

# **Modul Praktikum RPL Berorientasi Obyek**

**TI0231 - Semester Genap 2021/2022**

**Disusun oleh Antonius Rachmat C., S.Kom., M.Cs dan I Kadek Dendy Senapartha, S.T.,  
M.Eng.**



**UNIVERSITAS KRISTEN  
DUTA WACANA**

Copyright © 2022 Antonius Rachmat C dan I Kadek Dendy S

DIPUBLIKASIKAN OLEH PROGRAM STUDI INFORMATIKA UKDW

ANTON@TI.UKDW.AC.ID DAN DENDY.PARTHA@STAFF.UKDW.AC.ID

Licensed under the Creative Commons Attribution-NonCommercial 3.0 Unported License (the “License”). You may not use this file except in compliance with the License. You may obtain a copy of the License at <http://creativecommons.org/licenses/by-nc/3.0>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

*Yogyakarta, Agustus 2021*

## Kata Pengantar

Puji syukur kami panjatkan kepada Tuhan yang maha kuasa, atas berkat dan penyertaannya sehingga modul praktikum RPL BO semester genap 2021/2022 ini berhasil kami susun dengan baik. Modul praktikum ini digunakan untuk mata kuliah Praktikum Rekayasa Perangkat Lunak Berorientasi Obyek (TI0231) pada program studi Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana, Yogyakarta.

Modul praktikum ini disusun dengan pendekatan praktis, lebih banyak menekankan pada kegiatan praktek untuk mengimplementasikan konsep-konsep pemrograman berorientasi obyek. Bahasa pemrograman yang digunakan sebagai acuan dalam modul praktikum ini adalah Java, yang saat ini merupakan salah satu bahasa pemrograman yang paling populer dan banyak digunakan, misalnya pada platform Android.

Modul praktikum ini mengakomodasi 14 pertemuan dalam satu semester yang terbagi dalam 14 bab. Setiap bab terdiri dari tujuan praktikum, dasar teori, kegiatan praktikum dan latihan mandiri. Bagian tujuan praktikum pada suatu bab secara umum berisi mengenai hasil belajar yang ingin dicapai setelah mempelajari bab tersebut. Untuk landasan teori berisi teori-teori yang menunjang praktikum. Bagian kegiatan praktikum dibuat dengan pendekatan penyelesaian masalah, sehingga mahasiswa dapat berlatih menerapkan dasar teori dan konsep-konsep yang telah dipelajari sebelumnya. Bagian terakhir adalah latihan mandiri, berisi permasalahan-permasalahan yang harus dipecahkan oleh mahasiswa dengan membuat program sesuai dengan spesifikasi yang diberikan.

Modul praktikum ini telah disesuaikan berdasarkan kurikulum program studi Informatika dan telah mengalami beberapa penyesuaian sampai revisi yang terakhir ini tahun 2021 dimana matakuliah RPL dan PBO digabungkan. Matakuliah PBO menjadi praktikum RPL dengan nama Praktikum RPL-BO. Beberapa perbaikan telah dilakukan dan beberapa materi baru telah ditambahkan seperti Git, Maven, dan perubahan susunan materi dibandingkan dengan matakuliah PBO dulunya. Modul praktikum ini secara umum disusun dalam tiga bagian besar, yaitu dasar-dasar pemrograman berorientasi obyek, konsep bahasa pemrograman berorientasi obyek dan Java tingkat lanjut.

Kami harapkan modul praktikum ini dapat menjadi panduan dan membantu proses pembelajaran pada mata kuliah Praktikum Rekayasa Perangkat Lunak Berorientasi Obyek di program studi Informatika, FTI UKDW. Kami juga mengharapkan saran, masukan dan kritik mengenai modul praktikum ini, sehingga dapat terus diperbaiki dan terus disesuaikan dengan kebutuhan penyelenggaraan mata kuliah Praktikum RPL-BO. Selamat belajar, bereksplorasi dan meningkatkan kemampuan diri.

Yogyakarta, 30 Januari 2022

Penulis



## Daftar Isi

Sampul .....	i
Copyright .....	ii
Kata Pengantar .....	iii
Daftar Isi .....	v
Daftar Gambar .....	1
Daftar Tabel .....	1

### I

## Perkenalan Git, Maven, Java dan Pengenalan OOP

<b>1</b>	<b>Silabus, Git Versioning Control dan Github .....</b>	<b>3</b>
<b>1.1</b>	<b>Tujuan Praktikum .....</b>	<b>3</b>
<b>1.2</b>	<b>Alat dan Bahan .....</b>	<b>3</b>
<b>1.3</b>	<b>Silabus .....</b>	<b>4</b>
1.3.1	Materi Pertemuan .....	4
1.3.2	Aturan Praktikum .....	4
<b>1.4</b>	<b>Dasar Teori .....</b>	<b>4</b>
1.4.1	Pengenalan Version Control .....	4
1.4.2	Dasar-dasar Git .....	6
1.4.3	Github .....	15
<b>1.5</b>	<b>Kegiatan Praktikum .....</b>	<b>16</b>
1.5.1	Latihan Terbimbing .....	16

1.5.2	Latihan Mandiri	17
<b>2</b>	<b>Pengantar PBO dan Pemrograman Java</b>	<b>19</b>
<b>2.1</b>	<b>Tujuan Praktikum</b>	<b>19</b>
<b>2.2</b>	<b>Alat dan Bahan</b>	<b>19</b>
<b>2.3</b>	<b>Dasar Teori</b>	<b>20</b>
2.3.1	Kompilasi dan Menjalankan Program Java	20
2.3.2	IntelliJ IDEA	20
2.3.3	Class pada Java	20
2.3.4	Tipe Data dan Variabel pada Java	22
2.3.5	Struktur Kontrol Percabangan dan Perulangan pada Java	23
<b>2.4</b>	<b>Kegiatan Praktikum</b>	<b>24</b>
2.4.1	Membuat Program Hello World! Menggunakan IntelliJ	24
2.4.2	Latihan Terbimbing	25
2.4.3	Latihan Mandiri	26
<b>3</b>	<b>Maven</b>	<b>27</b>
<b>3.1</b>	<b>Tujuan Praktikum</b>	<b>27</b>
<b>3.2</b>	<b>Alat dan Bahan</b>	<b>27</b>
<b>3.3</b>	<b>Dasar Teori</b>	<b>27</b>
3.3.1	Tentang Maven	27
3.3.2	Instalasi Maven	29
3.3.3	Membuat Project Java dengan Maven	30
3.3.4	Struktur Direktori Project Maven	31
3.3.5	Mengenal file pom.xml	32
3.3.6	Kompilasi Program dengan Maven	33
<b>3.4</b>	<b>Kegiatan Praktikum</b>	<b>35</b>
3.4.1	Membuat dan mem-build project Java dengan Maven	35
<b>4</b>	<b>Class dan Object</b>	<b>37</b>
<b>4.1</b>	<b>Tujuan Praktikum</b>	<b>37</b>
<b>4.2</b>	<b>Alat dan Bahan</b>	<b>37</b>
<b>4.3</b>	<b>Dasar Teori</b>	<b>37</b>
4.3.1	Class dan Object	37
<b>4.4</b>	<b>Kegiatan Praktikum</b>	<b>39</b>
4.4.1	Mendefinisikan Class Tabungan	39
<b>4.5</b>	<b>Latihan Mandiri</b>	<b>40</b>
4.5.1	Pengecekan Method penarikan() dan penyetoran()	40
4.5.2	Implementasi Method transfer()	41
<b>5</b>	<b>Encapsulation, Static Field dan Static Method</b>	<b>43</b>
<b>5.1</b>	<b>Tujuan Praktikum</b>	<b>43</b>
<b>5.2</b>	<b>Alat dan Bahan</b>	<b>43</b>
<b>5.3</b>	<b>Dasar Teori</b>	<b>43</b>
5.3.1	Encapsulation	43
5.3.2	Access Specifier, Setter dan Getter	45

5.3.3	Instance Level dan Class Level	46
<b>5.4</b>	<b>Kegiatan Praktikum</b>	<b>46</b>
<b>5.5</b>	<b>Latihan Mandiri</b>	<b>48</b>
5.5.1	Matriks	48
5.5.2	Pembelian	48
<b>6</b>	<b>Constructor dan Overloading Method</b>	<b>51</b>
<b>6.1</b>	<b>Tujuan Praktikum</b>	<b>51</b>
<b>6.2</b>	<b>Alat dan Bahan</b>	<b>51</b>
<b>6.3</b>	<b>Dasar Teori</b>	<b>51</b>
6.3.1	Constructor	51
6.3.2	Overloading Constructor	52
6.3.3	Overloading Method	54
<b>6.4</b>	<b>Kegiatan Praktikum</b>	<b>55</b>
<b>6.5</b>	<b>Latihan Mandiri</b>	<b>57</b>
6.5.1	Melengkapi Class Waktu	57
6.5.2	Implementasi Class Rectangle	57

## II

## Konsep Pemrograman Berorientasi Obyek

<b>7</b>	<b>Inheritance dan Overriding</b>	<b>61</b>
<b>7.1</b>	<b>Tujuan Praktikum</b>	<b>61</b>
<b>7.2</b>	<b>Alat dan Bahan</b>	<b>61</b>
<b>7.3</b>	<b>Dasar Teori</b>	<b>61</b>
7.3.1	Inheritance	61
7.3.2	Constructor pada Inheritance	64
7.3.3	Overriding	66
<b>7.4</b>	<b>Kegiatan Praktikum</b>	<b>67</b>
<b>7.5</b>	<b>Latihan Mandiri</b>	<b>68</b>
<b>8</b>	<b>Abstract Class dan Interface</b>	<b>71</b>
<b>8.1</b>	<b>Tujuan Praktikum</b>	<b>71</b>
<b>8.2</b>	<b>Alat dan Bahan</b>	<b>71</b>
<b>8.3</b>	<b>Dasar Teori</b>	<b>71</b>
8.3.1	Abstract Class	71
8.3.2	Interface	74
<b>8.4</b>	<b>Kegiatan Praktikum</b>	<b>75</b>
<b>8.5</b>	<b>Latihan Mandiri</b>	<b>77</b>
<b>9</b>	<b>Polimorfisme</b>	<b>79</b>
<b>9.1</b>	<b>Tujuan Praktikum</b>	<b>79</b>
<b>9.2</b>	<b>Alat dan Bahan</b>	<b>79</b>

<b>9.3</b>	<b>Dasar Teori</b>	<b>79</b>
9.3.1	Definisi, Jenis Polimorfisme dan Kegunaannya	79
9.3.2	Penerapan Polimorfisme	80
<b>9.4</b>	<b>Kegiatan Praktikum</b>	<b>82</b>
<b>9.5</b>	<b>Latihan Mandiri</b>	<b>83</b>
9.5.1	Soal 1	83
9.5.2	Soal 2	83
<b>10</b>	<b>Penanganan Exception</b>	<b>85</b>
<b>10.1</b>	<b>Tujuan Praktikum</b>	<b>85</b>
<b>10.2</b>	<b>Alat dan Bahan</b>	<b>85</b>
<b>10.3</b>	<b>Dasar Teori</b>	<b>85</b>
10.3.1	Definisi, Jenis Exception dan Kegunaannya	85
10.3.2	Penerapan Exception	87
<b>10.4</b>	<b>Kegiatan Praktikum</b>	<b>89</b>
10.4.1	Exception Password	89
<b>10.5</b>	<b>Latihan Mandiri</b>	<b>93</b>
10.5.1	IncorrectFileName Exception	93

### III

## PBO dan Java Tingkat Lanjut

<b>11</b>	<b>String Manipulation pada Java</b>	<b>97</b>
<b>11.1</b>	<b>Tujuan Praktikum</b>	<b>97</b>
<b>11.2</b>	<b>Alat dan Bahan</b>	<b>97</b>
<b>11.3</b>	<b>Dasar Teori</b>	<b>97</b>
11.3.1	Class String pada Java	97
11.3.2	Sifat dan Karakteristik Class String	98
<b>11.4</b>	<b>Kegiatan Praktikum</b>	<b>99</b>
11.4.1	Mengecek Kesamaan Dua String	99
11.4.2	Menggunakan Method-method pada Class String	100
11.4.3	Penggunaan Class StringBuilder	100
<b>11.5</b>	<b>Latihan Mandiri</b>	<b>101</b>
<b>12</b>	<b>Java Collection</b>	<b>103</b>
<b>12.1</b>	<b>Tujuan Praktikum</b>	<b>103</b>
<b>12.2</b>	<b>Alat dan Bahan</b>	<b>103</b>
<b>12.3</b>	<b>Dasar Teori</b>	<b>103</b>
12.3.1	Array	103
12.3.2	Java Collection Framework	112
<b>12.4</b>	<b>Latihan Praktikum</b>	<b>119</b>
12.4.1	INFIX TO POSTFIX	119
12.4.2	POSTFIX EVALUATOR	120



<b>13</b>	<b>Java IO, Serialisasi, dan Deserialisasi</b>	<b>121</b>
13.1	Tujuan Praktikum	121
13.2	Alat dan Bahan	121
13.3	Dasar Teori	121
13.3.1	Java IO	121
13.3.2	Serialisasi dan Deserialisasi	128
<b>14</b>	<b>Package dan JDBC</b>	<b>131</b>
14.1	Tujuan Praktikum	131
14.2	Alat dan Bahan	131
14.3	Dasar Teori	131
14.3.1	Java Package	131
14.3.2	Java Database Connectivity (JDBC)	135

## Epilog

<b>Bibliography</b>	<b>143</b>
<b>Referensi</b>	<b>143</b>
<b>Daftar Index</b>	<b>145</b>
<b>Index</b>	<b>145</b>



## Daftar Gambar

1.1	Version Control Sistem Lokal	5
1.2	Version Control Sistem Terpusat	6
1.3	Version Control System Terdistribusi	7
1.4	Instalasi Git di Windows	8
1.5	Tampilan hasil perintah git –version	8
1.6	Tampilan hasil konfigurasi	9
1.7	Menambah file pada repo	10
1.8	Hasil Git Status	10
1.9	Hasil Git add	10
1.10	Hasil status commit pertama	11
1.11	Hasil status modifikasi kedua	11
1.12	Hasil status commit kedua	11
1.13	Catatan log perubahan	11
1.14	Catatan log perubahan	12
1.15	Catatan log perubahan	12
1.16	perubahan pada revisi berbeda	12
1.17	diff dari 2 file berbeda	13
1.18	Membatalkan Perubahan File yang Sudah dalam Kondisi staged	14
1.19	File yang Sudah dalam Kondisi Committed	14
1.20	Proses Push repository local ke repository remote	15
1.21	Hasil File di Github	15
1.22	Aplikasi Github Desktop for Windows	16

2.1	Kompilasi dan Menjalankan Program Java .....	20
2.2	Tampilan Awal IntelliJ IDEA .....	21
2.3	Class Diagram untuk Class Lampu .....	22
2.4	Instansiasi Object lampuDepan Berdasarkan Class Lampu .....	23
2.5	Membuat New Project pada IntelliJ IDEA .....	24
2.6	Output dari Program HelloWorld .....	25
2.7	Menentukan Bilangan Terbesar dari Parameter yang Diberikan .....	25
3.1	Overview konsep utama Maven .....	28
3.2	Struktur folder Maven .....	29
3.3	Penambahan maven pada PATH enviroment variable .....	30
3.4	Struktur folder project Maven .....	31
3.5	Struktur direktori maven .....	32
3.6	Hasil perintah mvn package .....	34
4.1	Class Diagram untuk Class Lampu .....	38
4.2	Class Diagram untuk Class Tabungan .....	39
5.1	Class Diagram untuk Class Tabungan .....	44
5.2	Class Tabungan Setelah Diberi Access Specifier .....	45
5.3	Class Tabungan Dengan Tambahan Implementasi No. Rekening .....	47
6.1	Class Diagram untuk Class Waktu .....	55
6.2	Overloading Constructor dan Overloading Method pada Class Waktu ..	56
6.3	Segiempat ABCD pada sistem koordinat Cartesius .....	57
7.1	Class Diagram untuk Class Rekening dan Nasabah .....	62
7.2	Relasi antara class Rekening, RekeningBisnis dan RekeningKeluarga ....	64
7.3	Pesan kesalahan pada class RekeningBisnis .....	65
7.4	Relasi antara class Rekening dan RekeningBisnis .....	67
8.1	Class BangunDatar dengan method luas() dan keliling() .....	72
8.2	Relasi inheritance class BangunDatar dengan subclassnya .....	72
8.3	Class BangunDatar sebagai abstract class .....	73
8.4	Interface Bentuk .....	74
8.5	Relasi antara interface Bentuk dengan class Segitiga .....	75
8.6	Relasi antara class Currency dan class IDR, USD dan Ringgit .....	76
9.1	Penerapan polimorfisme sederhana .....	80
9.2	Soal Mandiri .....	83
10.1	Contoh terjadi synchronous error .....	86
10.2	Contoh inline error handling .....	87
10.3	Contoh try catch .....	87

