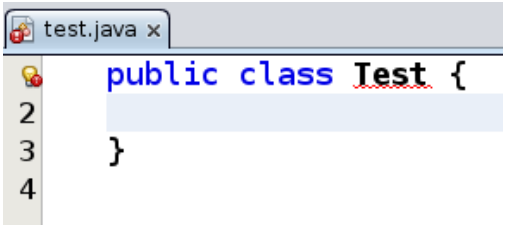
Editor

- Untuk pembahasan kita:
  - Disarankan menggunakan IDE NetBeans
  - Walaupun program yang dikembangkan berjalan tanpa GUI
  - NetBeans dapat didownload dari: http://www.netbeans.org

Nama File

- Apabila terdapat class yang dideklarasikan *public*, maka nama file harus sama dengan nama class
  - public: dapat diakses dari package atau class manapun
- Apabila tidak terdapat class yang dideklarasikan public, maka nama file bebas diberikan
- Dalam satu file, lebih dari satu class bisa dideklarasikan, namun hanya satu class yang boleh dideklarasikan public (dalam file tersebut)

Nama File




```
javac test.java
test.java:1: error: class Test is public, should be declared in a file named Test
.java
public class Test {
       ^
1 error
```

Perhatikanlah bahwa nama file adalah test.java, dan nama class adalah Test, sementara class Test dideklarasikan public (dan Java case-sensitive). Oleh karena itu:

- Gantilah nama file menjadi Test.java (disarankan)
- Ganti nama class menjadi test (tidak disarankan, karena ada konvensi nama class diawali dengan huruf kapital untuk setiap kata-nya)
- Sebagai alternatif, class Test dapat dideklarasikan non-public (default/package-private), dengan menghapus kata public dibaris pertama

Hasil Kompilasi

- Pada kompilasi yang berhasil untuk suatu file *.java, setiap class akan ditulis dalam <nama_class>.class.
- Ketika dijalankan, kita memanggil **nama class**, bukan nama file *.class tersebut.
- Berapa file *.class yang dihasilkan dalam contoh berikut?

Hasil Kompilasi

test.java

```
1  class TestSatu {
2      class TestSatuB {
3      }
4      class TestSatuC {
5      }
6      public static void main(String[] args) {
7          class TestSatuA {
8              class TestSatuAA {
9                  class TestSatuAAA {
10                     void method1() {
11                         class TestSatuAAAMethod1 {
12                             Runnable method1a() {
                                   return new Runnable() {
                                       public void run() {
15                                     }
16                                 };
17                             }
18                         }
19                     }
20                 }
21             }
22         }
23     }
24 }
25 class TestDua {
26     public static void main(String[] args) {
27     }
28 }
29 class TestTiga {
30     public static void main(String[] args) {
31     }
32 }
```

```
javac test.java
```

Hasil Kompilasi

```
TestDua.class
TestSatu$1TestSatuA.class
TestSatu$1TestSatuA$TestSatuAA.class
TestSatu$1TestSatuA$TestSatuAA$TestSatuAAA$1TestSatuAAAMethod1$1.class
TestSatu$1TestSatuA$TestSatuAA$TestSatuAAA$1TestSatuAAAMethod1.class
TestSatu$1TestSatuA$TestSatuAA$TestSatuAAA.class
TestSatu.class
TestSatu$TestSatuB.class
TestSatu$TestSatuC.class
TestTiga.class
```

Suatu program mungkin bisa membesar, dikerjakan kolaboratif, dan dibaca lagi dimasa mendatang. Akan lebih baik apabila dituliskan dengan dokumentasi (walaupun malas dilakukan), sesuai konvensi, dengan spasi yang memadai, dan dengan penggunaan file yang wajar.

Method main()

- Agar suatu program standalone dapat dijalankan, sebuah method main perlu dideklarasikan, sebagai berikut:

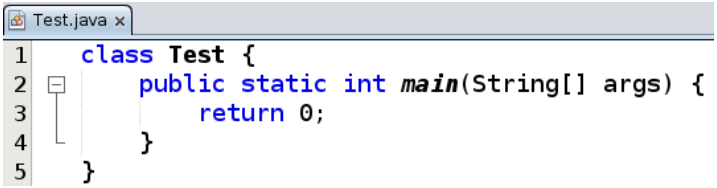
```
//salah satu dari ini
public static void main(String[] args) {
}

//atau ini
static public void main(String[] args) {
}
```

- Apabila dibutuhkan (misal untuk testing), kita bisa menambahkan method main() seperti contoh, untuk class-class yang tidak dirancang sebagai class utama aplikasi

Method main()

- Bagaimana kalau method main() yang diharapkan, tidak disediakan?



- Kompilasi tentu berhasil untuk kode tersebut

```
javac Test.java
```

- Mari kita coba jalankan:

```
java Test
Error: Main method must return a value of type void in class Test, please
define the main method as:
   public static void main(String[] args)
```

Method main()

- Bagaimana kalau terdapat lebih dari satu method main()?

Test.java ×

```
1   class Test {
2       public static void main(String[] args) {
3       }
4
5       static public void main(String[] args) {
6       }
7   }
```

```
javac Test.java
Test.java:5: error: method main(String[]) is already defined in class Test
    static public void main(String[] args) {
                       ^
1 error
```

Method main()

- Mari lebih serius! Bagaimana kalau terdapat lebih dari satu method main()?

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          System.out.println("Hello, World 0");
4      }
5
6      public static void main(String args) {
7          System.out.println("Hello, World 1");
8      }
9
10     public static void main(int[] args) {
11         System.out.println("Hello, World 2");
12     }
13
14     public static void main(Object args) {
15         System.out.println("Hello, World 3");
16     }
17
18     public static void main() {
19         System.out.println("Hello, World 4");
20     }
21 }
```

Method main()

- Kompilasi berhasil, method main() yang diharapkan akan dijalankan, karena method main() bisa di-overload

```
javac Test.java

java Test
Hello, World 0
```

- Overload: method-method dalam sebuah class yang memiliki nama sama, namun dengan signature yang berbeda

  (lebih lanjut: *Java Language Specification* 8.4.9)

  (apabila diperlukan, bacalah juga: covariant pada Java 5.0 atau yang lebih baru)

Method main()

- Overload? Contoh berikut bukanlah overloading!

Test.java

```
1   class Test {
2       public static void main(String[] args) {
3           System.out.println("Hello, World 0");
4       }
5
6       public static int main(String[] args) {
7           System.out.println("Hello, World 1");
8       }
9   }
```

```
javac Test.java
Test.java:6: error: method main(String[]) is already defined in class Test
    public static int main(String[] args) {
                      ^
1 error
```

Kompatibilitas

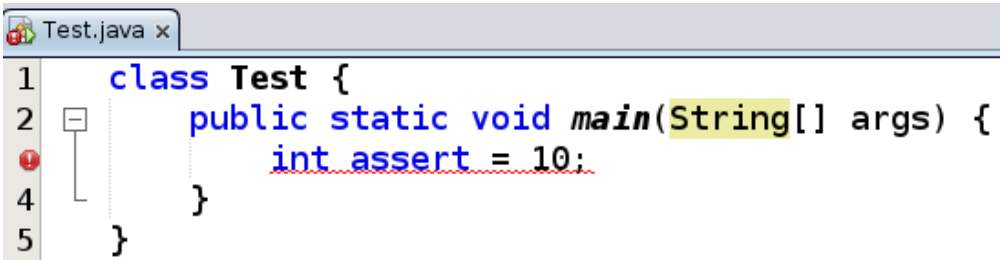
- Java memperhatikan kompatibilitas, baik pada level bahasa, virtual machine, ataupun pada API (dalam konteks penggunaan pustaka bawaan).
- Fitur baru pada level bahasa akan diusahakan agar kode yang ada tetap dapat dikompilasi dengan compiler versi baru (sebisa mungkin).
- Fitur yang akan dihapus pada pustaka bawaan akan ditandai deprecated, disarankan tidak digunakan, dan mungkin akan dihapus pada versi tertentu.

Kompatibilitas

- Kompatibilitas pada level source code bisa diset dengan opsi -source pada javac
  - Membangun program dengan source code yang kompatibel dengan Java versi tertentu. Contoh: menggunakan compiler Java versi 8, tapi source code yang diketikkan menggunakan aturan bahasa versi 1.3.
- Untuk menghasilkan file class untuk versi virtual machine tertentu, gunakan opsi -target pada javac
  - Perhatikanlah juga opsi -bootclasspath

Kompatibilitas

- Kode berikut dapat dikompilasi pada Java versi 1.3 (atau sebelumnya)



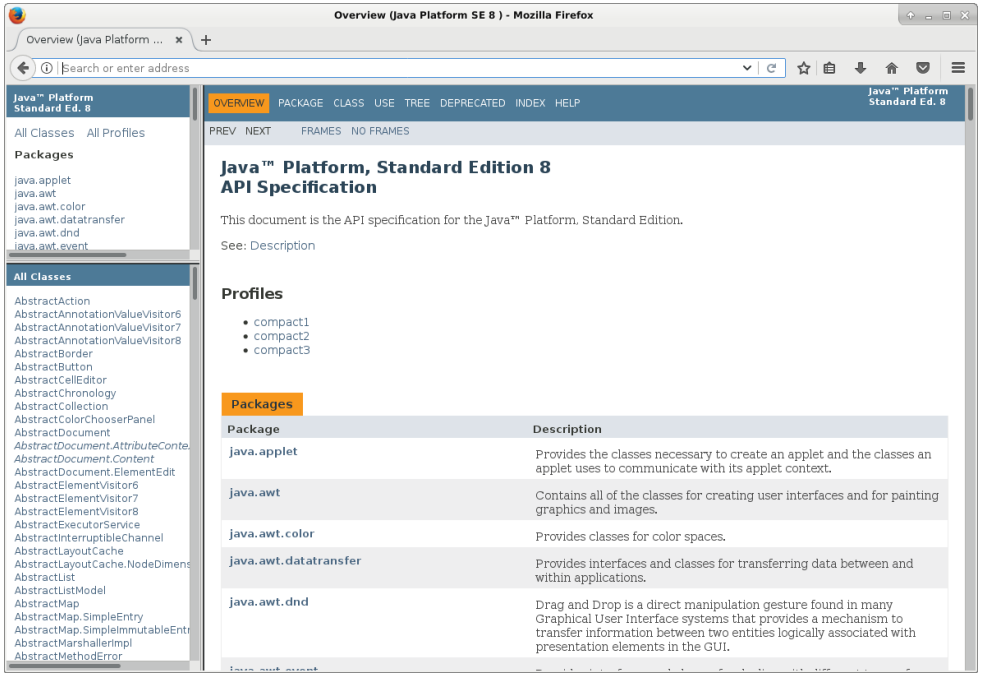
- Namun, gagal untuk versi 1.4 (atau yang lebih baru)

```
javac -Xlint:-options -source 1.4 Test.java
Test.java:3: error: as of release 1.4, 'assert' is a keyword, and may not
be used as an identifier
        int assert = 10;
            ^
  (use -source 1.3 or lower to use 'assert' as an identifier)
1 error
```

Dokumentasi

- API Specification: berisikan dokumentasi lengkap tentang class, interface, dan lainnya
  - Apabila menggunakan teks editor, maka dokumentasi ini wajib tersedia (dapat dibaca online ataupun di-download)
  - Apabila menggunakan IDE dengan auto-complete dan dokumentasi, tetap disarankan untuk dibaca
- Java Language Specification
  - Untuk memahami bahasa Java lebih lanjut

Dokumentasi



Struktur File

- Apabila file merupakan bagian dari package, maka definisi package merupakan statement pertama, diikuti oleh bagian import, dan deklarasi class
- Apabila bagian dari package default (tanpa package secara eksplisit, seperti contoh-contoh sebelumnya), maka bagian import merupakan bagian pertama, diikuti oleh deklarasi class
- Apabila bagian dari package default dan tidak ada statement import, maka deklarasi class merupakan bagian pertama
- Import semata-mata digunakan untuk menghemat pengetikan (tanpa harus mengacu ke fully-qualified name)
- Package java.lang secara otomatis diimport

Standard Output

- Untuk menulis ke standard output (terlihat di terminal atau command prompt), salah satu cara adalah dengan menggunakan method println, milik field out, dari class System, yang merupakan bagian dari package java.lang. Method ini di-overload.

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          System.out.println("Hello, World");
4
5          int i = 1;
6          System.out.println(i);
7      }
8  }
```

```
javac Test.java

java Test
Hello, World
1
```

Method main() dan static

- method main():
  - static: berada pada level class, tidak membutuhkan instance dari suatu class untuk dapat diakses
  - Method main(): dapat dijalankan tanpa instance dari class yang memiliki method main() tersebut di-instansiasi.

Method main() dan static

Test.java

```
1   class Test {
2       public static void hello() {
3           System.out.println("Hello");
4       }
5
6       public static void main(String[] args) {
7           hello();
8       }
9   }
```

```
javac Test.java

java Test
Hello
```

Andaikata hello() tidak static, maka:

Method main() dan static

Test.java

```
1   class Test {
2       public void hello() {
3           System.out.println("Hello");
4       }
5
6       public static void main(String[] args) {
7           hello();
8       }
9   }
```

```
javac Test.java
Test.java:7: error: non-static method hello() cannot be referenced from a
static context
        hello();
        ^
1 error
```

Method hello() tidak dapat di-refer dari method main() yang static. Dibutuhkan instance dari class Test.

Method main() dan static

Test.java

```
1   class Test {
2       public void hello() {
3           System.out.println("Hello");
4       }
5
6       public static void main(String[] args) {
7           Test test = new Test();
8           test.hello();
9       }
10  }
```

```
javac Test.java

java Test
Hello
```

Method static

- Gunakan nama class untuk memanggil method static

Test.java

```
1   class Test2 {
2       public static void helloStatic() {
3           System.out.println("Hello");
4       }
5
6       public void hello() {
7           System.out.println("Hello (non-static)");
8       }
9
10  }
11
12  class Test {
13      public static void main(String[] args) {
14          Test2.helloStatic();
15          new Test2().hello();
16      }
17  }
```

```
javac Test.java

java Test
Hello
Hello (non-static)
```

Method static

- Pemanggilan seperti ini akan membingungkan pembaca source code

Test.java

```
1   class Test2 {
2       public static void helloStatic() {
3           System.out.println("Hello");
4       }
5
6       public void hello() {
7           System.out.println("Hello (non-static)");
8       }
9
10  }
11
12  class Test {
13      public static void main(String[] args) {
14          Test2 test2 = new Test2();
15          test2.helloStatic();
16      }
17  }
```

Contoh Soal

- Kita telah membahas tentang menampilkan teks ke standard output. Cobalah untuk membuat program yang:
  - Menampilkan beberapa informasi sistem ke standard output
  - Informasi-informasi tersebut bisa didapatkan dengan membaca dokumentasi java.lang.System dan java.lang.Runtime

Bagian 2

Tipe Data dan Operator,

Class dan Objek,

Inisialisasi,

String,

Percabangan dan Perulangan,

Console

Tipe Data

- Tipe
  - Primitif: representasi nilai apa adanya, sesuai tipe
  - Reference
  - Null
    - tanpa nama
    - tipe untuk ekspresi null
    - anggap bahwa null adalah literal khusus → tipe reference apapun

      (*Java Language Specification* 4.1)

Tipe Data

- Primitif (8)
  - Numeric
    - Integral:
      - byte: -128 sampai 127 (8-bit)
      - short: -32.768 sampai 32.767 (16-bit)
      - int: 2.147.483.648 sampai 2.147.483.647 (32-bit)
      - long: -9.223.372.036.854.775.808 sampai 9.223.372.036.854.775.807 (64-bit)
        - diakhiri l atau L
      - char: '\u0000' sampai '\uffff' (0 sampai 65535, 16-bit unsigned, UTF-16 code unit)
        - satu karakter diantara kutip tunggal
    - Floating point
      - float: 32-bit IEEE 754, 1.40e-45f sampai 3.4028235e38f
        - diakhiri f atau F
      - double: 64-bit IEEE 754, 4.9e-324 sampai 1.7976931348623157e308
        - diakhiri digit, d, atau D
  - boolean: true, false

Tipe Data

- Untuk integer, representasi nilai dapat diberikan dalam desimal, oktal, heksadesimal, dan binari
  - Oktal: diawali dengan 0, digit 0 sampai 7
  - Heksadesimal: diawali dengan 0x atau 0X, digit 0 sampai 9, dan a (atau A) sampai f (atau F)
  - Binari: diawali dengan 0b atau 0B, digit 0 dan 1
- Penulisan nilai dapat dipisahkan dengan underscore (_), namun tidak dapat ditempatkan di awal atau diakhir
  - Contoh:
    - int i = 1_000_000_000
    - int j = 1_2_3_4_5

Tipe Data

- Reference
  - class
  - interface
  - type variable
  - array

Tipe Data

- Wrapper primitif ke reference
  - boolean → Boolean
  - byte → Byte
  - short → Short
  - char → Character
  - int → Integer
  - long → Long
  - float → Float
  - double → Double

Identifier

- Pengenal atau nama yang legal
- Aturan:
  - Membedakan huruf besar dan kecil (case-sensitive)
  - Panjang tidak dibatasi, terdiri dari kombinasi huruf (Java letter) dan angka (Java digit), harus dimulai dengan huruf (Java letter)
    - Java letter: unicode termasuk A-Z, a-z, _, $
  - Tidak dapat menggunakan keyword dalam bahasa Java

  (*Java Language Specification* 3.8)
- Identifier dituliskan di sebelah kanan tipenya
  - Contoh: int a, String s
- Assignment sederhana dengan operator =, dengan ekspresi di sisi kanan
  - Contoh: a = 10

  (*Java Language Specification* 15.26)

Konvensi

- Nama class dan interface: huruf pertama dalam setiap kata adalah huruf besar
  - Class: umumnya merupakan kata benda
  - Interface: umumnya merupakan kata sifat
  - Contoh: Thread (class), Runnable (interface), AutoCloseable (interface)
- Nama method: huruf pertama adalah huruf kecil, kata-kata berikutnya merujuk pada kesepakatan nama class/interface
  - Umumnya merupakan kata kerja
  - Contoh: getResultSet
- Nama field dan variabel: seperti nama method, dan umumnya merupakan kata benda atau singkatan
- Nama konstanta: semuanya huruf besar, dengan satu atau lebih kata atau singkatan dipisahkan dengan underscore
  - Contoh: MAX_VALUE

Keyword

abstract	continue	for	new	switch
assert	default	if	package	synchronized
boolean	do	goto	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Class dan Object

- Cetak biru atau template suatu objek
  - Class dideklarasikan dengan keyword class
  - Dapat memiliki field: mewakili state suatu objek
  - Dapat memiliki method: behavior, digunakan untuk interaksi dengan dunia luar
  - Class/Objek dan field/method dituliskan dengan tanda . (titik) sebagai pemisah
- Ketika suatu objek dibuat → instance dari class
- Di Java:
  - Setiap class merupakan turunan dari java.lang.Object (langsung ataupun tidak)
  - Setiap class hanya boleh memiliki satu superclass langsung (single inheritance)
  - Tanpa adanya superclass secara eksplisit, suatu class secara implisit merupakan subclass dari java.lang.Object

Package

- Program dapat terdiri dari nol (tidak secara eksplisit; tidak memiliki nama; default package) sampai sejumlah package dan subpackage, yang mana masing-masing memiliki penamaan tersendiri (diantaranya: untuk pengelompokan fungsi dan menghindari konflik nama)
- Penamaan dengan nama domain Internet organisasi (yang dibalik) ataupun penamaan sederhana, dengan pemisah subpackage berupa karakter . (titik)
  - Contoh:
    - com.noprianto.test
    - test
- Penamaan pada pustaka bawaan Java: diawali dengan java dan javax (misal: java.sql atau javax.swing)

Package

- Tanpa package secara eksplisit

Test.java

```
1   class Test {
2       public static void main(String[] args) {
3           System.out.println("Hello, World");
4       }
5   }
```

```
javac Test.java
```

Package

- Dengan penamaan sederhana

```
1 package test;
2
3 public class Test {
4     public static void main(String[] args) {
5         System.out.println("Hello, World");
6     }
7 }
```

javac test/Test.java

Sesuaikanlah dengan sistem operasi yang Anda gunakan

Package

- Dengan nama domain Internet (dibalik)

```
1 package com.noprianto.test;
2
3 public class Test {
4     public static void main(String[] args) {
5         System.out.println("Hello, World");
6     }
7 }
```

javac com/noprianto/test/Test.java

Sesuaikanlah dengan sistem operasi yang Anda gunakan

Package

- Penggunaan class dalam package: fully-qualified name

```
java test.Test
Hello, World

java com.noprianto.test.Test
Hello, World
```

Perhatikanlah bahwa kompilasi dengan javac dan menjalankan program dengan java, dalam contoh ini dilakukan di direktori yang mengandung direktori *test* dan *com*. Perhatikanlah juga bahwa untuk menjalankan program tersebut, dibutuhkan nama class dan bukan nama file.

Sebagai latihan, cobalah juga untuk menggunakan opsi -d pada saat kompilasi dengan javac.

Access Modifier

- Sebuah class (beserta field dan method-nya), dapat dirancang agar hanya dapat diakses oleh class itu sendiri, subclass, ataupun class manapun, baik dalam package yang sama ataupun berbeda
  - Akses pada field atau method suatu class
  - Menurunkan dari superclass
- Ada perbedaan antara modifier yang dapat diterapkan pada class dan field/method-nya
- Access modifier: public, protected, default (tanpa disebutkan, atau package-private), dan private
- Kita diharapkan dapat merancang dengan baik, agar suatu class memiliki fungsi/fokus spesifik (high cohesion) dan tidak banyak tergantung dengan class lain (loose coupling)
  - Class yang kita buat mungkin akan digunakan oleh programmer lain. Detil dan cara kerja class tidak perlu diakses/diketahui oleh pengguna class (encapsulation), sehingga, apabila perlu dilakukan perubahan, kode yang mengandalkan class tersebut masih dapat berfungsi.

Catatan: superclass dan subclass akan dibahas pada bagian 3

Access Modifier

- Pada class, kita bisa terapkan modifier public atau default (tanpa disebutkan)
- Pada anggota class, kita bisa terapkan modifier public, protected, default (tanpa disebutkan), dan private
- Hak akses:
  - Dari dalam class (class itu sendiri): semua bisa diakses
  - Dari class lain/subclass dalam package yang sama: semua bisa diakses, kecuali private
  - Dari subclass package lain: hanya public dan protected (subclass)
  - Dari class package lain: hanya public

Access Modifier

- Beberapa saran:
  - Yang tidak perlu diketahui dari luar class: private
  - Setter dan getter:
    - Setter: yang perlu diset dari luar, buat method public, dengan nama diawali set (contoh: setName). Di dalam method tersebut, kita bisa hanya sekedar mengassign ke field tertentu atau melakukan tindakan tambahan.
    - Getter: yang perlu didapatkan dari luar, buat method public, dengan nama diawali get (contoh: getName). Sama seperti setter, kita bisa sekedar mengembalikan nilai ataupun melakukan tindakan tambahan.
  - Apabila memungkinkan, fungsionalitas (API: Application Programming Interface) yang kita sediakan untuk pihak luar, dapat dibuat se-stabil mungkin. Hargailah juga kompatibilitas.

Inisialisasi

- Ketika langkah-langkah inisialisasi dalam class diperlukan, kita bisa tempatkan dalam:
  - Static initialization: pada saat class pertama di-load, dikerjakan berurutan sesuai terdefinisi (apabila lebih dari satu)
  - Instance initialization: setiap kali instance baru dari class tersebut dibuat, juga dikerjakan berurutan sesuai terdefinisi. Dapat digunakan apabila ada kode-kode yang akan selalu dikerjakan oleh *setiap* constructor.
  - Constructor
    - Nama harus sama dengan nama class
    - Tidak memiliki tipe kembalian (void juga tidak boleh, karena void merupakan tipe kembalian)
    - Tanpa argumen atau dengan argumen
    - Dapat dipanggil dari luar ataupun tidak (private)
    - Dapat memanggil constructor lain dalam class (this) ataupun constructor superclass (super). Salah satu dari keduanya harus merupakan statement pertama dalam suatu constructor (apabila digunakan).
    - Apabila superclass tidak memiliki constructor tanpa argumen, subclass perlu sediakan constructor, dan panggil constructor superclass yang tepat didalamnya
    - Apabila sebuah class tidak memiliki constructor, compiler akan sediakan

Inisialisasi

- Static initializer

```
static {
}
```

Test.java

```
1  class Test {
2      static {
3          System.out.println("Class Test di-load");
4          System.out.println("Hello");
5      }
6
7      static {
8          int i = 1;
9          System.out.println(i);
10     }
11
12     public static void main(String[] args) {
13         Test test1 = new Test();
14         Test test2 = new Test();
15     }
16 }
```

```
javac Test.java

java Test
Class Test di-load
Hello
1
```

Dua instance dibuat: setiap static init block hanya dikerjakan satu kali!

Inisialisasi

- Instance initializer

```
{
}
```

Test.java

```
1  class Test {
2      {
3          System.out.println("Instance test dibuat");
4          System.out.println("Hello");
5      }
6
7      {
8          int i = 1;
9          System.out.println(i);
10     }
11
12     public static void main(String[] args) {
13         Test test1 = new Test();
14         Test test2 = new Test();
15     }
16 }
```

```
javac Test.java

java Test
Instance test dibuat
Hello
1
Instance test dibuat
Hello
1
```

Dikerjakan setiap instance baru dibuat

Inisialisasi

- Constructor

Test.java

```
1  class Test {
2      Test() {
3          System.out.println("Constructor");
4      }
5
6      Test(String name) {
7          this();
8          System.out.println("Constructor lain telah dipanggil");
9          System.out.println("Constructor: " + name);
10     }
11
12     private Test(int i) {
13         System.out.println("Constructor private: " + i);
14     }
15
16     void Test() {
17         System.out.println("Method biasa, bukan Constructor");
18     }
19
20     public static void main(String[] args) {
21         Test test1 = new Test();
22         Test test2 = new Test("A");
23         Test test3 = new Test(10);
24         test1.Test();
25     }
26 }
```

```
javac Test.java

java Test
Constructor
Constructor
Constructor lain telah dipanggil
Constructor: A
Constructor private: 10
Method biasa, bukan Constructor
```

Perhatikanlah pemanggilan this() dan penggunaan tipe kembalian void!

Inisialisasi

- Urutan inisialisasi

Test.java

```
1   class Test {
2       Test() {
3           System.out.println("Constructor");
4       }
5
6       Test(String name) {
7           this();
8           System.out.println("Constructor lain telah dipanggil");
9           System.out.println("Constructor: " + name);
10      }
11
12      private Test(int i) {
13          System.out.println("Constructor private: " + i);
14      }
15
17      {
18          System.out.println("Instance initializer");
19      }
20
21      static {
22          System.out.println("Static initializer");
23      }
24
25      public static void main(String[] args) {
26          Test test1 = new Test();
27          Test test2 = new Test("A");
28          Test test3 = new Test(10);
29      }
    }
```

```
javac Test.java

java Test
Static initializer
Instance initializer
Constructor
Instance initializer
Constructor
Constructor lain telah dipanggil
Constructor: A
Instance initializer
Constructor private: 10
```

Instansiasi

- Dilakukan dengan kata kunci new, apabila berhasil, maka objek baru untuk class tersebut akan tersedia
- Class dapat dirancang agar hanya memiliki satu instance dalam suatu aplikasi (singleton design pattern)

Instansiasi

Test.java

```
1   class TestPrivate {
2       private TestPrivate() {
3           System.out.println("Private Constructor");
4       }
5   }
6
7   class Test {
8       public static void main(String[] args) {
9           TestPrivate test = new TestPrivate();
10      }
11  }
```

```
javac Test.java
Test.java:9: error: TestPrivate() has private access in TestPrivate
        TestPrivate test = new TestPrivate();
                           ^
1 error
```

Field

- Didefinisikan di dalam class, namun di luar method
  - Variabel instance: unik untuk setiap objek
  - Static atau variabel class: pada level class, dapat diubah oleh setiap objek (atau tanpa objek)
    - Modifier static
    - Dapat ditambahkan modifier final → konstanta
  - Dapat diterapkan access modifier

Field

Test.java

```
1   class Student {
2       private String name;
3
4       Student(String s) {
5           name = s;
6           System.out.println(name);
7       }
8   }
9
10  class Test {
11
12      public static void main(String[] args) {
13          Student student1 = new Student("A");
14          Student student2 = new Student("B");
15      }
16  }
```

```
javac Test.java

java Test
A
B
```

Kita membuat dua instance dari Student, student1 dan student2, dan masing-masing diberikan nama yang berbeda (A dan B). Kedua instance memiliki variabel name yang berbeda satu dengan lainnya

Field

Test.java

```
1   class Test {
2       public static int number = 1;
3
4       public static void main(String[] args) {
5           System.out.println(Test.number);
6
7           Test.number = 2;
8           System.out.println(Test.number);
9
10          Test test1 = new Test();
11          test1.number = 3;
12          System.out.println(Test.number);
13
14          Test test2 = new Test();
15          System.out.println(Test.number);
16      }
17  }
```

```
javac Test.java

java Test
1
2
3
3
```

Terdapat variabel number pada level class (static). Kita bisa ubah tanpa adanya instance (baris 7), atau dengan instance (baris 11, membingungkan ketika dibaca), dan perubahan akan berpengaruh pada instance lain (baris 15).

Field

Test.java

```
1   class Test {
2       public static final int MAX_LEVEL = 10;
3
4       public static void main(String[] args) {
5           System.out.println(MAX_LEVEL);
6
7           MAX_LEVEL = 11;
8       }
9   }
```

```
javac Test.java
Test.java:7: error: cannot assign a value to final variable MAX_LEVEL
        MAX_LEVEL = 11;
        ^
1 error
```

Kompilasi gagal karena kita mengassign sebuah nilai ke variabel yang final. Variabel ini dimaksudkan sebagai konstanta, dan sesuai konvensi, diberikan nama dengan huruf besar semua dengan underscore sebagai pemisah kata.

Nilai Default

- Setiap field akan diberikan nilai default sebagai berikut:
  - byte: nol → (byte) 0
  - short: nol → (short) 0
  - int: nol → 0
  - long: nol → 0L
  - float: nol (positif) → 0.0f
  - double: nol (positif) → 0.0d
  - char: karakter null → ‘\u0000’
  - boolean: false
  - Untuk reference: null

  (*Java Language Specification* 4.12.5)

Nilai Default

Test.java

```
1   class Test {
2       byte b;
3       short s;
4       int i;
5       long n;
6       float f;
7       double d;
8       char c;
9       boolean t;
10
11      public static void main(String[] args) {
12          Test test = new Test();
13          System.out.println(test.b);
14          System.out.println(test.s);
15          System.out.println(test.i);
16          System.out.println(test.n);
17          System.out.println(test.f);
18          System.out.println(test.d);
19          System.out.println(test.c);
20          System.out.println(test.t);
21      }
22  }
```

```
javac Test.java

java Test
0
0
0
0
0.0
0.0

false
```

Nilai Default

- Perhatikanlah bahwa variabel local harus secara eksplisit diberikan nilai (dengan inisialisasi atau assignment) sebelum digunakan. Tidak ada nilai default yang diberikan.

Test.java ×

```
1   class Test {
2       public static void main(String[] args) {
3           int i;
4       }
5   }
```

Kompilasi berhasil, walau variabel i tidak diberikan nilai, karena variabel tersebut tidak digunakan.

Test.java ×

```
1   class Test {
2       public static void main(String[] args) {
3           int i;
4
5           System.out.println(i);
6       }
7   }
```

```
javac Test.java
Test.java:5: error: variable i might not have been initialized
        System.out.println(i);
                           ^
1 error
```

Method

- Dapat menerima parameter ataupun tidak, dipisahkan koma untuk setiap parameter, dan parameter tersebut memiliki tipe (dengan tipe dituliskan terlebih dahulu)
  - Pemanggilan method diawali dengan (, diikuti dengan daftar argumen yang sesuai, diakhiri dengan)
  - atau () apabila tanpa argumen
- Dapat diterapkan access modifier
- Pada level class: dengan modifier static
- Mengembalikan nilai dengan tipe tertentu (dengan return)
  - Apabila tidak ada nilai yang dikembalikan, gunakanlah void
- Dapat mendeklarasikan exception yang dapat terjadi
- Dapat memiliki isi (diawali {, diakhiri }) atau tanpa implementasi (; setelah deklarasi) → lihatlah pembahasan pada bagian 3
- Dapat di-overload, di-override, atau didefinisikan ulang → lebih lanjut di bagian 3

  (Lebih lanjut, bacalah *Java Language Specification* 8.4)
- Terminologi (keduanya saling digunakan dalam manual ini):
  - Parameter: deklarasi method
  - Argumen: pada saat pemanggilan method

Method

Test.java

```
1   class Test {
2       void method1() {
3           String res = method2("Hello ", 123);
4           System.out.println(res);
5       }
6
7       public String method2(String s, int i) {
8           return s + i;
9       }
10
11      public static void main(String[] args) {
12          new Test().method1();
13      }
14  }
```

```
javac Test.java

java Test
Hello 123
```

Catatan:

- Method method1() tidak memiliki argumen dan tidak mengembalikan nilai. Di dalamnya, method2() dipanggil dengan dua argumen: String dan int, yang mengembalikan nilai dengan tipe String (yang kemudian kita tampilkan pada standard output).
- Method method1() memiliki hak akses default, method2() memiliki hak akses public, sementara method main() public dan static.

Operator

=	>	<	!	~	?	:				
==	<=	>=	!=	&&	||	++	--			
+	-	*	/	&	|	^	%	<<	>>	>>>
+=	-=	*=	/=	&=	|=	^=	%=	<<=	>>=	>>>=

Operator

- Equality
  - Diterapkan pada numerik, boolean, reference
  - Dengan operator == (sama dengan) atau != (tidak sama dengan)
    - Pada primitif: membandingkan nilai
    - Pada reference:
      - == → apabila kedua variabel null, maka true
      - == → apabila kedua variabel merujuk pada objek yang sama, maka true
  - Dengan method equals (milik java.lang.Object), yang mana perbandingan didasarkan pada *nilai* yang dianggap penting/menentukan
    - Gunakan operator ! untuk 'not' equal

(*Java Language Specification* 15.21)

Operator

Test.java

```
1   class Test {
2       public static void main(String[] args) {
3           String o = "o";
4
5           int a = 10;
6           int b = 20;
7           String c = "Hello";
8           String d = c;
9           String e = "H" + "e" + "l" + "l" + o;
10          System.out.println(a == b);
            System.out.println(c == d);
            System.out.println(c == e);
13          System.out.println(c.equals(e));
14          System.out.println(!c.equals(e));
15      }
16  }
```

```
javac Test.java

java Test
false
true
false
true
false
```

Catatan:

- Baris 10 membandingkan primitif a dan b, karena a tidak sama dengan b, maka false
- Baris 11 membandingkan apakah dua variabel merujuk pada objek yang sama
- Baris 12 bekerja dengan variabel e, yang pada dasarnya adalah String “Hello”, sama dengan c. Namun, e didapatkan dari penggabungan literal String dan variabel o pada saat runtime. Keduanya tidak lagi sama (String intern). Gantilah o dengan “o” dan bandingkanlah. (*Java Language Specification* 3.10.5)
- Baris 13 adalah contoh penggunaan method equals()

Operator

- Relasional
  - Diterapkan pada numerik: < (lebih kecil), <= (lebih kecil atau sama dengan), > (lebih besar), dan >= (lebih besar atau sama dengan)
  - Diterapkan pada reference atau null: instanceof → Dibahas pada bagian 3

(*Java Language Specification* 15.20)

Operator

Test.java

```
1  import java.io.Serializable;
2
3  class Test implements Serializable {
4      public static void main(String[] args) {
5          int a = 10;
6          int b = 10;
7
8          Test test = new Test();
9
10         System.out.println(a > b);
11         System.out.println(a >= b);
12         System.out.println(test instanceof Test);
13         System.out.println(test instanceof Object);
14         System.out.println(test instanceof Serializable);
15         System.out.println(null instanceof Test);
16     }
17 }
```

```
javac Test.java

java Test
false
true
true
true
true
false
```

Catatan:

- Pada baris 10 dan 11, kita membandingkan dua nilai numerik 10 > 10 (false) dan 10 >= 10 (true)
- Pada baris 12, kita membandingkan apakah objek test merupakan instance dari Test, yang mana adalah true (baris 8)
- Pada baris 13, kita mengetahui bahwa objek test juga adalah turunan dari Object
- Pada baris 14 – yang akan kita bahas lebih lanjut pada bagian 3 – karena test adalah instance dari Test (yang mengimplementasikan interface Serializable), maka test juga sebuah Serializable
- Pada baris 15, null bisa digunakan sebagai operand instanceof

Operator

- Penjumlahan
  - Diterapkan pada numerik primitif: + (penjumlahan) dan - (pengurangan)
  - Diterapkan pada String: + (penggabungan)

  (*Java Language Specification* 15.18)
- Multiplikatif
  - Diterapkan pada numerik primitif: * (perkalian), / (pembagian), % (sisa bagi)

  (*Java Language Specification* 15.17)
- Evaluasi dari kiri ke kanan, gunakanlah (dan) untuk pengelompokan ekspresi

Operator

- Increment/Decrement
  - Diterapkan pada numerik (tidak harus primitif): prefix ++, postfix ++, prefix --, postfix –
  - Prefix: dikerjakan sebelum nilai digunakan
    - Contoh: ++i, --i
  - Postfix: dikerjakan setelah nilai digunakan
    - Contoh: i++, i--
  - Tidak dapat diterapkan pada variabel yang final
- Minus
  - Diterapkan pada numerik (tidak harus primitif): -

(*Java Language Specification* 15.15)

Operator

Test.java

```
1   class Test {
2       public static void main(String[] args) {
3           int i = 0;
4           System.out.println(++i);
5           System.out.println(i++);
6           System.out.println(i);
7           System.out.println(-i);
8
9           System.out.println(--i);
10          System.out.println(i--);
11          System.out.println(i);
12
13          Integer j = 0;
14          System.out.println(j++);
15          System.out.println(-j);
16      }
17  }
```

```
javac Test.java

java Test
1
1
2
-2
1
1
0
0
-1
```

Catatan:

- Awalnya i bernilai 0. Pada baris 4, kita tambahkan dengan 1 dan tampilkan (output: 1). Pada baris 5, kita tampilkan (output: 1) dan tambahkan dengan 1. Pada baris 6, ketika ditampilkan, i bernilai 2. Pada baris 7, kita minuskan dan tampilkan (output: -2), namun tidak diassign ke variabel manapun.
- Pada baris 9, dengan i kembali ke 2, kita kurangi dengan 1 dan tampilkan (output: 1). Pada baris 10, kita tampilkan (output: 1) dan kurangi dengan 1. Pada baris 11, ketika ditampilkan, i bernilai 0.
- Pada baris 13, 14, dan 15, kita melihat bahwa kita bisa bekerja dengan reference

Operator

- Bitwise dan Logikal
  - Diterapkan pada numerik primitif: & (AND), ^ (exclusive OR), | (inclusive OR)
  - Diterapkan pada boolean atau Boolean: & (AND → true apabila kedua operand true), ^ (exclusive OR → true apabila kedua operand berbeda), | (inclusive OR → false apabila kedua operand false)

(*Java Language Specification* 15.22)

Operator

- Kondisional
  - Diterapkan pada boolean atau Boolean: && (AND), seperti &, namun operand di sebelah kanan hanya dievaluasi apabila operand di sebelah kiri bernilai true!
  - Diterapkan pada boolean atau Boolean: || (OR), seperti |, namun operand di sebelah kanan hanya dievaluasi apabila operand di sebelah kiri bernilai false!

(*Java Language Specification* 15.23 dan 15.24)

Operator

Test.java

```
1  class Test {
2      public static boolean testTrue() {
3          System.out.println("testTrue");
4          return true;
5      }
6
7      public static boolean testFalse() {
8          System.out.println("testFalse");
9          return false;
10     }
11
12     public static void main(String[] args) {
13         boolean t = false;
14         if (t == true & testTrue() == true) {
15             System.out.println("true true");
16         }
17
18         if (t == true && testTrue() == true) {
19             System.out.println("true true");
20         }
21     }
22 }
```

```
javac Test.java

java Test
testTrue
```

Catatan:

- Perhatikanlah bahwa t bernilai false
- Pada baris 14, method testTrue() dipanggil (dan menampilkan testTrue di standard output), walaupun t == true bernilai false. Ini karena kita menggunakan operator &.
- Pada baris 18, method testTrue() tidak dipanggil karena t == true sudah bernilai false, dan kita menggunakan operator &&
- Percabangan if akan dibahas setelah ini

String

- Instance dari class String merepresentasikan deretan code point Unicode
  - Literal dituliskan diantara kutip ganda. Contoh: “hello”
- Class java.lang.String diturunkan langsung dari java.lang.Object dan merupakan class final (tidak dapat diturunkan → dibahas di bagian 3)
- Perhatikanlah bahwa nilai sebuah String tidak dapat diubah begitu dibuat
- Untuk penggabungan string, Java menyediakan operator +

  (*Java Language Specification* 4.3.3)

  (Java *API Specification* java.lang.String)

String

- Beberapa contoh method:
  - Untuk mendapatkan panjang String, kita bisa gunakan method length()
  - Untuk mengetahui apakah sebuah String kosong (panjang 0), kita juga bisa gunakan isEmpty()
  - Untuk membandingkan String, kita bisa gunakan method equals() atau equalsIgnoreCase()

String

Test.java

```
1   class Test {
2       public static void main(String[] args) {
3           String s1 = "Hello";
4           String s2 = "hello";
5
6           System.out.println("Panjang " + s1 + ": " + s1.length());
7           System.out.println(s1 + " equals " + s2 + ": " + s1.equals(s2));
8           System.out.println(s1 + " equalsIgnoreCase " + s2 + ": "
9                   + s1.equalsIgnoreCase(s2));
10      }
11  }
```

```
javac Test.java

java Test
Panjang Hello: 5
Hello equals hello: false
Hello equalsIgnoreCase hello: true
```

String

- Untuk objek seperti String, namun dapat diubah langsung isinya, gunakanlah java.lang.StringBuilder
- Contoh method:
  - append() (di-overload)
  - delete()
  - insert() (di-overload)
  - length()
  - reverse()
  - toString()

String

Test.java

```
1   class Test {
2       public static void main(String[] args) {
3           StringBuilder builder = new StringBuilder("Hello");
4
5           builder.append(" world ");
6           builder.delete(0, 1);
7           builder.insert(0, "h");
8
9           String res = builder.toString();
10
11          System.out.println(res);
12      }
13  }
```

```
javac Test.java

java Test
hello world
```

Catatan:

- Pertama-tama, kita instansiasi sebuah StringBuilder dengan nilai awal adalah “Hello”
- Kemudian, kita tambahkan String “ world “
- Lalu, kita hapus dari indeks 0 sampai 1 (eksklusif)
- Dan, kita insert pada indeks 0 sebuah String “h”
- Walaupun dalam contoh ini tidak diperlukan, kita instansiasi sebuah String sesuai apa yang dikembalikan oleh toString()

Percabangan

- if-then

  if (expression) statement

- if-then-else

  if (expression) statementNoShortIf else statement

  if (expression) statementNoShortIf else statementNoShortIf

  (*Java Language Specification* 14.9)

- Catatan:
  - Expression: boolean atau Boolean
  - Statement bisa berupa satu baris statement atau sebuah blok (diantara { dan })
  - Gunakanlah (dan) untuk pengelompokan dalam ekspresi
  - Apabila expression bernilai true, maka statement dikerjakan

Percabangan

Test.java

```
1  class Test {
2      public static boolean testFalse() {
3          return false;
4      }
5
6      public static void main(String[] args) {
7          int a = 0;
8          if (a >= 0) {
9              System.out.println("a >= 0");
10         }
11
12         int b = 0;
13         if (b < 0) {
14             System.out.println("b < 0");
15         } else {
16             System.out.println("b tidak < 0");
17         }
18
19         if (testFalse() || (true | false)) {
20             System.out.println("method || (true | false)");
21         }
22
23         int c = 8;
24         if (c > 10) {
25             System.out.println("c > 10");
26         } else if (c > 5) {
27             System.out.println("c > 5");
28         } else if (c > 3) {
29             System.out.println("c > 3");
30         } else {
31             System.out.println("c lain");
32         }
33     }
34 }
```

```
javac Test.java

java Test
a >= 0
b tidak < 0
method || (true | false)
c > 5
```

Catatan:

- Baris 8 adalah contoh if-then
- Baris 13 adalah contoh if-then-else
- Baris 19 adalah contoh pemanggilan method (mengembalikan false), operator ||, dan penggunaan ( )
- Baris 24 adalah contoh if-then-else dengan if

Percabangan

- switch

  switch (expression) SwitchBlock

  (*Java Language Specification* 14.11)

- Catatan:
  - Tipe expression: char, byte, short, int, Character, Byte, Short, Integer, String atau Enum (Enum dibahas di bagian 3)
  - Di dalam switch block, bisa terdapat switch label yang ditandai case atau default:
    - case constantExpression:
      - Perhatikanlah bahwa ekspresi haruslah konstanta
    - case enumConstantName:
    - default:
      - Maksimum satu default
  - Apabila expression terevaluasi null, maka sebuah Exception (dibahas di bagian 3) akan terjadi
  - Apabila tidak ada case yang cocok, maka default (apabila ada) akan dikerjakan
  - Untuk setiap label, apabila tidak ada break, maka akan statement akan dilanjutkan ke label lain

Percabangan

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          int a = 0;
4
5          switch (a) {
6              case 0: {
7                          System.out.println("a == 0");
8                          System.out.println("jangan lupa break");
9                      }
10             case 1: System.out.println("a == 1, jangan lupa break");
11             default: System.out.println("a != 0 atau a != 1");
12         }
13     }
14 }
```

```
javac Test.java

java Test
a == 0
jangan lupa break
a == 1, jangan lupa break
a != 0 atau a != 1
```

Perhatikanlah bahwa kita lupa memberikan break

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          int a = 0;
4
5          switch (a) {
6              case 0: {
7                          System.out.println("a == 0");
8                          System.out.println("jangan lupa break");
9                          break;
10                     }
11             case 1: System.out.println("a == 1, jangan lupa break"); break;
12             default: System.out.println("a != 0 atau a != 1");
13         }
14     }
15 }
```

```
javac Test.java

java Test
a == 0
jangan lupa break
```

Percabangan

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          int a = 2;
4
5          switch (a) {
6              case 0: System.out.println("a == 0"); break;
7              case 1: System.out.println("a == 1"); break;
8          }
9      }
10 }
```

```
javac Test.java

java Test
```

Apabila diperlukan, gunakanlah default

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          int a = 2;
4
5          switch (a) {
6              case 0: System.out.println("a == 0"); break;
7              case 1: System.out.println("a == 1"); break;
8              default: System.out.println("a != 0, a != 1");
9          }
10     }
11 }
```

```
javac Test.java

java Test
a != 0, a != 1
```

Percabangan

Test.java ×

```
1   class Test {
2       public static void main(String[] args) {
3           Integer a = null;
4
5           switch (a) {
6               case 0: System.out.println("a == 0"); break;
7               case 1: System.out.println("a == 1"); break;
8           }
9       }
10  }
```

```
javac Test.java

java Test
Exception in thread "main" java.lang.NullPointerException
        at Test.main(Test.java:5)
```

Perhatikanlah bahwa a: null

Test.java ×

```
1   class Test {
2       public static void main(String[] args) {
3           long a = 0;
4
5           switch (a) {
6               case 0: System.out.println("a == 0"); break;
7               case 1: System.out.println("a == 1"); break;
8           }
9       }
10  }
```

Perhatikanlah bahwa a adalah sebuah long

```
javac Test.java
Test.java:5: error: incompatible types: possible lossy conversion from long to i
nt
        switch (a) {
                ^
1 error
```

Percabangan

Test.java ×

```
1   class Test {
2       public static void main(String[] args) {
3           String s = "hello";
4           String hello = "hello";
5
6           switch (s) {
7               case hello: System.out.println(hello); break;
8           }
9       }
10  }
```

```
javac Test.java
Test.java:7: error: constant string expression required
            case hello: System.out.println(hello); break;
                 ^
1 error
```

Perhatikanlah bahwa hello bukanlah konstanta. Dua contoh berikut bisa digunakan.

Test.java ×

```
1   class Test {
2       public static void main(String[] args) {
3           String s = "hello";
4
5           switch (s) {
6               case "hello": System.out.println("hello"); break;
7           }
8       }
9   }
```

Test.java ×

```
1   class Test {
2       public static void main(String[] args) {
3           String s = "hello";
4           final String hello = "hello";
5
6           switch (s) {
7               case hello: System.out.println(hello); break;
8           }
9       }
10  }
```

Perulangan

- while

  while (expression) statement

  (*Java Language Specification* 14.12)

- Catatan:
  - Expression: boolean atau Boolean
  - Statement bisa berupa satu baris statement atau sebuah blok (diantara { dan })
  - Gunakanlah (dan) untuk pengelompokan dalam ekspresi
  - Selama expression bernilai true, maka statement dikerjakan

Perulangan

Test.java

```
1   class Test {
2       public static void main(String[] args) {
3           int a = 0;
4
5           while (a < 3) System.out.println(++a);
6       }
7   }
```

```
javac Test.java

java Test
1
2
3
```

Perulangan

- do

  do statement while (expression)

  (*Java Language Specification* 14.13)

- Catatan:
  - Expression: boolean atau Boolean
  - Statement bisa berupa satu baris statement atau sebuah blok (diantara { dan })
  - Gunakanlah (dan) untuk pengelompokan dalam ekspresi
  - Selama expression bernilai true, maka statement dikerjakan. Atau, statement akan dikerjakan sampai expression bernilai false.
    - Karena pemeriksaan dilakukan di akhir, maka statement akan dikerjakan paling tidak satu kali

Perulangan

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          int a = 0;
4
5          do {
6              System.out.println(a);
7          } while (a > 0);
8      }
9  }
```

```
javac Test.java

java Test
0
```

Perhatikanlah bahwa a (bernilai 0) adalah false untuk kondisi a > 0. Namun statement tetap dikerjakan satu kali.

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          int a = 5;
4
5          do {
6              System.out.println(a--);
7          } while (a > 0);
8      }
9  }
```

```
javac Test.java

java Test
5
4
3
2
1
```

- a=5, tampil 5, -1 (4), true (>0)
- a=4, tampil 4, -1 (3), true (>0)
- a=3, tampil 3, -1 (2), true (>0)
- a=2, tampil 2, -1 (1), true (>0)
- a=1, tampil 1, -1 (0), false (>0), keluar

Perulangan

- for

  for (init; expression; update) statement

  (*Java Language Specification* 14.14.1)

- Catatan:
  - Init (opsional): deklarasi, inisialisasi variabel, evaluasi kiri-ke-kanan. Apabila lebih dari satu, maka dipisahkan koma.
  - Expression (opsional):
    - Apabila ada, maka dievaluasi (boolean atau Boolean) dan statement akan dikerjakan apabila hasil evaluasi adalah true
    - Apabila tidak ada, maka statement akan dikerjakan
  - Update (opsional):
    - Setelah statement berhasil dikerjakan, apabila ada, bagian ini akan dievaluasi kiri-ke-kanan
    - Apabila tidak ada, maka tidak ada aksi yang dilakukan

Perulangan

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          for (;;) {
4              System.out.println("loop");
5          }
6      }
7  }
```

```
javac Test.java

java Test
loop
loop
```

Statement akan dikerjakan terus menerus

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          for (int i=0; ;) {
4              System.out.println(i);
5          }
6      }
7  }
```

```
javac Test.java

java Test
0
0
```

Statement juga akan dikerjakan terus menerus

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          for (int i=0; i<3;) {
4              System.out.println(i);
5          }
6      }
7  }
```

```
javac Test.java

java Test
0
0
```

Statement juga akan dikerjakan terus menerus

Perulangan

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          for (int i=0; i<3; i++) {
4              System.out.println(i);
5          }
6      }
7  }
```

```
javac Test.java

java Test
0
1
2
```

- Pertama, i dideklarasikan, dan diberikan nilai 0
- Ekspresi i<3 dievaluasi true (0<3), dan i ditampilkan (0)
- Nilai i ditambah dengan 1 (menjadi 1)
- Ekspresi i<3 dievaluasi true (1<3), dan i ditampilkan (1)
- Nilai i ditambah dengan 1 (menjadi 2)
- Ekspresi i<3 dievaluasi true (2<3), dan i ditampilkan (2)
- Nilai i ditambah dengan 1 (menjadi 3)
- Ekspresi i<3 dievaluasi false (3<3), keluar

Perulangan

Test.java ×

```
1   class Test {
2       public static void main(String[] args) {
3           for (int i=0; i<3;) {
4               System.out.println(i++);
5           }
6       }
7   }
```

Output sama seperti contoh sebelumnya

Test.java ×

```
1   class Test {
2       public static void main(String[] args) {
3           for (int i=0, j=0; i<3; System.out.println(i + " " + j), i++, j += 2) {
4           }
5       }
6   }
```

```
javac Test.java

java Test
0 0
1 2
2 4
```

- Pertama, i dan j dideklarasikan dan keduanya diberikan nilai 0
- Kemudian dievaluasi apakah i<3 (true karena 0<3)
- Isi perulangan dikerjakan (dalam hal ini, tidak ada yang dikerjakan)
- Bagian update dikerjakan (tampilkan), i ditambahkan 1 dan j ditambahkan 2
- Perulangan dilanjutkan

Perulangan

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          for (int i=0; i<3; System.out.println(i++));
4      }
5  }
```

```
javac Test.java

java Test
0
1
2
```

Perhatikanlah titik koma setelah for

Perulangan

- Enhanced for

  for (FormalParameter: Expression) Statement

  (*Java Language Specification* 14.14.2)

- Catatan:
  - Tipe expression haruslah Iterable atau array (keduanya akan dibahas di bagian 3)
  - FormalParameter sesuai dengan formal parameter atau daftar argumen pada method

Perulangan

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          for (String arg: args) {
4              System.out.println(arg);
5          }
6      }
7  }
```

Test.java

```
1  class Test {
2      public static void main(String[] args) {
4          for (int i=0; i<args.length; i++) {
5              System.out.println(args[i]);
6          }
7      }
```

```
javac Test.java

java Test hello world
hello
world
```

Catatan:

- Kode yang di atas adalah contoh penggunaan enhanced for, sementara kode yang di bawah adalah for biasa
- Keduanya melakukan iterasi elemen dalam array String
- Array tersebut didapatkan dari command-line argumen
- Pada contoh enhanced for tersebut, kita tidak bekerja dengan indeks array

Perulangan

- Statement break
  - Dapat digunakan untuk menghentikan perulangan (while, do, for)
    - Pada contoh switch sebelumnya, break juga digunakan
  - Statement ini dapat bekerja dengan label (tidak dibahas dalam manual ini)

(*Java Language Specification* 14.15)

Perulangan

Test.java

```
1   class Test {
2       public static void main(String[] args) {
3           int a = 0;
4           while (true) {
5               if (a > 3) {
6                   break;
7               }
8               System.out.println(a++);
9           }
10
11          System.out.println();
12
13          int b = 0;
14          while (b <= 3) {
15              System.out.println(b++);
16          }
17      }
18  }
```

```
javac Test.java

java Test
0
1
2
3

0
1
2
3
```

Catatan:

- Pada baris 4, sebuah perulangan dilakukan. Perulangan ini akan dikerjakan karena ekspresi true. Di dalam perulangan, nilai a dicek, apakah lebih besar dari 3. Apabila benar, maka break digunakan untuk menghentikan perulangan. Apabila tidak, maka statement selanjutnya dikerjakan.
- Contoh tanpa break di baris 14

Perulangan

- Statement continue
  - Dapat digunakan untuk menghentikan iterasi dalam perulangan (while, do, for)
  - Statement ini dapat bekerja dengan label (tidak dibahas dalam manual ini)

(*Java Language Specification* 14.16)

Perulangan

Test.java

```
1   class Test {
2       public static void main(String[] args) {
3           int a = 0;
4           while (a <= 5) {
5               a++;
6
7               if (a % 2 == 0) {
8                   continue;
9               }
10
11              System.out.println(a);
12          }
13      }
14  }
```

```
javac Test.java

java Test
1
3
5
```

Catatan:

- Perhatikanlah bahwa pertama-tama, nilai a kita tambahkan dengan 1
- Kemudian, pada baris 7, kita cek apakah sisa bagi a dengan 2 adalah 0
  - Apabila benar, maka iterasi tersebut kita hentikan dengan continue (println() setelahnya tidak lagi dikerjakan).
    - Perulangan kemudian dilanjutkan dari awal

Console

- Class java.io.Console dapat digunakan untuk menampilkan teks ke layar ataupun mendapatkan input dari keyboard
- Ada atau tidaknya console bergantung pada platform dan bagaimana virtual machine Java dijalankan
  - Apabila dijalankan secara interaktif tanpa redireksi stdin atau stdout, maka umumnya akan terhubung ke keyboard dan display
- Console bisa didapatkan dengan System.console()
  - Apabila tidak tersedia, maka method akan mengembalikan null
- Console kita bahas di bagian ini untuk mendukung contoh soal

Console

Test.java

```
1  import java.io.Console;
2
3  class Test {
4      public static void main(String[] args) {
5          Console console = System.console();
6          if (console == null) {
7              System.out.println("Console tidak tersedia");
8          } else {
9              System.out.println("Console tersedia");
10         }
11     }
12 }
```

```
javac Test.java

java Test
Console tersedia

java Test < /dev/null
Console tidak tersedia
```

Catatan:

- Pada contoh pertama, program dijalankan (java Test) secara interaktif, tanpa redireksi stdin atau stdout. Dengan kondisi demikian dan console didukung pada platform yang digunakan, maka kita bisa lihat bahwa System.console() tidak mengembalikan null.
- Pada contoh kedua, program dijalankan dengan redireksi stdin (< /dev/null). Walaupun console didukung pada platform yang digunakan, namun redireksi tersebut menyebabkan System.console() mengembalikan null.

Console

- Menampilkan output
  - Kita bisa gunakan method printf ataupun format
  - Untuk contoh sederhana, kita bisa lewatkan String yang ingin ditulis apa adanya dengan printf
  - Contoh pemformatan dibahas pada bagian 7

Test.java

```
1  import java.io.Console;
2
3  class Test {
4      public static void main(String[] args) {
5          Console console = System.console();
6          if (console != null) {
7              console.printf("Console tersedia\n");
8          }
9      }
10 }
```

```
javac Test.java

java Test
Console tersedia
```

Escape sequence \n dapat digunakan untuk turun baris. Escape sequence lainnya bisa dibaca di Java Language Specification 3.10.6.