10.4	Contoh try catch finally .....	88
11.1	Perbedaan cara literal dan instansiasi object baru .....	99
12.1	Class Diagram HewanPeliharaan dan Anjing, Kucing, Kelinci .....	109
12.2	Interface Collection .....	113
12.3	Class konkrit Collection .....	114
14.1	Struktur Package .....	132
14.2	Relasi aplikasi dan JDBC .....	136
14.3	Arsitektur JDBC .....	136
14.4	Implementasi JDBC .....	137



## Daftar Tabel

7.1	Perbedaan class Rekening, RekeningBisnis dan RekeningKeluarga . . . . .	63
7.2	Perbedaan Overloading dan Overriding . . . . .	66
11.1	Beberapa methods di class String . . . . .	98





# Perkenalan Git, Maven, Java dan Pengenalan OOP

<b>1</b>	<b>Silabus, Git Versioning Control dan Github</b>	<b>3</b>
1.1	Tujuan Praktikum	
1.2	Alat dan Bahan	
1.3	Silabus	
1.4	Dasar Teori	
1.5	Kegiatan Praktikum	
<b>2</b>	<b>Pengantar PBO dan Pemrograman Java</b>	<b>19</b>
2.1	Tujuan Praktikum	
2.2	Alat dan Bahan	
2.3	Dasar Teori	
2.4	Kegiatan Praktikum	
<b>3</b>	<b>Maven</b>	<b>27</b>
3.1	Tujuan Praktikum	
3.2	Alat dan Bahan	
3.3	Dasar Teori	
3.4	Kegiatan Praktikum	
<b>4</b>	<b>Class dan Object</b>	<b>37</b>
4.1	Tujuan Praktikum	
4.2	Alat dan Bahan	
4.3	Dasar Teori	
4.4	Kegiatan Praktikum	
4.5	Latihan Mandiri	
<b>5</b>	<b>Encapsulation, Static Field dan Static Method</b>	<b>43</b>
5.1	Tujuan Praktikum	
5.2	Alat dan Bahan	
5.3	Dasar Teori	
5.4	Kegiatan Praktikum	
5.5	Latihan Mandiri	
<b>6</b>	<b>Constructor dan Overloading Method</b>	<b>51</b>
6.1	Tujuan Praktikum	
6.2	Alat dan Bahan	
6.3	Dasar Teori	
6.4	Kegiatan Praktikum	
6.5	Latihan Mandiri	



# 1. Silabus, Git Versioning Control dan Github

## 1.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Silabus dan Penjelasan Perkuliahan
2. Memahami Git dan cara kerjanya
3. Dapat menggunakan Git dan Github

## 1.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 8 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

### 1.3 Silabus

#### 1.3.1 Materi Pertemuan

No.	Materi	Tanggal
1	Perkenalan Git	7-11 Februari 2022
2	Dasar-Dasar OOP	14-18 Februari 2022
3	Maven	21-25 Februari 2022
4	Class vs Object dan Class diagram	28 Februari - 4 Maret 2022
5	Access modifier & Static field, Method, Encapsulation	7-11 Maret 2022
6	Constructor dan Overloading Method	14-18 Maret 2022
7	Inheritance & Overriding	21-25 Maret 2022
8	UTS	28 Maret - 8 April 2022
9	Abstract Class dan Interface	18-22 April 2022
10	Polimorfisme	25 - 29 April 2022
11	Penanganan Exception	9 - 13 Mei 2022
12	String Manipulation	16 - 20 Mei 2022
13	Java Collection	23 - 27 Mei 2022
14	Java I/O dan Serialisasi	30 Mei - 3 Juni 2022
15	Package dan JDBC	6-10 Juni 2022
16	UAS	13-24 Juni 2022

Secara detail silahkan dilihat pada RPS untuk tugas, penilaian, buku acuan, dan detail materi per pertemuan.

#### 1.3.2 Aturan Praktikum

Mahasiswa harus memenuhi aturan berikut:

- Tidak makan dan minum di dalam Lab
- Kehadiran minimal 75% untuk bisa UAS
- Tidak melakukan tindakan curang / melanggar aturan
- Tidak ada ujian / tes susulan
- Tidak ada remedi
- Tidak ada tambahan nilai dengan tugas apapun

### 1.4 Dasar Teori

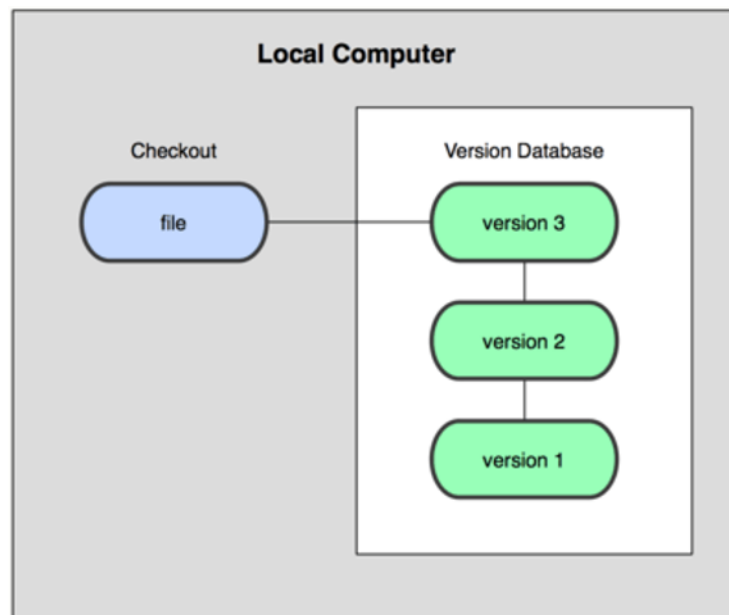
#### 1.4.1 Pengenalan Version Control

Version control adalah sebuah sistem yang mencatat setiap perubahan terhadap sebuah berkas/-file atau kumpulan berkas sehingga pada suatu saat dapat dilakukan proses kembali (back) terhadap salah satu versi dari berkas tersebut. Jika kita adalah seorang desainer grafis, desainer web, atau programmer dan kita ingin menyimpan setiap versi dari gambar, layout, atau kode program yang dibuat sebelumnya, maka Version Control System (VCS) merupakan sebuah solusi untuk digunakan. Sistem ini memungkinkan kita untuk mengembalikan berkas pada kondisi/keadaan sebelumnya, mengembalikan seluruh proyek pada keadaan sebelumnya, membandingkan perubahan setiap saat, melihat melihat siapa yang terakhir melakukan perubahan terbaru pada suatu berkas yang mungkin berpotensi menimbulkan masalah, siapa yang menerbitkan isu, atau peristiwa-peristiwa lainnya. Dengan menggunakan VCS dan jika kita tidak sengaja mengacaukan atau kehilangan berkas, maka kita dapat dengan mudah mengembalikannya.

##### Version Control Sistem Lokal

Kebanyakan orang melakukan pengontrolan versi dengan cara menyalin berkas-berkas pada direktori lain (mungkin dengan memberikan penanggalan pada direktori tersebut, jika mereka

rajin). Metode seperti ini sangat umum karena sangat sederhana, namun cenderung rawan terhadap kesalahan. Kita dapat dengan mudah lupa dimana direktorinya, selain itu dapat pula terjadi ketidaksengajaan penulisan pada berkas yang salah, atau menyalin berkas yang bukan dimaksud, atau ketidaksengajaan lainnya. Untuk mengatasi permasalahan ini, dikembangkan berbagai VCS lokal yang memiliki sebuah basis data sederhana untuk menyimpan semua perubahan pada berkas yang berada dalam cakupan revision control.



Gambar 1.1: Version Control Sistem Lokal

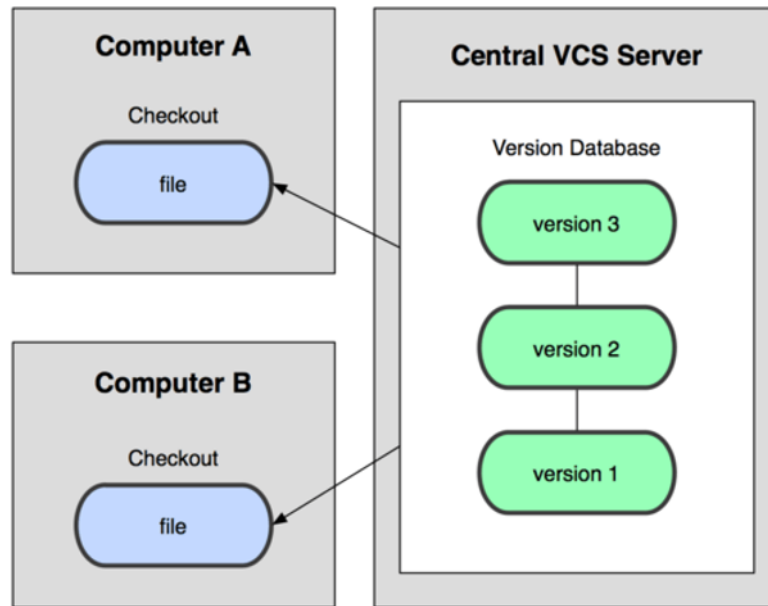
Sistem VCS lokal dapat dilihat pada Gambar 1.1. Contoh sistem VCS lokal adalah Revision Control System (RCS) yang mampu mengelola revisi dari banyak file. RCS mengotomasi penyimpanan, pengambilan, pencatatan, pemeriksaan, dan penggabungan dari sebuah revisi. Berguna untuk berkas yang sering direvisi seperti source code, program, dokumentasi, grafik, jurnal, dan surat.

### Version Control Sistem Terpusat

Permasalahan berikutnya yang dihadapi adalah para pengembang adalah perlunya melakukan kolaborasi dengan pengembang pada sistem lainnya. Untuk mengatasi permasalahan ini maka dibangunlah Centralized Version Control Systems (CVCSs). Sistem ini memiliki beberapa contoh, misalnya CVS, Subversion, dan Perforce. Sistem ini memiliki sebuah server untuk menyimpan setiap versi berkas, dan nantinya para client yang dapat melakukan checkout berkas dari server. Sistem seperti ini telah menjadi standard untuk version control. Cara kerja sistem VCS terpusat dapat dilihat pada Gambar 1.2.

Sistem seperti ini memiliki beberapa kelebihan, terutama jika dibandingkan dengan VCS lokal. Misalnya, setiap orang pada tingkat tertentu mengetahui apa yang orang lain lakukan pada proyek tertentu. Administrator memiliki kendali atas siapa yang dapat melakukan apa, dan jauh lebih mudah untuk mengelola sebuah CVCS dibandingkan menangani database lokal pada setiap client.

Walau demikian, sistem dengan model seperti ini juga memiliki kelemahan serius. Kelemahan nyata ini sama dengan yang direpresntasikan oleh sistem dengan server terpusat. Jika server mati, maka tidak ada seorangpun yang bisa berkolaborasi atau menyimpan perubahan terhadap apa yang mereka telah kerjakan. Jika harddisk yang menyimpan basisdata mengalami kerusakan, dan salinan yang benar belum tersimpan, kita akan kehilangan setiap perubahan dari proyek, kecuali



Gambar 1.2: Version Control Sistem Terpusat

snapshot yang dimiliki oleh setiap kolaborator pada komputernya masing-masing. VCS lokal juga mengalami nasib yang sama jika kita menyimpan seluruh history perubahan proyek pada satu tempat, kita mempunyai resiko kehilangan semuanya.

### Version Control System Terdistribusi

Untuk mengatasi permasalahan sebelumnya, Distributed Version Control Systems merupakan solusinya. Dalam sebuah DVCS (contohnya Git, Mercurial, Bazaar atau Darcs), client tidak hanya melakukan checkout untuk snapshot terakhir setiap berkas, namun client memiliki salinan penuh dari repositori tersebut. Jadi, jika server mati, dan sistem berkolaborasi melalui server tersebut, maka klien manapun dapat mengirimkan salinan repositori tersebut kembali ke server. Setiap checkout pada DVCS merupakan sebuah backup dari keseluruhan data (lihat Gambar 1.3).

Lebih jauh lagi, kebanyakan sistem seperti ini mampu menangani sejumlah remote repository dengan baik, jadi kita dapat melakukan kolaborasi dengan berbagai kelompok kolaborator dalam berbagai cara secara bersama-sama pada suatu proyek. Hal ini memungkinkan kita untuk menyusun beberapa jenis alur kerja yang tidak mungkin dilakukan pada sistem terpusat, seperti hierarchical model.

## 1.4.2 Dasar-dasar Git

### Instalasi Git

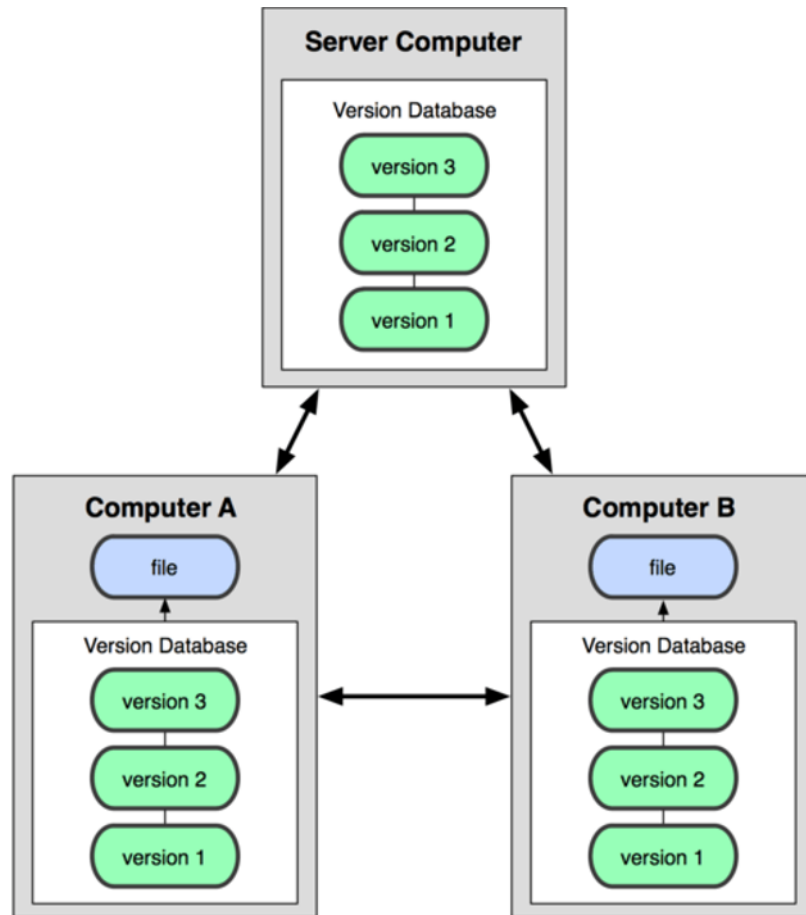
Git dikembangkan oleh pengembang Linux, yaitu Linus Torvalds. Untuk bisa menggunakannya, silahkan buka website resmi Git (<https://www.git-scm.com>). Kemudian unduh Git sesuai dengan arsitektur komputer kita. Jika menggunakan 64bit, unduh yang 64bit, begitu juga sebaliknya jika menggunakan 32bit, download versi 32bit. Kemudian lakukan instalasi seperti biasa. Contoh gambar instalasi Git pada Windows dapat dilihat pada Gambar .

Untuk mencobanya, silahkan buka **cmd** atau PowerShell, kemudian ketik perintah:

---

```
1 git --version
```

---



Gambar 1.3: Version Control System Terdistribusi

### Konfigurasi Awal

Ada beberapa konfigurasi yang harus dipersiapkan sebelum mulai menggunakan Git, seperti nama dan email. Silahkan lakukan konfigurasi dengan perintah berikut ini.

```
1 git config --global user.name "usernameAnda"
2 git config --global user.email "emailAnda"
```

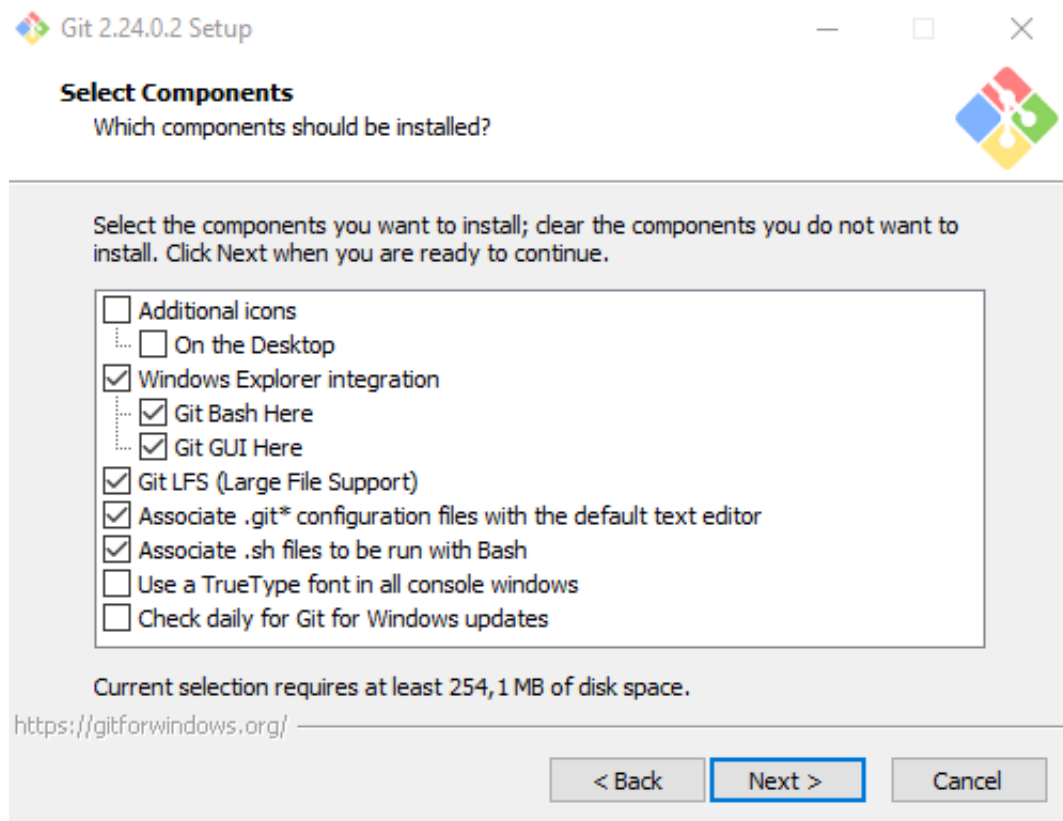
Contoh:

```
1 git config --global user.name "dendy prtha"
2 git config --global user.email "dendy.prtha@staff.ukdw.ac.id"
```

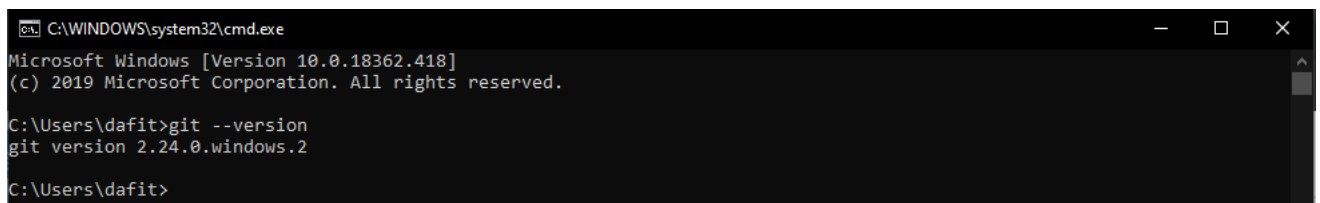
Kemudian periksa konfigurasinya dengan perintah:

```
1 git config --list
```

Apabila berhasil tampil seperti gambar 1.6, berarti konfigurasi berhasil.



Gambar 1.4: Instalasi Git di Windows



Gambar 1.5: Tampilan hasil perintah git --version

### Membuat Repository Baru

Repository (repository) dalam bahasa Indonesia artinya gudang. Repository merupakan istilah yang digunakan untuk direktori proyek yang menggunakan Git. Jika kita memiliki sebuah direktori dengan nama **proyek-01** dan di dalamnya sudah menggunakan git, maka kita sudah punya repository bernama **proyek-01**. Pembuatan repository dapat dilakukan dengan perintah `git init nama-dir`. Contoh pada drive D anda, buka cmd dan ketikkan:

```
1       git init belajar-git
```

Perintah tersebut akan membuat direktori bernama `belajar-git` di drive D. Kalau direktorinya sudah ada, maka Git akan melakukan inisialisasi di dalam direktori tersebut. Perintah `git init` akan membuat sebuah direktori bernama `.git` di dalam proyek kita. Direktori ini digunakan Git sebagai database untuk menyimpan perubahan yang kita lakukan. Hati-hati! Kalau kita menghapus direktori ini, maka semua rekaman atau catatan yang dilakukan oleh Git akan hilang. Contoh lain untuk membuat repository pada direktori saat ini (working directory) adalah sebagai berikut.



```
C:\Users\dendy>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/development/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=wincred
user.name=dendy prtha
user.email=dendy.prtha@staff.ukdw.ac.id
core.autocrlf=input
credential.https://github.com.helper=wincred
credential.https://github.com.username=prtha.dendy@gmail.com
```

Gambar 1.6: Tampilan hasil konfigurasi

---

```
1 git init .
```

---

Tanda titik (.) artinya kita akan membuat repository pada direktori tempat kita berada saat ini. Misal kita sedang berada di D:

praktikum-pbo

pertemuan1

contoh1, maka project git adalah pada contoh1.

### **.gitignore**

.gitignore merupakan sebuah file yang berisi daftar nama-nama file dan direktori yang akan diabaikan oleh Git. Perubahan apapun yang kita lakukan terhadap file dan direktori yang sudah masuk ke dalam daftar .gitignore tidak akan dicatat oleh Git. Cara menggunakan .gitignore, buat saja sebuah file bernama .gitignore dalam root direktori proyek/repository. Pembuatan file .gitignore sebaiknya dilakukan di awal pembuatan repository. Contoh .gitignore pada proyek-proyek tertentu, bisa dilihat di repository ini (<https://github.com/github/gitignore>)

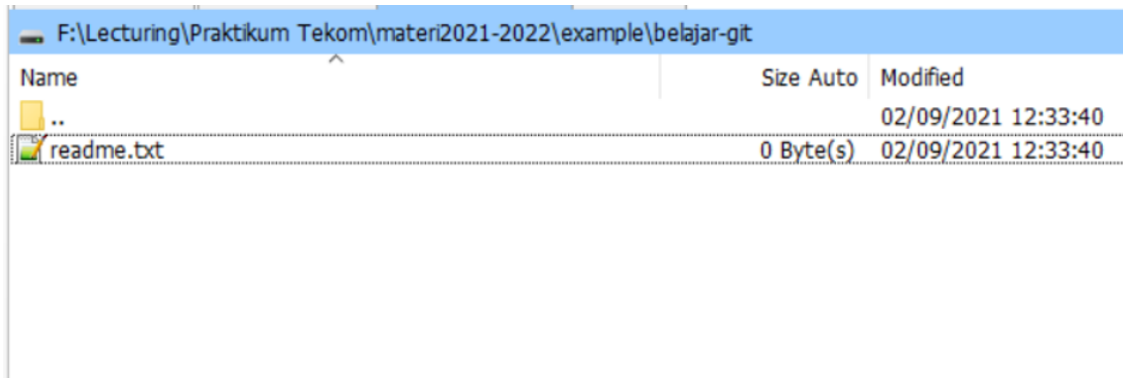
### **Simpan Perubahan Revisi dengan Git Commit**

Sebelumnya kita sudah membuat repository kosong. Belum ada apa-apa di sana. Sekarang coba tambahkan sebuah file baru pada repository Anda. Setelah ditambahkan, coba ketik perintah git status untuk melihat status repositorynya. Berdasarkan keterangan pada gambar 1.8, saat ini kita berada cabang (branch) master dan ada tiga file yang belum ditambahkan ke Git.

### **Tiga Kelompok Kondisi File dalam Git**

Sebelum kita membuat revisi, kita akan berkenalan dulu dengan tiga kondisi file dalam Git.

1. **Modified** : Modified adalah kondisi dimana revisi atau perubahan sudah dilakukan, tetapi belum ditandai dan belum disimpan di version control.
2. **Staged** : Staged adalah kondisi dimana revisi sudah ditandai, tetapi belum disimpan di version control. Untuk mengubah kondisi file dari modified ke staged gunakan perintah git add nama-file.



Name	Size Auto	Modified
..		02/09/2021 12:33:40
readme.txt	0 Byte(s)	02/09/2021 12:33:40

Gambar 1.7: Menambah file pada repo

```
F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        readme.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Gambar 1.8: Hasil Git Status

3. Committed: Committed adalah kondisi dimana revisi sudah disimpan di version control. perintah untuk mengubah kondisi file dari staged ke committed adalah git commit.

### Membuat Revisi Pertama

Sekarang kita akan sudah tahu kondisi-kondisi file dalam Git. Selanjutnya, silahkan ubah kondisi file readme.txt tadi menjadi staged dengan perintah git add. Hasilnya dapat dilihat pada gambar 1.8

```
F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git add readme.txt

F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   readme.txt
```

Gambar 1.9: Hasil Git add

Setelah itu, ubah kondisi file tersebut ke committed agar semua perubahan disimpan oleh Git. Gunakan perintah pada gambar 1.10

### Membuat Revisi Kedua

Silahkan modifikasi isi file readme.txt. Setelah itu ketik lagi perintah git status. Kondisinya sekarang berada dalam modified. Hasilnya dapat dilihat pada gambar 1.11

```
F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git commit -m "commit pertama"
[master (root-commit) db4d08d] commit pertama
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 readme.txt
```

Gambar 1.10: Hasil status commit pertama

```
F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Gambar 1.11: Hasil status modifikasi kedua

Terlihat di sana, file readme.txt sudah dimodifikasi. Kondisinya sekarang berada dalam modified. Lakukan commit lagi seperti revisi pertama. Hasilnya dapat dilihat pada gambar 1.12

```
F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git add readme.txt
F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git commit -m "menambahkan isi"
[master 1d4d1e5] menambahkan isi
1 file changed, 1 insertion(+)
```

Gambar 1.12: Hasil status commit kedua

Dengan demikian, revisi kedua sudah disipkan oleh Git. Mungkin anda belum tahu maksud dari argumen -m, argumen tersebut untuk menambahkan pesan setiap menyimpan revisi. Sekarang Git sudah mencatat dua revisi yang sudah kita lakukan. Kita bisa ibaratkan revisi-revisi ini sebagai checkpoint pada Game. Apabila nanti ada kesalahan, kita bisa kembali ke checkpoint ini.

### Melihat Catatan Log Revisi

Kita sudah membuat dua revisi pada repositori belajar-git. Sekarang bagaimana caranya kita melihat catatan log dari revisi-revisi tersebut? Git sudah menyediakan perintah git log untuk melihat catatan log perubahan pada repositori. Gambar 1.13 adalah contoh penggunaannya.

```
F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git log
commit 1d4d1e50983b7d06ae9050c0bfc61ee6c4064993 (HEAD -> master)
Author: dendy prtha <dendy.prtha@staff.ukdw.ac.id>
Date: Thu Sep 2 12:45:58 2021 +0700

    menambahkan isi

commit db4d08d0e0eac8ff744cd5ad37b36360491c8076
Author: dendy prtha <dendy.prtha@staff.ukdw.ac.id>
Date: Thu Sep 2 12:41:24 2021 +0700

    commit pertama
```

Gambar 1.13: Catatan log perubahan

Untuk menampilkan log yang lebih pendek, kita bisa menambahkan argumen --oneline. Untuk

melihat log pada revisi tertentu, kita bisa memasukkan nomer revisi/commit. Untuk melihat revisi pada file tertentu, kita dapat memasukkan nama filenya. Perhatikan gambar 1.14.

```
F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git log readme.txt
commit 1d4d1e50983b7d06ae9050c0bfc61ee6c4064993 (HEAD -> master)
Author: dendy prtha <dendy.prtha@staff.ukdw.ac.id>
Date: Thu Sep 2 12:45:58 2021 +0700

    menambahkan isi

commit db4d08d0e0eac8ff744cd5ad37b36360491c8076
Author: dendy prtha <dendy.prtha@staff.ukdw.ac.id>
Date: Thu Sep 2 12:41:24 2021 +0700

    commit pertama
```

Gambar 1.14: Catatan log perubahan

Misalkan dalam repositori dikerjakan oleh banyak orang. Maka kita dapat melihat revisi apa saja yang dilakukan oleh orang tertentu dengan perintah pada gambar 1.15

```
F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git log --author="dendy prtha"
commit 1d4d1e50983b7d06ae9050c0bfc61ee6c4064993 (HEAD -> master)
Author: dendy prtha <dendy.prtha@staff.ukdw.ac.id>
Date: Thu Sep 2 12:45:58 2021 +0700

    menambahkan isi

commit db4d08d0e0eac8ff744cd5ad37b36360491c8076
Author: dendy prtha <dendy.prtha@staff.ukdw.ac.id>
Date: Thu Sep 2 12:41:24 2021 +0700

    commit pertama
```

Gambar 1.15: Catatan log perubahan

### Melihat Perbandingan Revisi dengan Git Diff

Kita sudah belajar cara melihat log revisi di repositori. Sekarang kita kan pelajari perintah git diff, fungsinya untuk melihat perbedaan perubahan di revisi. Gunakan perintah seperti gambar 1.16 untuk melihat perubahan yang dilakukan pada revisi tertentu.

```
F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git diff db4d08d0e0eac8ff744cd5ad37b36360491c8076
diff --git a/readme.txt b/readme.txt
index e69de29..18244e5 100644
--- a/readme.txt
+++ b/readme.txt
@@ -0,0 +1 @@
+ belajar GIT dengan semangat dan penuh kebahagiaan
\ No newline at end of file
```

Gambar 1.16: perubahan pada revisi berbeda

Lihatlah hasil di atas, simbol plus (+) artinya kode yang ditambahkan. Sedangkan kalau ada kode yang dihapus simbolnya akan menggunakan minus (-). Apabila kita melakukan banyak perubahan, maka akan banyak sekali tampil output. Karena itu, kita mungkin hanya perlu melihat perubahan untuk file tertentu saja. Untuk melihat perbandingan perubahan pada file tertentu, gunakan perintah seperti pada gambar 1.17. Contoh: lakukan modifikasi file readme.txt, lalu ketikkan perintah git diff readme.txt

```

F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git diff readme.txt
diff --git a/readme.txt b/readme.txt
index 18244e5..dbb1433 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1,1 @@
- belajar GIT dengan semangat dan penuh kebahagiaan
\ No newline at end of file
+ belajar GIT dengan semangat dan perut yang kenyang.
\ No newline at end of file

```

Gambar 1.17: diff dari 2 file berbeda

Kita juga dapat membandingkan perbedaan yang ada pada dua revisi yang berbeda. Perintah yang digunakan adalah `git diff <nomer commit1> <nomer commit2>`.

### Membatalkan Perubahan

Kita sudah belajar cara melihat perbedaan di setiap revisi. Sekarang kita akan belajar, cara membatalkan sebuah revisi. Terkadang pada perubahan yang kita lakukan terjadi kesalahan dan kita ingin mengembalikannya seperti keadaan sebelumnya. Maka kita perlu menyuruh git untuk mengembalikannya. Ada beberapa perintah yang digunakan diantaranya: `git checkout`, `git reset`, dan `git revert`. Jika revisi kita belum staged ataupun committed, kita bisa mengembalikannya menggunakan perintah

---

```
1 git checkout <nama-file>
```

---

Hati-hati! Terkadang perintah ini sangat berbahaya, karena akan menghapus perubahan yang baru saja dilakukan. Bila kita sudah merubah banyak hal, maka itu akan sia-sia setelah menjalankan perintah ini.

### Membatalkan Perubahan File yang Sudah dalam Kondisi staged

Kondisi staged merupakan kondisi file yang sudah di add (`git add`), namun belum disimpan (`git commit`) ke dalam Git. Sebagai contoh, kita lakukan perubahan lagi di file `readme.txt` seperti pada contoh sebelumnya dan setelah itu kita ubah kondisinya menjadi staged dengan perintah `git add`. Perhatikan gambar 1.18.

Nah, file `readme.txt` sudah masuk ke dalam kondisi staged. Untuk mengubahnya menjadi kondisi modified, kita bisa menggunakan perintah `git reset`. Sekarang file `readme.txt` sudah dalam kondisi modified, kita bisa membatalkan perubahannya dengan perintah `git checkout` seperti contoh sebelumnya.

### Membatalkan Perubahan File yang Sudah dalam Kondisi Committed

Sekarang bagaimana kalau filenya sudah dalam kondisi committed dan kita ingin mengembalikannya? Untuk melakukan ini, kita harus mengetahui nomer commit, kemudian mengembalikan perubahannya seperti pada nomer commit tersebut. Misalkan, kita ubah kembali file `readme.txt` dan kemudian kita commit. Sekarang kita akan melihat nomer commit dengan perintah `git log`.