Console

- Mendapatkan input
  - Kita bisa gunakan method readLine(), yang mengembalikan String. Method ini di-overload (salah satunya menerima pemformatan).
  - Secara sederhana, kita bisa lewatkan String yang ingin digunakan sebagai prompt ke readLine()

Test.java

```
1  import java.io.Console;
2
3  class Test {
4      public static void main(String[] args) {
5          Console console = System.console();
6          if (console != null) {
7              String name = console.readLine("Masukkan nama Anda: ");
8              System.out.println("Hello, " + name);
9          }
10     }
11 }
```

```
javac Test.java

java Test
Masukkan nama Anda: test
Hello, test
```

Console

- Mendapatkan input berupa password (tidak ditampilkan ketika sedang diketikkan)
  - Kita bisa gunakan method readPassword(), yang mengembalikan char[] (bukan String). Method ini di-overload (salah satunya menerima pemformatan).
  - Secara sederhana, kita bisa lewatkan String yang ingin digunakan sebagai prompt ke readPassword()
  - Untuk keperluan manual ini, password yang didapatkan dapat diubah ke String dengan melewatkan char[] ke salah satu constructor String.

Console

Test.java

```
1  import java.io.Console;
2
3  class Test {
4      public static void main(String[] args) {
5          Console console = System.console();
6          if (console != null) {
7              char[] password = console.readPassword("Password: ");
8              String passwordString = new String(password);
9              System.out.println("Password Anda: " + passwordString);
10         }
11     }
12 }
```

```
javac Test.java

java Test
Password:
Password Anda: test
```

Contoh Soal

- Kita telah membahas tentang console, perulangan, percabangan, sejumlah tipe data, operator, class/objek, String, dan lainnya. Cobalah untuk membuat program yang:
  - Meminta input berupa nama, password, ataupun informasi lainnya
  - Memeriksa apakah password sesuai kriteria
  - Apabila input tidak diisikan, maka program tidak dapat dilanjutkan (kembali meminta)
  - Dan, pada akhirnya menampilkan berbagai informasi yang relevan ke standard output

Bagian 3

abstract, final,

interface,

exception,

array, generics, collection,

bekerja dengan file

Final

- Sebuah variabel bisa dideklarasikan final, dimana nilai hanya bisa diberikan sekali pada variabel tersebut

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          final int a;
4          a = 1;
5
6          System.out.println(a);
7      }
8  }
```

```
javac Test.java

java Test
1
```

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          final int a;
4          a = 1;
5
6          System.out.println(a);
7
8          a = 2;
9      }
10 }
```

```
javac Test.java
Test.java:8: error: variable a might already have been assigned
        a = 2;
        ^
1 error
```

Final

- Apabila sebuah variabel final adalah reference ke sebuah objek, state objek tersebut bisa berubah. Final dalam hal ini diartikan bahwa variabel tersebut selalu merefer ke objek yang sama.

  (*Java Language Specification* 4.12.4)

Test.java

```
1  class Test {
2      public static void main(String[] args) {
3          final StringBuilder builder = new StringBuilder("Hello");
4          System.out.println(builder.toString());
5
6          builder.append(" world");
7          System.out.println(builder.toString());
8      }
9  }
```

```
javac Test.java

java Test
Hello
Hello world
```

Perhatikanlah bahwa builder, sebuah reference ke StringBuilder, adalah final. Diberikan nilai awal, state builder berubah dengan method append().

Final

Test.java

```
1   class Test {
2       public static void main(String[] args) {
3           final StringBuilder builder1 = new StringBuilder("1");
4           StringBuilder builder2 = new StringBuilder("2");
5           StringBuilder builder3 = new StringBuilder("3");
6
7           System.out.println(builder3);
8           builder3 = builder2;
9           System.out.println(builder3);
10
11          builder1 = builder2;
12      }
13  }
```

```
javac Test.java
Test.java:11: error: cannot assign a value to final variable builder1
        builder1 = builder2;
        ^
1 error
```

Perhatikanlah bahwa builder1 adalah final, sementara builder2 dan builder3 adalah tidak. Pada baris 8, builder3 kini merefer ke objek yang sama yang direfer oleh builder2. Kita bisa lakukan ini karena builder3 tidak final. Pada baris 11, kita tidak bisa melakukan yang serupa karena builder1 adalah final.

Final

- Pada field, baik variabel class (static) atau instance (non-static), bisa dideklarasikan final.
  - Sebelumnya, kita telah melihat contoh penggunaan field yang static dan final (konstanta)
- Blank final: deklarasi tanpa inisialisasi
  - Apabila digunakan, sebuah class variabel blank final harus diberikan nilai di static initializer
  - Apabila digunakan, sebuah instance variabel blank final, juga harus diberikan nilai, baik lewat constructor atau initializer

Final

Test.java

```
1  class Test {
2      public static final int MAX_LEVEL;
3
4      public static void main(String[] args) {
5      }
6  }
```

```
javac Test.java
Test.java:2: error: variable MAX_LEVEL not initialized in the default constructo
r
    public static final int MAX_LEVEL;
                            ^
1 error
```

Test.java

```
1  class Test {
2      public static final int MAX_LEVEL;
3
4      static {
5          MAX_LEVEL = 10;
6      }
7
8      public static void main(String[] args) {
9          System.out.println(MAX_LEVEL);
10     }
11 }
```

```
javac Test.java

java Test
10
```

Final

Test.java

```
1  class Test {
2      private final int MIN_LEVEL;
3      private final int MAX_LEVEL;
4
5      public static void main(String[] args) {
6      }
7  }
```

```
javac Test.java
Test.java:2: error: variable MIN_LEVEL not initialized in the default constructo
r
    private final int MIN_LEVEL;
                      ^
Test.java:3: error: variable MAX_LEVEL not initialized in the default constructo
r
    private final int MAX_LEVEL;
                      ^
2 errors
```

Test.java

```
1  class Test {
2      private final int MIN_LEVEL;
3      private final int MAX_LEVEL;
4
5      {
6          MIN_LEVEL = 1;
7      }
8
9      Test() {
10         MAX_LEVEL = 10;
11     }
12
13     public static void main(String[] args) {
14         System.out.println(new Test().MIN_LEVEL);
15         System.out.println(new Test().MAX_LEVEL);
16     }
17 }
```

Perhatikanlah bahwa MIN_LEVEL diset di initializer dan MAX_LEVEL diset di constructor. Ini hanyalah sebuah contoh.

Final

- Dalam pemrograman berorientasi objek, sebuah class yang lebih umum dapat diturunkan menjadi class yang lebih spesifik
- Di Java, sebuah class dapat diturunkan, kecuali class tersebut dideklarasikan final
  - Dengan dideklarasikan final, diartikan bahwa definisi class tersebut telah komplit dan/atau tidak ada subclass yang diinginkan atau dibutuhkan

  (*Java Language Specification* 8.1.1.2)

Final

Test.java

```
1  class Test extends String {
2      public static void main(String[] args) {
3      }
4  }
```

```
javac Test.java
Test.java:1: error: cannot inherit from final String
class Test extends String {
                   ^
1 error
```

Perhatikanlah bahwa java.lang.String adalah final dan oleh karenanya tidak dapat diturunkan. Di Java, untuk menurunkan dari sebuah class, kita gunakan keyword extends.

Final

- Modifier final juga dapat diterapkan pada method. Dalam hal ini, method tersebut tidak dapat di-override oleh subclass.
  - Override: mendefinisikan ulang method, agar spesifik terhadap suatu subclass (terdapat sejumlah aturan)

Test.java

```
1  class Test1 {
2      public void method1() {
3          System.out.println("method1 in Test1");
4      }
5  }
6
7  class Test extends Test1 {
9      public void method1() {
10         System.out.println("method1 in Test");
11     }
12
13     public static void main(String[] args) {
14         Test test = new Test();
15         test.method1();
16     }
}
```

```
javac Test.java

java Test
method1 in Test
```

Perhatikanlah bahwa kita meng-override method1() milik Test1 di Test.

Final

Test.java

```
1  class Test1 {
2      public final void method1() {
3          System.out.println("method1 in Test1");
4      }
5  }
6
7  class Test extends Test1 {
9      public void method1() {
10         System.out.println("method1 in Test");
11     }
12
13     public static void main(String[] args) {
14         Test test = new Test();
15         test.method1();
16     }
}
```

Akan tetapi, ceritanya akan berbeda ketika method1() di Test1 dideklarasikan final.

Class Test1 sendiri tetap bisa diturunkan (karena tidak final), hanya method1() di dalam Test1 tidak bisa di-override (karena final).

```
javac Test.java
Test.java:8: error: method1() in Test cannot override method1() in Test1
    public void method1() {
                ^
  overridden method is final
1 error
```

Abstract

- Pada class, abstract diartikan sebagai class yang belum lengkap

  (*Java Language Specification* 8.1.1.1)

- Class yang abstract dapat memiliki method yang tidak abstract

Test.java

```
1  abstract class TestAbstract {
2      public void method1() {
3          System.out.println("method1 in TestAbstract");
4      }
5
6      public void method2() {
7          System.out.println("method2 in TestAbstract");
8      }
9  }
10
11 class Test {
12     public static void main(String[] args) {
13     }
14 }
```

Abstract

- Class yang abstract tidak dapat diinstansiasi

Test.java

```
1   abstract class TestAbstract {
2       public void method1() {
3           System.out.println("method1 in TestAbstract");
4       }
5   }
6
7   class Test {
8       public static void main(String[] args) {
9           TestAbstract test = new TestAbstract();
10      }
11  }
```

```
javac Test.java
Test.java:9: error: TestAbstract is abstract; cannot be instantiated
        TestAbstract test = new TestAbstract();
                            ^
1 error
```

Abstract

- Class yang abstract dapat diturunkan, dan class turunannya juga dapat berupa class yang abstract

Test.java

```
1  abstract class TestAbstract {
2      public void method1() {
3          System.out.println("method1 in TestAbstract");
4      }
5  }
6
7  abstract class TestAbstract2 extends TestAbstract {
8  }
9
10 abstract class TestAbstract3 extends TestAbstract2 {
11 }
12
13 class Test {
14     public static void main(String[] args) {
15     }
16 }
```

Abstract

- Method yang abstract tidak memiliki isi, dimana deklarasi diakhiri dengan ; (bukan {})

Test.java

```
1  class TestAbstract {
2      public abstract void method1();
3  }
4
5  class Test {
6      public static void main(String[] args) {
7      }
8  }
```

```
javac Test.java
Test.java:1: error: TestAbstract is not abstract and does not override abstract
method method1() in TestAbstract
class TestAbstract {
^
1 error
```

- Apabila ada method yang abstract dalam suatu class, class tersebut harus abstract

Test.java

```
1  abstract class TestAbstract {
2      public abstract void method1();
3  }
4
5  class Test {
6      public static void main(String[] args) {
7      }
8  }
```

Abstract

- Class yang menurunkan dari class yang abstract, selama class tersebut tidak abstract (sehingga dapat diinstansiasi), harus menyediakan implementasi method yang abstract (termasuk milik superclass class yang abstract)

Test.java ×

```
1   abstract class TestAbstract1 {
2       public abstract void method1();
3   }
4
5   abstract class TestAbstract2 extends TestAbstract1 {
6       public abstract void method2();
7   }
8
9   abstract class TestAbstract3 extends TestAbstract2 {
10      public abstract void method3();
11  }
12
13  class Test extends TestAbstract3 {
14      public static void main(String[] args) {
15      }
16  }
```

```
javac Test.java
Test.java:13: error: Test is not abstract and does not override abstract method
method3() in TestAbstract3
class Test extends TestAbstract3 {
^
1 error
```

Abstract

Test.java

```
1   abstract class TestAbstract1 {
2       public abstract void method1();
3   }
4
5   abstract class TestAbstract2 extends TestAbstract1 {
6       public abstract void method2();
7   }
8
9   abstract class TestAbstract3 extends TestAbstract2 {
10      public abstract void method3();
11  }
12
13  class Test extends TestAbstract3 {
14      public void method3() {}
15
16      public static void main(String[] args) {
17      }
18  }
```

```
javac Test.java
Test.java:13: error: Test is not abstract and does not override abstract method
method2() in TestAbstract2
class Test extends TestAbstract3 {
^
1 error
```

Abstract

Test.java

```
1  abstract class TestAbstract1 {
2      public abstract void method1();
3  }
4
5  abstract class TestAbstract2 extends TestAbstract1 {
6      public abstract void method2();
7  }
8
9  abstract class TestAbstract3 extends TestAbstract2 {
10     public abstract void method3();
11 }
12
13 class Test extends TestAbstract3 {
14     public void method1() {}
15     public void method2() {}
16     public void method3() {}
17
18     public static void main(String[] args) {
19         Test test = new Test();
20         test.method3();
21     }
22 }
```

Interface

- Sebuah class hanya dapat menurunkan dari satu superclass secara langsung. Class turunan adalah class yang memiliki fungsi, tugas, atau karakteristik yang lebih spesifik. Dengan demikian, sebuah hirarki class harusnya menggambarkan garis turunan yang jelas, berdasarkan kriteria tertentu.
- Bagaimana kalau terdapat beberapa class yang tidak saling terkait (tidak dalam satu garis turunan), namun dapat memiliki sejumlah karakteristik yang sama?
  - Contoh: String (java.lang.Object → java.lang.String) dan JButton (java.lang.Object → java.awt.Component → java.awt.Container → javax.swing.JComponent → javax.swing.AbstractButton → javax.swing.Jbutton) adalah dua class yang tidak dalam satu garis turunan.
  - Namun, keduanya memiliki satu karakteristik yang sama, dalam artian, sama-sama mengimplementasikan interface java.io.Serializable

Interface

- Sebuah class dapat mengimplementasikan lebih dari satu interface. Dengan demikian, sejumlah class yang tidak terkait, dapat memiliki sejumlah karakteristik yang sama.
- Sebagaimana halnya kita merancang class supaya memiliki fokus atau fungsi yang spesifik, kita juga melakukan hal yang serupa dengan interface (barangkali secara lebih spesifik lagi :))

Interface

- Interface berfungsi sebagai sebuah kesepakatan, yang mana harus ditaati oleh setiap class yang mengimplementasikan interface tersebut
- Kesepakatan ini bisa saja hanya dari sisi identitas, atau terdapat method yang harus diimplementasikan

Interface

- Interface dideklarasikan dengan keyword interface
- Berbeda dengan class yang hanya dapat menurunkan dari satu superclass langsung, interface dapat menurunkan lebih dari satu superinterface secara langsung (multiple inheritance; juga dengan keyword extends)
  - Namun interface tidak dapat mengimplementasikan interface lain
- Sebuah interface secara implisit adalah abstract
- Method dalam interface secara implisit adalah public dan abstract (tidak dapat memiliki isi)
- Field dalam interface secara implisit adalah public, static, dan final
- Sebagaimana halnya dengan class, access modifier dapat diterapkan pada interface

  (Java Language Specification 9, 9.1, 9.2, 9.3)

Interface

Test.java

```
1   interface Extensible {
2   }
3
4   interface Programmable {
5   }
6
7   interface Scriptable extends Extensible, Programmable {
8   }
9
10  class Test implements Scriptable{
11      public static void main(String[] args) {
12      }
13  }
```

Catatan:

- Anggaplah kita ingin sebuah program dapat di-extend (Extensible) dan di-program lebih lanjut (Programmable). Anggaplah mekanismenya berbeda sama sekali dan oleh karena itu, keduanya tidak dalam satu garis turunan.
- Kita juga ingin program kita dapat dikembangkan lebih lanjut lewat scripting (Scriptable), yang mana dalam hal ini memiliki karakteristik dari Extensible dan Programmable, sehingga kita menurunkan dari dua interface tersebut (multiple inheritance, baris 7).
- Perhatikanlah bawa Test mengimpementasikan interface Scriptable (baris 10)

Interface

Test.java ×

```
1   interface Extensible {
2       String FORMAT = "jar";
3   }
4
5   interface Programmable {
6   }
7
8   interface Scriptable extends Extensible, Programmable {
9   }
10
11  class Test implements Scriptable {
12      public static void main(String[] args) {
13          System.out.println(FORMAT);
14          FORMAT = "class";
15      }
16  }
```

Perhatikanlah field FORMAT. Field ini tidak dideklarasikan secara eksplit sebagai public, static, dan final. Namun implikasinya demikian, karena:

- Dapat diakses dari konteks static (main)
- Tidak dapat diberikan ulang sebuah nilai (pesan kesalahan pada baris 14)

```
javac Test.java
Test.java:14: error: cannot assign a value to final variable FORMAT
        FORMAT = "class";
        ^
1 error
```

Interface

Test.java

```
1   interface Extensible {
2       String FORMAT = "jar";
3       void load();
4   }
5
6   interface Programmable {
7   }
8
9   interface Scriptable extends Extensible, Programmable {
10  }
11
12  class Test implements Scriptable {
13      public static void main(String[] args) {
14          System.out.println(FORMAT);
15      }
16  }
```

```
javac Test.java
Test.java:12: error: Test is not abstract and does not override abstract method
load() in Extensible
class Test implements Scriptable {
^
1 error
```

Class Test dalam hal ini perlu meng-override method load() milik Extensible, karena Test mengimplementasi Scriptable (dan Scriptable juga menurunkan dari Extensible)

Interface

Test.java

```
1   interface Extensible {
2       String FORMAT = "jar";
3       void load();
4   }
5
6   interface Programmable {
7   }
8
9   interface Scriptable extends Extensible, Programmable {
10  }
11
12  class Test implements Scriptable {
13      void load() {
14          System.out.println("LOADING");
15      }
16
17      public static void main(String[] args) {
18          System.out.println(FORMAT);
19      }
20  }
```

Catatan:

- Ingatlah bahwa method dalam interface adalah public dan abstract
- Walau implementasi load() disediakan di baris 13 (memenuhi sisi abstract), jangan lupa bahwa load() secara implisit adalah public
- Dan, salah satu aturan override adalah tidak boleh memberikan akses yang lebih terbatas
  - Method load() → public
  - Di baris 13 → default
    - Lebih terbatas

```
javac Test.java
Test.java:13: error: load() in Test cannot implement load() in Extensible
    void load() {
         ^
  attempting to assign weaker access privileges; was public
1 error
```

Interface

Test.java ×

```
1  interface Extensible {
2      String FORMAT = "jar";
3      void load();
4  }
5
6  interface Programmable {
7  }
8
9  interface Scriptable extends Extensible, Programmable {
10 }
11
12 class Test implements Scriptable {
13     public void load() {
14         System.out.println("LOADING");
15     }
16
17     public static void main(String[] args) {
18     }
19 }
```

Tambahkan public di baris 13, dan kompilasi berhasil. Kita telah memenuhi kesepakatan yang disyaratkan oleh interface Extensible.

Interface

Test.java

```
1   interface Extensible {
2       String FORMAT = "jar";
3       void load();
4   }
5
6   interface Programmable {
7       long register();
8   }
9
10  interface Scriptable extends Extensible, Programmable {
11  }
12
13  class Test implements Scriptable {
14      public void load() {
15          System.out.println("LOADING");
16      }
17
18      public static void main(String[] args) {
19      }
20  }
```

Perhatikanlah bahwa kita juga harus meng-override register() milik Programmable, karena Scriptable juga menurunkan dari Programmable.

```
javac Test.java
Test.java:13: error: Test is not abstract and does not override abstract method
register() in Programmable
class Test implements Scriptable {
^
1 error
```

Interface

Test.java

```
1   interface Extensible {
2       String FORMAT = "jar";
3       void load();
4   }
5
6   interface Programmable {
7       long register();
8   }
9
10  interface Scriptable extends Extensible, Programmable {
11  }
12
13  class Test implements Scriptable {
14      public void load() {
15          System.out.println("LOADING");
16      }
17
18      public long register() {
19          return 1;
20      }
21
22      public static void main(String[] args) {
23      }
24  }
```

Kali ini, kita meng-override public abstract long register() dengan benar (di baris 18), dan kompilasi berhasil.

Interface

Test.java

```
1  interface Extensible {
2      String FORMAT = "jar";
3      void load();
4  }
5
6  interface Programmable {
7      long register();
8  }
9
10 interface Scriptable extends Extensible, Programmable {
11     String getLanguage();
12 }
13
14 class Test implements Scriptable, java.io.Serializable {
15     public void load() {
16         System.out.println("LOADING");
17     }
18
19     public long register() {
20         return 1;
21     }
22
23     public String getLanguage() {
24         return "";
25     }
26
27     public static void main(String[] args) {
28     }
29 }
```

Catatan:

- Perhatikanlah bahwa kita telah memenuhi kesepakatan yang disyaratkan oleh Extensible, Programmable, dan Scriptable.
- Perhatikanlah juga bahwa di baris 14, kita mengimplementasikan lebih dari satu interface secara langsung, yaitu Scriptable (contoh sebelumnya) dan java.io.Serializable (baru).
- Terdapat interface yang tidak memiliki method dan berfungsi untuk memberikan identitas (untuk kegunaan tertentu). Salah satunya adalah Serializable tersebut.

Program to an Interface

- Kita bisa merancang API (contoh: suatu method) agar tidak merujuk langsung pada implementasi/class tertentu. Sebagai gantinya, kita gunakan interface.
- Dengan demikian, apabila diperlukan, kita bisa menggunakan/mengganti implementasi lain (dari suatu interface yang sama) dengan perubahan yang minim

Program to an Interface

Test.java ×

```
1   interface Extensible {
2   }
3
4   class Program1 implements Extensible {
5   }
6
7   class Program2 implements Extensible {
8   }
9
10  class Test{
11      public static void main(String[] args) {
12          Program1 program1 = new Program1();
13          Program2 program2 = new Program2();
14      }
15  }
```

Pada kedua contoh ini, mari kita asumsikan bahwa check() dan check2() hanya membutuhkan sebuah class yang mengimplementasikan Extensible. Bagaimana kalau kita ingin mengganti tipe dari program1 dan program2 (baris 12, 13 di gambar kiri) atau ingin melewatkan tipe lain (juga Extensible) ke check() atau check2() (di gambar kanan)?

Test.java ×

```
1   interface Extensible {
2   }
3
4   class Program1 implements Extensible {
5   }
6
7   class Program2 implements Extensible {
8   }
9
10  class Test{
11      public static void check(Program1 program) {
12      }
13
14      public static void check2(Program2 program) {
15      }
16
17      public static void main(String[] args) {
18          Program1 program1 = new Program1();
19          Program2 program2 = new Program2();
20
21          check(program1);
22          check2(program2);
23      }
24  }
```

Program to an Interface

Test.java

```
1  interface Extensible {
2  }
3
4  class Program1 implements Extensible {
5  }
6
7  class Program2 implements Extensible {
8  }
9
10 class Program3 implements Extensible {
11 }
12
13 class Test{
14     public static void check(Program3 program) {
15     }
16
17     public static void check2(Program3 program) {
18     }
19
20     public static void main(String[] args) {
21         Program3 program1 = new Program3();
22         Program3 program2 = new Program3();
23
24         check(program1);
25         check2(program2);
26     }
27 }
```

Kita ingin mengganti tipe dari program1 dan program2 (sekarang baris 21 dan 22) ke Program3 (class baru, baris 10):

- Maka, kita harus mengganti tipe di baris 21 dan 22, baik untuk deklarasi tipe (sebelah kiri nama variabel) ataupun instansiasi (sebelah kanan new).
- Lalu, jangan lupa bahwa kita juga harus mengganti tipe argumen (sekarang baris 14 dan 17) menjadi Program3.
- Ini semua disebabkan karena sebuah class baru, Program3 (baris 10). Padahal, ke depan, beberapa class baru mungkin akan ditambahkan, dan kita mungkin akan mengganti lagi tipe dari program1 dan program2.

Program to an Interface

Test.java

```
1  interface Extensible {
2  }
3
4  class Program1 implements Extensible {
5  }
6
7  class Program2 implements Extensible {
8  }
9
10 class Test{
11     public static void check(Extensible program) {
12     }
13
14     public static void check2(Extensible program) {
15     }
16
17     public static void main(String[] args) {
18         Extensible program1 = new Program1();
19         Extensible program2 = new Program2();
20
21         check(program1);
22         check2(program2);
23     }
24 }
```

Bagaimana kalau sejak awal, kita menggunakan tipe Extensible di method check() dan check2() (baris 11 dan 14)?

Bagaimana juga kalau sejak awal, kita mendeklarasikan tipe program1 dan program2 sebagai Extensible (di baris 18 dan 19)?

Tentu saja, ketika kita menentukan bahwa implementasi dari program1 adalah Program1, maka instansiasi pada baris 18 adalah new Program1(). Hal yang sama juga dilakukan pada program2 (new Program2() pada baris 19).

Program to an Interface

Test.java ×

```
1  interface Extensible {
2  }
3
4  class Program1 implements Extensible {
5  }
6
7  class Program2 implements Extensible {
8  }
9
10 class Program3 implements Extensible {
11 }
12
13 class Test{
14     public static void check(Extensible program) {
15     }
16
17     public static void check2(Extensible program) {
18     }
19
20     public static void main(String[] args) {
21         Extensible program1 = new Program3();
22         Extensible program2 = new Program3();
23
24         check(program1);
25         check2(program2);
26     }
27 }
```

Andaikata class Program3 (juga Extensible) dibuat sekarang (di baris 10), dan kita memutuskan untuk mengganti implementasi program1 dan program2 ke Program3:

- Kita cukup mengganti instansiasi new Program1() ke new Program3() di baris 21
- Dan mengganti new Program2() ke new Program3() di baris 22

Perhatikanlah bahwa tidak ada perubahan pada deklarasi method check() dan check2().

Class baru mungkin akan ditambahkan dan kita mungkin akan mengganti implementasi. Namun, selama masih Extensible, perubahan yang dilakukan akan minim.

Operator instanceof

- Operator instanceof adalah sebuah operator untuk membandingkan tipe

  RelationalExpression instanceof ReferenceType

  (*Java Language Specification* 15.20.2)
- Catatan:
  - RelationalExpression harus merupakan reference atau null
  - ReferenceType harus merujuk pada reference yang tipenya tersedia sepenuhnya pada saat runtime (*Java Language Specification* 4.7)
  - Hasil evaluasi true apabila nilai dari RelationalExpression tidak null dan reference dapat di-cast (konversi) ke ReferenceType tanpa terjadinya ClassCastException (Exception dibahas setelah ini)

Operator instanceof

Test.java ×

```
1  class Test{
2      public static void main(String[] args) {
3          int i;
4          System.out.println(i instanceof Integer);
5      }
6  }
```

```
javac Test.java
Test.java:4: error: unexpected type
        System.out.println(i instanceof Integer);
                           ^
  required: reference
  found:    int
1 error
```

Test.java ×

```
1  class Test{
2      public static void main(String[] args) {
3          int i;
4          System.out.println(i instanceof int);
5      }
6  }
```

```
javac Test.java
Test.java:4: error: unexpected type
        System.out.println(i instanceof int);
                           ^
  required: reference
  found:    int
Test.java:4: error: unexpected type
        System.out.println(i instanceof int);
                                        ^
  required: class or array
  found:    int
2 errors
```

Test.java ×

```
1  class Test{
2      public static void main(String[] args) {
3          System.out.println(null instanceof Object);
4      }
5  }
```

```
javac Test.java

java Test
false
```

Perhatikanlah bahwa instanceof bekerja dengan reference

Operator instanceof

Test.java

```
1  interface Extensible {
2  }
3
4  class Program1 implements Extensible {
5  }
6
7  class Program2 {
8  }
9
10 class Test{
11     public static void main(String[] args) {
12         System.out.println("Hello" instanceof String);
13         System.out.println("Hello" instanceof Object);
14         System.out.println("Hello" instanceof java.io.Serializable);
15         System.out.println((new Program1()) instanceof Extensible);
16         System.out.println((new Program2()) instanceof Extensible);
17     }
18 }
```

```
javac Test.java

java Test
true
true
true
true
false
```

Catatan:

- "Hello" adalah instance dari String karena "Hello" adalah sebuah String, yang mana juga merupakan sebuah Object
- Karena class String mengimplementasikan Serializable, maka perbandingan pada baris 14 juga bernilai true
- Class Program1 mengimplementasikan Extensible, dan oleh karenanya, perbandingan pada baris 15 bernilai true. Namun, beda ceritanya dengan Program2.