Kita akan mengembalikan kondisi file `readme.txt`, seperti pada commit sebelumnya. Maka kita bisa menggunakan perintah berikut:

---

```
1 git checkout <no revisi> readme.txt
```

---

```

F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.txt

no changes added to commit (use "git add" and/or "git commit -a")

F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git add readme.txt

F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   readme.txt

F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git reset readme.txt
Unstaged changes after reset:
M   readme.txt

F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.txt

no changes added to commit (use "git add" and/or "git commit -a")

```

Gambar 1.18: Membatalkan Perubahan File yang Sudah dalam Kondisi staged

```

F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git add readme.txt

F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git commit -m "belajar git greget!"
[master 7c0eea2] belajar git greget!
1 file changed, 1 insertion(+), 1 deletion(-)

F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git log
commit 7c0eea240e4b0cdec1a66f806f2728f3aa838027 (HEAD -> master)
Author: dendy prtha <dendy.prtha@staff.ukdw.ac.id>
Date: Thu Sep 2 14:06:57 2021 +0700

    belajar git greget!

commit 1d4d1e50983b7d06ae9050c0bfc61ee6c4064993
Author: dendy prtha <dendy.prtha@staff.ukdw.ac.id>
Date: Thu Sep 2 12:45:58 2021 +0700

    menambahkan isi

commit db4d08d0e0eac8ff744cd5ad37b36360491c8076
Author: dendy prtha <dendy.prtha@staff.ukdw.ac.id>
Date: Thu Sep 2 12:41:24 2021 +0700

    commit pertama

```

Gambar 1.19: File yang Sudah dalam Kondisi Committed

Seperti mesin waktu, kita sudah mengembalikan keadaan file index.html seperti keadaan saat commit tersebut. Namun, saat ini kondisi readme.txt dalam keadaan staged. Kita bisa kembalikan ke dalam kondisi modified dengan perintah git reset.

Pada contoh tersebut, kita sudah berhasil mengembalikan file readme.txt ke dalam keadaan seperti commit sebelumnya. Perlu diperhatikan, pengembalian kondisi file ke versi sebelumnya



akan mengubah file tersebut menjadi kondisi modified. Jika memang kita ingin mengembalikan isi file ke versi sebelumnya, anda perlu melakukan add dan commit file tersebut.

### 1.4.3 Github

#### Membuat Repository di Github

Terlebih dahulu, Anda perlu membuat akun Github dengan menggunakan akun email anda. Kemudian silahkan buka Github, kemudian buat sebuah repository dengan nama belajar-git. Bila berhasil, maka github akan menghasilkan repository kosong.

Silahkan buka kembali repository lokal yang pernah kita buat, yaitu belajar-git. Kita akan upload ke Github. Sebelum kita bisa upload semua revisi yang ada di repository lokal, kita harus menambahkan remote-nya terlebih dahulu. Remote dapat kita tambahkan dengan perintah berikut

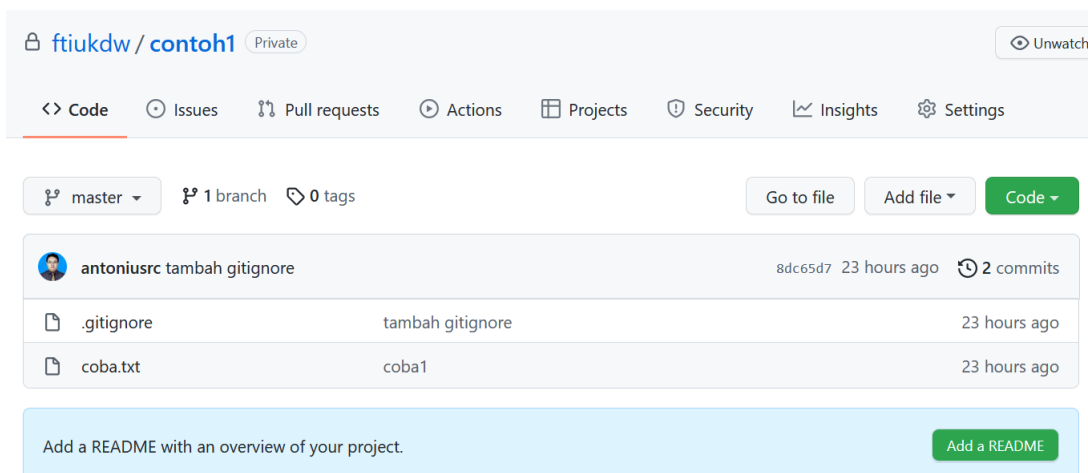
```
1 git remote add github https://github.com/<username anda>/belajar-git.git
```

Setelah itu, silahkan ketik perintah `git remote -v` untuk melihat remote apa saja yang sudah ditambahkan. Selanjutnya kita bisa melakukan push atau mengirim revisi ke repository remote (Github) dengan perintah `git push <nama remote> <nama cabang tujuan>`. Contoh : Sekarang lihat

```
F:\Lecturing\Praktikum Tekom\materi2021-2022\example\belajar-git>git push github master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (10/10), 852 bytes | 213.00 KiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/dendy-senapatha/belajar-git.git
 * [new branch]      master -> master
```

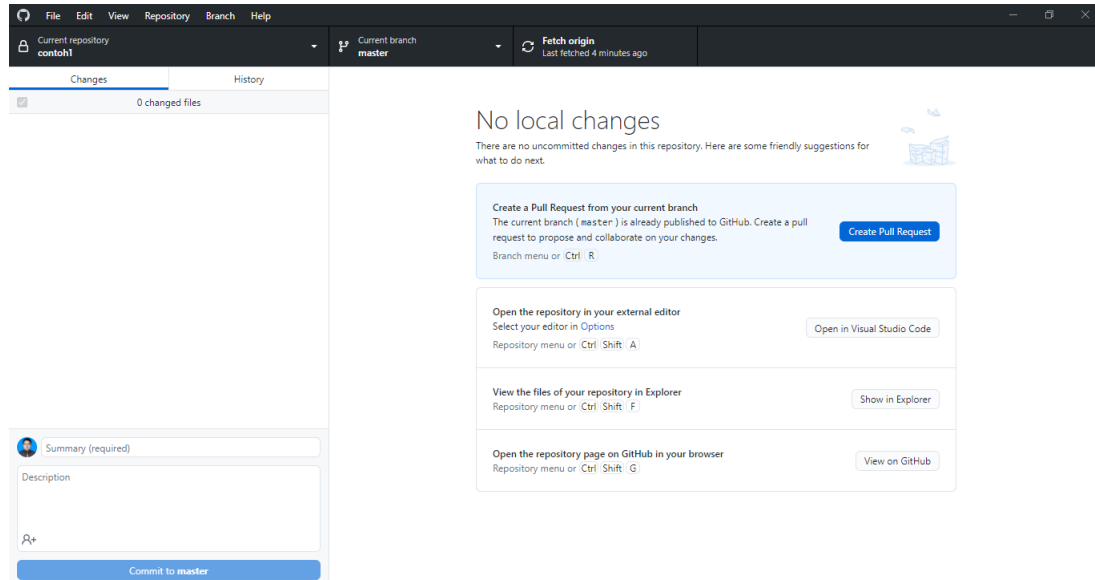
Gambar 1.20: Proses Push repository lokal ke repository remote

ke repository Github kita, pasti semuanya sudah ter-upload disana. Contoh hasil push dapat dilihat pada Gambar 1.21



Gambar 1.21: Hasil File di Github

Untuk memudahkan proses manajemen project-project di Github, silahkan digunakan program Github Desktop for Windows, Mac, atau Linux. Kita dapat mendownload di <https://desktop.github.com/> secara gratis. Dengan menggunakan aplikasi tersebut maka proses manajemen project akan jauh lebih mudah karena semua berbasis visual (GUI based).



Gambar 1.22: Aplikasi Github Desktop for Windows

## 1.5 Kegiatan Praktikum

### 1.5.1 Latihan Terbimbing

1. Buatlah sebuah repository public baru di Github anda masing-masing dengan nama rplbo-guided1
2. Tambahkan readme, .gitignore untuk Java, dan license Mozilla Public License 2.0
3. Tuliskan kode Java berikut dan simpan dengan nama Contoh<NIM>.java, misalnya Contoh22002529.java pada folder repo.

```

1      public class Contoh<NIM> {
2          public static void main(String args[]) {
3              int x=10;
4              int y=25;
5              int z=x+y;
6
7              System.out.println("Hasil penjumlahan"
8                  + " dari x+y = " + z);
9              System.out.println("Hasil akar kuadrat "
10                 + "dari z = " + Math.sqrt(z));
11
12              String namadepan = "Antonius";
13              System.out.println("Nama depan saya adalah = " + namadepan);
14              System.out.println("Nama lengkap saya "
15                 + "adalah = " + namadepan + " Rachmat");

```



```
16         }  
17     }
```

---

4. Lakukan fetch dan pull dari Github terlebih dahulu untuk memastikan data terbaru
5. Lakukan staging, commit, dan push ke GitHub tersebut!
6. Lihat hasilnya di repo Github

### 1.5.2 Latihan Mandiri

1. Clone repository dari Github untuk RPLBO Tugas 1: <https://github.com/antoniusrc/rplbo-tugas1>
2. Buatlah folder sesuai dengan nim anda <nim>
3. Buatlah sebuah file yang berisi kode program Java yang digunakan untuk menghitung penjumlahan  $1 + 2 + 3 + \dots + n$  bilangan (n adalah 3 digit terakhir nim anda). Output dari program adalah hasil penjumlahannya. Beri nama <nim>.java di dalam folder tersebut.
4. Push ke Git dan berikan keterangan "<nim> program java"
5. Outputkan juga TERIMA KASIH <NIM> dan <>Nama Anda> sebagai bagian akhir setelah hasil penjumlahan dikeluarkan.
6. Push ke Git dan berikan keterangan "<nim> output Java"



## 2. Pengantar PBO dan Pemrograman Java

### 2.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Memahami penggunaan IntelliJ sebagai Integrated Development Environment (IDE) untuk pengembangan program dalam bahasa pemrograman Java.
2. Dapat menggunakan IntelliJ untuk membuat project baru, menambahkan file ke dalam project, mengubah isi file, melakukan kompilasi dan menjalankan program.
3. Dapat membuat program dalam bahasa pemrograman Java dalam bentuk mendefinisikan method main(), mendefinisikan dan mengakses variabel serta dapat mengimplementasikan struktur kontrol percabangan.
4. Memahami konsep dan dapat menggunakan bahasa pemrograman Java.
5. Dapat mendefinisikan Class, menambahkan atribut dan method serta melakukan instansiasi Object dari Class yang dibuat.
6. Dapat menampilkan output di console/terminal dan menerima masukan dari keyboard.

### 2.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 4 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

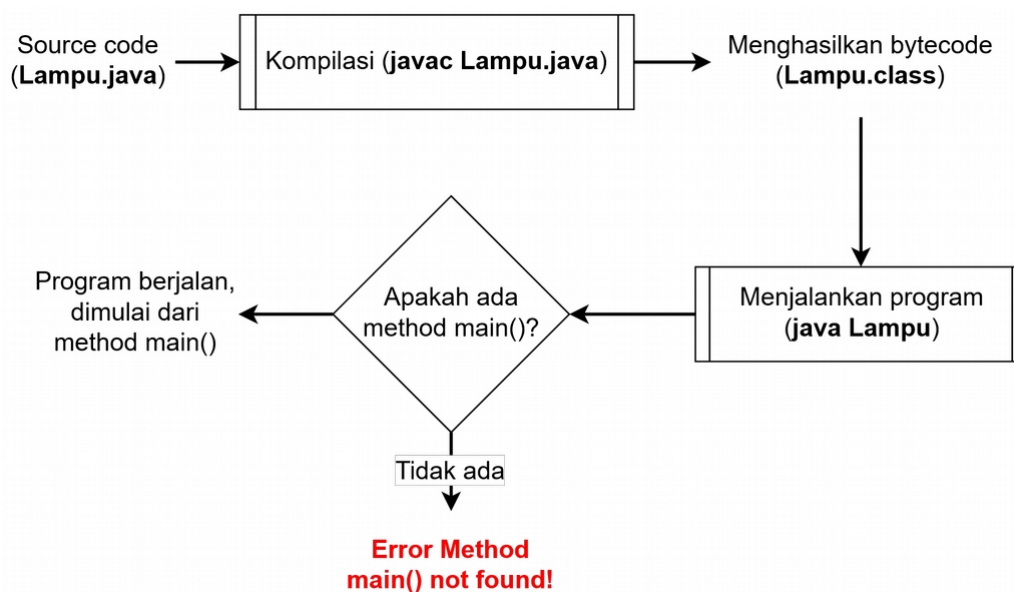
- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 8).
- IntelliJ Ultimate 64 bit (Students Key).

## 2.3 Dasar Teori

### 2.3.1 Kompilasi dan Menjalankan Program Java

Proses kompilasi pada Java menggunakan `javac` untuk mengubah source code menjadi bytecode. Pada proses kompilasi dilakukan pemeriksaan sintaks maupun pustaka-pustaka yang diperlukan apakah sudah tersedia atau tidak. Jika proses kompilasi berhasil, akan dihasilkan file bytecode (file `.class`). File bytecode tersebut dapat dijalankan menggunakan perintah Java. Untuk lebih jelasnya, perhatikan Gambar 4.1.

! Supaya program Java dapat dijalankan harus ada method `main()` di class tersebut. Jika tidak ada method `main()` maka akan muncul pesan kesalahan *"Error: Main method not found in class..."*.



Gambar 2.1: Kompilasi dan Menjalankan Program Java

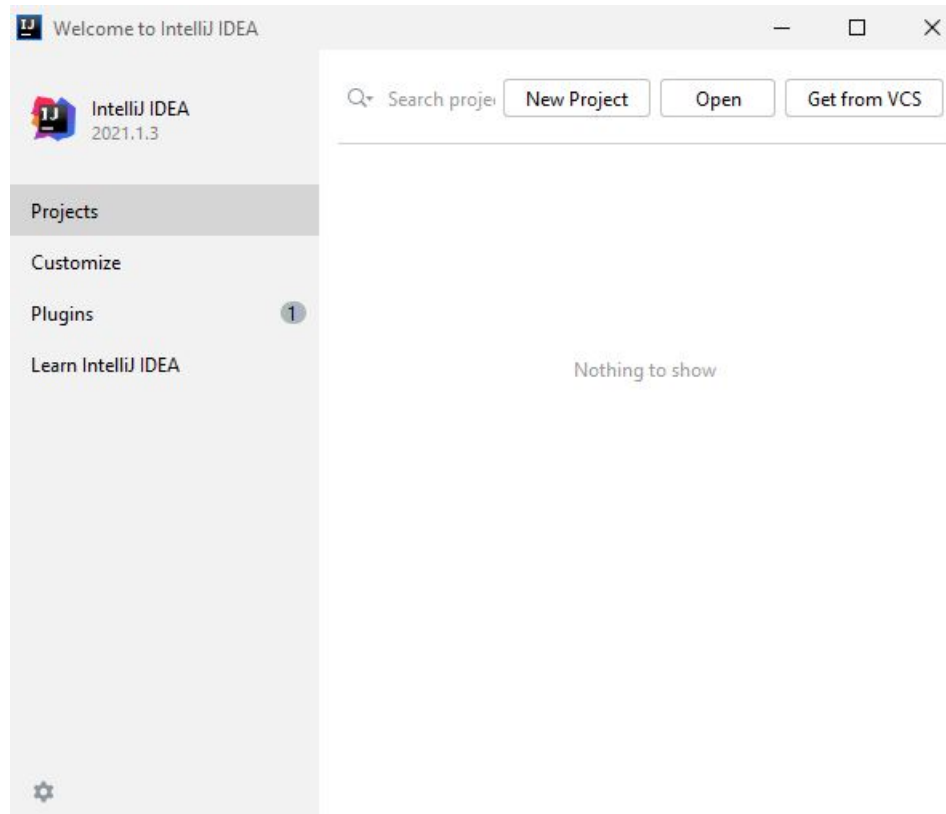
### 2.3.2 IntelliJ IDEA

IntelliJ IDEA adalah salah satu *Integrated Development Environment* (IDE) yang banyak digunakan untuk pengembangan aplikasi dalam bahasa pemrograman Java. IntelliJ IDEA dapat didownload di <https://www.jetbrains.com/idea/> secara gratis atau berbayar. Saat modul ini ditulis, Anda dapat menggunakan IntelliJ IDEA berbayar secara gratis dengan cara melakukan registrasi dan mendaftar untuk mendapatkan lisensi sebagai pelajar atau pengajar di <https://www.jetbrains.com/community/education/#students>. Saat pertama kali IntelliJ dijalankan, memiliki tampilan seperti pada Gambar 4.2.

### 2.3.3 Class pada Java

Java merupakan bahasa pemrograman yang mendukung paradigma Object Oriented Programming (OOP). Beberapa karakteristik yang diperlukan oleh suatu bahasa pemrograman yang mendukung paradigma OOP adalah:

- Semua masalah dilihat dari sudut pandang object.
- Program tersusun dari beberapa object yang saling berinteraksi.



Gambar 2.2: Tampilan Awal IntelliJ IDEA

- Setiap object memiliki alokasi memori yang mungkin tersusun dari object-object lainnya.
- Setiap object memiliki tipe (type).
- Semua object bertipe sama secara umum dapat menerima pesan yang sama.

Class merupakan template/blueprint dari Object. Object diciptakan dari Class, sehingga akan memiliki karakteristik yang sama dengan Class tersebut. Jika ada beberapa Object diciptakan dari Class yang sama, maka Object-object tersebut akan memiliki karakteristik dan perilaku yang sama.

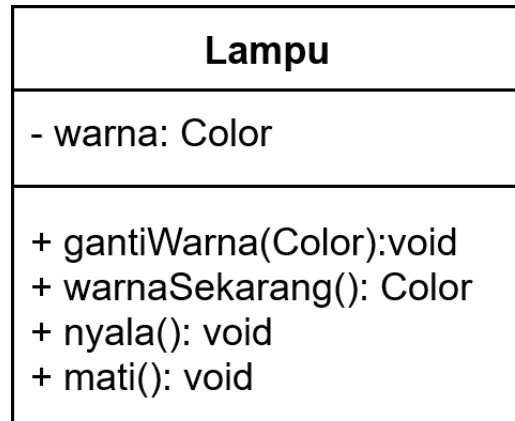
Secara umum, sebuah Class terdiri dari beberapa bagian berikut:

- Atribut/State/Field
- Constructor
- Method/Behavior

Sebagai contoh, perhatikan ilustrasi dari sebuah Class Lampu yang dapat dilihat pada Gambar 2.3. Diagram yang menggambarkan Class Lampu tersebut dinamakan sebagai Class Diagram, yang merupakan bagian dari Diagram UML (Unified Modelling Language). Dari diagram tersebut dapat dilihat bahwa Class tersebut bernama Lampu, memiliki atribut warna dan 4 methods, yaitu **gantiWarna()**, **warnaSekarang()**, **nyala()** dan **mati()**.

Atribut menggambarkan karakteristik, keadaan (state) maupun nilai yang terkait dengan Class tersebut. Pada Class Lampu memiliki atribut warna yang menunjukkan warna lampu tersebut, yang dapat bernilai merah, hijau, biru, kuning maupun warna-warna lainnya. Sedangkan method menunjukkan operasi-operasi yang dapat dilakukan oleh Object yang dibuat dari Class tersebut. Pada Class Lampu, sebuah lampu dapat diganti warnanya (**gantiWarna(Color)**), dapat diambil informasi warna sekarang (**warnaSekarang()**), dapat dinyalakan (**nyala()**) dan dimatikan (**mati()**).

Dari Class Lampu tersebut, kita dapat mendefinisikan beberapa Object yang memiliki karakteristik sesuai dengan yang didefinisikan oleh Class Lampu. Setiap Object yang dibuat dari Class



Gambar 2.3: Class Diagram untuk Class Lampu

Lampu masing-masing memiliki atribut dan method yang sama seperti pada Class Lampu, tetapi nilai dari atribut serta method dapat berbeda antar Object. Sebagai contoh jika dibuat tiga Object dari Class Lampu, bisa saja lampu pertama berwarna merah, lampu kedua berwarna biru dan lampu ketiga berwarna kuning. Saat lampu pertama dimatikan, lampu kedua dan ketiga tidak terpengaruh oleh operasi tersebut.

#### 2.3.4 Tipe Data dan Variabel pada Java

Secara umum tipe data pada Java dibagi menjadi dua macam, yaitu tipe data primitives dan tipe data references. Tipe data primitives pada Java antara lain:

- byte (number, berukuran 1 byte).
- short (number, berukuran 2 bytes).
- int (number, berukuran 4 bytes).
- long (number, berukuran 8 bytes).
- float (floating point number, berukuran 4 bytes).
- double (floating point number, berukuran 8 bytes).
- char (karakter, berukuran 2 bytes).
- boolean (nilai true atau false, berukuran 1 byte).

Tipe data references adalah Object yang dibuat dari Class tertentu. Sedangkan Class ada yang merupakan Class bawaan dari Java dan ada Class yang dibuat oleh anda sendiri atau orang lain (selain bawaan Java).

Java merupakan strong typed language, yang berarti setiap variabel harus didefinisikan terlebih dahulu sebelum dapat digunakan. Untuk mendefinisikan variabel bertipe primitives dapat dilakukan sebagai berikut (contoh):

---

```

1 byte myData = 10;
2 boolean javaIsEasy = true;
3 float myPBOGrade = 87.34;
4 String myname = "Antonius";

```

---

Sedangkan untuk mendefinisikan sebuah object harus melalui tahap instansiasi, baru kemudian object tersebut dapat dioperasikan. Berikut ini adalah contoh pembuatan object dari class Lampu:

---

```

1 Lampu lampuDepan = new Lampu();
2 lampuDepan.nyala();

```

---

---

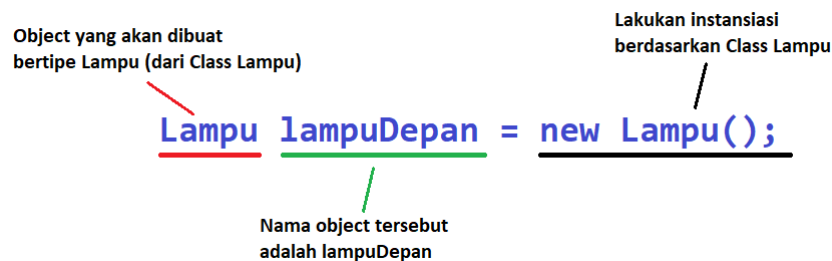
```

3 lampuDepan.mati();
4 Lampu lampuBelakang = new Lampu();
5 lampuBelakang.nyala();

```

---

Contoh tersebut menggambarkan instansiasi object lampuDepan dan lampuBelakang dari class Lampu. Object lampuDepan kemudian dinyalakan dan dimatikan, sehingga kondisi terakhir lampuDepan adalah mati. Pada object lampuBelakang setelah diinstansiasi, kemudian dinyalakan, sehingga kondisi terakhir lampuBelakang adalah menyala. Proses instansiasi dilakukan dengan menggunakan keyword new yang diikuti oleh nama class dari object yang akan dibuat. Untuk lebih jelasnya, perhatikan Gambar 2.4.



Gambar 2.4: Instansiasi Object lampuDepan Berdasarkan Class Lampu

### 2.3.5 Struktur Kontrol Percabangan dan Perulangan pada Java

Bahasa pemrograman Java dikembangkan dengan menggunakan C/C++ sebagai salah satu referensi, sehingga sintaksnya mirip sekali dengan bahasa pemrograman C/C++. Secara umum struktur kontrol percabangan pada Java terdiri dari:

- If – else.
- Switch – case.

Berikut ini adalah contoh penggunaan percabangan if-else pada Java:

---

```

1 int tinggi = 166;
2 bool lulusSeleksiTNI = false;
3 if(tinggi > 160)
4     lulusSeleksiTNI = true;

```

---

Demikian juga dengan struktur kontrol perulangan di Java yang sama persis dengan C/C++, terdiri dari:

- for
- do...while
- while

Berikut ini adalah contoh penggunaan perulangan for pada Java:

---

```

1 for(int i=0; i<10; i++){
2     System.out.println("Perulangan ke - " + i);
3 }

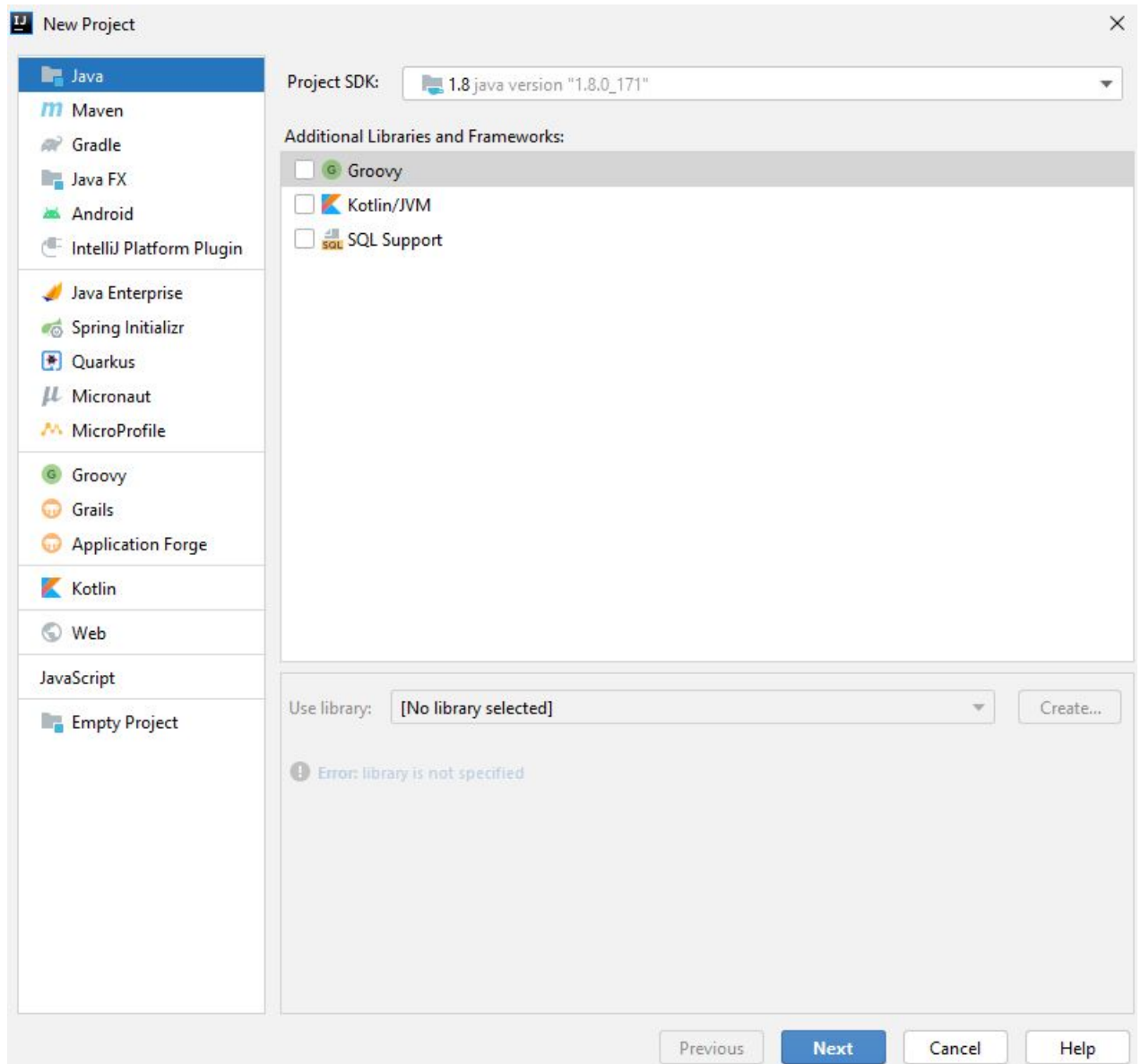
```

---

## 2.4 Kegiatan Praktikum

### 2.4.1 Membuat Program Hello World! Menggunakan IntelliJ

Pada latihan ini anda akan membuat program yang menampilkan tulisan Hello World! menggunakan IntelliJ IDEA. Jalankan aplikasi IntelliJ pada komputer anda masing-masing, kemudian buatlah sebuah project baru dengan mengakses menu **NewProject > Java** (lihat Gambar 2.5).



Gambar 2.5: Membuat New Project pada IntelliJ IDEA

Isikan nama project dengan **HelloWorld**. Kemudian klik tombol **Finish**

Project tersebut masih kosong, anda perlu menambahkan class baru dengan cara klik kanan pada folder **src**, kemudian pilih menu **New > Java Class**.

Isi nama Class dengan **HelloWorld** dan kemudian tekan Enter

Pada Class HelloWorld.java Anda akan mendefinisikan method **main()**, yang merupakan titik awal dari suatu program Java. Method yang pertama kali dijalankan dari project anda adalah method main(). Jika beberapa class memiliki method main(), anda harus menentukan method



main() dari class mana yang akan dijalankan pertama kali.

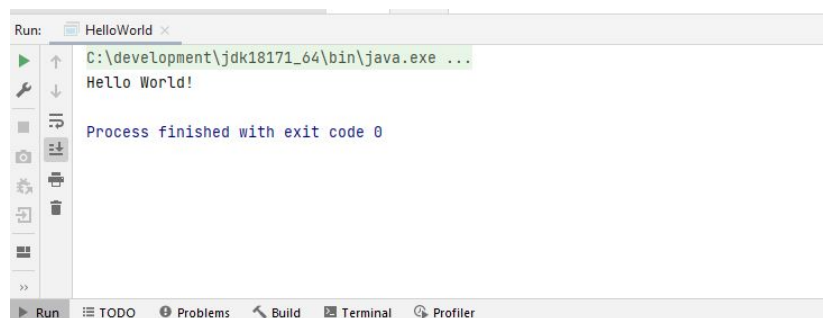
Isikan dengan class HelloWorld dengan kode berikut :

```

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

```

Untuk menjalankan program tersebut klik tombol **Run..** yang ada pada menu **Run > Run..** . Output yang dihasilkan oleh program yang dijalankan dapat dilihat pada bagian Run (Kiri bawah), seperti ditunjukkan pada Gambar 2.6.

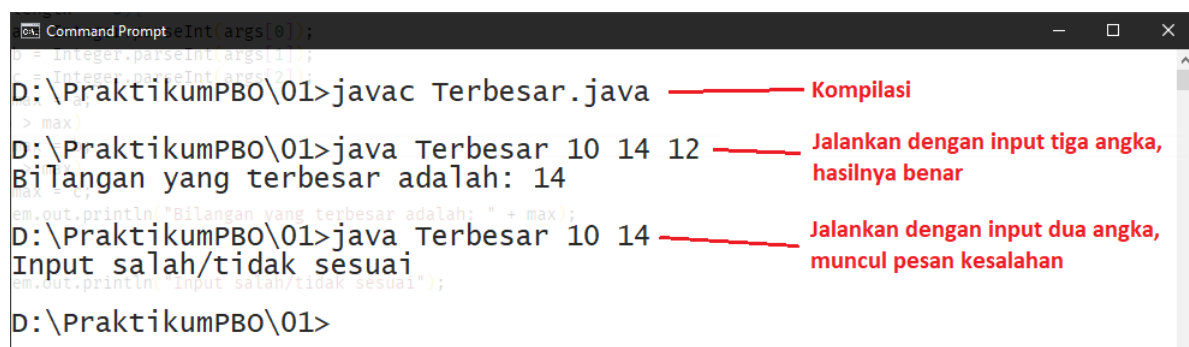


Gambar 2.6: Output dari Program HelloWorld

### 2.4.2 Latihan Terbimbing

#### Kompilasi dan Menjalankan Program Menggunakan Command Prompt/Terminal - 1

Buatlah program yang dapat menentukan bilangan terbesar dari tiga bilangan yang dimasukkan sebagai parameter saat program dijalankan menggunakan key. Contoh tampilan dan hasil dari program tersebut dapat dilihat pada Gambar 2.7.



Gambar 2.7: Menentukan Bilangan Terbesar dari Parameter yang Diberikan

Beberapa hal yang anda perlu ketahui untuk dapat mengerjakan latihan ini:

- Parameter yang dimasukkan saat program dijalankan akan masuk ke dalam variabel args pada method main().
- Variabel args berupa array of String, ukuran dari args tergantung dari jumlah parameter yang diberikan.

- Program ini mengharuskan tiga buah parameter, sehingga lakukan pengecekan apakah variabel args berukuran 3 dengan menggunakan `if(args.length == 3)`. Tampilkan pesan kesalahan jika jumlah parameter tidak sama dengan 3.
- Variabel args adalah array of String, sehingga semua parameter yang dimasukkan akan berupa String. Untuk dapat dibandingkan sebagai angka dalam mencari bilangan terbesar, anda perlu mengubahnya menjadi int menggunakan method `Integer.parseInt()`. Berikut ini adalah contoh pengubahan String menjadi int:

---

```
1      String param1 = "10";
2      int angka = Integer.parseInt(param1);
3      //sekarang angka bernilai 10.
```

---

### Kompilasi dan Menjalankan Program Menggunakan IntelliJ IDEA - 2

Pada latihan ini anda akan membuat sebuah project yang memiliki method `main()` lebih dari satu (terdapat di beberapa Class). Anda akan belajar bagaimana menjalankan method `main()` yang diinginkan dengan menggunakan IntelliJ IDEA.