Operator instanceof

- Perhatikanlah bahwa perbandingan ini dapat juga ditolak pada saat kompilasi

Test.java ×

```
1   class Test{
2       public static void main(String[] args) {
            System.out.println("Hello" instanceof Integer);
4       }
5   }
```

```
javac Test.java
Test.java:3: error: incompatible types: String cannot be converted to Integer
        System.out.println("Hello" instanceof Integer);
                           ^
1 error
```

Exception dan Error

- Pada saat sebuah program melanggar *aturan,* atau karena memang diminta, atau karena terjadi hal luar biasa yang seharusnya tidak terjadi, maka Exception atau Error akan terjadi
- Sebuah class, java.lang.Throwable (turunan langsung dari java.lang.Object), adalah superclass dari semua Exception dan Error
- Ini adalah topik yang luas dan kompleks, dan manual ini hanya akan membahas sebagian kecil diantaranya
  - Sebagai tindak lanjut, bacalah juga API Specification untuk Throwable ataupun salah satu dari subclassnya
  - Java Language Specification membahas tentang Exception pada bab 11

Exception dan Error

- Terdapat dua turunan langsung dari Throwable:
  - java.lang.Exception: program dapat melakukan recovery
  - java.lang.Error: program tidak diharapkan untuk melakukan recovery
- Exception dan Error dibagi menjadi dua bagian besar yang harus dipahami:
  - Unchecked exception:
    - java.lang.Error (dan turunannya)
    - java.lang.RuntimeException (dan turunannya). Perhatikanlah bahwa RuntimeException adalah turunan langsung dari java.lang.Exception.
  - Checked exception:
    - Semua class diluar kategori unchecked exception
    - Yaitu, semua turunan dari Throwable selain:
      - java.lang.Error (dan turunannya)
      - java.lang.RuntimeException (dan turunannya)

Exception dan Error

- Unchecked exception:
  - Tidak perlu dideklarasikan dalam klausa throws dari method ataupun constructor apabila exception bisa terjadi dalam method atau constructor tersebut
  - Tidak diperiksa pada saat kompilasi

Exception dan Error

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           throw new RuntimeException("runtime exception");
4       }
5   }
```

```
javac Test.java

java Test
Exception in thread "main" java.lang.RuntimeException: runtime exception
        at Test.main(Test.java:3)
```

Perhatikanlah bahwa kita melempar sebuah unchecked exception (RuntimeException), lengkap dengan informasi pesan kesalahan

Exception dan Error

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           try {
4               throw new RuntimeException("runtime exception");
5           } catch (RuntimeException e) {
6               System.out.println("Terjadi RuntimeException dengan pesan: " +
7                       e.getMessage());
8           }
9       }
10  }
```

```
javac Test.java

java Test
Terjadi RuntimeException dengan pesan: runtime exception
```

Walaupun hanya sebagai contoh, perhatikanlah bahwa apa yang kita lempar tersebut kita tangkap dalam blok catch (try/catch), yang mana di dalamnya bisa mulai dilakukan recovery (atau sekedar menampilkan pesan kesalahan)

Exception dan Error

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           try {
4               System.out.println(1/0);
5           } catch (NullPointerException e) {
6               System.out.println("Terjadi NullPointerException");
7           }
8       }
9   }
```

```
javac Test.java

java Test
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Test.main(Test.java:4)
```

Perhatikanlah bahwa kita mencoba untuk menangkap sebuah exception yang mungkin terjadi, dengan harapan dapat menampilkan pesan kesalahan. Akan tetapi, kita salah menangkap exception.

Exception dan Error

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           System.out.println(1/0);
4       }
5   }
```

```
javac Test.java

java Test
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Test.main(Test.java:3)
```

Perhatikanlah bahwa program sebelumnya, dengan exception handling yang tidak tepat, memiliki efek yang sama seperti contoh ini (tanpa exception handling)

Exception dan Error

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           System.out.println(1/0);
4
5           System.out.println("Hello, World");
6       }
7   }
```

```
javac Test.java

java Test
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Test.main(Test.java:3)
```

Perhatikanlah bahwa ketika exception terjadi dalam contoh ini, statement pada baris 5 tidak lagi dikerjakan

Exception dan Error

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           try {
4               System.out.println(1/0);
5           } catch (ArithmeticException e) {
6               System.out.println("Terjadi kesalahan: " + e.getMessage());
7           }
8
9           System.out.println("Hello, World");
10      }
11  }
```

```
javac Test.java

java Test
Terjadi kesalahan: / by zero
Hello, World
```

Kini, kita telah menangkap exception yang tepat, sehingga error recovery bisa dilakukan. Kesalahan yang terjadi ditampilkan lengkap dengan pesan kesalahan.

Disarankan untuk menangkap exception yang spesifik (tidak catch Exception atau Throwable)

Exception dan Error

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           try {
4               String hello = null;
5               System.out.println(hello.length()/0);
6           } catch (NullPointerException e) {
7               System.out.println("Terjadi kesalahan: " + e.getClass());
8           } catch (ArithmeticException e) {
9               System.out.println("Terjadi kesalahan: " + e.getClass());
10          }
11
12          System.out.println("Hello, World");
13      }
14  }
```

```
javac Test.java

java Test
Terjadi kesalahan: class java.lang.NullPointerException
Hello, World
```

Kita bisa mencoba menangkap lebih dari satu exception seperti dalam contoh

Exception dan Error

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           try {
4               String hello = "Hello";
5               System.out.println(hello.length()/0);
6           } catch (NullPointerException e) {
7               System.out.println("Terjadi kesalahan: " + e.getClass());
8           } catch (ArithmeticException e) {
9               System.out.println("Terjadi kesalahan: " + e.getClass());
10          }
11
12          System.out.println("Hello, World");
13      }
14  }
```

```
javac Test.java

java Test
Terjadi kesalahan: class java.lang.ArithmeticException
Hello, World
```

Perhatikanlah bahwa hello kini tidak lagi null (baris 5), sehingga exception lain yang terjadi

Exception dan Error

Test.java ×

```
1   class Test{
2       public static void main(String[] args) {
3           try {
4               String hello = "Hello";
5               System.out.println(hello.length()/0);
6           } catch (ArithmeticException e) {
7               System.out.println("Terjadi kesalahan Arithmetic");
8           } catch (Exception e) {
9               System.out.println("Terjadi kesalahan lain: " + e.getClass());
10          }
11      }
12  }
```

```
javac Test.java

java Test
Terjadi kesalahan Arithmetic
```

Test.java ×

```
1   class Test{
2       public static void main(String[] args) {
3           try {
4               String hello = "Hello";
5               System.out.println(hello.length()/0);
6           } catch (Exception e) {
7               System.out.println("Terjadi kesalahan lain: " + e.getClass());
8           } catch (ArithmeticException e) {
9               System.out.println("Terjadi kesalahan Arithmetic");
10          }
11      }
12  }
```

```
javac Test.java
Test.java:8: error: exception ArithmeticException has already been caught
        } catch (ArithmeticException e) {
          ^
1 error
```

Perhatikanlah bahwa dalam contoh seperti ini, urutan menentukan
→ dari spesifik ke umum

Exception dan Error

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           try {
4               String hello = "Hello";
5               System.out.println(hello.length()/0);
6           } catch (NullPointerException | ArithmeticException e) {
7               System.out.println("Terjadi kesalahan: " + e.getClass());
8           }
9
10          System.out.println("Hello, World");
11      }
12  }
```

```
javac Test.java

java Test
Terjadi kesalahan: class java.lang.ArithmeticException
Hello, World
```

Multi-catch bisa digunakan selama tidak dalam satu garis turunan. Dalam hal ini, kedua exception tersebut sama-sama adalah turunan langsung dari java.lang.RuntimeException

Exception dan Error

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           try {
4               String hello = "Hello";
5               System.out.println(hello.length()/0);
6           } catch (RuntimeException | ArithmeticException e) {
7               System.out.println("Terjadi kesalahan: " + e.getClass());
8           }
9       }
10  }
```

```
javac Test.java
Test.java:6: error: Alternatives in a multi-catch statement cannot be related by
 subclassing
        } catch (RuntimeException | ArithmeticException e) {
                                    ^
  Alternative ArithmeticException is a subclass of alternative RuntimeException
1 error
```

Perhatikanlah bahwa java.lang.ArithmeticException adalah turunan langsung dari java.lang.RuntimeException

Exception dan Error

Test.java ×

```
1   class Test{
2       public static void delay() {
4           Thread.sleep(1000);
        }
5
6       public static void main(String[] args) {
7           delay();
8       }
9   }
```

```
javac Test.java
Test.java:3: error: unreported exception InterruptedException; must be caught or
 declared to be thrown
        Thread.sleep(1000);
                     ^
1 error
```

Perhatikanlah bahwa Thread.sleep() dideklarasikan throws InterruptedException, dan exception ini adalah checked exception (dalam hal ini, sebuah turunan langsung java.lang.Exception). Kita perlu tangani sendiri atau deklarasikan bahwa ini akan dilempar lagi (untuk ditangkap oleh yang lainnya).

Exception dan Error

Test.java

```
1   class Test{
2       public static void delay() throws InterruptedException {
3           Thread.sleep(1000);
4       }
5
6       public static void main(String[] args) {
7           delay();
8       }
9   }
```

```
javac Test.java
Test.java:7: error: unreported exception InterruptedException; must be caught or
 declared to be thrown
        delay();
             ^
1 error
```

Perhatikanlah bahwa method delay() mendeklarasikan InterruptedException dilempar lagi (dengan keyword throws). Pesan kesalahan kompilasi tidak lagi di dalam method tersebut. Akan tetapi, kompilasi masih gagal karena tidak ada yang menangkap exception tersebut.

Exception dan Error

Test.java

```
1  class Test{
2      public static void delay() throws InterruptedException {
3          Thread.sleep(1000);
4      }
5
6      public static void main(String[] args) {
7          try {
8              delay();
9          } catch (InterruptedException e) {
10             System.out.println(e);
11         }
12
13         System.out.println("Hello, World");
14     }
15 }
```

Kita tangani exception tersebut

Test.java

```
1  class Test{
2      public static void delay() throws InterruptedException {
3          Thread.sleep(1000);
4      }
5
6      public static void main(String[] args) throws InterruptedException {
7          delay();
8
9          System.out.println("Hello, World");
10     }
11 }
```

Atau, kita deklarasikan untuk dilempar kembali

Exception dan Error

Test.java

```
1  import java.io.IOException;
2
3  class Test{
4      public static void readFile() {
5          try {
6              throw new IOException();
7          } catch (IOException e) {
8          }
9      }
10
11     public static void readFile2() throws IOException {
12         throw new IOException();
13     }
14
15     public static void main(String[] args) {
16
17     }
18 }
```

Catatan:

- Perhatikanlah bahwa java.io.IOException adalah checked exception
- Kita tangani sendiri dalam method readFile()
- Atau kita deklarasikan dengan throws supaya ditangani oleh yang lain
- Bedakanlah antara throws dan throw
  - throws: digunakan untuk mendeklarasikan checked exception yang dapat di-throw oleh sebuah method atau constructor (dipisahkan koma)
  - throw: menyebabkan sebuah exception di-throw

Exception dan Error

- Ketika suatu method di-override oleh subclass, beberapa aturan tentang exception berikut berlaku:
  - Method yang meng-override tidak boleh throw checked exception yang baru atau lebih umum. Hanya boleh lebih sedikit atau lebih spesifik (atau tidak dideklarasikan/throw sama sekali).
  - Method yang meng-override boleh throw unchecked exception

Exception dan Error

Test.java

```
1
2  import java.io.IOException;
3  import java.sql.SQLException;
4
5  class TestException1 {
6      public void test() throws IOException, SQLException {
7          int a = 0;
8          if (a == 0) {
9              throw new IOException();
10         } else {
11             throw new SQLException();
12         }
13     }
14 }
15
16 class TestException2 extends TestException1 {
17     public void test() {
18     }
19 }
20
21 class Test{
22     public static void main(String[] args) {
23     }
24 }
```

Catatan:

- Method test() milik TestException1 di-override di TestException2
- Kita lihat bahwa method yang meng-override boleh tidak throw exception

Exception dan Error

Test.java

```
1  import java.io.IOException;
2  import java.sql.SQLException;
3
4  class TestException1 {
5      public void test() throws IOException, SQLException {
6          int a = 0;
7          if (a == 0) {
8              throw new IOException();
9          } else {
10             throw new SQLException();
11         }
12     }
13 }
14
15 class TestException2 extends TestException1 {
16     public void test() {
17         throw new RuntimeException();
18     }
19 }
20
21 class Test{
22     public static void main(String[] args) {
23         new TestException2().test();
24     }
25 }
```

Method yang meng-override boleh throw unchecked exception

```
javac Test.java

java Test
Exception in thread "main" java.lang.RuntimeException
        at TestException2.test(Test.java:17)
        at Test.main(Test.java:23)
```

Exception dan Error

Test.java

```
1  import java.io.FileNotFoundException;
2  import java.io.IOException;
3  import java.sql.SQLException;
4
5  class TestException1 {
6      public void test() throws IOException, SQLException {
7          int a = 0;
8          if (a == 0) {
9              throw new IOException();
10         } else {
11             throw new SQLException();
12         }
13     }
14 }
15
16 class TestException2 extends TestException1 {
17     public void test() throws FileNotFoundException {
18     }
19 }
20
21 class Test{
22     public static void main(String[] args) {
23     }
24 }
```

Method yang meng-override boleh throw exception yang lebih spesifik.

Perhatikanlah bahwa java.io.FileNotFoundException adalah turunan dari java.io.IOException.

Exception dan Error

Test.java

```
1  import java.io.IOException;
2  import java.sql.SQLException;
3
4  class TestException1 {
5      public void test() throws IOException, SQLException {
6          int a = 0;
7          if (a == 0) {
8              throw new IOException();
9          } else {
10             throw new SQLException();
11         }
12     }
13 }
14
15 class TestException2 extends TestException1 {
16     public void test() throws Exception {
17     }
18 }
19
20 class Test{
21     public static void main(String[] args) {
22     }
23 }
```

Method yang meng-override tidak boleh throw Exception yang baru atau lebih umum.

Perhatikanlah bahwa Exception adalah superclass dari IOException ataupun SQLException.

```
javac Test.java
Test.java:16: error: test() in TestException2 cannot override test() in TestExce
ption1
    public void test() throws Exception {
                ^
  overridden method does not throw Exception
1 error
```

Exception dan Error

- finally: dikerjakan, terjadi exception atau tidak
  - Dapat dipergunakan misal untuk menutup resource yang dibuka sebelumnya

```
try {
    System.out.println("Hello");
} finally {
    System.out.println("Finally dikerjakan");
}

try {
    System.out.println(1);
} catch (ArithmeticException e) {
    System.out.println(e.getMessage());
} finally {
    System.out.println("Finally dikerjakan");
}

try {
    System.out.println(1/0);
} catch (ArithmeticException e) {
    System.out.println(e.getMessage());
} finally {
    System.out.println("Finally dikerjakan");
}
```

```
java Test
Hello
Finally dikerjakan
1
Finally dikerjakan
/ by zero
Finally dikerjakan
```

Exception dan Error

- Kita dapat menggunakan try-with-resources untuk class yang implementasikan java.lang.AutoCloseable, sehingga resource yang dibuka tidak perlu ditutup secara manual
  - Contoh penggunaan dibahas di bagian 5, ketika bekerja dengan sistem database

Array

- Array adalah objek, dibuat secara dinamis, dan dapat di-assign ke variabel dengan tipe Object
- Array dapat berisikan sejumlah elemen atau kosong. Setiap elemen diakses dengan indeks integer, dimulai dari 0.
- Setiap elemen dalam array (primitif atau reference) memiliki tipe yang sama
- Panjang array bukanlah bagian dari tipe array tersebut
- Array char bukanlah String
- Untuk multi dimensi, array di Java adalah array dari array
- Variabel instance dengan tipe array, yang tidak secara eksplisit diberikan nilai, akan memiliki nilai default null. Sementara elemen dalam array akan diberikan nilai default sebagaimana tipe-nya.
- Java Language Specification membahas tentang array di bab 10 dan 15.10

Array

- Deklarasi

  tipe, diikuti oleh satu atau lebih pasangan [dan], diikuti oleh nama variabel (lihatlah variasi berikut)

  int[] a;

  int[][] b;

  String[][] c;

  String d[]; ← posisi [] bisa dibelakang nama variabel

  String[] e[]; ← sama dengan String[][] e;

Array

Test.java

```
1  class Test{
2      public static void main(String[] args) {
3          int[] a;
4          int[][] b;
5          String[][] c;
6          String d[];
7          String[] e[];
8      }
9  }
```

Array

- Pembuatan

  Dengan keyword new, diikuti dengan tipe array tersebut, diikuti dengan [, jumlah elemen, dan]

  Contoh:

  a = new int[5];

  b = new int[5][];

  c = new String[5][];

  d = new String[5];

  e = new String[5][];
- Dapat dilakukan sekaligus dengan deklarasi.
  - Contoh: float[] f = new float[5];

Array

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           int[] a;
4           a = new int[5];
5
6           int[][] b;
7           b = new int[5][];
8
9           String[][] c;
10          c = new String[5][];
11
12          String d[];
13          d = new String[5];
14
15          String[] e[];
16          e = new String[5][];
17
18          float[] f = new float[5];
19      }
20  }
```

Array

- Inisialisasi
  - Dengan initializer: diawali {, dipisahkan koma untuk setiap element, diakhiri }
    Contoh:
    int[] a = {1,2,3};
  - Dengan memberikan nilai lewat indeks

```
int[] b = new int[3];
b[0] = 1;
b[1] = 2;
b[2] = 3;
```

    (dapat dilakukan juga lewat perulangan)

Array

Test.java

```
1  class Test{
2      public static void main(String[] args) {
3          int[] a = {1,2,3};
4
5          int[] b = new int[3];
6          b[0] = 1;
7          b[1] = 2;
8          b[2] = 3;
9
10         for (int i: a) {
11             System.out.println(i);
12         }
13
14         System.out.println();
15
16         for (int i: a) {
17             System.out.println(i);
18         }
19     }
20 }
```

```
javac Test.java

java Test
1
2
3

1
2
3
```

Array

- Inisialisasi array dari array
  - int[][] a = new int[3][];
  - Untuk setiap elemen dalam array tersebut (yang juga merupakan array):
    - a[0] = new int[3];
    - a[1] = new int[4];
    - a[2] = new int[] {8, 9, 10, 11, 12, 13};
      Perhatikanlah bahwa panjang setiap elemen tersebut bisa berbeda
  - Kemudian, lakukanlah inisialisasi elemen array seperti biasa
    a[0][0] = 1;
    dan seterusnya
    (atau, lakukanlah dalam perulangan)

Array

Test.java

```
1  class Test{
2      public static void main(String[] args) {
3          int[][] a = new int[3][];
4
5          a[0] = new int[3];
6          a[1] = new int[4];
7          a[2] = new int[] {8, 9, 10, 11, 12, 13};
8
9          a[0][0] = 1;
10         a[0][1] = 2;
11         a[0][2] = 3;
12
13         a[1][0] = 4;
14         a[1][1] = 5;
15         a[1][2] = 6;
16         a[1][3] = 7;
17
18         for (int[] i: a) {
19             for (int j: i) {
20                 System.out.print(j + " ");
21             }
22             System.out.println();
23         }
24     }
25 }
```

```
javac Test.java

java Test
1 2 3
4 5 6 7
8 9 10 11 12 13
```

Array

- Exception ketika akses pada indeks diluar batasan array
  - Ingatlah bahwa indeks dimulai dari 0
  - java.lang.ArrayIndexOutOfBoundsException

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           int[] a = new int[3];
4
5           a[0] = 1;
6           a[1] = 2;
7           a[2] = 3;
8
9           System.out.println(a[3]);
10      }
11  }
```

```
javac Test.java

java Test
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
        at Test.main(Test.java:9)
```

Array

- Mengetahui panjang array:
  - Field: public final int length

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           int[] a = new int[3];
4
5           a[0] = 1;
6           a[1] = 2;
7           a[2] = 3;
8
9           System.out.println(a.length);
10      }
11  }
```

```
javac Test.java

java Test
3
```

Array

- Mendapatkan representasi String dari isi sebuah array
  - Gunakanlah java.util.Arrays.toString(). Method ini di-overload.
  - Kita tidak menggunakan method toString() dari objek array itu sendiri

Test.java

```
1  import java.util.Arrays;
2
3  class Test{
4      public static void main(String[] args) {
5          int[] a = new int[3];
6
7          a[0] = 1;
8          a[1] = 2;
9          a[2] = 3;
10
11         System.out.println(a.toString());
12         System.out.println(Arrays.toString(a));
13     }
14 }
```

```
javac Test.java

java Test
[I@4aa298b7
[1, 2, 3]
```

Array

- Pengurutan: java.util.Arrays.sort()
- Fungsi-fungsi lain: lihatlah juga pada dokumentasi class java.util.Arrays

Test.java

```
1  import java.util.Arrays;
2
3  class Test{
4      public static void main(String[] args) {
5          int[] a = new int[3];
6
7          a[0] = 3;
8          a[1] = 1;
9          a[2] = 2;
10
11         System.out.println(Arrays.toString(a));
12         Arrays.sort(a);
13         System.out.println(Arrays.toString(a));
14     }
15 }
```

```
javac Test.java

java Test
[3, 1, 2]
[1, 2, 3]
```

Pengantar Generic

- Sebuah class, constructor, atau method adalah generic apabila class, constructor, atau method tersebut mendeklarasikan satu atau lebih type variable
  - Type variable tersebut dikenal sebagai type parameter dari class, constructor, atau method tersebut
- Pengantar generic di manual ini hanya dimaksudkan sebagai persiapan untuk pembahasan tentang collection
  - Dengan fokus pengantar hanya pada type-safety
- Generic programming dan generic di Java adalah topik yang kompleks dan membutuhkan pembahasan yang mendalam (di luar cakupan manual ini)

Pengantar Generic

- Sebelum Java 5.0

Test.java

```
1  import java.util.ArrayList;
2  import java.util.List;
3
4  class Test{
5      public static void main(String[] args) {
6          List list = new ArrayList();
7          list.add("Hello");
8          list.add(new Integer(10));
9
10         String s1 = (String) list.get(0);
11         String s2 = (String) list.get(1);
12     }
13 }
```

Catatan:

- Dalam hal ini, kita tidak bisa menentukan bahwa list tersebut hanya menerima String
- Pada baris 8, kita berhasil menambahkan sebuah Integer
- Sementara, kita mengharapkan String ketika ingin mendapatkan isi dari list pada baris 10 dan 11
- Yang mana, terjadi ClassCastException karena Integer tidak dapat di-cast ke String (di baris 11)

```
javac -Xlint:-options -source 1.4 Test.java

java Test
Exception in thread "main" java.lang.ClassCastException: java.lang.Integer canno
t be cast to java.lang.String
        at Test.main(Test.java:11)
```

Pengantar Generic

- Sebelum Java 5.0

Test.java

```
1  import java.util.ArrayList;
2  import java.util.List;
3
4  class Test{
5      public static void main(String[] args) {
6          List list = new ArrayList();
7          list.add("Hello");
8          list.add(new Integer(10));
9
10         for (int i=0; i<list.size(); i++) {
11             Object o = list.get(i);
12             if (o instanceof String) {
13                 String s = (String) o;
14                 System.out.println(s + ": " + s.length());
15             } else {
16                 System.out.println("Bukan String");
17             }
18         }
19     }
20 }
```

```
javac -Xlint:-options -source 1.4 Test.java

java Test
Hello: 5
Bukan String
```

Catatan:

- Di dalam perulangan, kita mendapatkan kembali isi dari list sebagai Object (baris 11)
- Kemudian, kita memeriksa apakah objek tersebut adalah instance dari String (baris 12)
- Apabila benar, barulah kita cast ke String (baris 13) dan panggil method dari String tersebut (baris 14)

Pengantar Generic

- Java 5.0 dan setelahnya

Test.java

```
1  import java.util.ArrayList;
2  import java.util.List;
3
4  class Test{
5      public static void main(String[] args) {
6          List<String> list = new ArrayList<>();
7          list.add("Hello");
8          list.add(new Integer(10));
9      }
10 }
```

```
javac Test.java
Test.java:8: error: no suitable method found for add(Integer)
        list.add(new Integer(10));
             ^
    method Collection.add(String) is not applicable
      (argument mismatch; Integer cannot be converted to String)
    method List.add(String) is not applicable
      (argument mismatch; Integer cannot be converted to String)
Note: Some messages have been simplified; recompile with -Xdiags:verbose to get
full output
1 error
```

Catatan:

- Kita membuat sebuah list yang hanya bisa menerima String, dengan List<String> list (baris 6)
- Di baris 8, kita menambahkan sebuah Integer
- Dan, kompilasi gagal
- Perhatikanlah penggunaan diamond operator <> pada baris 6, yang memungkinkan type inference (sejak Java 7)

Pengantar Generic

- Java 5.0 dan setelahnya

Test.java

```
1  import java.util.ArrayList;
2  import java.util.List;
3
4  class Test{
5      public static void main(String[] args) {
6          List<String> list = new ArrayList<>();
7          list.add("Hello");
8
9          for (String s: list) {
10             System.out.println(s + ": " + s.length());
11         }
12     }
13 }
```

```
javac Test.java

java Test
Hello: 5
```

Catatan:

- Karena kompilasi berhasil, pemeriksaan dan cast tidak lagi diperlukan
- Perhatikanlah bahwa kode menjadi lebih singkat dan exception tidak lagi terjadi (gara-gara cast yang gagal)

Collection

- Sebuah collection adalah sebuah objek yang mewakili sekumpulan objek (elemen)
- Java menyediakan sejumlah class collection dengan struktur data dan algoritma siap pakai, yang mengimplementasikan interface tertentu
- Terbagi atas interface mendasar berikut:
  - java.util.Collection
  - java.util.Map
- Manual ini hanya akan membahas beberapa implementasi collection. Untuk selengkapnya, bacalah tentang Java Collections Framework, yang dapat diakses dari dokumentasi interface java.util.Collection atau java.util.Map.

Collection

- java.util.Collection memiliki beberapa subinterface berikut (dalam konteks collection):
  - java.util.List
  - java.util.Set
  - java.util.SortedSet
  - java.util.NavigableSet
  - java.util.Queue
  - java.util.concurrent.BlockingQueue
  - java.util.concurrent.TransferQueue
  - java.util.Deque
  - java.util.concurrent.BlockingDeque

Collection

- java.util.Map memiliki beberapa subinterface berikut (dalam konteks collection):
  - java.util.SortedMap
  - java.util.NavigableMap
  - java.util.concurrent.ConcurrentMap
  - java.util.concurrent.ConcurrentNavigableMap

Collection

- java.util.List
  - Sebuah ordered collection
  - Elemen dapat diposisikan tertentu
  - Elemen dapat diakses dengan indeks (integer, dimulai dari 0)
  - Elemen dapat dicari
  - Umumnya mengijinkan duplikasi (termasuk null, apabila null diijinkan)
  - Beberapa method (umumnya di-overload):
    - add(): menambahkan elemen
    - get(): mendapatkan elemen
    - remove(): menghapus elemen
    - size(): mendapatkan jumlah elemen dalam List
    - contains(): memeriksa apakah List mengandung suatu elemen
  - Implementasi yang dibahas dalam manual ini: java.util.ArrayList

Collection

- java.util.ArrayList
  - Berbeda dengan array, ukuran ArrayList dapat berkembang secara dinamis
  - ArrayList mengijinkan null
  - Alternatif: java.util.Vector (synchronized)

Collection

Test.java

```
1  import java.util.ArrayList;
2  import java.util.List;
3
4  class Test{
5      public static void main(String[] args) {
6          List<String> list = new ArrayList<>();
7          list.add("World");
8          list.add(null);
9          list.add(0, "Hello");
10         list.add("test");
11
12         list.remove(list.size()-1);
13
14         for (int i=0; i<list.size(); i++) {
15             String s = list.get(i);
16             System.out.println(s);
17         }
18
19         System.out.println();
20
21         for (String s: list) {
22             System.out.println(s);
23         }
24     }
25 }
```

```
javac Test.java

java Test
Hello
World
null

Hello
World
null
```

Catatan:

- Setelah list dibuat (baris 6), kita menambahkan “World” (indeks 0)
- Kemudian, kita menambahkan null yang diijinkan (indeks 1)
- Lalu, pada indeks 0, kita masukkan “Hello” (baris 9). Dengan demikian, list kita berisi: “Hello”, “World”, null
- Kemudian, kita tambahkan “test” (baris 10)
- Lalu, kita hapus elemen dengan indeks [size dari list dikurangi 1], yang mana merupakan elemen terakhir, atau “test”
- Pada baris 14, kita melakukan iterasi dengan for biasa. Di baris 15, kita mendapatkan elemen dengan indeks (method get())
- Enhanced for disarankan (baris 21)

Collection

Test.java

```
1  import java.util.ArrayList;
2  import java.util.List;
3
4  class Test{
5      public static void main(String[] args) {
6          List<List<String>> lists = new ArrayList<>();
7
8          List<String> list1 = new ArrayList<>();
9          list1.add("Hello");
10         list1.add("World");
11
12         List<String> list2 = new ArrayList<>();
13         list2.add("1");
14         list2.add("2");
15
16         lists.add(list1);
17         lists.add(list2);
18
           for (List<String> list: lists) {
               for (String s: list) {
21                 System.out.print(s + " ");
22             }
23             System.out.println();
24         }
25     }
26 }
```

```
javac Test.java

java Test
Hello World
1 2
```

Contoh List dari List<String>

Collection

- java.util.Map
  - Memetakan key ke value, dimana duplikasi key tidak dimungkinkan
  - Terdapat sejumlah implementasi. Ada yang memungkinkan order dari sebuah Map berdasarkan aturan tertentu.
  - Sejumlah implementasi mengijinkan null – baik pada key dan/atau value – sementara sejumlah implementasi lainnya tidak mengijinkan
  - Beberapa method:
    - put(): memetakan key ke value
    - get(): mengembalikan value sesuai dengan key
    - remove(): menghapus pemetaan
    - size(): mendapatkan jumlah pemetaan key ke value
    - keySet(): mengembalikan key dalam Map sebagai sebuah Set
    - values(): mengembalikan value dalam Map sebagai sebuah Collection
    - containsKey(): memeriksa apakah Map mengandung key tertentu
    - containsValue(): memeriksa apakah Map mengandung value tertentu
  - Implementasi yang dibahas dalam manual ini: java.util.HashMap dan java.util.LinkedHashMap

Collection

- java.util.HashMap
  - Sebuah hash table
  - Mengijinkan null untuk key ataupun value
  - Order map tidak dijamin
  - Alternatif: java.util.Hashtable (synchronized)

Collection

Test.java

```
1  import java.util.HashMap;
2  import java.util.Map;
3
4  class Test{
5      public static void main(String[] args) {
6          Map<String, Integer> map = new HashMap<>();
7
8          map.put("a", 1);
9          map.put("b", 2);
10
11         map.put("c", 3);
12         map.remove("c");
13
14         System.out.println("Ukuran map: " + map.size());
15
16         System.out.println("Daftar key: ");
           for (String k: map.keySet()) {
18             System.out.println(k);
19         }
20
21         System.out.println("Daftar value: ");
           for (Integer v: map.values()) {
23             System.out.println(v);
24         }
25
26         System.out.println("Daftar key -> value: ");
           for (String k: map.keySet()) {
28             System.out.println(k + " -> " + map.get(k));
29         }
30     }
31 }
```