Buatlah sebuah project baru di Eclipse dengan nama **Terbimbing02**. Tambahkan dua Class ke dalam project tersebut. Beri nama Class tersebut masing-masing: `Output1` dan `Output2`. Untuk setiap Class, pastikan anda menambahkan method `main()` ke dalamnya. Pada class `Output1`, jika dijalankan akan menampilkan tulisan: **"Output ini dari Class Output1"**. Sedangkan class `Output2` jika dijalankan akan menampilkan tulisan **"Output ini dari Class Output2"**.

#### 2.4.3 Latihan Mandiri

Buatlah program melanjutkan dari Latihan Terbimbing No. 1. Kali ini dibuat agar input adalah n bilangan dan program akan menampilkan bilangan terkecilnya. Contoh:

---

```
1      javac Terkecil.java 5 3 10 1 4 2
2      Bilangan terkecil adalah 1
```

---

**SELAMAT MENERJAKAN.**

## 3. Maven

### 3.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Memahami konsep dan dapat menggunakan Maven.
2. Dapat menggunakan Maven untuk membuat project Java sederhana.

### 3.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 4 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 8).
- IntelliJ Ultimate 64 bit (Students Key).

### 3.3 Dasar Teori

#### 3.3.1 Tentang Maven

Maven adalah salah satu build tool yang dibuat oleh Apache yang sering digunakan dalam proyek aplikasi Java. Dengan Maven kita bisa:

- Kompilasi source code dengan mudah
- Melakukan Testing
- Menginstal library yang dibutuhkan

Tujuan dari Maven adalah mempermudah proses pengembangan perangkat lunak sehingga :

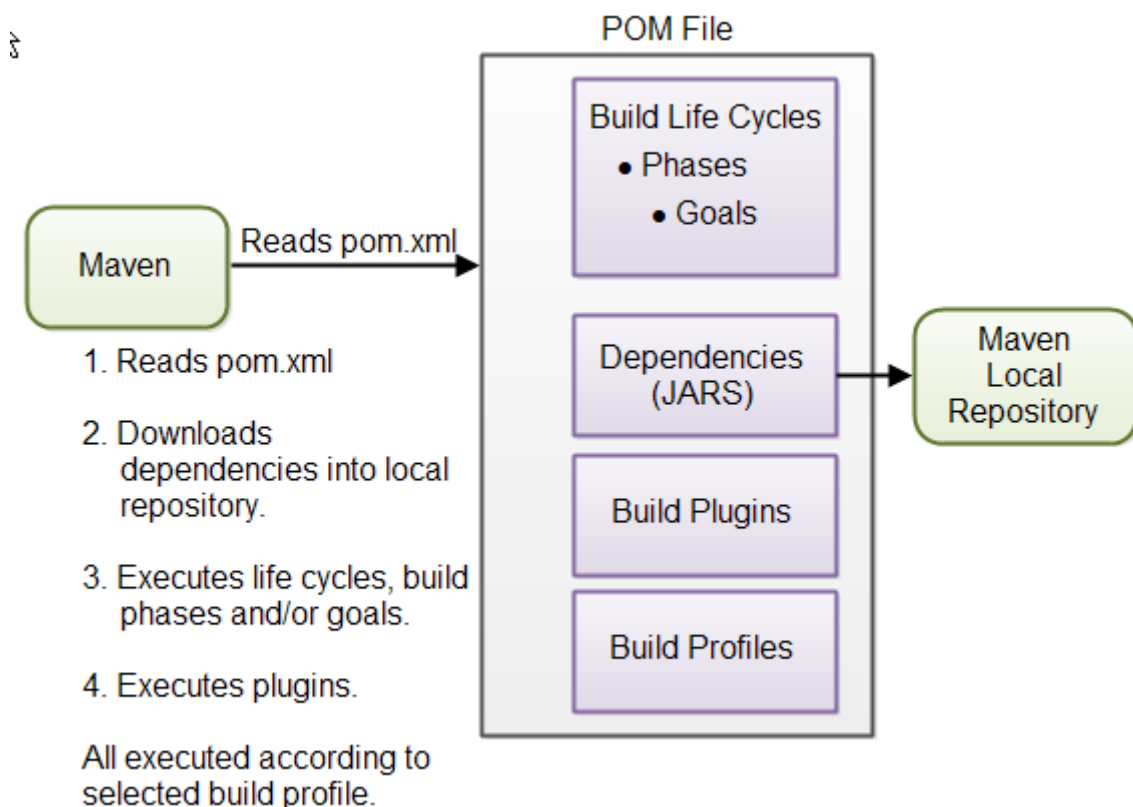
- Mempermudah proses building project
- Menyediakan build system yang seragam

- Menyediakan informasi proyek aplikasi yang berkualitas
- Mendorong proses pengembangan aplikasi menggunakan praktik yang lebih baik.

Selain digunakan untuk Java, Maven juga dapat digunakan untuk mengelola project dengan bahasa lain seperti Kotlin, Ruby, Scala, dan sebagainya. Maven adalah program berbasis teks yang digunakan pada command line. Namun, bisa juga digunakan dalam IDE seperti IntelliJ, Eclipse, dan Netbeans.

### Core Concept

Maven menggunakan file POM (Project Object Model), yang merupakan file utama yang merepresentasikan project resources, seperti source code, test code, dependencies, dll. File POM harus terletak di root directory project. Perhatikan gambar 3.1



Gambar 3.1: Overview konsep utama Maven

### Build Life Cycles, Phases, dan Goals

Process build Maven terbagi menjadi beberapa *life cycle*, *phases*, dan *goals*. Sebuah *build life cycle* terdiri dari sederet *build phases*, dan setiap *build phase* terdiri dari sederet *goals*. Saat perintah maven dijalankan, maka perintah maven akan diteruskan agar dieksekusi. Perintah ini merupakan nama dari *build life cycle*, *phases*, atau *goals*. Jika *life cycle* akan dieksekusi, semua *build phases* dalam *life cycle* tersebut akan dieksekusi.

### Dependencies dan Repositories

Salah satu *goal* yang dieksekusi oleh Maven adalah melakukan pengecekan *dependencies* yang dibutuhkan oleh project. Dependencies adalah file JAR eksternal (java library) yang digunakan oleh project kita. Jika dependencies yang digunakan project tidak ditemukan pada repository lokal maven,

makan maven akan mendownloadnya secara otomatis dari *maven central repository* dan menyimpannya pada local repository. *Maven local repository* adalah sebuah folder khusus pada hardisk kita, yang digunakan untuk menampung semua *dependency* yang digunakan oleh setiap project maven. Kita bisa mengkonfigurasi letak repository lokal maven atau menambah/mengurangi/mengubah Maven central repository.

### Build Plugins

Build plugins digunakan untuk menambahkan *goals* tambahan pada *build phase*. Jika kita ingin menambah set action pada project yang tidak tersedia pada *build phases* dan *goals* standard Maven, kita bisa menambahkannya pada file POM. Maven memiliki plugins standard yang dapat digunakan dan kita dapat menambahkan plugin custom versi kita sendiri.

### Build Profiles

Build profile digunakan jika kita ingin mem-build project kita dengan cara yang berbeda. sebagai contoh, kita ingin mem-build project untuk kebutuhan yang berbeda, yaitu build untuk *development environment* dan *production environment*. Kedua build package ini tentu akan menggunakan konfigurasi yang berbeda. Untuk mempermudah proses build kedua package ini dapat dikonfigurasi melalui file POM. Dan saat mengeksekusi perintah maven, kita dapat menspesifikan build profile mana yang akan digunakan.

## 3.3.2 Instalasi Maven

Untuk melakukan instalasi Maven, dibutuhkan JDK yang sudah diset pada *environment variable*. Maven dapat di download dari situs <https://maven.apache.org/download.html>, dengan format .zip atau .tar.gz. Setelah Maven berhasil didownload, ekstrak lah file zip, struktur folder dari maven dapat dilihat dari gambar 5.2.

Name	Date modified	Type
bin	05/06/2021 4:18	File folder
boot	05/06/2021 4:18	File folder
conf	05/06/2021 4:18	File folder
lib	05/06/2021 4:19	File folder
LICENSE	08/11/2019 2:32	File
NOTICE	08/11/2019 2:32	File
README.txt	08/11/2019 2:32	TXT File

Gambar 3.2: Struktur folder Maven

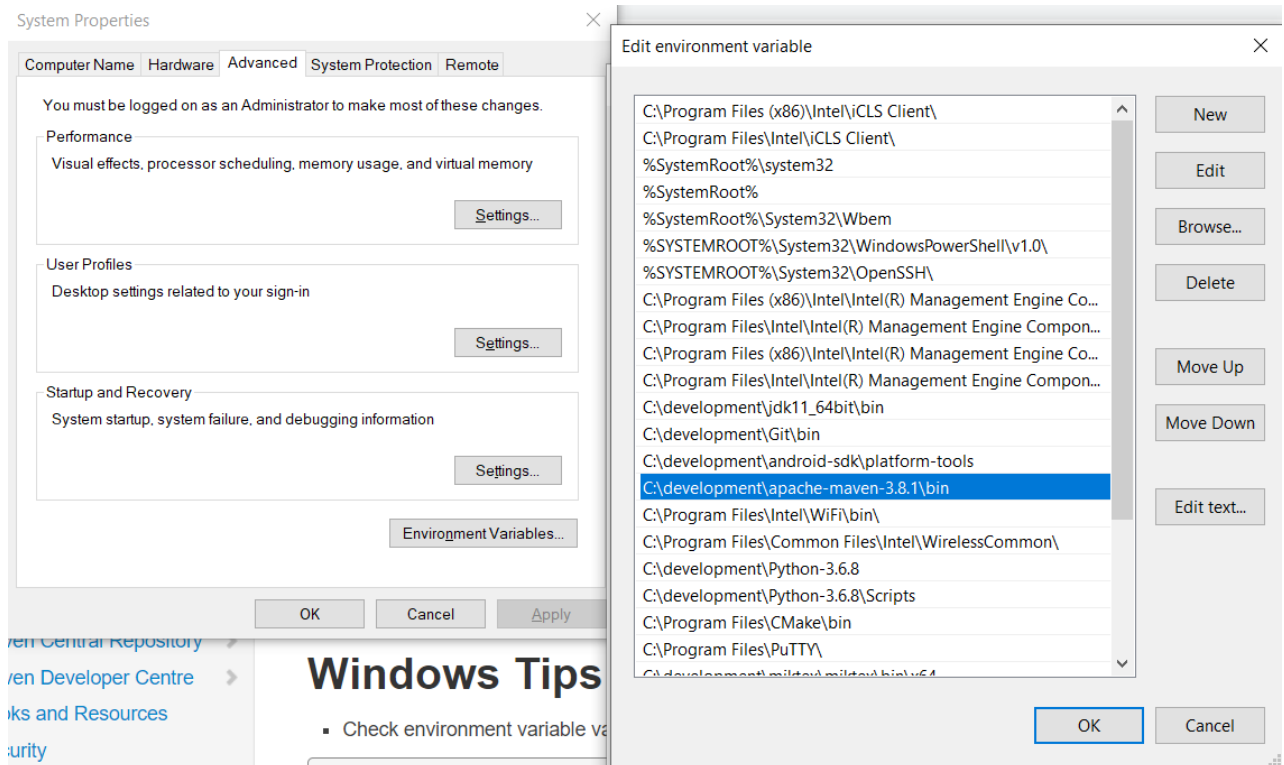
Tambahkan lokasi folder bin Maven kedalam PATH environment variable agar dapat digunakan secara langsung melalui command prompt. Perhatikan gambar 3.3

Lakukan pengecekan dengan cara membuka Command Prompt baru, dan ketikkan perintah

```
1 mvn -v
```

Apabila berhasil, akan menampilkan output sebagai berikut

```
1 Apache Maven 3.8.1 (05c21c65bdfed0f71a2f2ada8b84da59348c4c5d)
2 Maven home: C:\development\apache-maven-3.8.1\bin\..
3 Java version: 11.0.12, vendor: Oracle Corporation, runtime: C:\development\jdk11_64
```



Gambar 3.3: Penambahan maven pada PATH environment variable

```

4      Default locale: en_ID, platform encoding: Cp1252
5      OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

```

### 3.3.3 Membuat Project Java dengan Maven

Ada dua cara yang bisa kita lakukan untuk membuat proyek Java dengan Maven:

- Menggunakan Command Line atau Terminal.
- Menggunakan IDE seperti Netbeans, Eclipse, dan IntelliJ.

#### Membuat Project Maven dengan Command Line

Mari kita coba menggunakan Terminal. Ketikkan perintah berikut ini, dan tekan Enter:

```

1      mvn archetype:generate
2      -DgroupId=com.rplbo.app
3      -DartifactId=rplbo-app
4      -DarchetypeArtifactId=maven-archetype-quickstart
5      -DinteractiveMode=false
6      -DarchetypeVersion=1.4

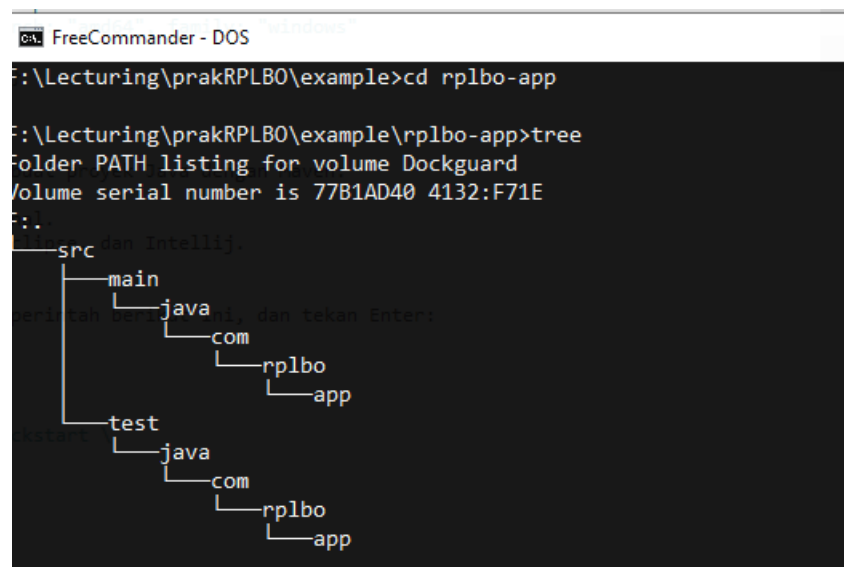
```

Keterangan:

- Argumen `archetype:generate` artinya kita akan membuat sebuah archetype;
- Argumen `-DgroupId=com.rplbo.app` adalah namespace atau nama package dari aplikasi;
- Argumen `-DartifactId=rplbo-app` adalah nama direktori dari aplikasi;
- Argumen `-DarchetypeArtifactId=maven-archetype-quickstart` artifact atau template yang akan digunakan;

- Arugmen -DinteractiveMode=false untuk mematikan mode interaktif.

Setelah itu, coba masuk ke direktori rplbo-app kemudian lihat struktur direktorinya. Hasilnya dapat dilihat pada gambar 3.4



Gambar 3.4: Struktur folder project Maven

### Membuat Project Maven dengan IntelliJ

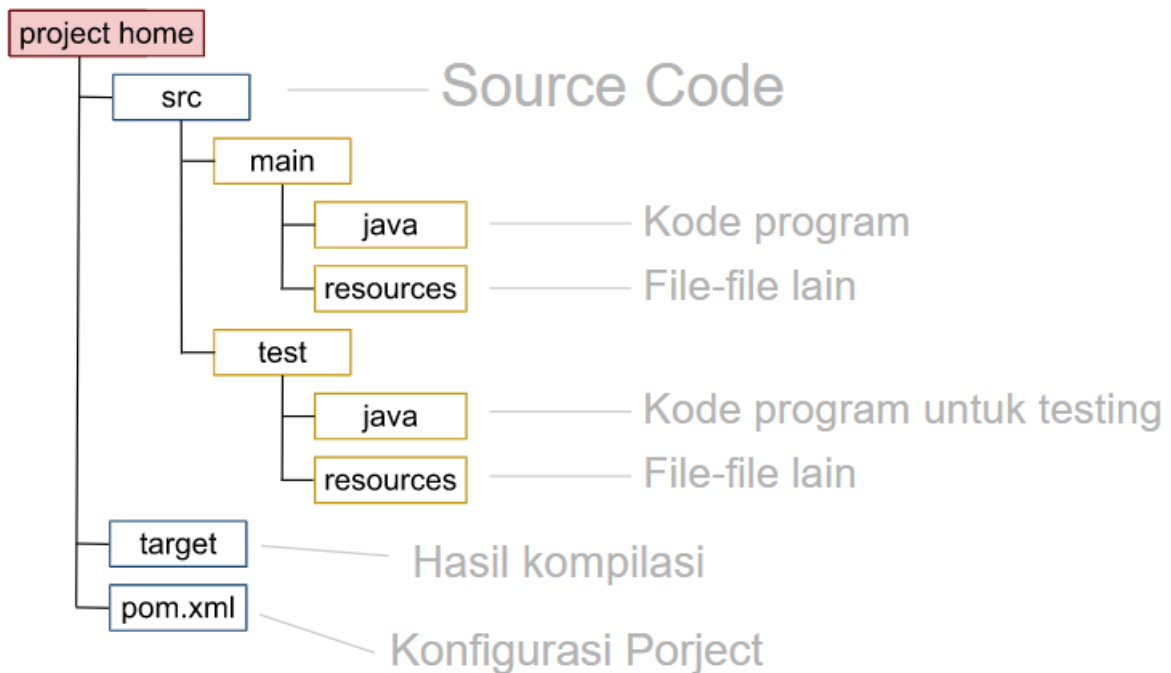
- Klik New Project pada Welcome Screen. Atau tekan File | New | Project dari menu utama.
- Pilih Maven dari option yang terletak sisi kiri
- Pilih JDK yang akan digunakan oleh project. Atau pilih archetype tertentu bila ingin menggunakan template untuk membuat project, lalu tekan Next. NB: Archetype adalah project templating toolkit yang digunakan untuk meng-generate project secara konsisten.
- pada tampilan berikutnya, isikan Maven Coordinate yang akan ditambahkan pada file pom.xml, isian tersebut yaitu :
  - GroupId : package dari project baru
  - artifactId : nama dari project
  - Version : versi dari project baru. Secara default akan diisi secara otomatis
 Apabila semua telah terisi, tekan Next.
- Pada tampilan berikutnya, tentukan nama dan lokasi project. Tekan Finish.

IntelliJ IDEA akan membuat project Maven yang terdapat file pom.xml, yang didalamnya terdiri dari compiler dan versi java yang digunakan, dedicated tool window, dan semua dependencies project yang dibutuhkan untuk bekerja.

#### 3.3.4 Struktur Direktori Project Maven

Gambar 3.5 adalah struktur direktori yang akan dibuatkan oleh Maven:

Terdapat beberapa file dan direktori yang harus dipahami. Berikut ini penjelasannya:



Gambar 3.5: Struktur direktori maven

Nama File atau Direktori	keterangan
project home	berisi file pom.xml dan semua direktori
src/main/java	berisi kode program
src/main/resources	berisi file pendukung untuk project seperti gambar, dokumen, dll.
src/test/java	berisi kode program untuk melakukan testing
src/test/resources	berisi file pendukung untuk melakukan testing
target	berisi file hasil kompilasi seperti .jar dan .class
pom.xml	berisi konfigurasi project

### 3.3.5 Mengenal file pom.xml

POM adalah singkatan dari Project Object Model. File pom.xml merupakan file XML yang berisi konfigurasi dari project. Perhatikan contoh berikut.

```

1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4     http://maven.apache.org/maven-v4_0_0.xsd">
5     <modelVersion>4.0.0</modelVersion>
6     <groupId>com.rplbo.app</groupId>
7     <artifactId>rplbo-app</artifactId>
8     <packaging>jar</packaging>
9     <version>1.0-SNAPSHOT</version>
10    <properties>
11        <maven.compiler.source>1.7</maven.compiler.source>
12        <maven.compiler.target>1.7</maven.compiler.target>
13    </properties>

```



```

14     <name>rplbo-app</name>
15     <url>http://maven.apache.org</url>
16     <dependencies>
17         <dependency>
18             <groupId>junit</groupId>
19             <artifactId>junit</artifactId>
20             <version>4.11</version>
21             <scope>test</scope>
22         </dependency>
23     </dependencies>
24 </project>

```

Bagian penting yang harus kita pahami adalah di bagian `<dependencies>`. Bagian ini tempat kita menentukan library apa yang akan kita gunakan dalam project. Contohnya: di sini sudah ada library junit untuk melakukan unit testing.

```

1 <dependencies>
2     <dependency>
3         <groupId>junit</groupId>
4         <artifactId>junit</artifactId>
5         <version>4.11</version>
6         <scope>test</scope>
7     </dependency>
8 </dependencies>

```

### 3.3.6 Kompilasi Program dengan Maven

Kita dapat melakukan kompilasi aplikasi dengan perintah:

```

1 mvn package

```

Perintah ini akan membuat direktori baru bernama target yang berisi file hasil kompilasi berupa .jar dan .class. Hasil eksekusi dapat dilihat pada gambar 3.6.

Berikut ini adalah perintah-perintah standard yang langsung dapat digunakan pada maven.

Nama perintah	keterangan
mvn clean	menghapus directory target, tempat dimana maven melakukan builds project
mvn validate	memvalidasi apakah project sudah dalam format yang benar dan semua informasi telah tersedia. Perintah ini akan memastikan semua dependency terdownload.
mvn compile	mengcompile source code project
mvn test	mengeksekusi test code menggunakan framework unit testing yang digunakan.
mvn package	packing semua kode yang sudah dcompile kedalam format JAR atau WAR
mvn verify	mengeksekusi semua integration test yang ada pada project
mvn install	mem-build semua project sesuai konfigurasi yg ada pada POM.xml dan meng-install hasilnya pada local repository maven.
mvn site	mengeksekusi build life cycles site
mvn deploy	meng-copy package akhir ke remote repository agar dapat disharingkan

Untuk menjalankan program JAR gunakan perintah:

```

F:\Lecturing\prakRPLB0\example\rplbo-app>mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.rplbo.app:rplbo-app >-----
[INFO] Building rplbo-app 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ rplbo-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory F:\Lecturing\prakRPLB0\example\rplbo-app\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ rplbo-app ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ rplbo-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory F:\Lecturing\prakRPLB0\example\rplbo-app\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ rplbo-app ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to F:\Lecturing\prakRPLB0\example\rplbo-app\target\test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ rplbo-app ---
[INFO] Surefire report directory: F:\Lecturing\prakRPLB0\example\rplbo-app\target\surefire-reports

-----
T E S T S
-----
Running com.rplbo.app.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ rplbo-app ---
[INFO] Building jar: F:\Lecturing\prakRPLB0\example\rplbo-app\target\rplbo-app-1.0-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 4.895 s
[INFO] Finished at: 2022-01-31T16:24:48+07:00
[INFO]

```

Gambar 3.6: Hasil perintah mvn package

---

```
1 java -cp target/rplbo-app-1.0-SNAPSHOT.jar com.rplbo.app.App
```

---

atau

---

```
1 java -jar target/rplbo-app-1.0-SNAPSHOT.jar
```

---

Jika sudah memiliki MANIFEST dan main classnya.

### 3.4 Kegiatan Praktikum

#### 3.4.1 Membuat dan mem-build project Java dengan Maven

1. Buatlah project baru menggunakan maven dengan cara memasukan semua properti secara langsung melalui command prompt. Adapun parameter yang harus diinputkan sebagai berikut :
  - (a) groupId=com.<NIM>.app
  - (b) artifactId=<NIM>-app
  - (c) archetypeArtifactId=maven-archetype-quickstart
  - (d) interactiveMode=false
2. Tambahkan Class berikut ini ke project Anda di

src\main\java\com\<NIM>\app

---

```

1  import java.util.Scanner;
2
3  public class PerulanganWhile {
4      public static void main(String[] args) {
5
6          // membuat variabel dan scanner
7          boolean running = true;
8          int counter = 0;
9          String jawab;
10         Scanner scan = new Scanner(System.in);
11
12         while( running ) {
13             System.out.println("Apakah anda ingin keluar?");
14             System.out.print("Jawab [ya/tidak]> ");
15
16             jawab = scan.nextLine();
17
18             // cek jawabannya, kalau ya maka berhenti mengulang
19             if(jawab.equalsIgnoreCase("ya")){
20                 running = false;
21             }
22
23             counter++;
24         }
25
26         System.out.println("Anda sudah melakukan perulangan sebanyak "
27             + counter + " kali");
28     }
29 }

```

---

3. Lakukan compile menggunakan Maven
4. tambahkan kode berikut ini pada POM.xml

---

```

1  <project>
2  ...
3  <build>

```

```
4     <plugins>
5         <plugin>
6             <groupId>org.apache.maven.plugins</groupId>
7             <artifactId>maven-jar-plugin</artifactId>
8             <configuration>
9                 <archive>
10                     <manifest>
11                         <addClasspath>true</addClasspath>
12                         <mainClass>com.<NIM>.app.PerulanganWhile</mainClass>
13                     </manifest>
14                 </archive>
15             </configuration>
16         </plugin>
17     </plugins>
18 </build>
19 ...
20 </project>
```

5. Buat JAR menggunakan Maven (compile dan package), dan eksekusi jar tersebut dengan perintah **java -jar target/<nama file JAR>**

## 4. Class dan Object

### 4.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Memahami dan dapat menjelaskan perbedaan antara Class dan Object.
2. Dapat mendefinisikan Class, menambahkan atribut dan method serta melakukan instansiasi Object dari Class yang dibuat.
3. Dapat menampilkan output di console/terminal dan menerima masukan dari keyboard.

### 4.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 8 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 17).
- IntelliJ Ultimate 64 bit (Students Key).

### 4.3 Dasar Teori

#### 4.3.1 Class dan Object

Java merupakan bahasa pemrograman yang mendukung paradigma Object Oriented Programming (OOP). Beberapa karakteristik yang diperlukan oleh suatu bahasa pemrograman yang mendukung paradigma OOP adalah:

- Semuanya adalah object.
- Program tersusun dari beberapa object yang saling berinteraksi.
- Setiap object memiliki alokasi memori yang mungkin tersusun dari object-object lainnya.

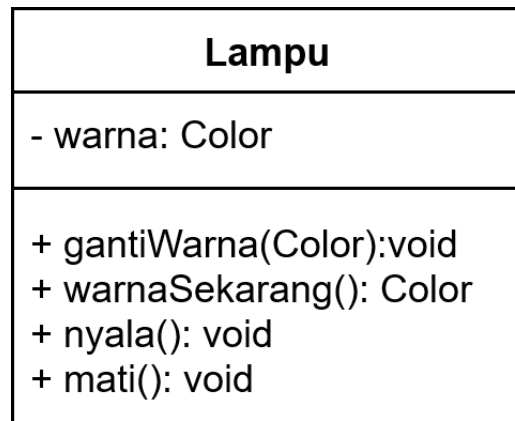
- Setiap object memiliki tipe (type).
- Semua object bertipe sama secara umum dapat menerima pesan yang sama.

Class merupakan template/blueprint dari Object. Object diciptakan dari Class, sehingga akan memiliki karakteristik yang sama dengan Class tersebut. Jika ada beberapa Object diciptakan dari Class yang sama, maka Object-object tersebut akan memiliki karakteristik dan perilaku yang sama. Secara umum, sebuah Class terdiri dari beberapa bagian berikut:

- Atribut/State.
- Constructor.
- Method/Behavior.

! Class perlu dibuat dulu, kemudian baru dapat membuat Object berdasarkan Class tersebut.

Sebagai contoh, perhatikan ilustrasi dari sebuah Class Lampu yang dapat dilihat pada Gambar 4.1. Diagram yang menggambarkan Class Lampu tersebut dinamakan sebagai Class Diagram, yang merupakan bagian dari Diagram UML (Unified Modelling Language). Dari diagram tersebut dapat dilihat bahwa Class tersebut bernama Lampu, memiliki atribut warna dan 4 methods, yaitu **gantiWarna()**, **warnaSekarang()**, **nyala()** dan **mati()**.



Gambar 4.1: Class Diagram untuk Class Lampu

Atribut menggambarkan karakteristik, keadaan (state) maupun nilai yang terkait dengan Class tersebut. Pada Class Lampu memiliki atribut warna yang menunjukkan warna lampu tersebut, yang dapat bernilai merah, hijau, biru, kuning maupun warna-warna lainnya. Sedangkan method menunjukkan operasi-operasi yang dapat dilakukan oleh Object yang dibuat dari Class tersebut. Pada Class Lampu, sebuah lampu dapat diganti warnanya (**gantiWarna(Color)**), dapat diambil informasi warna sekarang (**warnaSekarang()**), dapat dinyalakan (**nyala()**) dan dimatikan (**mati()**).

Dari Class Lampu tersebut, kita dapat mendefinisikan beberapa Object yang memiliki karakteristik sesuai dengan yang didefinisikan oleh Class Lampu. Setiap Object yang dibuat dari Class Lampu masing-masing memiliki atribut dan method yang sama seperti pada Class Lampu, tetapi nilai dari atribut serta method dapat berbeda antar Object. Sebagai contoh jika dibuat tiga Object dari Class Lampu, bisa saja lampu pertama berwarna merah, lampu kedua berwarna biru dan lampu ketiga berwarna kuning. Saat lampu pertama dimatikan, lampu kedua dan ketiga tidak terpengaruh oleh operasi tersebut.

! Dari satu Class, dapat dibuat banyak Object. Masing-masing Object tersebut memiliki state dan behavior seperti yang telah didefinisikan di Class. Nilai dari state setiap Object dapat berbeda-beda dan tidak saling mempengaruhi satu sama lain (independen).

### 4.3.2 Tambahan: Struktur Kontrol pada Java

Contoh program operator bitwise pada Java.

---

```

1      public class OperatorBitwise {
2          public static void main(String[] args) {
3              int a = 60;    /* 60 = 0011 1100 */
4              int b = 13;    /* 13 = 0000 1101 */
5              int c = 0;
6
7              c = a & b;      /* 12 = 0000 1100 */
8              System.out.println("a & b = " + c);
9
10             c = a | b;      /* 61 = 0011 1101 */
11             System.out.println("a | b = " + c);
12
13             c = a ^ b;      /* 49 = 0011 0001 */
14             System.out.println("a ^ b = " + c);
15
16             c = ~a;         /* -61 = 1100 0011 */
17             System.out.println("~a = " + c);
18
19             c = a << 2;     /* 240 = 1111 0000 */
20             System.out.println("a << 2 = " + c);
21
22             c = a >> 2;     /* 215 = 1111 */
23             System.out.println("a >> 2 = " + c);
24
25             c = a >>> 2;    /* 215 = 0000 1111 */
26             System.out.println("a >>> 2 = " + c);
27
28             boolean suka = true;
29             String jawaban;
30
31             // menggunakan operator ternary
32             jawaban = suka ? "iya" : "tidak";
33
34             // menampilkan jawaban
35             System.out.println(jawaban);
36         }
37     }

```

---

Contoh program percabangan pada Java.

---

```

1      import java.util.Scanner;
2
3      public class HitungGrade {
4          public static void main(String[] args) {
5              // membuat variabel dan scanner
6              int nilai;

```

```

7         String grade;
8         Scanner scan = new Scanner(System.in);
9
10        // mengambil input
11        System.out.print("Inputkan nilai: ");
12        nilai = scan.nextInt();
13
14        // higung gradenya
15        if ( nilai >= 90 ) {
16            grade = "A";
17        } else if ( nilai >= 80 ){
18            grade = "B+";
19        } else if ( nilai >= 70 ){
20            grade = "B";
21        } else if ( nilai >= 60 ){
22            grade = "C+";
23        } else if ( nilai >= 50 ){
24            grade = "C";
25        } else if ( nilai >= 40 ){
26            grade = "D";
27        } else {
28            grade = "E";
29        }
30
31        // cetak hasilnya
32        System.out.println("Grade: " + grade);
33    }
34 }

```

Contoh program percabangan pada Java (2).

```

1     import java.util.Scanner;
2
3     public class LampuLalulintas {
4         public static void main(String[] args) {
5             // membuat variabel dan Scanner
6             String lampu;
7             Scanner scan = new Scanner(System.in);
8
9             // mengambil input
10            System.out.print("Inputkan nama warna: ");
11            lampu = scan.nextLine();
12
13            switch(lampu){
14                case "merah":
15                    System.out.println("Lampu merah, berhenti!");
16                    break;
17                case "kuning":
18                    System.out.println("Lampu kuning, harap hati-hati!");

```



```

19         break;
20         case "hijau":
21             System.out.println("Lampu hijau, silahkan jalan!");
22             break;
23         default:
24             System.out.println("Warna lampu salah!");
25     }
26 }
27 }

```

Contoh program perulangan pada Java.

```

1     import java.util.Scanner;
2
3     public class PerulanganWhile {
4         public static void main(String[] args) {
5             // membuat variabel dan scanner
6             boolean running = true;
7             int counter = 0;
8             String jawab;
9             Scanner scan = new Scanner(System.in);
10
11             while( running ) {
12                 System.out.println("Apakah anda ingin keluar?");
13                 System.out.print("Jawab [ya/tidak]> ");
14
15                 jawab = scan.nextLine();
16
17                 // cek jawabannya, kalau ya maka berhenti mengulang
18                 if( jawab.equalsIgnoreCase("ya") ){
19                     running = false;
20                 }
21
22                 counter++;
23             }
24
25             System.out.println("Anda sudah melakukan perulangan sebanyak " + counter);
26         }
27     }

```

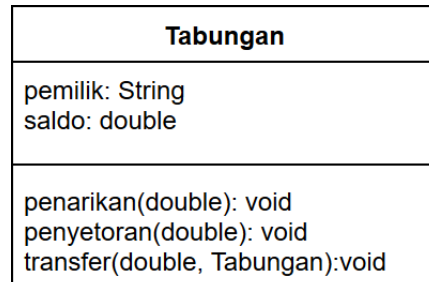
## 4.4 Kegiatan Praktikum

### 4.4.1 Mendefinisikan Class Tabungan

Pada latihan ini anda diminta untuk membuat project dengan Eclipse. Aplikasi yang akan dibuat adalah simulasi rekening tabungan yang dapat melakukan operasi penarikan, penyetoran dan transfer. Secara umum, Class diagram dari Class Tabungan tersebut dapat dilihat pada Gambar 1.