```
javac Test.java

java Test
Ukuran map: 2
Daftar key:
a
b
Daftar value:
1
2
Daftar key -> value:
a -> 1
b -> 2
```

Catatan:

- Kita membuat sebuah Map dari String ke Integer, dan memetakan (dengan put()) “a” ke 1, “b” ke 2, dan “c” ke 3 (baris 6, 8, 9, 11). Kemudian, kita hapus pemetaan “c” (baris 12). Dengan demikian, jumlah pemetaan adalah 2 (baris 14).
- Daftar key dan value masing-masing didapatkan dengan keySet() dan values() (baris 17 dan 22)
- Untuk mendapatkan value yang dipetakan oleh suatu key, kita menggunakan get()

Collection

Test.java

```
1  import java.util.HashMap;
2  import java.util.Map;
3
4  class Test{
5      public static void main(String[] args) {
6          Map<String, Map<String, Map<Integer, String>>> map = new HashMap<>();
7
8
9          Map<Integer, String> map1 = new HashMap<>();
10         map1.put(1, "a");
11         map1.put(2, "b");
12
13         Map<String, Map<Integer, String>> map2 = new HashMap<>();
14         map2.put("hello", map1);
15
16         map.put("test", map2);
17
18         for (String k: map.keySet()) {
19             System.out.println(k);
20             for (String k2: map.get(k).keySet()) {
21                 Map<Integer, String> temp = map.get(k).get(k2);
22                 System.out.println("\t" + k2);
23                 for (Integer k3: temp.keySet()) {
24                     System.out.println("\t\t" + k3 + "->" + temp.get(k3));
25                 }
26             }
27         }
28
29     }
30 }
```

```
javac Test.java

java Test
test
        hello
                1->a
                2->b
```

Contoh:

- Map dari String ke
  - Map dari String ke
    - Map dari Integer ke String

Collection

- java.util.LinkedHashMap
  - Kombinasi hash table dan doubly-linked list
  - Mengijinkan null untuk key ataupun value
  - Order map dijamin:
    - Insertion order
    - Access order

Collection

- java.util.Collections: class untuk bekerja dengan collection. Sebagai contoh: reverse, shuffle, sort, dan lainnya.

Test.java

```
1  import java.util.ArrayList;
2  import java.util.Collections;
3  import java.util.List;
4
5  class Test{
6      public static void main(String[] args) {
7          List<String> list = new ArrayList<>();
8          list.add("C");
9          list.add("A");
10         list.add("B");
11         System.out.println(list);
12
13         Collections.sort(list);
14         System.out.println(list);
15
16         Collections.reverse(list);
17         System.out.println(list);
18
19         Collections.shuffle(list);
20         System.out.println(list);
21         Collections.shuffle(list);
22         System.out.println(list);
23     }
24 }
```

```
javac Test.java

java Test
[C, A, B]
[A, B, C]
[C, B, A]
[A, C, B]
[C, A, B]
```

BigInteger dan BigDecimal

- java.math.BigInteger: dapat digunakan ketika kita membutuhkan integer yang lebih dari batasan long

Test.java ×

```
1   class Test{
2       public static void main(String[] args) {
3           long number1 = 92233720368547758O7L;
            long number2 = 92233720368547758O8L;
5       }
6   }
```

```
javac Test.java
Test.java:4: error: integer number too large: 9223372036854775808
        long number2 = 92233720368547758O8L;
                       ^
1 error
```

BigInteger dan BigDecimal

Test.java

```
1  import java.math.BigInteger;
2
3  class Test{
4      public static void main(String[] args) {
5          BigInteger number = new BigInteger("9223372036854775808");
6          System.out.println(number);
7          System.out.println();
8
9          number = number.pow(8);
10         System.out.println(number);
11         System.out.println();
12
13         number = number.add(number);
14         System.out.println(number);
15         System.out.println();
16
17         number = number.subtract(new BigInteger("10"));
18         System.out.println(number);
19         System.out.println();
20
21         number = number.multiply(number);
22         System.out.println(number);
23         System.out.println();
24
25         number = number.divide(new BigInteger("2"));
26         System.out.println(number);
27         System.out.println();
28     }
29 }
```

```
javac Test.java

java Test
9223372036854775808

52374249726338269920211035149241586435466272736689036631732661889538140742474792
87813232147721446651441418694604096113614747610473416628885325644143001 6

10474849945267653984042207029848317287093254547337807326346532377907628148494958
57562646429544289330288283738920819222722949522094683325777065128828600 32

10474849945267653984042207029848317287093254547337807326346532377907628148494958
57562646429544289330288283738920819222722949522094683325777065128828600 22

10972248137587377366511872502374418540148785271864664140224003976912394763519345
89433035139907272558722656945067574422348991636772548929580644820743648924562923
99438706232040876652107321264493983735575210594502839076063822281173462654602241
41694333763944337715309380734238381103422920720567168346445840484

54861240687936886832559362511872092700743926359323320701120019884561973817596729
47165175699536362793613284725337872111744958183862744647903224103718244622814619
97193531160204383260536606322469918677876052972514195380319111405867313273011207
08471668819721688576546903671191905517114603602835841732229202 42
```

Contoh penggunaan method pow(), add(), subtract(), multiple(), dan divide()

BigInteger dan BigDecimal

- java.math.BigDecimal: digunakan ketika kita membutuhkan bilangan desimal yang besar/kecil, dengan presisi dan pembulatan yang dapat ditentukan (misal dalam aplikasi keuangan)

Test.java

```
1   class Test{
2       public static void main(String[] args) {
3           System.out.println(0.1 + 0.1 + 0.1);
4           System.out.println(0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1);
5           System.out.println(0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1
6                   + 0.1 + 0.1);
7       }
8   }
```

```
javac Test.java

java Test
0.30000000000000004
0.7999999999999999
0.9999999999999999
```

BigInteger dan BigDecimal

Test.java

```
1
2  import java.math.BigDecimal;
3
4  class Test{
5      public static void main(String[] args) {
6          BigDecimal number1 = new BigDecimal("0.1");
7
8          BigDecimal number2 = number1.add(number1).add(number1);
9          System.out.println(number2);
10
11         BigDecimal number3 = number2.add(number1).multiply(new BigDecimal(2));
12         System.out.println(number3);
13
14         BigDecimal number4 = number3.add(number1.
15                 multiply(new BigDecimal(3))).subtract(number1);
16         System.out.println(number4);
17     }
18 }
```

```
javac Test.java

java Test
0.3
0.8
1.0
```

Contoh penggunaan method add(), subtract(), dan multiple()

BigInteger dan BigDecimal

Test.java

```
1  import java.math.BigDecimal;
2
3  class Test{
4      public static void main(String[] args) {
5          BigDecimal number1 = new BigDecimal("1");
6          BigDecimal number2 = new BigDecimal("3");
7          BigDecimal number3 = number1.divide(number2);
8          System.out.println(number3);
9      }
10 }
```

```
javac Test.java

java Test
Exception in thread "main" java.lang.ArithmeticException: Non-terminating decima
l expansion; no exact representable decimal result.
        at java.math.BigDecimal.divide(BigDecimal.java:1690)
        at Test.main(Test.java:7)
```

Contoh penggunaan method divide() yang gagal (terjadi exception), karena memiliki ekspansi desimal yang tidak terbatas

BigInteger dan BigDecimal

Test.java

```
1  import java.math.BigDecimal;
2  import java.math.RoundingMode;
3
4  class Test{
5      public static void main(String[] args) {
6          System.out.println(BigDecimal.valueOf(1).
7                  divide(BigDecimal.valueOf(3), RoundingMode.HALF_UP));
8
9          System.out.println(BigDecimal.valueOf(1).
10                 divide(BigDecimal.valueOf(3), RoundingMode.HALF_DOWN));
11
12         System.out.println(BigDecimal.valueOf(1).
13                 divide(BigDecimal.valueOf(3), RoundingMode.UP));
14
15         System.out.println(BigDecimal.valueOf(1).
16                 divide(BigDecimal.valueOf(3), RoundingMode.DOWN));
17
18         System.out.println(BigDecimal.valueOf(1).
19                 divide(BigDecimal.valueOf(3), RoundingMode.CEILING));
20
21         System.out.println(BigDecimal.valueOf(1).
22                 divide(BigDecimal.valueOf(3), RoundingMode.FLOOR));
23     }
24 }
```

```
javac Test.java

java Test
0
0
1
0
1
0
```

Perhatikanlah bahwa dengan divide() yang di-overload, yang membutuhkan RoundingMode, pembagian tidak lagi terjadi exception. Tapi, perhatikanlah bahwa 1/3 adalah 0 (kita menggunakan beberapa metode pembulatan).

BigInteger dan BigDecimal

Test.java

```
1  import java.math.BigDecimal;
2  import java.math.RoundingMode;
3
4  class Test{
5      public static void main(String[] args) {
6          BigDecimal number1 = new BigDecimal(1);
7          BigDecimal number3 = new BigDecimal(3);
8
9          for (int scale=0; scale<10; scale++) {
10             System.out.println(number1.divide(number3, scale,
11                     RoundingMode.HALF_UP));
12         }
13     }
14 }
```

```
javac Test.java

java Test
0
0.3
0.33
0.333
0.3333
0.33333
0.333333
0.3333333
0.33333333
0.333333333
```

Perhatikanlah bahwa kita menggunakan method divide() yang di-overload, yang membutuhkan scale dan RoundingMode. Dengan demikian, pembagian tidak lagi terjadi exception, dan scale bisa kita tentukan seperti dalam contoh output.

BigInteger dan BigDecimal

Test.java

```
1  import java.math.BigDecimal;
2  import java.math.MathContext;
3  import java.math.RoundingMode;
4
5  class Test{
6      public static void main(String[] args) {
7          BigDecimal number1 = new BigDecimal(1);
8          BigDecimal number3 = new BigDecimal(3);
9          BigDecimal number10 = new BigDecimal(10);
10         BigDecimal number30 = new BigDecimal(30);
11
12         BigDecimal result1 = number1.divide(number3, 3, RoundingMode.HALF_UP);
13         System.out.println(result1);
14
15         MathContext context = new MathContext(3, RoundingMode.HALF_UP);
16         BigDecimal result2 = number1.divide(number3, context);
17         System.out.println(result2);
18         BigDecimal result3 = number10.divide(number30, context);
19         System.out.println(result3);
20     }
21 }
```

```
javac Test.java

java Test
0.333
0.333
0.333
```

Sebagai alternatif, kita bisa mempergunakan MathContext dan method divide yang di-overload, yang membutuhkan sebuah MathContext

Pengantar File dan Path

- Kita dapat bekerja dengan file (baik pada isi file ataupun operasi pada file), filesystem, ataupun hal-hal terkait lainnya (yang bisa sangat kompleks)
- Pada manual ini, kita hanya membahas beberapa contoh bekerja dengan file teks, yang diperlukan untuk mendukung pembahasan lain
- Bacalah juga API Specification untuk class ataupun interface yang digunakan dalam pembahasan pengantar ini

Pengantar File dan Path

- java.io.File: sebuah class, representasi abstrak file dan direktori
- java.nio.file.Path: sebuah interface, disarankan ketika bekerja dengan file (dibandingkan dengan java.io.File)
  - java.nio.file.Paths: sebuah class, digunakan untuk membuat objek java.nio.file.Path
  - java.nio.file.Files: sebuah class, digunakan untuk bekerja dengan objek java.nio.file.Path
- Interoperabilitas File ke Path dan sebaliknya:
  - Method toPath() milik File
  - Method toFile() milik Path

Pengantar File dan Path

Test.java

```
1  import java.io.IOException;
2  import java.nio.file.Files;
3  import java.nio.file.Path;
4  import java.nio.file.Paths;
5
6  class Test{
7      public static void main(String[] args) {
8          Path path = Paths.get("test.txt");
9          System.out.println(Files.exists(path));
10
11         try {
12             Files.createFile(path);
13             System.out.println(Files.exists(path));
14
15             Files.delete(path);
16             System.out.println(Files.exists(path));
17         } catch (IOException e) {
18             System.out.println(e);
19         }
20     }
21 }
```

```
javac Test.java

java Test
false
true
false
```

Catatan:

- Pertama-tama, ketika bekerja dengan file, sejumlah method mendeklarasikan bahwa IOException (sebuah checked exception) dapat terjadi. Oleh karena itu, kita mungkin perlu tangani exception ini.
- Pada baris 8, kita membuat sebuah objek Path dan memeriksa apakah file yang diwakili oleh objek tersebut ditemukan di file sistem. Karena tidak ada, maka false dikembalikan.
- Pada baris 12, kita membuat file tersebut dan memeriksa apakah file yang diwakili tersebut sekarang ditemukan, di baris 13. Pembuatan berhasil dan true dikembalikan.
- Pada baris 15, kita menghapus file dan sekali lagi memeriksanya di baris 16.

Pengantar File dan Path

Test.java

```
1  import java.io.IOException;
2  import java.nio.file.Files;
3  import java.nio.file.Path;
4  import java.nio.file.Paths;
5
6  class Test{
7      public static void main(String[] args) throws IOException {
8          Path dir = Paths.get("test");
9          Files.createDirectories(dir);
10
11         Path path = Paths.get(dir.toString(), "test.txt");
12         System.out.println(Files.exists(path));
13
14         Files.createFile(path);
15         System.out.println(Files.exists(path));
16     }
17 }
```

```
javac Test.java

java Test
false
true
```

Catatan:

- Kita dapat membuat direktori (dan subdirektori) dengan method createDirectories() milik Files (baris 9)
- Objek Path bisa dibuat dengan mengkombinasikan direktori (baris 11)
- Apabila file telah ditemukan, exception java.nio.file.FileAlreadyExistsException akan terjadi

Pengantar File dan Path

Test.java

```
1  import java.io.IOException;
2  import java.nio.file.Files;
3  import java.nio.file.Path;
4  import java.nio.file.Paths;
5  import java.util.ArrayList;
6  import java.util.List;
7
8  class Test{
9      public static void main(String[] args) throws IOException {
10         List<String> list = new ArrayList<>();
11         list.add("Hello");
12         list.add("World");
13
14         Path path = Paths.get("test.txt");
15         if (!Files.exists(path)) {
16             Files.createFile(path);
17         }
18         System.out.println("Ukuran file: " + Files.size(path));
19
20         Files.write(path, list);
21         System.out.println("Ukuran file: " + Files.size(path));
22     }
23 }
```

```
javac Test.java

java Test
Ukuran file: 0
Ukuran file: 12
```

Catatan:

- Kita membuat sebuah List dari String, dan menambahkan “Hello” dan “World” (baris 10-12)
- Sebuah objek Path, mewakili file “test.txt”, dibuat dan diperiksa apakah ditemukan di file sistem. Apabila tidak, maka file kita buat (baris 14-17).
- Pada baris 18, kita dapatkan ukuran file dengan method size() milik Files, yang dalam contoh ini mengembalikan 0 (file baru saja dibuat). Dapat terjadi exception java.nio.file.NoSuchFileException apabila file tidak ditemukan.
- Pada baris 20, dengan method write() milik Files, kita tulis isi dari List ke dalam file teks

Pengantar File dan Path

Test.java

```
2  import java.nio.file.Files;
3  import java.nio.file.Path;
4  import java.nio.file.Paths;
5  import java.nio.file.StandardOpenOption;
6  import java.util.ArrayList;
7  import java.util.List;
8
9  class Test{
10     public static void main(String[] args) throws IOException {
11         List<String> list = new ArrayList<>();
12         list.add("Hello");
13         list.add("World");
14
15         Path path = Paths.get("test.txt");
16         if (!Files.exists(path)) {
17             Files.createFile(path);
18         }
19         System.out.println("Ukuran file: " + Files.size(path));
20
21         Files.write(path, list, StandardOpenOption.APPEND);
22         System.out.println("Ukuran file: " + Files.size(path));
23     }
24 }
```

```
javac Test.java

java Test
Ukuran file: 0
Ukuran file: 12

java Test
Ukuran file: 12
Ukuran file: 24

java Test
Ukuran file: 24
Ukuran file: 36
```

Untuk menambahkan pada akhir file, kita bisa gunakan StandardOpenOption.APPEND sebagai argumen pada method write() milik Files. Beberapa opsi lainnya tersedia dan dapat dibaca di dokumentasi API.

Pengantar File dan Path

Test.java

```
1  import java.io.IOException;
2  import java.nio.file.Files;
3  import java.nio.file.Path;
4  import java.nio.file.Paths;
5  import java.util.ArrayList;
6  import java.util.List;
7
8  class Test{
9      public static void main(String[] args) throws IOException {
10         Path path = Paths.get("test.txt");
11         if (Files.exists(path)) {
12             System.out.println("Ukuran file: " + Files.size(path));
13
14             List<String> list = new ArrayList<>();
15             list = Files.readAllLines(path);
16
17             for (String s: list) {
18                 System.out.println(s);
19             }
20         }
21     }
22 }
```

```
javac Test.java

java Test
Ukuran file: 36
Hello
World
Hello
World
Hello
World
```

Untuk membaca dari file teks, kita bisa gunakan method readAllLines() milik Files, yang akan mengembalikan apa yang dibaca ke sebuah List dari String

Pengantar File dan Path

Test.java

```
1  import java.io.IOException;
2  import java.nio.file.Files;
3  import java.nio.file.Path;
4  import java.nio.file.Paths;
5  import java.nio.file.StandardCopyOption;
6
7  class Test{
8      public static void main(String[] args) throws IOException {
9          Path src = Paths.get("test.txt");
10         if (Files.exists(src)) {
11             Path dest = Paths.get("copy.txt");
12             Files.copy(src, dest, StandardCopyOption.REPLACE_EXISTING);
13         }
14     }
15 }
```

```
javac Test.java

java Test
```

Untuk mengopi file, salah satu caranya adalah dengan menggunakan method copy milik Files. Tanpa adanya StandardCopyOption.REPLACE_EXISTING yang dilewatkan, sebuah exception java.nio.file.FileAlreadyExistsException akan terjadi apabila file telah ditemukan di file sistem.

Properties

- java.util.Properties (yang diturunkan dari java.util.Hashtable) dapat digunakan untuk menyimpan pengaturan (dalam bentuk pemetaan key→value, String ke String), yang pada akhirnya dapat disimpan ke file atau dibaca dari file
  - Untuk membuat pemetaan baru, kita bisa gunakan setProperty()
  - Untuk membaca dari pemetaan yang ada, kita bisa gunakan getProperty
  - Untuk menulis ke file, kita bisa gunakan method store() (di-overload, salah satunya mengharapkan sebuah java.io.Writer)
  - Untuk membaca dari file, kita bisa gunakan method load() (di-overload, salah satunya mengharapkan java.io.Reader)

Properties

Test.java

```
1  import java.util.Properties;
2
3  class Test{
4      public static void main(String[] args) {
5          Properties prop = new Properties();
6          prop.setProperty("name", "Hello World");
7          prop.setProperty("year", "2018");
8
9          System.out.println(prop.getProperty("name"));
10         System.out.println(prop.getProperty("year"));
11     }
12 }
```

```
javac Test.java

java Test
Hello World
2018
```

Properties

Test.java

```
1  import java.io.IOException;
2  import java.nio.file.Files;
3  import java.nio.file.Path;
4  import java.nio.file.Paths;
5  import java.util.Properties;
6
7  class Test{
8      public static void main(String[] args) throws IOException {
9          Properties prop = new Properties();
10         prop.setProperty("name", "Hello World");
11         prop.setProperty("year", "2018");
12
13         Path path = Paths.get("test.properties");
14         prop.store(Files.newBufferedWriter(path), "komentar");
15     }
16 }
```

```
javac Test.java

java Test
```

Catatan:

- Perhatikanlah bahwa newBufferedWriter() milik Files mengembalikan sebuah java.io.BufferedWriter, yang juga merupakan sebuah java.io.Writer
- Sementara, store() milik Properties mengharapkan sebuah java.io.Writer

Contoh isi file test.properties

```
#komentar
#Mon Sep 10 14:49:00 WIB 2018
name=Hello World
year=2018
```

Properties

Test.java

```
1  import java.io.IOException;
2  import java.nio.file.Files;
3  import java.nio.file.Path;
4  import java.nio.file.Paths;
5  import java.util.Properties;
6
7  class Test{
8      public static void main(String[] args) throws IOException {
9          Path path = Paths.get("test.properties");
10
11         Properties prop = new Properties();
12         prop.load(Files.newBufferedReader(path));
13
14         System.out.println(prop.getProperty("name"));
15         System.out.println(prop.getProperty("year"));
16     }
17 }
```

Catatan:

- Kita menggunakan file test.properties yang dihasilkan sebelumnya
- Perhatikanlah bahwa load() milik Properties mengharapkan sebuah java.io.Reader
- Sementara, newBufferedReader() milik Files mengembalikan sebuah java.io.BufferedReader, yang juga merupakan sebuah java.io.Reader

```
javac Test.java

java Test
Hello World
2018
```

Contoh Soal

- Kita telah membahas tentang file, Properties, dan lainnya. Cobalah untuk membuat program yang:
  - Meminta sejumlah input kepada user
  - Sebagian dari input tersebut disimpan ke file properties (misal: nama)
  - Ketika program dijalankan kembali, apa yang disimpan di dalam file properties tersebut dapat dibaca kembali (misal: nama)
  - Sejumlah exception mungkin akan terjadi. Pastikanlah bahwa ini telah ditangani dengan baik.

Bagian 4

Enum,

Nested Class, Static Nested Class,

Inner Class, Anonymous Inner Class,

Pengantar GUI dengan Swing,

Komponen, Layout Manager, Event, Dialog,

SwingWorker

Enum

- Merupakan sebuah tipe, yang memungkinkan kita untuk mendefinisikan sebuah enumerasi nilai/konstanta
- Dapat diterapkan access modifier. Enum dapat dideklarasikan top level ataupun di dalam class, dan aturan access modifier berlaku.
- Enum tidak dapat dideklarasikan abstract
- Superclass langsung enum adalah java.lang.Enum
- Konstanta enum dapat diikuti oleh argumen, yang akan dilewatkan ke constructor

  (*Java Language Specification* 8.9)

Enum

- Tanpa enumerasi, kita mungkin akan bekerja dengan integer seperti contoh berikut

Test.java

```
1   class Test{
2       public static final int RED = 1;
3       public static final int GREEN = 2;
4       public static final int BLUE = 3;
5
6       public static void main(String[] args) {
7           int color = 1;
8           if (color == RED) {
9               System.out.println("R");
10          }
11      }
12  }
```

```
javac Test.java

java Test
R
```

Perhatikanlah bahwa kita bisa saja memberikan nilai integer lainnya ke variabel color, selama masih dalam batasan tipe data

Enum

- Tanpa enumerasi, kita juga mungkin akan bekerja dengan String seperti contoh berikut

Test.java

```
1   class Test{
2       public static final String RED = "R";
3       public static final String GREEN = "G";
4       public static final String BLUE = "B";
5
6       public static void main(String[] args) {
7           String color = "R";
8           if (color.equals(RED)) {
9               System.out.println(color);
10          }
11      }
12  }
```

```
javac Test.java

java Test
R
```

Perhatikanlah bahwa serupa seperti contoh sebelumnya, kita dapat memberikan nilai apapun ke dalam variabel color selama masih dalam batasan (termasuk null)

Enum

- Dengan enum, contoh sebelumnya dapat dituliskan lebih terbaca dan pasti

Test.java

```
1   enum Colors {
2       RED, GREEN, BLUE
3   }
4
5   class Test{
6       public static void main(String[] args) {
7           Colors color = Colors.RED;
8           if (color == Colors.RED) {
9               System.out.println(color);
10          }
11      }
12  }
```

```
javac Test.java

java Test
RED
```

Perhatikanlah bahwa kita hanya bisa memberikan nilai RED, GREEN, BLUE untuk color. Kita tidak bisa memberikan nilai diluar enumerasi yang telah didefinisikan.

Enum

Test.java

```
1   enum Colors {
2       RED, GREEN, BLUE
3   }
4
5   class Test{
6       public static void main(String[] args) {
7           Colors color = 1;
8           Colors color2 = "R";
9       }
10  }
```

```
javac Test.java
Test.java:7: error: incompatible types: int cannot be converted to Colors
        Colors color = 1;
                       ^
Test.java:8: error: incompatible types: String cannot be converted to Colors
        Colors color2 = "R";
                        ^
2 errors
```

Enum

- Kita bisa memberikan nilai khusus untuk setiap konstanta enum, yang kemudian akan dilewatkan ke constructor

Test.java

```
1   enum Colors {
2       RED("R"), GREEN("G"), BLUE("B");
3       private final String code;
4       Colors(String code) {
5           this.code = code;
6       }
7
8       public String getCode() {
9           return code;
10      }
11  }
12
13  class Test{
14      public static void main(String[] args) {
15          Colors color = Colors.RED;
16          if (color == Colors.RED) {
17              System.out.println(color.getCode());
18          }
19      }
20  }
```

```
javac Test.java

java Test
R
```

Nested Class

- Nested class: class yang dideklarasikan di dalam class (enclosing class) atau interface lain → menjadi bagian dari class seperti halnya field/method
  - Static nested class: seperti class top-level yang ditempatkan di dalam class lain
  - Non-static → inner class:
    - Local (inner class yang memiliki nama)
    - Anonymous (inner class yang tidak memiliki nama)

  (*Java Language Specification* 8.1.3, 14.3, 15.9.5)
- Contoh alasan penggunaan:
  - Sebuah class hanya digunakan sekali, untuk kebutuhan spesifik (misal: event listener ketika sebuah tombol di-klik)
  - Sebuah class merupakan bagian dari class lain
- Pembahasan nested class di dalam manual ini hanyalah sebagai pengantar untuk pembahasan lainnya

Nested Class

- Akses pada field/method instance (yang non-static):
  - Static nested class: tidak memiliki akses
  - Inner class: memiliki akses, termasuk ke yang private
- Access modifier dapat diberikan pada nested class: public protected default/package-private private
  - Kecuali inner class yang dideklarasikan di dalam sebuah method
- Akses dari luar terhadap nested class:
  - Static nested class: membutuhkan nama outer class
  - Inner class: membutuhkan reference ke instance dari outer class

Static Nested Class

Test.java

```
1   class Outer {
2       private static String NAME = "Outer";
3       private int x = 10;
4
5       protected static class Nested {
6           void methodInNested() {
7               System.out.println("method In Nested, akses static: " + NAME);
8           }
9       }
10  }
11
12
13  public class Test{
14      public static void main(String[] args) {
15          Outer.Nested nested = new Outer.Nested();
16          nested.methodInNested();
17      }
18  }
```

```
javac Test.java

java Test
method In Nested, akses static: Outer
```

Catatan:

- Perhatikanlah bahwa class Nested bisa diberikan contoh access modifier protected, sementara top-level seperti Outer tidak bisa
- Method dalam Nested bisa mengakses NAME (static), namun tidak bisa x (non-static)
- Di baris 15, kita bisa membuat instance Nested tanpa adanya instance dari Outer
- Di baris 15, kita tetap membutuhkan nama Outer ketika mengakses Nested

Inner Class

Test.java

```
1  class Outer {
2      private static String NAME = "Outer";
3      private int x = 10;
4
5      class Inner {
6          private int x = 20;
7          void methodInInner() {
8              System.out.println("x: " + this.x);
9              System.out.println("x outer: " + Outer.this.x);
10         }
11     }
12
13     void methodInOuter() {
14         Inner inner = new Inner();
15         inner.methodInInner();
16     }
17 }
18
19
20 public class Test{
21     public static void main(String[] args) {
22         Outer outer = new Outer();
23         outer.methodInOuter();
24     }
25 }
```

```
javac Test.java

java Test
x: 20
x outer: 10
```

Catatan:

- Perhatikanlah bahwa inner class bisa mengakses field x yang private (baris 9)
- Di dalam Inner, untuk mengakses x milik sendiri, dapat dengan this.x, namun untuk mengakses x milik Outer, kita gunakan Outer.this.x (baris 8 dan 9)
- Di dalam methodInOuter() (baris 13-15), kita membuat instance dari Inner dan memanggil methodnya
- Class Outer dapat diinstansiasi seperti biasa (baris 22-23)

Inner Class

Test.java

```
1   class Outer {
2       private static String NAME = "Outer";
3       private int x = 10;
4
5       class Inner {
6           private int x = 20;
7           void methodInInner() {
8               System.out.println("x: " + this.x);
9               System.out.println("x outer: " + Outer.this.x);
10          }
11      }
12  }
13
14
15  public class Test{
16      public static void main(String[] args) {
17          Outer outer = new Outer();
18          Outer.Inner inner = outer.new Inner();
19          inner.methodInInner();
20      }
21  }
```

```
javac Test.java

java Test
x: 20
x outer: 10
```

Catatan:

- Kita memodifikasi Outer sehingga instansiasi Inner dilakukan dari luar
- Pertama, kita instansiasi Outer terlebih dahulu (baris 17)
- Setelah itu, sebagai Outer.Inner, kita gunakan instance dari outer untuk membuat instance dari Inner (**outer.new**, bukan hanya new, baris 18)
- Selanjutnya, instance dari Inner bisa digunakan seperti biasa (baris 19)

Inner Class (Method)

Test.java

```
1  class Outer {
2      void methodInOuter(int x) {
3          class Inner {
4              void methodInInner() {
5                  System.out.println("x: " + x);
6              }
7          }
8
9          Inner inner = new Inner();
10         inner.methodInInner();
11     }
12
13 }
14
15
16 public class Test{
17     public static void main(String[] args) {
18         Outer outer = new Outer();
19         outer.methodInOuter(1000);
20     }
21 }
```

```
javac -source 1.7 Test.java
warning: [options] bootstrap class path not set in conjunction with -source 1.7
Test.java:5: error: local variable x is accessed from within inner class; needs
to be declared final
                System.out.println("x: " + x);
                                           ^
1 error
1 warning
```

Catatan:

- Perhatikanlah bahwa Inner kini dideklarasikan di dalam sebuah method (baris 3)
- Class Inner tersebut diinstansiasi, juga di dalam method (baris 9)
- Setelah diinstansiasi, sebuah method milik Inner dipanggil (baris 10)
- Kita membuat sebuah instance dari Outer (baris 18), dan memanggil method methodInOuter() (baris 19), di mana di dalam method tersebut sebuah inner class dideklarasikan, diinstansiasi, dan dipanggil methodnya
- Pada saat memanggil methodInOuter(), kita melewatkan sebuah integer 1000, yang diterima sebagai variabel lokal x
- Perhatikanlah bahwa kompilasi gagal

Inner Class (Method)

- Menggunakan compiler Java versi 8, untuk kode yang sama, kompilasi berhasil

```
javac -source 1.8 Test.java

java Test
x: 1000
```

- Untuk compiler versi sebelumnya, deklarasikanlah int x sebagai final (final int x)

Anonymous Inner Class

- Tidak memiliki nama
- Hanya dapat meng-extend satu superclass atau mengimplementasi satu interface, namun tidak bisa keduanya
- Dibuat dengan keyword new, diikuti oleh nama class atau interface

Anonymous Inner Class

Test.java

```
1  interface Extensible {
2      void load();
3  }
4
5  class Script {
6      void run() {
7          System.out.println("Run");
8      }
9  }
10
11 class Outer {
12     Script method1InOuter() {
13         return new Script() {
14         };
15     }
16
17     Extensible method2InOuter() {
18         return new Extensible() {
19             public void load() {
20                 System.out.println("Load");
21             }
22         };
23     }
24 }
25
26 public class Test{
27     public static void main(String[] args) {
28         Outer outer = new Outer();
29         Script script = outer.method1InOuter();
30         Extensible ex = outer.method2InOuter();
31
32         script.run();
33         ex.load();
34     }
35 }
```

```
javac Test.java

java Test
Run
Load
```

Catatan:

- Kita memiliki sebuah interface Extensible (baris 1), yang di dalamnya dideklarasikan sebuah method load() (baris 2), dan class harus sediakan implementasinya
- Kita juga memiliki sebuah class Script (baris 5), yang di dalamnya sudah terdapat method run() (baris 6-8)
- Perhatikanlah bahwa method method1InOuter() mengembalikan sebuah Script (baris 12), yang mana, pada saat return (baris 13), kita membuat sebuah instance dari anonymous inner class (new Script, tanpa nama, subclass, {}). Perhatikanlah bahwa ; di baris 14 adalah wajib.
- Hal serupa terjadi pada method2InOuter, yang mengembalikan sebuah implementasi dari Extensible. Perhatikanlah kita ‘new’ sebuah interface (baris 18, new instance dari class yang implementasikan, tanpa nama), sekaligus menyediakan implementasi method load(). Token ; pada baris 22 adalah penting.

Anonymous Inner Class

Test.java

```
1  class Script {
2      void run() {
3          System.out.println("Run");
4      }
5  }
6
7  public class Test{
9      public static void test(Script s) {
10         s.run();
11     }
12
13     public static void main(String[] args) {
14         Script s = new Script();
15         test(s);
16
18         test(new Script() {
19             void run() {
20                 System.out.println("Run in subclass");
21             }
22         });
23     }
24 }
```

```
javac Test.java

java Test
Run
Run in subclass
```

Catatan:

- Kita memiliki sebuah class Script (baris 1) yang memiliki sebuah method run()
- Di dalam class Test, kita memiliki sebuah static method test() yang membutuhkan sebuah instance dari Script dilewatkan sebagai argumen (di dalam method tersebut, kita hanya memanggil method run() milik instance dari Script)
- Di baris 13, kita membuat instance dari Script dan melewatkannya ke test() di baris 14
- Yang menarik adalah di baris 16, dimana pada saat memanggil method test(), kita melewatkan instance yang merupakan subclass dari Script (tapi tidak memiliki nama, new Script() {}) yang sekaligus meng-override method run()

Nested Interface

- Kita dapat memiliki nested interface seperti pada contoh berikut (penggunaannya tidak dibahas di dalam manual ini)

Test.java

```
1   interface Extensible {
2       interface Extensible2 {
3       }
4   }
5
6   public class Test{
7       interface Interface1 {
8           interface Interface2 {
9           }
10      }
11      public static void main(String[] args) {
12      }
13  }
```

Pengantar GUI dengan Swing

- Swing dapat digunakan untuk membuat graphical user interface yang dapat berjalan pada platform yang mendukung Java
- Swing merupakan bagian dari pustaka bawaan (package javax.swing)
- Manual ini membahas dasar-dasar bekerja dengan Swing tanpa menggunakan GUI designer
  - IDE NetBeans datang dengan GUI designer yang sangat mempermudah bekerja dengan Swing
  - Sebagian besar kode di dalam bagian ini dikompilasi dengan compiler versi 8. Sesuaikanlah apabila menggunakan compiler versi sebelumnya.

Pengantar GUI dengan Swing

- Semua komponen Swing dan class terkait harus diakses dalam event dispatch thread (karena Swing tidak thread-safe, sebagaimana umumnya GUI toolkit)
- Setelah merancang user interface, kita umumnya merespon berbagai event yang terjadi (misal ketika sebuah tombol di klik)
  - Semua event yang dihasilkan dari user dikerjakan dalam event dispatch thread

  (Dokumentasi API package javax.swing)

Pengantar GUI dengan Swing

- Method main() tidak dikerjakan di dalam event dispatch thread dan kita butuh mentransfer kontrol ke thread tersebut pada saat merancang dan menampilkan GUI
- Kita bisa gunakan invokeLater() dari class javax.swing.SwingUtilities, yang akan menjadwalkan sebuah instance dari class yang mengimplementasikan interface java.lang.Runnable, untuk dikerjakan di event dispatch thread
  - java.lang.Runnable: diimplementasikan oleh class dimana instancenya dimaksudkan untuk dikerjakan oleh suatu thread. Sebuah method run() tanpa argumen harus disediakan.
  - Kita bisa gunakan anonymous inner class

Pengantar GUI dengan Swing

Test.java

```
1  import javax.swing.SwingUtilities;
2
3  public class Test{
4      public void show() {
5      }
6
7      public static void main(String[] args) {
           SwingUtilities.invokeLater(new Runnable() {
               public void run() {
                   new Test().show();
               }
           });
13     }
14 }
```

Di dalam method show(), kita dapat merancang dan menampilkan GUI (dikerjakan di event dispatch thread)

Pengantar GUI dengan Swing

Test.java

```
1  import javax.swing.JFrame;
2  import javax.swing.SwingUtilities;
3
4  public class Test{
5      public void show() {
6          JFrame frame = new JFrame("Hello World");
7          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
8          frame.pack();
9          frame.setVisible(true);
10     }
11
12     public static void main(String[] args) {
           SwingUtilities.invokeLater(new Runnable() {
               public void run() {
15                 new Test().show();
16             }
17         });
18     }
19 }
```

← frame telah diubah ukurannya

Catatan:

- Ketika dijalankan, sebuah frame berukuran kecil akan ditampilkan (dapat diubah ukurannya)
- Klik pada tombol untuk menutup frame tersebut dapat dilakukan, yang pada akhirnya menyebabkan keluar dari aplikasi
- JFrame adalah sebuah top-level window dengan title dan border
- Di baris 7, kita tentukan apa yang terjadi ketika user menutup frame
- Di baris 8, kita memanggil method pack() yang diturunkan dari java.awt.Window (dibahas setelah ini)
- Dan di baris 9, kita tampilkan frame

Pengantar GUI dengan Swing

Test.java

```
2  import javax.swing.JFrame;
3  import javax.swing.JLabel;
4  import javax.swing.SwingUtilities;
5
6  public class Test{
7      public void show() {
8          JFrame frame = new JFrame("Hello World");
9          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
10
11         JLabel label = new JLabel("Hello World");
12         label.setPreferredSize(new Dimension(200, 50));
13
14         frame.getContentPane().add(label);
15         frame.pack();
16
17         frame.setVisible(true);
18     }
19
20     public static void main(String[] args) {
           SwingUtilities.invokeLater(new Runnable() {
               public void run() {
23                 new Test().show();
24             }
25         });
26     }
27 }
```

Catatan:

- Kita membuat sebuah JLabel, yang dapat digunakan untuk menampilkan teks (dalam hal ini: “Hello World”, baris 11)
- Kita set ukuran label yang diinginkan dengan setPreferredSize() milik javax.swing.JComponent, dengan melewatkan sebuah java.awt.Dimension (width 200, height 50)
- Kita tambahkan label tersebut dengan add() milik java.awt.Container (baris 14)
- Method pack() milik java.awt.Window akan menyebabkan window diberikan ukuran sesuai ukuran yang diinginkan (oleh komponen di dalamnya)

Contoh Komponen Swing

- Kita telah melihat contoh penggunaan JLabel. Swing datang dengan sejumlah komponen lain yang dapat digunakan untuk membuat user interface yang fungsional
  - Kita hanya akan membahas beberapa contoh komponen saja
  - IDE NetBeans dapat digunakan untuk drag-and-drop berbagai komponen tanpa mengetik sebaris kode pun
- Perhatikanlah bahwa constructor komponen-komponen Swing umumnya di-overload
  - Di dalam manual ini, kita hanya menggunakan salah satu constructor saja untuk setiap komponen yang dibahas
- Apabila diperlukan, event listener sederhana akan langsung dibahas
  - Contoh: apa yang akan dilakukan ketika sebuah tombol di-klik
  - Berbagai event yang tersedia dan cara menanganinya dapat dipilih dan ditentukan dengan mudah dengan IDE NetBeans, tanpa mengetik sebaris kode pun (kecuali untuk isi dari event listenernya)

JButton

Test.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3  import javax.swing.JButton;
4  import javax.swing.JFrame;
5  import javax.swing.SwingUtilities;
6
7  public class Test{
8      public void show() {
9          JFrame frame = new JFrame("Hello World");
10         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11
12         JButton button = new JButton("Hello");
           button.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent e) {
15                 JButton source = (JButton) e.getSource();
16                 source.setText("World");
17             }
18         });
19
20         frame.getContentPane().add(button);
21         frame.pack();
22         frame.setVisible(true);
23     }
24
25     public static void main(String[] args) {
           SwingUtilities.invokeLater(new Runnable() {
               public void run() {
28                 new Test().show();
29             }
30         });
31     }
32 }
```

Catatan:

- JButton dapat digunakan sebagai tombol yang dapat di-klik. Kita membuat sebuah tombol dengan label “Hello” (baris 12)
- Sebagai contoh sederhana, kita menambahkan sebuah java.awt.event.ActionListener, dengan addActionListener() milik javax.swing.AbstractButton
- Interface java.awt.event.ActionListener mendefinisikan sebuah method actionPerformed() yang menerima sebuah argumen class java.awt.event.ActionEvent
- Kita mendapatkan source event dengan method getSource() milik class java.util.EventObject, dimana event awalnya terjadi
- Perhatikanlah bahwa getSource() mengembalikan java.lang.Object, yang kita cast ke JButton (baris 15)
- Label pada JButton diset dengan setText()

JCheckBox

Test.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3  import javax.swing.JCheckBox;
4  import javax.swing.JFrame;
5  import javax.swing.SwingUtilities;
6
7  public class Test{
8      public void show() {
9          JFrame frame = new JFrame("Hello World");
10         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11
12         JCheckBox check = new JCheckBox("Hello");
13         check.addActionListener(new ActionListener() {
14             public void actionPerformed(ActionEvent e) {
15                 JCheckBox source = (JCheckBox) e.getSource();
16                 System.out.println(source.isSelected());
17             }
18         });
19
20         frame.getContentPane().add(check);
21         frame.pack();
22         frame.setVisible(true);
23     }
24
25     public static void main(String[] args) {
26         SwingUtilities.invokeLater(new Runnable() {
27             public void run() {
28                 new Test().show();
29             }
30         });
31     }
32 }
```

```
java Test
true
false
true
```

Catatan:

- JCheckBox dapat dipilih (check) atau tidak, dan state ini dapat diketahui dengan method isSelected() milik javax.swing.AbstractButton
- Perhatikanlah bahwa contoh penggunaan JButton juga dapat diterapkan di sini

JRadioButton dan ButtonGroup

Test.java

```
1  import javax.swing.ButtonGroup;
2  import javax.swing.JFrame;
3  import javax.swing.JPanel;
4  import javax.swing.JRadioButton;
5  import javax.swing.SwingUtilities;
6
7  public class Test{
8      public void show() {
9          JFrame frame = new JFrame("Hello World");
10         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11
12         JRadioButton radioHello = new JRadioButton("Hello");
13         JRadioButton radioWorld = new JRadioButton("World");
14
15         ButtonGroup group = new ButtonGroup();
16         group.add(radioHello);
17         group.add(radioWorld);
18
19         JPanel panel = new JPanel();
20         panel.add(radioHello);
21         panel.add(radioWorld);
22
23         frame.getContentPane().add(panel);
24         frame.pack();
25         frame.setVisible(true);
26     }
27
28     public static void main(String[] args) {
           SwingUtilities.invokeLater(new Runnable() {
               public void run() {
31                 new Test().show();
32             }
33         });
34     }
35 }
```

Catatan:

- Kita membuat dua JRadioButton, memberikan masing-masing sebuah label (baris 12 dan 13)
- Kemudian, kita buat sebuah javax.swing.ButtonGroup (baris 15) dan menambahkan kedua radio button tersebut (dengan method add()), agar hanya salah satu radio button yang dapat dipilih
- Di baris 19, kita membuat sebuah JPanel (sebuah container), dan menambahkan kedua radio button tersebut ke dalamnya (baris 20-21)
- Barulah, di baris 23, kita menambahkan panel tersebut ke frame

JTextField

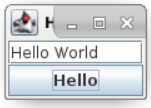
Test.java

```
1  import java.awt.BorderLayout;
2  import java.awt.event.ActionEvent;
3  import java.awt.event.ActionListener;
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6  import javax.swing.JTextField;
7  import javax.swing.SwingUtilities;
8
9  public class Test{
10     public void show() {
11         JFrame frame = new JFrame("Hello World");
12         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13
14         JTextField text = new JTextField();
15
16         JButton button = new JButton("Hello");
           button.addActionListener(new ActionListener() {
19             public void actionPerformed(ActionEvent e) {
20                 System.out.println(text.getText());
21             }
           });
22
23         frame.getContentPane().add(text, BorderLayout.CENTER);
24         frame.getContentPane().add(button, BorderLayout.SOUTH);
25         frame.pack();
26         frame.setVisible(true);
27     }
28
29     public static void main(String[] args) {
           SwingUtilities.invokeLater(new Runnable() {
32             public void run() {
33                 new Test().show();
34             }
           });
35     }
36 }
```

Catatan:

- Perhatikanlah bahwa kini kita menambahkan dua komponen: JTextField dan JButton (baris 14 dan 16)
- Sebuah action listener untuk JButton disediakan, yang mana akan mendapatkan apa yang diisikan di JTextField (dengan getText() milik javax.swing.text.JtextComponent) dan menampilkannya ke standard output (baris 19)
- Dalam contoh ini, kita menggunakan method add() untuk menambahkan komponen, yang argumen keduanya adalah sebuah konstanta milik java.awt.BorderLayout (dibahas di bagian Layout Manager)

```
java Test
Hello World
Hello World
Hello World
```



JPasswordField

Test.java

```
1  import java.awt.BorderLayout;
2  import java.awt.event.ActionEvent;
3  import java.awt.event.ActionListener;
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6  import javax.swing.JPasswordField;
7  import javax.swing.SwingUtilities;
8
9  public class Test {
10     public void show() {
11         JFrame frame = new JFrame("Hello World");
12         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13
14         JPasswordField password = new JPasswordField();
15         JButton button = new JButton("Hello");
           button.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent e) {
                   System.out.println(password.getPassword());
               }
19         });
20
21         frame.getContentPane().add(password, BorderLayout.CENTER);
22         frame.getContentPane().add(button, BorderLayout.SOUTH);
23         frame.pack();
24         frame.setVisible(true);
25     }
26
27     public static void main(String[] args) {
28         SwingUtilities.invokeLater(new Runnable() {
               public void run() {
31                 new Test().show();
32             }
33         });
34     }
35 }
```

Catatan:

- Kita menambahkan sebuah JPasswordField (baris 14)
- Untuk mendapatkan password yang diisikan, kita bisa menggunakan method getPassword() (mengembalikan char[])
- Apabila diperlukan, bacalah juga pembahasan tentang JTextField sebelumnya

```
java Test
Hello
Hello
```

JTextArea

Test.java

```
1  import java.awt.BorderLayout;
2  import java.awt.event.ActionEvent;
3  import java.awt.event.ActionListener;
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6  import javax.swing.JTextArea;
7  import javax.swing.SwingUtilities;
8
9  public class Test {
10     public void show() {
11         JFrame frame = new JFrame("Hello World");
12         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13
14         JTextArea text = new JTextArea(25, 80);
15         JButton button = new JButton("Hello");
16         button.addActionListener(new ActionListener() {
17             public void actionPerformed(ActionEvent e) {
18                 System.out.println(text.getText());
19             }
20         });
21
22         frame.getContentPane().add(text, BorderLayout.CENTER);
23         frame.getContentPane().add(button, BorderLayout.SOUTH);
24         frame.pack();
25         frame.setVisible(true);
26     }
27
28     public static void main(String[] args) {
29         SwingUtilities.invokeLater(new Runnable() {
30             public void run() {
31                 new Test().show();
32             }
33         });
34     }
35 }
```

Catatan:

- Kita menggunakan JTextArea untuk bekerja dengan teks yang dapat lebih dari satu baris
- Di baris 14, kita buat sebuah JTextArea dengan 25 baris dan 80 kolom
- Untuk bekerja dengan isi teks, kita bisa menggunakan method getText() ataupun setText() milik javax.swing.text.JTextComponent

```
java Test
hello
world
:)
```

JScrollPane

Test.java

```
1  import java.awt.BorderLayout;
2  import java.awt.event.ActionEvent;
3  import java.awt.event.ActionListener;
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6  import javax.swing.JScrollPane;
7  import javax.swing.JTextArea;
8  import javax.swing.SwingUtilities;
9
10 public class Test {
11     public void show() {
12         JFrame frame = new JFrame("Hello World");
13         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14
15         JTextArea text = new JTextArea(4, 10);
16         JScrollPane scroll = new JScrollPane(text);
17
18         JButton button = new JButton("Hello");
           button.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent e) {
21                 System.out.println(text.getText());
22             }
23         });
24
25         frame.getContentPane().add(scroll, BorderLayout.CENTER);
26         frame.getContentPane().add(button, BorderLayout.SOUTH);
27         frame.pack();
28         frame.setVisible(true);
29     }
30
31     public static void main(String[] args) {
           SwingUtilities.invokeLater(new Runnable() {
               public void run() {
34                 new Test().show();
35             }
36         });
37     }
38 }
```

Catatan:

- Pada contoh JTextArea sebelumnya, kita tidak dapat bekerja dengan scrollbar ketika teks yang ditampung lebih dari area yang ditampilkan
- Di contoh ini, kita gunakan sebuah JScrollPane (baris 16), yang ketika dibuat, menerima sebuah java.awt.Component
- Perhatikanlah bahwa kita add JScrollPane tersebut (baris 25)

```
java Test
Hello World 1
Hello World 2
Hello World 3
Hello World 4
Hello World 5
```

JEditorPane

Test.java

```
1  import java.awt.BorderLayout;
2  import javax.swing.JEditorPane;
3  import javax.swing.JFrame;
4  import javax.swing.JScrollPane;
5  import javax.swing.SwingUtilities;
6
7  public class Test {
8      public void show() {
9          JFrame frame = new JFrame("Hello World");
10         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11
12         JEditorPane editor = new JEditorPane();
13         editor.setContentType("text/html");
14         editor.setEditable(false);
15         editor.setText("<b>Hello</b> <u>World</u>");
16         JScrollPane scroll = new JScrollPane(editor);
17
18         frame.getContentPane().add(scroll, BorderLayout.CENTER);
19         frame.pack();
20         frame.setVisible(true);
21     }
22
23     public static void main(String[] args) {
           SwingUtilities.invokeLater(new Runnable() {
26             public void run() {
27                 new Test().show();
28             }
29         });
30     }
```

Catatan:

- Sebuah JEditorPane dapat digunakan untuk menampilkan atau mengedit sejumlah tipe konten
- Dalam contoh ini, kita hanya menampilkan konten berupa text/html (mendukung HTML 3.2). Teks tidak dapat diedit karena setEditable(false) milik javax.swing.text.JTextComponent.

```
java Test
```

JSpinner

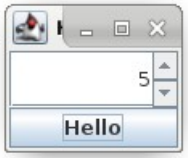
Test.java

```
1  import java.awt.BorderLayout;
2  import java.awt.event.ActionEvent;
3  import java.awt.event.ActionListener;
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6  import javax.swing.JSpinner;
7  import javax.swing.SpinnerNumberModel;
8  import javax.swing.SwingUtilities;
9
10 public class Test {
11     public void show() {
12         JFrame frame = new JFrame("Hello World");
13         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14
15         JSpinner spinNumber = new JSpinner(new SpinnerNumberModel());
16
17         JButton button = new JButton("Hello");
           button.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent e) {
20                 System.out.println(spinNumber.getModel().getValue());
21             }
22         });
23
24         frame.getContentPane().add(spinNumber, BorderLayout.CENTER);
25         frame.getContentPane().add(button, BorderLayout.SOUTH);
26         frame.pack();
27         frame.setVisible(true);
28     }
29
30     public static void main(String[] args) {
           SwingUtilities.invokeLater(new Runnable() {
               public void run() {
33                 new Test().show();
34             }
35         });
36     }
37 }
```

Catatan:

- Sebuah JSpinner bisa bekerja dengan interface javax.swing.SpinnerModel
- Beberapa class model disertakan: javax.swing.SpinnerNumberModel, javax.swing.SpinnerListModel, javax.swing.SpinnerDateModel
- Untuk mendapatkan model, kita bisa gunakan method getModel()

```
java Test
1
2
3
```



JProgressBar

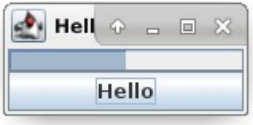
Test.java

```
3  import java.awt.event.ActionListener;
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6  import javax.swing.JProgressBar;
7  import javax.swing.SwingUtilities;
8
9  public class Test {
10     public void show() {
11         JFrame frame = new JFrame("Hello World");
12         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13
14         JProgressBar progress = new JProgressBar();
15         progress.setValue(50);
16
17         JButton button = new JButton("Hello");
           button.addActionListener(new ActionListener() {
20             public void actionPerformed(ActionEvent e) {
21                 System.out.println(progress.getValue());
22             }
23         });
24
25         frame.getContentPane().add(progress, BorderLayout.CENTER);
26         frame.getContentPane().add(button, BorderLayout.SOUTH);
27         frame.pack();
28         frame.setVisible(true);
29     }
30
31     public static void main(String[] args) {
32         SwingUtilities.invokeLater(new Runnable() {
33             public void run() {
34                 new Test().show();
35             }
36         });
37     }
}
```

Catatan:

- Kita membuat sebuah JProgressBar dengan constructor tanpa argumen, yang mana nilai awal dan minimum adalah 0, dan nilai maksimum adalah 100 (baris 14)
- Di baris 15, kita set progress nya menjadi 50 dengan setValue()
- Untuk mendapatkan nilai progress, kita gunakan getValue() (baris 20)
- JProgressBar akan dibahas lagi lebih lanjut ketika kita membahas tentang javax.swing.SwingWorker

```
java Test
50
50
```



JTabbedPane

Test.java

```
1  import java.awt.BorderLayout;
2  import javax.swing.JButton;
3  import javax.swing.JFrame;
4  import javax.swing.JLabel;
5  import javax.swing.JPanel;
6  import javax.swing.JTabbedPane;
7  import javax.swing.SwingUtilities;
8
9  public class Test {
10     public void show() {
11         JFrame frame = new JFrame("Hello World");
12         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13         JButton button1 = new JButton("Button 1");
14         JLabel label1 = new JLabel("Label 1");
15         JPanel panel1 = new JPanel();
16         panel1.add(label1);
17         panel1.add(button1);
18         JButton button2 = new JButton("Button 2");
19         JPanel panel2 = new JPanel();
20         panel2.add(button2);
21         JButton button3 = new JButton("Button 3");
22         JPanel panel3 = new JPanel();
23         panel3.add(button3);
24         JTabbedPane tab = new JTabbedPane();
25         tab.addTab("Tab 1", panel1);
26         tab.addTab("Tab 2", panel2);
27         tab.addTab("Tab 3", panel3);
28         frame.getContentPane().add(tab, BorderLayout.CENTER);
29         frame.pack();
30         frame.setVisible(true);
31     }
32     public static void main(String[] args) {
33         SwingUtilities.invokeLater(new Runnable() {
34             public void run() {
35                 new Test().show();
36             }
37         });
38     }
39 }
```

Catatan:

- Pertama, kita membuat satu JButton dan satu JLabel (baris 13-14), dan menempatkan mereka di dalam sebuah JPanel (baris 15-17)
- Kita ulangi langkah serupa, dimana kita membuat JButton baru dan menempatkannya di JPanel (baris 18-20 dan 21-23)
- Di baris 24, kita membuat sebuah JTabbedPane. Dengan method addTab() yang menerima String title dan java.awt.Component, kita tambahkan setiap panel yang kita buat sebelumnya (baris 25-27).
- JTabbedPane tersebut kemudian kita tambahkan ke frame

JComboBox

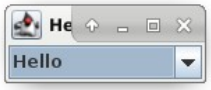
Test.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3  import javax.swing.JComboBox;
4  import javax.swing.JFrame;
5  import javax.swing.SwingUtilities;
6
7  public class Test {
8      public void show() {
9          JFrame frame = new JFrame("Hello World");
10         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11
12         JComboBox<String> combo = new JComboBox<>();
           combo.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent e) {
15                 int selected = combo.getSelectedIndex();
16                 if (selected > -1) {
17                     System.out.println(combo.getItemAt(selected));
18                 }
19             }
20         });
21
22         combo.addItem("Hello");
23         combo.addItem("World");
24
25         frame.getContentPane().add(combo);
26         frame.pack();
27         frame.setVisible(true);
28     }
29
30     public static void main(String[] args) {
           SwingUtilities.invokeLater(new Runnable() {
               public void run() {
33                 new Test().show();
34             }
35         });
36     }
37 }
```

Catatan:

- Perhatikanlah bahwa JComboBox adalah generic. Di baris 12, kita membuat sebuah JComboBox yang berisi String.
- Di baris 13, kita menambahkan sebuah action listener. Di dalamnya, kita mendapatkan indeks terpilih dengan getSelectedIndex() (-1 apabila tidak ada yang dipilih). Untuk mendapatkan item pada posisi tertentu, kita bisa gunakan getItemAt() (dapat mengembalikan null).

```
java Test
Hello
World
```



JComboBox

Test.java

```
1  import javax.swing.JComboBox;
2  import javax.swing.JFrame;
3  import javax.swing.SwingUtilities;
4
5  class Script {
6      private final String name;
7      private final String format;
8      Script(String name, String format) {
9          this.name = name;
10         this.format = format;
11     }
12     public String toString () { return name + " - " + format; }
13 }
14
15 public class Test {
16     public void show() {
17         JFrame frame = new JFrame("Hello World");
18         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19
20         JComboBox<Script> combo = new JComboBox<>();
21         combo.addItem(new Script("A", "jar"));
22         combo.addItem(new Script("B", "class"));
23
24         frame.getContentPane().add(combo);
25         frame.pack();
26         frame.setVisible(true);
27     }
28
29     public static void main(String[] args) {
30         SwingUtilities.invokeLater(new Runnable() {
31             public void run() {
32                 new Test().show();
33             }
34         });
35     }
36 }
```

Catatan:

- Kita membuat sebuah class Script (baris 5), yang instance nya akan ditambahkan ke dalam combo box JComboBox<Script> (baris 21-22)
- Di dalam class Script, kita override method toString() untuk menampilkan field yang dirasa perlu untuk ditampilkan dalam combo box (baris 12)

JOptionPane

- javax.swing.JOptionPane menyediakan berbagai dialog siap pakai yang dapat digunakan diantaranya untuk:
  - Menampilkan pesan (berbagai variasi)
  - Meminta input, konfirmasi, pilihan, ataupun lainnya (berbagai variasi)
- Manual ini hanya membahas contoh sederhana penggunaan beberapa method static di dalam class ini. Bacalah juga dokumentasi API untuk informasi selengkapnya.

JOptionPane

Test.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3  import javax.swing.JButton;
4  import javax.swing.JFrame;
5  import javax.swing.JOptionPane;
6  import javax.swing.JTextArea;
7  import javax.swing.SwingUtilities;
8
9  public class Test {
10     public void show() {
11         JFrame frame = new JFrame("Hello World");
12         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13
14         JButton button = new JButton("Hello");
           button.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent e) {
17                 JOptionPane.showMessageDialog(frame, button.getText());
18                 JOptionPane.showMessageDialog(frame, button);
19                 JTextArea text = new JTextArea("Hello World");
20                 text.setEditable(false);
21                 JOptionPane.showMessageDialog(frame, text, "title",
22                         JOptionPane.ERROR_MESSAGE);
23
24             }
25         });
26         frame.getContentPane().add(button);
27         frame.pack();
28         frame.setVisible(true);
29     }
30
31     public static void main(String[] args) {
           SwingUtilities.invokeLater(new Runnable() {
               public void run() {
34                 new Test().show();
35             }
36         });
37     }
38 }
```

JButton

JTextArea

Perhatikanlah bahwa showMessageDialog() dapat menerima message berupa Object

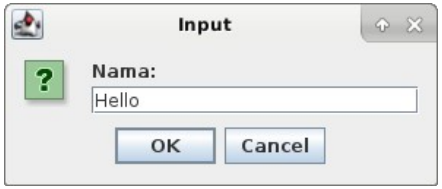
JOptionPane

```
public void actionPerformed(ActionEvent e) {
    String name = JOptionPane.showInputDialog(frame, "Nama:", "");

    int confirm = JOptionPane.showConfirmDialog(
            frame, "Ganti Password?");

    if (confirm == JOptionPane.YES_OPTION) {
        JPasswordField password = new JPasswordField();
        Object[] buttons = {"Button 1", "Button 2"};
        int confirmPassword = JOptionPane.showOptionDialog(
                frame, password,
                "Password",
                JOptionPane.YES_NO_OPTION,
                JOptionPane.PLAIN_MESSAGE,
                new ImageIcon("test.png"),
                buttons,
                buttons[1]);

        if (confirmPassword == JOptionPane.YES_OPTION) {
            System.out.println(password.getPassword());
        }
    }
}
```



Kita dapat menggunakan showInputDialog() untuk meminta input. Untuk konfirmasi, kita bisa gunakan showConfirmDialog(). Perhatikanlah bahwa kita bisa mengganti icon, pilihan tombol yang ada, serta tombol default.

JFileChooser

Test.java

```
1  import java.awt.BorderLayout;
2  import java.awt.event.ActionEvent;
3  import java.awt.event.ActionListener;
4  import java.io.File;
5  import javax.swing.JButton;
6  import javax.swing.JFileChooser;
7  import javax.swing.JFrame;
8  import javax.swing.SwingUtilities;
9
10 public class Test {
11     public void show() {
12         JFrame frame = new JFrame("Hello World");
13         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14
15         JButton button = new JButton("Hello");
16         button.addActionListener(new ActionListener() {
17             public void actionPerformed(ActionEvent e) {
18                 JFileChooser chooser = new JFileChooser();
19                 int ret = chooser.showOpenDialog(frame);
20                 if (ret == JFileChooser.APPROVE_OPTION) {
21                     File file = chooser.getSelectedFile();
22                     System.out.println(file.getName());
23                 }
24             }
25         });
26
27         frame.getContentPane().add(button, BorderLayout.SOUTH);
28         frame.pack();
29         frame.setVisible(true);
30     }
31     public static void main(String[] args) {
32         SwingUtilities.invokeLater(new Runnable() {
33             public void run() {
34                 new Test().show();
35             }
36         });
37     }
38 }
```

Catatan:

- Pertama, kita membuat instance dari JFileChooser (baris 18)
- Untuk menampilkan dialog untuk membuka file, kita bisa menggunakan method showOpenDialog()
- Apabila user mengkonfirmasi memilih file, maka showOpenDialog() akan mengembalikan APPROVE_OPTION (sebuah int)
- File terpilih kemudian didapatkan dengan getSelectedFile(), yang mengembalikan java.io.File
- Method showSaveDialog() dapat digunakan untuk menampilkan dialog untuk menyimpan file

```
test.png
Test.class
```

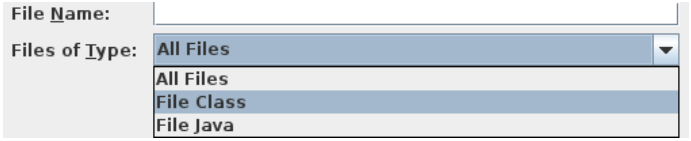
JFileChooser

```
FileFilter filter1 = new FileNameExtensionFilter("File Class", "class");
FileFilter filter2 = new FileNameExtensionFilter("File Java", "java");

JFileChooser chooser = new JFileChooser();
chooser.addChoosableFileFilter(filter1);
chooser.addChoosableFileFilter(filter2);

chooser.setDialogTitle("Buka File");

int ret = chooser.showOpenDialog(frame);
if (ret == JFileChooser.APPROVE_OPTION) {
    File file = chooser.getSelectedFile();
    System.out.println(file.getName());
}
```



Catatan:

- Class javax.swing.filechooser.FileNameExtensionFilter dapat digunakan sebagai filter berupa ekstensi nama file (tidak case-sensitive)
- Method addChoosableFileFilter() bisa digunakan untuk menambahkan filter yang dapat dipilih oleh user
- Judul dialog dapat diubah dengan setDialogTitle()
- Bacalah lebih lanjut dokumentasi class JFileChooser untuk pengaturan lainnya

WindowListener dan WindowAdapter

- Menerima event window (aktivasi, sudah ditutup, sedang ditutup, tidak lagi aktif, dari minimized ke normal, dari normal ke minimized, pertama kali dibuka)
  - Implement interface java.awt.event.WindowListener
  - Atau extend java.awt.event.WindowAdapter dan override method yang dibutuhkan saja
- Manual ini hanya membahas contoh konfirmasi ketika sebuah JFrame sedang ditutup

WindowListener dan WindowAdapter

Test.java

```
1  import java.awt.Dimension;
2  import java.awt.event.WindowAdapter;
3  import java.awt.event.WindowEvent;
4  import javax.swing.JFrame;
5  import javax.swing.JOptionPane;
6  import javax.swing.SwingUtilities;
7
8  public class Test {
9      public void show() {
10         JFrame frame = new JFrame("Hello World");
11         frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
12         frame.addWindowListener(new WindowAdapter() {
13             public void windowClosing(WindowEvent e) {
14                 int confirm = JOptionPane.showConfirmDialog(frame, "Tutup ?");
15                 if (confirm == JOptionPane.YES_OPTION) {
16                     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17                 }
18             }
19         });
20
21         frame.setPreferredSize(new Dimension(400, 200));
22         frame.pack();
23         frame.setVisible(true);
24     }
25     public static void main(String[] args) {
26         SwingUtilities.invokeLater(new Runnable() {
27             public void run() {
28                 new Test().show();
29             }
30         });
31     }
32 }
```

Catatan:

- Kita menggunakan setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE) agar frame tidak melakukan apa-apa ketika ditutup (default: HIDE_ON_CLOSE)
- Kita menambahkan window listener berupa sebuah subclass dari WindowAdapter, meng-override windowClosing(), dan menampilkan konfirmasi
- Apabila konfirmasi ya, maka setDefaultCloseOperation() di-set ke Jframe.EXIT_ON_CLOSE (atau: System.exit())

Layout Manager

- Layout Manager menentukan bagaimana posisi dan ukuran komponen-komponen di dalam sebuah container
  - Dengan demikian, kita tidak perlu menentukan posisi absolut sebuah komponen, dan mengatur posisinya ketika container diubah ukurannya, ataupun ketika properti sistem (seperti font) berubah
  - Apabila memang sangat diperlukan, posisi absolut komponen tetap bisa ditentukan dengan tidak menggunakan layout manager

Layout Manager

- Tersedia sejumlah layout manager di Swing (implementasi interface java.awt.LayoutManager) seperti:
  - java.awt.BorderLayout
  - javax.swing.BoxLayout
  - java.awt.FlowLayout
  - java.awt.GridBagLayout
  - java.awt.GridLayout
  - javax.swing.ScrollPaneLayout
  - javax.swing.SpringLayout

    (dan lainnya)

Layout Manager

- Container dapat memiliki default layout manager
  - javax.swing.JPanel: java.awt.FlowLayout
  - Content pane (misal dari JFrame): java.awt.BorderLayout
- Kita dapat mengubah layout manager default tersebut

Layout Manager

- Kita dapat mengetikkan sendiri kode untuk menempatkan komponen ke layout manager (seperti dengan method add() di beberapa contoh sebelumnya)
- Atau, disarankan menggunakan GUI Builder seperti yang disediakan oleh IDE NetBeans ketika bekerja dengan layout komponen yang kompleks
- Manual ini hanya membahas sekilas contoh penggunaan java.awt.BorderLayout
  - Pada penggunaan JPanel, kita sudah menggunakan java.awt.FlowLayout ketika menambahkan komponen di beberapa contoh sebelumnya

Layout Manager

- java.awt.BorderLayout:
  - Komponen ditempatkan dalam lima region: NORTH, SOUTH, EAST, WEST, CENTER
    - Posisi relatif disediakan, tapi tidak dicontohkan dalam manual ini
  - Setiap region hanya dapat menampung paling banyak satu komponen (walaupun komponen ini dapat berupa container)
  - Method add() menerima argumen kedua berupa konstanta region atau CENTER apabila tidak ditentukan secara eksplisit

Layout Manager

Test.java

```
1  import java.awt.BorderLayout;
2  import javax.swing.JButton;
3  import javax.swing.JFrame;
4  import javax.swing.JLabel;
5  import javax.swing.JPanel;
6  import javax.swing.JTextArea;
7  import javax.swing.SwingUtilities;
8
9  public class Test {
10     public void show() {
11         JFrame frame = new JFrame("Hello World");
12         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13
14         JLabel label1 = new JLabel("Label 1");
15
16         JTextArea text = new JTextArea(10, 20);
17         JButton button1 = new JButton("Button 1");
18         JPanel panel = new JPanel();
19         panel.add(text);
20         panel.add(button1);
21
22         JButton button2 = new JButton("Button 2");
23
24         frame.getContentPane().add(label1, BorderLayout.NORTH);
25         frame.getContentPane().add(panel, BorderLayout.CENTER);
26         frame.getContentPane().add(button2, BorderLayout.SOUTH);
27
28         frame.pack();
29         frame.setVisible(true);
30     }
31     public static void main(String[] args) {
32         SwingUtilities.invokeLater(new Runnable() {
33             public void run() {
34                 new Test().show();
35             }
36         });
37     }
38 }
```

Di dalam contoh ini, label1 (JLabel) ditempatkan pada region BorderLayout.NORTH, panel (JPanel, berisi text dan button1) ditempatkan pada region BorderLayout.CENTER, dan button2 (JButton) ditempatkan pada region BorderLayout.SOUTH.

SwingWorker

- Bekerja dengan Swing:
  - Tugas-tugas yang membutuhkan waktu lama untuk diselesaikan, tidak boleh dijalankan di event dispatch thread
    - Apabila dilakukan, user interface dapat menjadi tidak responsif
  - Komponen Swing hanya boleh diakses di event dispatch thread

(Dokumentasi class javax.swing.SwingWorker)

SwingWorker

- Mari simulasikan tugas yang membutuhkan waktu lama dan sekaligus mengakses komponen Swing
  - Perulangan dengan penundaan waktu → simulasi tugas
  - Progress dalam perulangan ditampilkan pada progress bar → akses pada komponen Swing
- Tugas tersebut dikerjakan dalam event dispatch thread, ketika sebuah tombol di klik

SwingWorker

Test.java

```
1  import java.awt.BorderLayout;
2  import java.awt.event.ActionEvent;
3  import java.awt.event.ActionListener;
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6  import javax.swing.JProgressBar;
7  import javax.swing.SwingUtilities;
8
9  public class Test {
10     public void show() {
11         JFrame frame = new JFrame("Hello World");
12         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13         JProgressBar progress = new JProgressBar();
14         JButton button = new JButton("Hello");
15         button.addActionListener(new ActionListener() {
17             public void actionPerformed(ActionEvent e) {
18                 for (int i=1; i<=5; i++) {
19                     try {
20                         Thread.sleep(1000);
21                         progress.setValue(i * 20);
22                     } catch (InterruptedException ex) {
23                     }
24                 }
25             }
26         });
27         frame.getContentPane().add(progress, BorderLayout.CENTER);
28         frame.getContentPane().add(button, BorderLayout.SOUTH);
29         frame.pack();
30         frame.setVisible(true);
31     }
32     public static void main(String[] args) {
33         SwingUtilities.invokeLater(new Runnable() {
34             public void run() {
35                 new Test().show();
36             }
37         });
38     }
}
```

Catatan:

- Ketika tombol di klik, apa yang diinginkan adalah: untuk kurang lebih setiap detik (1000 mili detik), progress bar akan diupdate (20, 40, 60, 80, 100)
- Yang terjadi adalah: setelah tombol di klik, kurang lebih selama 5 detik, user interface menjadi tidak responsif (klik pada tombol tidak direspon, resize window tidak direspon, dan lainnya), dan pada akhirnya, progress bar mencapai nilai 100 (langsung dari 0, tidak bertahap)
- Ingatlah bahwa kita mencoba untuk melakukan tugas yang berat di dalam event dispatch thread, dan apa yang terjadi adalah normal

SwingWorker

- javax.swing.SwingWorker
  - Sebuah class abstract (tidak dapat diinstansiasi, kita perlu bekerja dengan subclass)
  - Dapat digunakan sebagai alat bantu untuk mengerjakan tugas yang membutuhkan waktu lama, di thread lain
  - Didesain untuk dijalankan hanya sekali
- Cara kerja:
  - Pada thread aktif (umumnya event dispatch thread): method execute() milik instance SwingWorker (subclass) dipanggil, dan sebuah worker thread akan dijadwalkan
  - Pada thread worker: method doInBackground() akan dikerjakan (melakukan tugas-tugas yang membutuhkan waktu). Di dalam method ini, kita bisa publikasikan hasil sementara dengan method publish().
  - Pada event dispatch thread: method process() dan done() akan dikerjakan

SwingWorker

Test.java

```
1  import java.awt.BorderLayout;
2  import java.awt.event.ActionEvent;
3  import java.awt.event.ActionListener;
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6  import javax.swing.SwingUtilities;
7  import javax.swing.SwingWorker;
8
9  public class Test {
10     public void show() {
11         JFrame frame = new JFrame("Hello World");
12         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13         JButton button = new JButton("Hello");
14         button.addActionListener(new ActionListener() {
15             public void actionPerformed(ActionEvent e) {
16                 SwingWorker worker = new SwingWorker<Void, Void>() {
17                     protected Void doInBackground() throws Exception {
18                         for (int i=1; i<=5; i++) {
19                             Thread.sleep(1000);
20                             System.out.println(i*20);
21                         }
22                         return null;
23                     }
24                 };
25                 worker.execute();
26             }
27         });
28         frame.getContentPane().add(button, BorderLayout.SOUTH);
29         frame.pack();
30         frame.setVisible(true);
31     }
32     public static void main(String[] args) {
33         SwingUtilities.invokeLater(new Runnable() {
34             public void run() {
35                 new Test().show();
36             }
37         });
38     }
39 }
```

Catatan:

- Dalam contoh ini, tidak ada komponen user interface yang diupdate, tapi ketika tombol di klik, simulasi tugas dilakukan (dan user interface tetap responsif)
- Void pertama pada SwingWorker artinya kita tidak membutuhkan nilai kembalian doInBackground(). Sementara Void kedua artinya kita tidak membutuhkan hasil sementara.
- Perhatikanlah bahwa kita menggunakan Void (bukan void) dan return null (baris 16, 17, 22)
- Jangan lupa untuk menjadwalkan dengan method execute (baris 25)

```
20
40
60
80
100
```

SwingWorker

```
public void show() {
    JFrame frame = new JFrame("Hello World");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JProgressBar progress = new JProgressBar();

    JButton button = new JButton("Hello");
    button.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            SwingWorker worker = new SwingWorker<Void, Integer>() {
                protected Void doInBackground() throws Exception {
                    for (int i=1; i<=5; i++) {
                        Thread.sleep(1000);
                        publish(i * 20);
                    }
                    return null;
                }

                protected void process(List<Integer> chunks) {
                    int last = chunks.get(chunks.size()-1);
                    progress.setValue(last);
                }
            };
            worker.execute();
        }
    });

    frame.getContentPane().add(progress, BorderLayout.CENTER);
    frame.getContentPane().add(button, BorderLayout.SOUTH);

    frame.pack();
    frame.setVisible(true);
}
```

Catatan:

- Hanya method show() yang ditampilkan
- Perhatikanlah bahwa dalam contoh ini, kita tidak membutuhkan hasil dari doInBackground() (Void, return null). Akan tetapi, kita membutuhkan hasil sementara (nilai untuk progress bar, Integer).
- Publikasi hasil sementara dilakukan dengan method publish() di dalam doInBackground(). Beberapa nilai bisa dipublikasikan sebelum process() dipanggil (oleh karenanya, dikerjakan sekaligus).
- Apa yang dipublikasikan kemudian dikerjakan pada method process() (di event dispatch thread, bisa akses komponen Swing). Perhatikanlah kita ambil hasil yang dipublikasikan terakhir dari List).

SwingWorker

```
public void actionPerformed(ActionEvent e) {
    SwingWorker worker = new SwingWorker<Integer, Integer>() {
        protected Integer doInBackground() throws Exception {
            Integer ret = 0;
            for (int i=1; i<=5; i++) {
                Thread.sleep(1000);
                publish(i * 20);
                ret += i;
            }
            return ret;
        }

        protected void process(List<Integer> chunks) {
            int last = chunks.get(chunks.size()-1);
            progress.setValue(last);
        }

        protected void done() {
            Integer ret = 0;

            try {
                ret = get();
            } catch (InterruptedException | ExecutionException e) {
            }
            button.setText(ret.toString());
        }
    };
    worker.execute();
}
```

Catatan:

- Perhatikanlah bahwa doInBackground() kini mengembalikan sebuah Integer (hasil penjumlahan). Berbeda dengan contoh sebelumnya, kita new SwingWorker<Integer, Integer>(){}.
- Nilai yang dikembalikan tersebut bisa didapatkan dengan method get(), dimana exception InterruptedException dan ExecutionException dapat terjadi
- Kita memanggil get() di dalam method done(). Method done() dikerjakan setelah doInBackground() selesai dikerjakan. Yang menarik adalah done() dikerjakan di event dispatch thread sehingga kita boleh mengakses komponen Swing (mengubah label button sesuai representasi String kembalian get()).

Contoh Soal

- Kita telah membahas pengenalan GUI dengan Swing. Cobalah untuk membuat salah satu program GUI berikut, yang:
  - Dapat memilih sebuah file teks (*.txt) menggunakan dialog, kemudian membaca, dan menampilkan isinya
  - Dapat melakukan kalkulasi yang membutuhkan waktu relatif lama, dimana perkembangan selama kalkulasi diinformasikan secara visual (misal lewat sebuah progress bar). Pada akhirnya, hasil kalkulasi ditampilkan. Tentu saja, user interface harus tetap responsif.

Bagian 5

Bekerja dengan sistem database relasional

JDBC

- Java Database Connectivity
- API untuk bekerja dengan sistem database (relasional)
- JDBC driver tersedia untuk berbagai sistem database relasional populer
- Sejak JDBC 4.0 (Java 6, tahun 2006), driver untuk sistem database bekerja sesuai mekanisme Service Provider
  - Driver yang kompatibel menyertakan file META-INF/services/java.sql.Driver. Ini merupakan file teks yang berisi nama implementasi interface java.sql.Driver.
- Manual ini hanya membahas pengantar untuk bekerja dengan sistem database relasional. Bacalah dokumentasi API untuk pembahasan selengkapnya.

JDBC

- Driver umumnya didistribusikan dalam satu file jar
  - Driver menyediakan implementasi sejumlah interface
- Pastikanlah file jar ini termasuk dalam class path, pada saat program yang ingin menggunakan driver tersebut dijalankan (akan bekerja dengan sistem database tersebut)

JDBC

- Untuk program yang akan bekerja dengan sejumlah sistem database, pastikanlah semua driver telah termasuk dalam class path, ketika program dijalankan
- Driver tidak dibutuhkan pada saat program dikompilasi

JDBC

- Manual ini akan membahas contoh bekerja dengan sistem database relasional Apache Derby
- Cara kerja dengan sistem database lain umumnya sama, kecuali:
  - Nama driver
  - Parameter koneksi ke database yang bisa berbeda
  - Sintaks dari query dan pemanggilan/penggunaan fungsi yang tersedia pada level database juga dapat berbeda
  - Terdapat penggunaan fitur spesifik database

Apache Derby

- Free/Open Source Software
- Ditulis dengan Java
- Bisa bekerja secara embedded (tidak membutuhkan server database dijalankan) ataupun client/server (server perlu dijalankan terlebih dahulu)
- Tidak diperlukan instalasi secara system-wide dan dapat bekerja dari portable storage device
- Informasi selengkapnya: db.apache.org/derby/
- Pastikanlah file jar driver yang dibutuhkan telah tersedia
  - Sebagai alternatif: Interpreter bahasa pemrograman Singkong (https://nopri.github.io/Singkong.jar) mengandung driver-driver JDBC berikut: Apache Derby (Embedded Driver dan Client Driver) dan PostgreSQL. Network Server Apache Derby juga disertakan dalam file tersebut.
  - Singkong.jar juga datang bersama database tool yang diantaranya dapat digunakan untuk menjalankan query dan mendapatkan hasilnya
  - Apabila menggunakan Singkong.jar, Class.forName() untuk nama class driver diperlukan

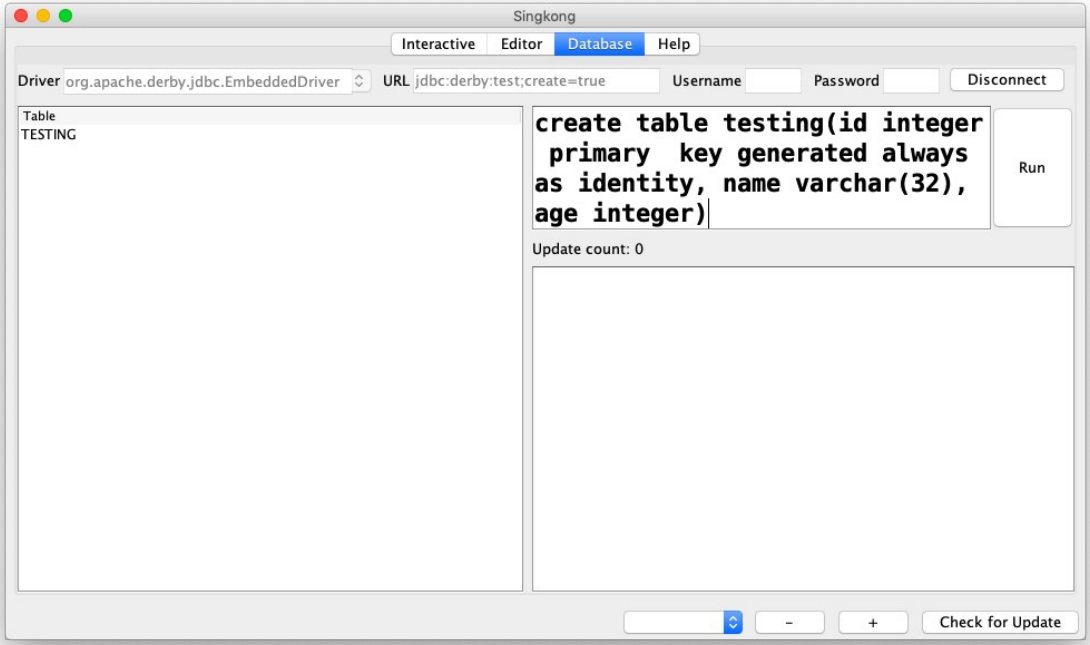
Apache Derby

- Kita akan bekerja dengan satu database, dengan nama test, secara embedded
  - Direktori test akan dibuat di direktori aktif
- Interpreter Singkong (Singkong.jar) akan digunakan dalam contoh ini
- Untuk membuat database tersebut:
  - java -jar Singkong.jar
  - Interpreter Singkong akan dijalankan. Aktiflah pada tab Database.
    - Driver (dipilih): org.apache.derby.jdbc.EmbeddedDriver
    - URL (diisikan): jdbc:derby:test;create=true
    - Username dapat dikosongkan
    - Password dapat dikosongkan
    - Kliklah tombol Connect
  - Database akan dibuat dan kita segera dapat bekerja dengan SQL query
  - Keluarlah dari Interpreter Singkong setelah bekerja dengan database embedded tersebut (misal setelah membuat database dan membuat tabel di halaman berikut).

Ketika bekerja dengan Derby secara embedded, hanya satu Java Virtual Machine yang dapat membuka sebuah database pada satu waktu. Koneksi lain dari Java Virtual Machine yang berbeda tidak dapat dilakukan. Untuk kebutuhan banyak koneksi pada satu database dengan Java Virtual Machine berbeda, gunakanlah Network Server Apache Derby.

Contoh Table

- Kita bekerja dengan contoh table sederhana berikut



Koneksi Database

- Apabila Anda bekerja dengan sistem database lain, atau menggunakan database Apache Derby dengan nama database yang berbeda, atau menggunakan database Apache Derby secara client/server, maka sesuaikanlah URL-nya
- URL yang kita gunakan, setelah database dibuat:
  - jdbc:derby:test
- Nama class driver: org.apache.derby.jdbc.EmbeddedDriver

Koneksi Database

- Kita menggunakan class java.sql.DriverManager dan salah satu method berikut:
  - getConnection(String url)
  - getConnection(String url, Properties info)
  - getConnection(String url, String user, String password)
- Method tersebut mengembalikan sebuah instance dari class yang mengimplementasikan interface java.sql.Connection
- Apabila menggunakan Singkong.jar, Class.forName() untuk nama class driver diperlukan. Semua contoh selanjutnya di dalam bagian pembahasan ini akan menggunakan Class.forName(). Sesuaikanlah apabila diperlukan.

Koneksi Database

```
import java.sql.Connection;
import java.sql.DriverManager;

public class Test {
    public static void main(String[] args) {
        String url = "jdbc:derby:test";
        String user = "";
        String password = "";
        Connection conn = DriverManager.getConnection(url, user, password);
    }
}
```

```
javac Test.java
Test.java:10: error: unreported exception SQLException; must be caught or declar
ed to be thrown
        Connection conn = DriverManager.getConnection(url, user, password);
                                                      ^
1 error
```

SQLException

- Perhatikanlah bahwa ketika bekerja dengan JDBC, banyak method mendeklarasikan kemungkinan terjadinya checked exception java.sql.SQLException
  - Dalam manual ini, akan dicontohkan bagaimana kita menangani SQLException yang terjadi
  - Namun, untuk menjaga agar kode cukup singkat (sehingga dapat terbaca), apabila tidak disebutkan untuk ditangani, maka exception tersebut akan di-throw lagi

SQLException

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Test {
    public static void main(String[] args) throws ClassNotFoundException {
        String url = "jdbc:derby:test";
        String user = "";
        String password = "";
        Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
        try (Connection conn =
            DriverManager.getConnection(url, user, password)) {
            System.out.println("Koneksi database berhasil");
        } catch (SQLException e) {
            System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
        }
    }
}
```

Kita menggunakan try-with-resources dimana Connection dideklarasikan di dalam try

```
javac Test.java
```

← Kompilasi berhasil

Classpath

Walaupun kompilasi berhasil, akan terjadi exception ketika menjalankan program (dengan: java Test). Baik class tidak dapat ditemukan (apabila menggunakan Class.forName()) atau driver yang sesuai tidak ditemukan untuk URL tersebut.

Kita perlu memberitahu virtual machine bahwa terdapat lokasi lain class yang perlu ikut dicari (classpath), dengan opsi -cp (daftar direktori, file jar, file zip; sesuaikanlah pemisah direktori/file dengan sistem operasi yang digunakan). Menempatkan file yang berisi driver (misal: Singkong.jar) saja tidaklah cukup.

Classpath

- Secara default, pada contoh-contoh sebelumnya, pencarian class secara default adalah di direktori aktif (diwakili dengan .)
- Ketika kita akan meng-override lokasi pencarian class dengan opsi -cp, maka kita perlu ikutkan direktori aktif dalam pencarian, apabila memang ini yang kita inginkan
  - Pemisah daftar path di macOS atau Linux adalah :
  - Pemisah daftar path di Windows adalah ;
  - Contoh: -cp .:Singkong.jar
    - Digunakan dalam contoh-contoh pembahasan tentang database di dalam manual ini
    - Sesuaikanlah apabila diperlukan

Classpath

```
java -cp .:Singkong.jar Test
Koneksi database berhasil
```

Apabila terdapat kesalahan, maka exception akan terjadi. Dalam SQLException, detil pesan kesalahan bisa berbeda untuk sistem database yang berbeda.

Statement dan PreparedStatement

- java.sql.Statement
  - Sebuah interface
  - Digunakan untuk menjalankan perintah SQL statik
- java.sql.PreparedStatement
  - Sebuah interface, turunan dari java.sql.Statement
  - Digunakan untuk menjalankan statement SQL yang dikompilasi terlebih dahulu
  - Kita bisa bekerja dengan parameter IN
- java.sql.CallableStatement
  - Sebuah interface, turunan dari java.sql.PreparedStatement
  - Digunakan untuk menjalankan stored procedure
    - Tidak dibahas dalam manual ini

Statement dan PreparedStatement

```
1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.ResultSet;
4  import java.sql.SQLException;
5  import java.sql.Statement;
6
7  public class Test {
8      public static void main(String[] args) throws ClassNotFoundException {
9          String url = "jdbc:derby:test";
10         String user = "";
11         String password = "";
12         Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
13         try (Connection conn =
14             DriverManager.getConnection(url, user, password);
15             Statement stmt = conn.createStatement();
16             ResultSet rs = stmt.executeQuery("select * from testing")) {
17         } catch (SQLException e) {
18             System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
19         }
20     }
21 }
```

Kita gunakan createStatement() milik java.sql.Connection (baris 15) dan query diberikan setelahnya (baris 16) dengan executeQuery() milik java.sql.Statement

Statement dan PreparedStatement

```
 1 import java.sql.Connection;
 2 import java.sql.DriverManager;
 3 import java.sql.PreparedStatement;
 4 import java.sql.ResultSet;
 5 import java.sql.SQLException;
 6
 7 public class Test {
 8     public static void main(String[] args) throws ClassNotFoundException {
 9         String url = "jdbc:derby:test";
10         String user = "";
11         String password = "";
12         Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
13         try (Connection conn =
14             DriverManager.getConnection(url, user, password);
15             PreparedStatement stmt = conn.prepareStatement(
16                 "select * from testing where id=? and name=?")) {
17             stmt.setInt(1, 1);
18             stmt.setString(2, "Hello");
19         } catch (SQLException e) {
20             System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
21         }
22     }
23 }
```

Catatan:

- Kita menggunakan prepareStatement() dan bukan createStatement()
- Pada saat prepareStatement(), kita lewatkan argumen berupa query, dengan ? sebagai placeholder parameter
- Gunakanlah method setXxx milik java.sql.PreparedStatement (contoh: setInt() dan setString()) dengan indeks dimulai dari 1 (bukan 0) untuk memberikan nilai parameter IN
- Query dapat dijalankan dengan method executeQuery() milik java.sql.PreparedStatement

ResultSet

- java.sql.ResultSet
  - Sebuah interface
  - Digunakan sebagai representasi hasil dari query
  - Kursor: awalnya diposisikan sebelum baris pertama
  - Beberapa method yang mungkin digunakan dalam manual ini:
    - first(): memindahkan kursor ke baris pertama,
      - mengembalikan true apabila kursor berada pada baris valid; false apabila tidak ada lagi baris
    - last(): memindahkan kursor ke baris terakhir; kembalian sama seperti first()
    - next(): memindahkan kursor ke baris berikutnya (forward); kembalian sama seperti first()
    - getXxx(): mendapatkan nilai sesuai kolom
      - Contoh: getInt() dan getString()
      - Method-method ini di-overload sehingga dapat menerima parameter berupa columnIndex (int) atau columnLabel (String)

Select

```
 7 public class Test {
 8     public static void main(String[] args) throws ClassNotFoundException {
 9         String url = "jdbc:derby:test";
10         String user = "";
11         String password = "";
12         Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
13         try (Connection conn =
14             DriverManager.getConnection(url, user, password);
15             PreparedStatement stmt = conn.prepareStatement(
16                 "select * from testing where id=? and name=?")) {
17             stmt.setInt(1, 1);
18             stmt.setString(2, "Hello");
19             try (ResultSet rs = stmt.executeQuery()) {
20                 while (rs.next()) {
21                     System.out.println(rs.getInt("id"));
22                     System.out.println(rs.getString("name"));
23                 }
24             }
25         } catch (SQLException e) {
26             System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
27         }
28     }
29 }
```

SQL: dengan select, kita ingin mendapatkan semua kolom (select *) dari table testing (from testing) dimana id dan name mengandung nilai tertentu (operator and, where id=? and name=?)

- ? pertama akan diberikan nilai 1 (baris 17)
- ? kedua akan diberikan nilai “Hello” (baris 18)
- Query dikerjakan di baris 19, mengembalikan objek ResultSet (tidak pernah null)
- Perhatikanlah bahwa pengutipan untuk string tidak perlu dilakukan secara manual

Dengan perulangan while, kita pindahkan kursor ke baris berikutnya (next(), forward). Selama terdapat baris valid (mengembalikan true), maka kita dapatkan nilai untuk kolom id dan name di dalam perulangan. Karena table kita masih kosong, maka perulangan tidak dikerjakan.

Select

```
1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.ResultSet;
4  import java.sql.SQLException;
5  import java.sql.Statement;
6
7  public class Test {
8      public static void main(String[] args) throws ClassNotFoundException {
9          String url = "jdbc:derby:test";
10         String user = "";
11         String password = "";
12         Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
13         try (Connection conn =
14             DriverManager.getConnection(url, user, password);
15             Statement stmt = conn.createStatement();
16             ResultSet rs = stmt.executeQuery("select * from testing")) {
17             while (rs.next()) {
18                 System.out.println("id: " + rs.getInt("id"));
19                 System.out.println("name: " + rs.getString("name"));
20             }
21         } catch (SQLException e) {
22             System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
23         }
24     }
25 }
```

Contoh untuk menampilkan seluruh isi table testing (kolom id dan name)

Insert

```
 1 import java.sql.Connection;
 2 import java.sql.DriverManager;
 3 import java.sql.PreparedStatement;
 4 import java.sql.SQLException;
 5
 6 public class Test {
 7     public static void main(String[] args) throws ClassNotFoundException {
 8         String url = "jdbc:derby:test";
 9         String user = "";
10         String password = "";
11         Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
12         try (Connection conn =
13             DriverManager.getConnection(url, user, password);
14             PreparedStatement stmt = conn.prepareStatement(
15                 "insert into testing(name) values(?)")) {
16             stmt.setString(1, "Hello");
17             int res = stmt.executeUpdate();
18             System.out.println(res);
19         } catch (SQLException e) {
20             System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
21         }
22     }
23 }
```

```
javac Test.java
java -cp .:Singkong.jar Test
1
java -cp .:Singkong.jar Test
1
```

← program dijalankan dua kali, dan setiap kalinya berhasil menambahkan satu baris (executeUpdate() mengembalikan 1)

SQL: dengan insert, kita tambahkan ke table testing untuk kolom name (insert into testing(name)) sebuah nilai yang diwakilkan dengan sebuah ?

- Di baris 16, kita berikan nilai “Hello” untuk ? pertama
- Berbeda dengan contoh sebelumnya, query dikerjakan dengan executeUpdate(), yang mengembalikan int
- Nilai kembalian:
  - Row count
  - 0 untuk query yang tidak mengembalikan apa-apa

Update

```
 1 import java.sql.Connection;
 2 import java.sql.DriverManager;
 3 import java.sql.PreparedStatement;
 4 import java.sql.SQLException;
 5
 6 public class Test {
 7     public static void main(String[] args) throws ClassNotFoundException {
 8         String url = "jdbc:derby:test";
 9         String user = "";
10         String password = "";
11         Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
12         try (Connection conn =
13             DriverManager.getConnection(url, user, password);
14             PreparedStatement stmt = conn.prepareStatement(
15                 "update testing set name=? where id=?")) {
16             stmt.setString(1, "Hello World");
17             stmt.setInt(2, 1);
18             int res = stmt.executeUpdate();
19             System.out.println(res);
20         } catch (SQLException e) {
21             System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
22         }
23     }
24 }
```

```
javac Test.java

java -cp .:Singkong.jar Test
1
```

SQL: dengan update, kita ingin meng-update table testing (update testing), dan mengubah isi kolom name dengan sebuah nilai (set name=?) dimana id sama dengan nilai tertentu (where id=?)

- Di baris 16, kita berikan nilai “Hello World” untuk ? pertama
- Di baris 17, kita berikan nilai 1 untuk ? kedua
- Isi kolom name akan diganti dengan “Hello World” apabila isi kolom id=1
- Sama seperti insert, kita menggunakan method executeUpdate()

Delete

```
 1 import java.sql.Connection;
 2 import java.sql.DriverManager;
 3 import java.sql.PreparedStatement;
 4 import java.sql.SQLException;
 5
 6 public class Test {
 7     public static void main(String[] args) throws ClassNotFoundException {
 8         String url = "jdbc:derby:test";
 9         String user = "";
10         String password = "";
11         Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
12         try (Connection conn =
13             DriverManager.getConnection(url, user, password);
14             PreparedStatement stmt = conn.prepareStatement(
15                 "delete from testing where id > ?")) {
16             stmt.setInt(1, 100);
17             int res = stmt.executeUpdate();
18             System.out.println(res);
19         } catch (SQLException e) {
20             System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
21         }
22     }
23 }
```

```
javac Test.java

java -cp .:Singkong.jar Test
0
```

← Dalam contoh, kita hanya isikan dua baris ke dalam table testing dengan id ditambahkan otomatis (dari 1). Dengan demikian, tidak ada id dengan nilai > 100, dan executeUpdate() mengembalikan 0.

SQL: dengan delete, kita ingin menghapus baris dari table testing (delete from testing) dimana id lebih besar dari nilai tertentu (where id > ?)

- Di baris 16, kita berikan nilai 100 untuk ? pertama
- Baris-baris di dalam table testing akan dihapus untuk setiap baris dimana nilai kolom id > 100
- Sama seperti insert dan update, kita menggunakan method executeUpdate()

Label Kolom

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

public class Test {
    public static void main(String[] args) throws ClassNotFoundException {
        String url = "jdbc:derby:test";
        String user = "";
        String password = "";
        Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
        try (Connection conn =
            DriverManager.getConnection(url, user, password);
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("select * from testing")) {
            ResultSetMetaData md = rs.getMetaData();
            for (int i=1; i<=md.getColumnCount(); i++) {
                System.out.print(md.getColumnLabel(i) + "\t");
            }
            System.out.println();
        } catch (SQLException e) {
            System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
        }
    }
}
```

```
javac Test.java

java -cp .:Singkong.jar Test
ID      NAME    AGE
```

Catatan:

- Interface java.sql.ResultSetMetaData berisi sejumlah method untuk mendapatkan metadata resultset
- Jumlah kolom dalam resultset bisa didapatkan dengan getColumnCount()
- Label setiap kolom (klausa SQL as, default ke nama kolom) bisa didapatkan dengan getColumnLabel() dengan argumen berupa nomor kolom
- Bacalah juga dokumentasi interface ini untuk metadata lainnya

Label Kolom

```
try (Connection conn =
    DriverManager.getConnection(url, user, password);
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("select * from testing")) {
    List<String> columns = new ArrayList<>();
    ResultSetMetaData md = rs.getMetaData();
    for (int i=1; i<=md.getColumnCount(); i++) {
        String label = md.getColumnLabel(i);
        System.out.print(label + "\t");
        columns.add(label);
    }
    System.out.println();
    while (rs.next()) {
        for (String column: columns) {
            Object obj = rs.getObject(column);
            System.out.print(obj + "\t");
        }
        System.out.println();
    }
} catch (SQLException e) {
    System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
}
```

```
javac Test.java

java -cp .:Singkong.jar Test
ID      NAME    AGE
1       Hello World     null
2       Hello   null
```

Perhatikanlah bahwa kita menggunakan method getObject(String columnLabel) milik java.sql.ResultSet. Apabila nilai adalah SQL NULL, maka null dikembalikan.

Query yang Dinamis

- Ketika kita tidak mengetahui jenis SQL query yang diberikan (bisa select, bisa insert/update/delete), method execute() milik java.sql.Statement bisa digunakan
  - Method ini di-overload
    - kita menggunakan execute(String sql)
  - Mengembalikan boolean
    - true apabila hasil pertama adalah sebuah ResultSet
    - false apabila merupakan update count atau tidak ada hasil

Query yang Dinamis

```
13          Console console = System.console();
14          if (console != null) {
15              String sql = console.readLine("SQL: ");
16
17              String url = "jdbc:derby:test";
18              String user = "";
19              String password = "";
20              Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
21              try (Connection conn =
22                  DriverManager.getConnection(url, user, password);
23                  Statement stmt = conn.createStatement()) {
24                  boolean ret = stmt.execute(sql);
25                  if (ret) {
26                      try (ResultSet rs = stmt.getResultSet()) {
27                          List<String> columns = new ArrayList<>();
28                          ResultSetMetaData md = rs.getMetaData();
29                          for (int i=1; i<=md.getColumnCount(); i++) {
30                              String label = md.getColumnLabel(i);
31                              System.out.print(label + "\t");
32                              columns.add(label);
33                          }
34                          System.out.println();
35                          while (rs.next()) {
36                              for (String column: columns) {
37                                  Object obj = rs.getObject(column);
38                                  System.out.print(obj + "\t");
39                              }
40                              System.out.println();
41                          }
42                      }
43                  } else {
44                      int update = stmt.getUpdateCount();
45                      System.out.println("Update count: " + update);
46                  }
47              } catch (SQLException e) {
48                  System.out.println("Terjadi kesalahan SQL: " + e.getMe
49              }
50          }
51      }
```

Catatan:

- Hanya cuplikan dari class Test yang ditampilkan
- Method getResultSet() milik java.sql.Statement bisa digunakan untuk mendapatkan hasil sebagai objek ResultSet
  - Apabila merupakan update count atau tidak ada hasil lagi, mengembalikan null
- Method getUpdateCount() milik java.sql.Statement bisa digunakan untuk mendapatkan hasil sebagai update count
  - Apabila merupakan sebuah ResultSet atau tidak ada hasil lagi, maka nilai -1 dikembalikan
- Contoh ini tidak melakukan pemeriksaan tambahan yang mungkin diperlukan

Query yang Dinamis

```
javac Test.java

java -cp .:Singkong.jar Test
SQL: select id, name from testing
ID      NAME
1       Hello World
2       Hello

java -cp .:Singkong.jar Test
SQL: insert into testing(name) values('test')
Update count: 1

java -cp .:Singkong.jar Test
SQL: error
Terjadi kesalahan SQL: Syntax error: Encountered "error" at line 1, column 1.
```

Generated Key

```
 8 public class Test {
 9     public static void main(String[] args) throws ClassNotFoundException {
10         String url = "jdbc:derby:test";
11         String user = "";
12         String password = "";
13         Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
14         try (Connection conn =
15             DriverManager.getConnection(url, user, password);
16             PreparedStatement stmt = conn.prepareStatement(
17                 "insert into testing(name) values(?)",
18                 Statement.RETURN_GENERATED_KEYS)) {
19             stmt.setString(1, "Hello World");
20             int ret = stmt.executeUpdate();
21             if (ret > 0) {
22                 try (ResultSet rs = stmt.getGeneratedKeys()) {
23                     if (rs.next()) {
24                         System.out.println("Generated " + rs.getInt(1));
25                     }
26                 }
27             }
28         } catch (SQLException e) {
29             System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
30         }
31     }
32 }
```

Catatan:

- Kita ingin mendapatkan generated key setelah insert dilakukan
- Perhatikanlah bahwa kita menggunakan prepareStatement(String sql, int autoGeneratedKeys), dengan autoGeneratedKeys adalah RETURN_GENERATED_KEYS milik java.sql.Statement (baris 16-18)
- Ketika executeUpdate() > 0 (baris 21), kita bisa dapatkan sebuah ResultSet dengan getGeneratedKeys() milik java.sql.Statement (baris 22)
- Apabila next() berhasil (baris 23), kita dapatkan generated key dalam kolom pertama (baris 24). Dalam contoh tabel kita, dimana id adalah integer, kita gunakan getInt(int columnIndex).

```
javac Test.java

java -cp .:Singkong.jar Test
Generated 4
```

Transaksi

- Secara default, koneksi ke database berada dalam mode auto-commit
  - Setiap statement SQL akan dijalankan dan di-commit sebagai transaksi individual
- Apabila beberapa statement merupakan sebuah transaksi (harus berhasil semua atau gagal semua), maka kita perlu meminta agar tidak berada dalam mode auto-commit dengan setAutoCommit(false) milik java.sql.Connection
  - Commit dapat dilakukan dengan commit() dan rollback (batalkan) dapat dilakukan dengan rollback()
- Savepoint – yang menandai point tertentu dalam transaksi – didukung, namun tidak dibahas dalam manual ini

Transaksi

```
 6 public class Test {
 7     public static void main(String[] args) throws ClassNotFoundException {
 8         String url = "jdbc:derby:test";
 9         String user = "";
10         String password = "";
11         Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
12         try (Connection conn =
13             DriverManager.getConnection(url, user, password);
14             PreparedStatement stmt = conn.prepareStatement(
15                 "insert into testing(name) values(?)")) {
16             for (int i=0; i<3; i++) {
17                 String s = "Hello " + i;
18                 stmt.setString(1, s);
19                 stmt.executeUpdate();
20 
21                 if (i == 1) {
22                     throw new SQLException("Sengaja");
23                 }
24             }
25         } catch (SQLException e) {
26             System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
27         }
28     }
29 }
```

```
javac Test.java

java -cp .:Singkong.jar Test
Terjadi kesalahan SQL: Sengaja
```

Catatan:

- Di baris 21-23, kita sengaja throw sebuah SQLException, yang menyebabkan perulangan kita (dimulai baris 16) tidak selesai
- Walau demikian, ketika i bernilai 0 dan 1, statement insert masih sempat dikerjakan
- Sehingga, baris “Hello 0” dan “Hello 1” akan ditambahkan ke tabel, namun “Hello 2” tidak (karena terjadi exception)
- Yang kita inginkan adalah semua insert berhasil dilakukan atau tidak ada insert sama sekali yang terjadi
- Ketika terjadi kesalahan, kita ingin batalkan insert sebelumnya

Transaksi

```
12          try (Connection conn =
13              DriverManager.getConnection(url, user, password)) {
14              try (PreparedStatement stmt = conn.prepareStatement(
15                  "insert into testing(name) values(?)")) {
16                  conn.setAutoCommit(false);
17                  for (int i=0; i<3; i++) {
18                      String s = "Hello " + i;
19                      stmt.setString(1, s);
20                      stmt.executeUpdate();
21
22                      if (i == 1) {
23                          throw new SQLException("Sengaja");
24                      }
25                  }
26                  conn.commit();
27              } catch (SQLException e) {
28                  System.out.println("Terjadi kesalahan SQL, rollback ");
29                  conn.rollback();
30              }
31          } catch (SQLException e) {
32              System.out.println("Terjadi kesalahan SQL: " + e.getMessage());
33          }
```

Catatan:

- Perhatikanlah bahwa conn dipindahkan ke blok try luar supaya catch yang dalam bisa mengakses variabel tersebut (dalam contoh ini)
- Di baris 16, kita disable auto-commit
- Di baris 22-24, kita tetap throw sebuah SQLException
- Ketika exception ini terjadi, maka semua yang terjadi sebelumnya dalam sesi ini dibatalkan dengan rollback() (baris 29)
- Andaikata tidak ada exception yang terjadi (baris 22-24 bisa dikomentari), maka keseluruhan insert akan di-commit dengan commit() (baris 26)

```
javac Test.java

java -cp .:Singkong.jar Test
Terjadi kesalahan SQL, rollback
```

Contoh Soal

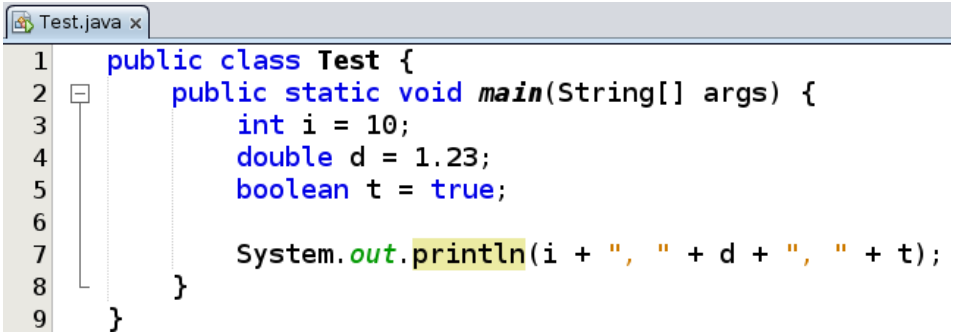
- Kita telah bekerja dengan sistem database relasional. Cobalah untuk membuat program yang:
  - Dapat menampilkan semua data dalam tabel
  - Dapat menghapus atau meng-update data yang ditampilkan tersebut berdasarkan id yang dipilih
  - Dapat menambahkan data baru → tampilkan id data yang ditambahkan tersebut
  - Dapat menangani dengan baik setiap exception yang terjadi

Bagian 6

Pembahasan tambahan

Format

- Contoh-contoh sebelumnya menggunakan pola berikut ketika menampilkan ke standard output:




```
javac Test.java

java Test
10, 1.23, true
```

Format

- Kita bisa gunakan class java.util.Formatter seperti contoh sederhana berikut:

Test.java

```
1   public class Test {
2       public static void main(String[] args) {
3           int i = 10;
4           double d = 1.23;
5           boolean t = true;
6
7           String output = String.format("Formatter: %d, %.2f, %b", i, d, t);
8           System.out.println(output);
9       }
10  }
```

```
javac Test.java

java Test
Formatter: 10, 1.23, true
```

Format

- Kita dapat memformat tampilan berbagai tipe data
- Untuk setiap tipe data, terdapat sejumlah pengaturan (seperti menampilkan nama bulan secara lengkap untuk tanggal/waktu, mengatur presisi, jumlah karakter, dan lainnya)
- Kita membutuhkan:
  - Format string
  - Daftar argumen

Format

Test.java

```
1  import java.math.BigInteger;
2
3  public class Test {
4      public static void main(String[] args) {
5          int i = -123456789;
6          double d = 1.23;
7          BigInteger n = new BigInteger("12345678901234567890123456789");
8
9          String output = String.format("Contoh: %(0,20d dan %10.4f dan %,d",
10                 i, d, n);
11         System.out.println(output);
12     }
13 }
```

```
javac Test.java

java Test
Contoh: (0000000123,456,789) dan     1.2300 dan 123,456,789,012,345,678,901,234,
567,890
```

Format

- Di contoh sebelumnya, kita memformat tampilan untuk int, double, dan BigInteger
  - Variabel i:
    - (: bilangan negatif ditampilkan dalam (dan)
    - 0: padding awal dengan 0
    - ,: diberikan pemisah
    - Ditampilkan dalam minimum 20 karakter
  - Variabel d:
    - Ditampilkan dalam minimum 10 karakter
    - Presisi: 4
  - Variabel n:
    - ,: diberikan pemisah

Format

- Umum, karakter, numerik:

  %[argument_index$][flags][width][.precision]conversion
- Tanggal/waktu:

  %[argument_index$][flags][width]conversion

  (Dokumentasi API java.util.Formatter)

Tanggal dan Waktu

- Java versi 8 datang dengan API baru untuk bekerja dengan tanggal dan waktu
- Contoh berikut membutuhkan Java versi 8

Tanggal dan Waktu

Test.java

```
1  import java.time.LocalDate;
2  import java.time.LocalDateTime;
3  import java.time.LocalTime;
4  import java.time.Month;
5
6  public class Test {
7      public static void main(String[] args) {
8          LocalDateTime t1 = LocalDateTime.now();
9          System.out.println(t1);
10
11         LocalDate t2 = LocalDate.of(2018, Month.JANUARY, 1);
12         System.out.println(t2);
13
14         LocalTime t3 = LocalTime.parse("00:01:02");
15         System.out.println(t3);
16     }
17 }
```

Catatan:

- Dalam contoh tersebut, kita mendapatkan waktu saat ini dengan method now() dari class java.time.LocalDateTime (baris 8)
- Kita juga bisa mendapatkan sebuah java.time.LocalDate dengan method of() (baris 11)
- java.time.LocalTime bisa didapatkan juga dari CharSequence, dengan method parse()

Tanggal dan Waktu

Test.java

```
1  import java.time.LocalDateTime;
2
3  public class Test {
4      public static void main(String[] args) {
5          LocalDateTime t1 = LocalDateTime.now();
6          System.out.println(t1);
7
8          LocalDateTime t2 = t1.plusDays(1).plusHours(1).plusMinutes(1);
9          System.out.println(t2);
10     }
11 }
```

Kita dapat menambahkan waktu dengan berbagai method yang tersedia. Bacalah dokumentasi API untuk class terkait.

Tanggal dan Waktu

- Ketika bekerja dengan Java versi 7 atau sebelumnya, gunakanlah:
  - java.util.Date

Test.java ×

```
1  import java.util.Date;
2
3  public class Test {
4      public static void main(String[] args) {
5          Date date = new Date();
6          System.out.println(date);
7      }
8  }
```

Interpreter Bahasa Pemrograman Singkong

- Interpreter Bahasa Pemrograman Singkong: Singkong.jar, dapat didownload dari https://nopri.github.io
  - Buku Singkong: singkong.pdf juga dapat didownload gratis
  - Cara penggunaan interpreter Singkong dapat dibaca di buku tersebut
- Interpreter Singkong kompatibel dengan Java versi 5.0 atau yang lebih baru
- Singkong.jar perlu ditambahkan ke class path pada saat kompilasi dan menjalankan program Java, yang ingin menggunakan interpreter Singkong

Interpreter
Bahasa Pemrograman Singkong

Interpretasi

Isi file Test.java

```
import com.noprianto.singkong.Singkong;


public class Test {
    public static void main(String[] args) {
        String code = "var list = [1,2,3] println(list)";
        String output = Singkong.evaluatorString(code);
        System.out.println(output);
    }
}
```

Kompilasi dan menjalankan program:

```
javac -cp Singkong.jar Test.java
java -cp Singkong.jar:. Test
[1, 2, 3]
```

Interpreter
Bahasa Pemrograman Singkong

Interpretasi, Built-in

Isi file Test.java

```
import com.noprianto.singkong.Singkong;


public class Test {
    public static void main(String[] args) {
        String code = "println([1,2,3]) system()";
        String output = Singkong.evaluatorString(code, new
String[]{"system"});
        System.out.println(output);
    }
}
```

Kompilasi dan menjalankan program:

```
javac -cp Singkong.jar Test.java
java -cp Singkong.jar:. Test
[1, 2, 3]
ERROR: built-in function "system" is disabled
```

Interpreter
Bahasa Pemrograman Singkong

Interpretasi, Environment

Isi file Test.java

```
import java.util.Map;
import java.util.HashMap;
import com.noprianto.singkong.Singkong;

public class Test {
    public static void main(String[] args) {
        Map<String, Object> map = new HashMap<String,
Object>();
        map.put("hello", "Hello, World");
        map.put("test", true);

        String code = "println(hello) println(test)";
        String output = Singkong.evaluatorString(code,
            Singkong.environmentFromMap(map));

        System.out.println(output);
    }
}
```

Kompilasi dan menjalankan program:

```
javac -cp Singkong.jar Test.java
java -cp Singkong.jar:. Test
Hello, World
true
```

Interpreter
Bahasa Pemrograman Singkong

Interpretasi, Environment, Built-in

Isi file Test.java

```
import java.util.Map;
import java.util.HashMap;
import com.noprianto.singkong.Singkong;

public class Test {
    public static void main(String[] args) {
        Map<String, Object> map = new HashMap<String,
Object>();
        map.put("hello", "Hello, World");

        String code = "println(hello) info()";
        String output = Singkong.evaluatorString(code,
            Singkong.environmentFromMap(map),
            new String[]{"system", "info"});

        System.out.println(output);
    }
}
```

Kompilasi dan menjalankan program:

```
javac -cp Singkong.jar Test.java
java -cp Singkong.jar:. Test
Hello, World
ERROR: built-in function "info" is disabled
```

Interpreter
Bahasa Pemrograman Singkong

Interpretasi, Environment, PrintStream

Isi file Test.java

```
import java.io.ByteArrayOutputStream;
import java.io.PrintStream;
import java.util.HashMap;
import java.util.Map;
import com.noprianto.singkong.Singkong;

public class Test {
    public static void main(String[] args) {
        String result = "";

        Map<String, Object> map = new HashMap<String,
Object>();
        map.put("hello", "Hello, World");
        map.put("test", true);

        ByteArrayOutputStream outputStream = new
ByteArrayOutputStream();
        try {
            PrintStream output = new PrintStream(outputStream);
            Singkong.evaluatorString("println(hello)
println(test)",
                    Singkong.environmentFromMap(map), output);
            result = outputStream.toString();
        } catch (Exception e) {
        }

        System.out.println(result);
    }
}
```

Kompilasi dan menjalankan program:

```
javac -cp Singkong.jar Test.java
java -cp Singkong.jar:. Test
Hello, World
true
```

Interpreter
Bahasa Pemrograman Singkong

Interpretasi, Environment, Built-in, PrintStream

Isi file Test.java

```
import java.io.ByteArrayOutputStream;
import java.io.PrintStream;
import java.util.HashMap;
import java.util.Map;
import com.noprianto.singkong.Singkong;

public class Test {
    public static void main(String[] args) {
        String result = "";

        Map<String, Object> map = new HashMap<String,
Object>();
        map.put("hello", "Hello, World");

        ByteArrayOutputStream outputStream = new
ByteArrayOutputStream();
        try {
            PrintStream output = new PrintStream(outputStream);
            Singkong.evaluatorString("println(hello) info()",
                    Singkong.environmentFromMap(map), output,
                    new String[]{"system", "info"});
            result = outputStream.toString();
        } catch (Exception e) {
        }

        System.out.println(result);
    }
}
```

Kompilasi dan menjalankan program:

```
javac -cp Singkong.jar Test.java
java -cp Singkong.jar:. Test
Hello, World
ERROR: built-in function "info" is disabled
```

Interpreter
Bahasa Pemrograman Singkong

Kode program Singkong juga dapat memanggil method Java.

Untuk informasi selengkapnya, bacalah juga pembahasan tentang **Memanggil Method Java** dan **Embedding Singkong** di Buku Singkong.

Referensi

- The Java ® Language Specification, Java SE 7 Edition
- Java™ Platform, API Specification
- Mengenal dan Menggunakan Bahasa Pemrograman Singkong

Java® adalah bahasa pemrograman *general-purpose* (dapat digunakan untuk membangun aplikasi berbagai domain), mendukung beberapa paradigma pemrograman (termasuk *object-oriented* berbasis *class*), dengan sistem tipe *strong* dan *static*, dan termasuk bahasa pemrograman *high-level*.

Manual pendamping training ini difokuskan pada dasar-dasar bahasa pemrograman Java untuk digunakan di desktop (termasuk bekerja dengan GUI dan sistem database relasional). Materi dirancang agar siap diterapkan dan setiap bagian pembahasan dapat diakhiri dengan contoh soal.

Dr. Noprianto menyukai pemrograman, memiliki sertifikasi Java (OCP), dan telah menulis beberapa buku cetak: Python (2002), Debian GNU/Linux (2006), OpenOffice.org (2006), Java (2018), dan Singkong (2020). Beliau juga menulis beberapa buku elektronik gratis: wxWidgets (2006), Python (2009), SQLiteBoy (2014), dan OpenERP (rekan penulis, 2014).

Noprianto lulus dari jurusan teknik informatika (2003), manajemen sistem informasi (2015), dan doktor ilmu komputer (2019) dari Universitas Bina Nusantara.

Noprianto mengembangkan bahasa pemrograman Singkong dan interpreternya sejak 2019.

Buku dan softwarenya dapat didownload dari https://nopri.github.io

---

ISBN 978-602-52770-0-9

PT. Stabil Standar Sinergi
Email: info@stabilstandar.com

9 786025 277009