Beberapa hal yang perlu diperhatikan dalam implementasi yang anda buat:

- Saldo tabungan tidak boleh negatif.



Gambar 4.2: Class Diagram untuk Class Tabungan

- Operasi penarikan tidak boleh menyebabkan saldo menjadi negatif. Jumlah uang yang akan diambil juga tidak boleh negatif.
- Operasi penyetoran tidak boleh menyebabkan saldo menjadi negatif atau berkurang. Jumlah yang akan disetorkan tidak boleh negatif.

Jalankan Eclipse, kemudian buatlah project baru (**File > New > Project**), isikan **Rekening** sebagai nama project tersebut. Kemudian buatlah sebuah Class baru dengan nama **Tabungan**. Ubahlah isi class Tabungan sehingga menjadi seperti berikut:

```

1 public class Tabungan {
2     String pemilik;
3     double saldo;
4
5     String getPemilik() { return pemilik; }
6
7     double getSaldo() { return saldo; }
8
9     void penarikan(double jumlah){
10         if(jumlah > 0 && saldo >= jumlah){
11             saldo = saldo - jumlah;
12         }
13     }
14
15     void penyetoran(double jumlah){
16         if(jumlah > 0)
17             saldo = saldo + jumlah;
18     }
19 }

```

Tambahkan sebuah Class baru ke dalam project dengan nama **Main**. Kemudian buatlah implementasi dari method main() di dalam Class Main seperti berikut:

```

1 public class Main {
2     public static void main(String[] args){
3         Tabungan t1 = new Tabungan();
4         t1.saldo = 1000;
5         t1.pemilik = "Antonius RC";
6     }

```

```
7      //tampilkan informasi tabungan
8      System.out.println("Tabungan: " + t1.pemilik);
9      System.out.println("Saldo: " + t1.saldo);
10
11     //lakukan operasi penyetoran sebesar 250
12     t1.penyetoran(250);
13
14     //tampilkan saldo setelah penyetoran
15     System.out.println("Saldo: " + t1.saldo);
16
17     //lakukan operasi penarikan sebesar 450
18     t1.penarikan(450);
19
20     //tampilkan saldo setelah penarikan
21     System.out.println("Saldo: " + t1.saldo);
22 }
23 }
```

Jika implementasi anda benar, seharusnya dihasilkan output sebagai berikut:

```
Tabungan: Antonius RC
Saldo: 10000
Saldo: 1250
Saldo: 800
```

## 4.5 Latihan Mandiri

### 4.5.1 Pengecekan Method penarikan() dan penyetoran()

Pada class Main, tambahkan pengecekan implementasi method penarikan() dan penyetoran dengan menggunakan urutan proses sebagai berikut:

1. Buat object A dari class Tabungan. Ubah saldo A menjadi 300 dan nama pemilik adalah "A".
2. Buat object B dari class Tabungan. Ubah saldo B menjadi 1200 dan nama pemilik adalah "B".
3. Tampilkan informasi dari A dan B (nama pemilik dan saldonya)
4. Lakukan operasi penarikan sejumlah 150 dari rekening A.
5. Lakukan operasi penyetoran sejumlah 300 ke rekening B.
6. Lakukan operasi penarikan sejumlah -500 dari rekening B.
7. Lakukan operasi penyetoran sejumlah -1000 ke rekening A.
8. Tampilkan informasi dari A dan B (nama pemilik dan saldonya).
9. **Saldo A di akhir seharusnya 150 dan saldo B seharusnya 1500.**

### 4.5.2 Implementasi Method transfer()

Buatlah implementasi dari method transfer(), kemudian lakukan pengujian dengan urutan sebagai berikut:

1. Buat object A, B, dan C dari class Tabungan.
2. Ubah saldo A: 1200, B: 3000 dan C: 1500.
3. A transfer 200 ke B.
4. B transfer 2400 ke C.
5. C transfer 1100 ke A.

6. Tampilkan saldo A, B dan C (**seharusnya A: 2100, B: 800 dan C: 2800**)
7. B transfer 1000 ke C.
8. C transfer -2000 ke A
9. Tampilkan saldo A, B dan C (**seharusnya A: 2100, B: 800 dan C: 2800**)
10. A transfer 2000 ke A
11. Tampilkan saldo A (**seharusnya tetap 2100**)

**SELAMAT MENGERJAKAN.**

## 5. Encapsulation, Static Field dan Static Method

### 5.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Memahami dan dapat mengimplementasikan konsep Encapsulation.
2. Memahami dan dapat menggunakan access specifier, setter dan getter.
3. Memahami dan dapat menggunakan static field dan static method.

### 5.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 8 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 17).
- IntelliJ Ultimate 64 bit (Students Key).

### 5.3 Dasar Teori

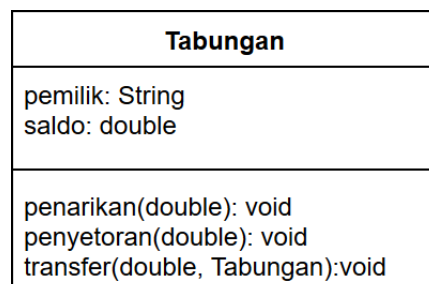
#### 5.3.1 Encapsulation

**Encapsulation** merupakan salah satu bagian utama dari konsep pemrograman berorientasi objek, selain **Inheritance**, **Abstraction** dan **Polymorphism**. Secara umum encapsulation adalah mekanisme untuk mengimplementasikan prinsip **Information Hiding**. Information Hiding berarti melindungi integritas data yang dimiliki dengan cara membatasi akses atau bahkan dengan cara menyembunyikan data sehingga tidak dapat diakses begitu saja. Pada pemrograman berorientasi objek, encapsulation berupa pembatasan akses data tetapi tetap menyediakan mekanisme yang

terkontrol untuk mengakses data yang dianggap penting dengan tujuan untuk menjaga integritas data.

Sebagai contoh, saldo adalah informasi penting yang ada pada suatu tabungan. Saldo tidak boleh diubah dengan sembarang. Saldo hanya boleh berupa melalui proses-proses yang telah disediakan, antara lain dengan melakukan operasi **penyetoran**, **penarikan** maupun **transfer**. Operasi-operasi tersebut dapat mengubah saldo dengan lebih terkontrol. Misalnya pada operasi penyetoran, seharusnya menyebabkan jumlah saldo bertambah. Karena itu perlu dipastikan jumlah uang yang disetorkan bernilai positif, bukan negatif.

Perhatikan Gambar 5.1 yang merupakan Class diagram untuk class Tabungan, seperti yang telah dibahas pada pertemuan sebelumnya.



Gambar 5.1: Class Diagram untuk Class Tabungan

Pada class Tabungan terdapat informasi pemilik dan jumlah saldo dari tabungan tersebut. Selain itu juga disediakan 3 methods yang dapat diakses untuk mengubah saldo, yaitu penarikan(), penyetoran() dan transfer(). Perhatikan potongan kode program berikut ini:

---

```

1 Tabungan t1 = new Tabungan();
2 t1.saldo = 1000;
3 t1.pemilik = "Anton";
4 t1.penarikan(2000);

```

---

Pada potongan kode program tersebut, dilakukan instansiasi object t1 dari class Tabungan (baris 1). Kemudian field saldo diubah menjadi 1000 (baris 2) dan pemilik menjadi "Anton" (baris 3). Selanjutnya dilakukan operasi penarikan sejumlah 2000 (baris 4). Operasi penarikan ini tidak valid karena jumlah saldo yang dimiliki 1000, tetapi ingin diambil 4000. Pada method penarikan() telah dibuat implementasi pengecekan dan validasi operasi penarikan, sehingga nilai saldo dapat selalu dijaga dengan baik jika dipanggil operasi penarikan.

Ada satu hal yang perlu diperhatikan, kode pada baris 2 bertujuan untuk mengisi nilai awal saldo dengan nilai 1000. Akan tetapi hal tersebut bisa disalahgunakan dengan cara sebagai berikut:

---

```

1 Tabungan t1 = new Tabungan();
2 t1.saldo = -1000;

```

---

Pada potongan kode program tersebut, saldo dapat dengan mudah diubah menjadi negatif. Hal ini tentu saja bertentangan dengan tujuan yang ingin dicapai, yaitu nilai saldo yang selalu dijaga integritas dan validitasnya. Field saldo perlu diproteksi sehingga tidak dapat diubah secara langsung. Proteksi tersebut diimplementasikan dalam bentuk **access specifier**.

### 5.3.2 Access Specifier, Setter dan Getter

Access specifier pada Java merupakan mekanisme untuk menentukan apakah suatu field atau method dapat diakses oleh class tertentu berdasarkan posisi/letaknya. Terdapat empat macam access specifier pada Java, yaitu:

- **Private.** Field atau method yang ditandai dengan access specifier private hanya dapat diakses dari dalam class itu sendiri.
- **Package (default).** Field atau method yang ditandai dengan access specifier package/default hanya dapat diakses dari dalam class itu sendiri dan dari class-class lain yang berada dalam package yang sama.
- **Protected.** Field atau method yang ditandai dengan access specifier protected sama seperti pada package, tetapi dengan tambahan dapat diakses oleh class-class turunannya.
- **Public.** Field atau method yang ditandai dengan access specifier public dapat diakses dari mana saja (tidak ada batasan).

Pada implementasi class Tabungan, lebih baik jika seluruh field yang dimiliki diberi access specifier private sehingga tidak dapat diakses secara langsung. Kemudian seluruh method yang ada diberi access specifier public, sehingga dapat diakses secara langsung dari mana saja. Perubahan tersebut dapat dilihat pada Gambar 5.2.

Tabungan
- pemilik: String - saldo: double
+ penarikan(double): void + penyetoran(double): void + transfer(double, Tabungan): void

Gambar 5.2: Class Tabungan Setelah Diberi Access Specifier

Getter (atau accessor) merupakan method di dalam suatu class yang dibuat dengan tujuan untuk mengakses suatu field tertentu yang telah diberi access specifier private. Getter umumnya berupa method yang diberi nama "get" + nama field yang ingin diberi akses. Setter (atau mutator) merupakan method di dalam suatu class yang dibuat dengan tujuan untuk mengubah nilai suatu field yang telah diberi access specifier private. Setter umumnya berupa method dengan nama "set" + nama field yang ingin diubah. Pada class Tabungan, terdapat dua field yang diproteksi private, yaitu saldo dan pemilik. Informasi saldo dan pemilik suatu tabungan umumnya diperlukan, sehingga perlu ditambahkan getter untuk field saldo dan pemilik.

Potongan kode program berikut ini menunjukkan getter untuk field saldo dan pemilik pada class Tabungan:

```

1 public class Tabungan {
2     private String pemilik;
3     private double saldo;
4     public String getPemilik(){
5         return pemilik;
6     }
7     public double getSaldo(){
8         return saldo;
9     }

```

```

10 .....//bagian ini tidak ditampilkan
11 }

```

Class Tabungan tidak membutuhkan setter untuk field saldo dan pemilik. Field pemilik tidak boleh diubah, sedangkan field saldo hanya dapat diubah melalui proses-proses yang telah disediakan sebelumnya, yaitu penarikan, penyetoran dan transfer.

! Tidak semua field yang ada perlu dibuatkan getter dan setternya.

### 5.3.3 Instance Level dan Class Level

Pada setiap object yang dibuat berdasarkan class Tabungan akan memiliki field saldo dan pemilik. Setiap object tersebut dapat memiliki nilai saldo dan pemilik yang berbeda-beda. Perubahan nilai saldo pada suatu object tidak akan mempengaruhi nilai saldo pada object-object lainnya. Field saldo disini merupakan field pada instance level. Demikian juga pada method penarikan() yang akan mengubah saldo pada suatu object tertentu. Method penarikan() merupakan method pada instance level.

Field pada class level dapat ditemui di seluruh object dan nilainya akan selalu sama. Perubahan nilai field pada class level di suatu object akan secara otomatis mengubah nilai tersebut di seluruh object yang dibuat dari class yang sama. Selain itu field pada class level dapat diakses secara langsung melalui class-nya.

Method yang mengakses field pada class level juga harus disesuaikan. Pada Java keyword untuk membuat field pada class level adalah **static**. Dengan kata kunci static maka semua field yang dibuat static akan dikenal diseluruh program (pada JVM). Field pada class level boros memory karena memiliki sifat seperti variabel global. Contoh penggunaan:

```

1 private static int counter;
2
3 public static void setCounter(int c){ ... }
4 public static int getCounter(){ ... }

```

Misalnya pada class Tabungan ingin ditambahkan suatu fitur yaitu nomor rekening, dengan ketentuan sebagai berikut:

- No. rekening bersifat auto-increment setiap ada rekening baru.
- No. rekening tidak boleh berubah. Untuk memenuhi kebutuhan baru tersebut, berikut ini beberapa hal yang perlu dipertimbangkan sebelum membuat implementasinya:
- No rekening harus diberi access specifier private, sehingga tidak bisa diubah langsung.
- Kemudian sediakan getter sehingga nomor rekening tetap dapat diakses jika diperlukan.
- Perlu ada pencatatan sudah sampai nomor berapa rekening-rekening sebelumnya, sehingga bisa menentukan nomor rekening yang baru.

Implementasi yang akan dibuat untuk memenuhi kebutuhan baru tersebut dapat dilihat pada Gambar 5.3.

## 5.4 Kegiatan Praktikum

Pada bagian ini anda akan membuat implementasi dari Class Tabungan berdasarkan class diagram pada Gambar 5.3. Buatlah project baru pada Eclipse, kemudian buat Class Tabungan dengan isi sebagai berikut:



Tabungan
- pemilik: String - saldo: double - noRekening: int - nextId: int
+ Tabungan(String, double) + getPemilik(): String + getSaldo(): double + getNoRekening(): int + penarikan(double): void + penyetoran(double): void + transfer(double, Tabungan): void

Gambar 5.3: Class Tabungan Dengan Tambahan Implementasi No. Rekening

```

1 public class Tabungan {
2     private String pemilik;
3     private double saldo;
4     private int noRekening;
5     private static int nextId = 1;
6
7     public static void setNextId(int newId) {
8         nextId = newId;
9     }
10    public static int getNextId() {
11        return nextId;
12    }
13    public int getNoRekening() {
14        return noRekening;
15    }
16    public Tabungan(String pemilik, double saldo) {
17        this.pemilik = pemilik;
18        this.saldo = saldo;
19        noRekening = nextId;
20        nextId++;
21    }
22    public double getSaldo() {
23        return saldo;
24    }
25    public String getPemilik() {
26        return pemilik;
27    }
28 }

```

Kemudian buatlah sebuah Class Main dengan method main() di dalamnya, lalu coba lakukan beberapa hal berikut ini secara berurutan:

1. Buat tiga object dari class Tabungan, beri nama: t1, t2 dan t3.

2. Tampilkan nomor rekening dari t1, t2 dan t3. **Output yang diharapkan: 1, 2, 3.**
3. Akses method getNextId() dari t1 dan Tabungan, bandingkan hasilnya. **Mengapa demikian?**
4. Akses method setNextId() dengan parameter newId = 10. Kemudian buat sebuah object baru dari class Tabungan (t4). Tampilkan nomor rekening dari t4. **Output yang diharapkan: 10.**
5. Akses method getNextId() dari Tabungan, **nilainya seharusnya adalah 11.**

## 5.5 Latihan Mandiri

### 5.5.1 Matriks

Matriks adalah sebuah sekumpulan bilangan yang disusun dalam baris dan kolom yang digunakan dalam Matematika. Anda diminta untuk membuat abstraksi tentang Matriks dan melakukan enkapsulasinya. Matriks yang dibuat memiliki ukuran 3 x 3.

1. baris = 3 (konstanta)
2. kolom = 3 (konstanta)
3. `int[][] data = new int[baris][kolom];`

Method:

1. getter baris
2. getter kolom
3. getter dan setter data array 2 dimensi secara keseluruhan
4. getter dan setter data untuk baris dan kolom tertentu (i,j)
5. jumlahMatriksSkalar dengan input nilai skalar (angka integer) untuk menjumlahkan matriks dengan nilai skalar
6. kurangMatriksSkalar dengan input nilai skalar (angka integer) untuk mengurangi matriks dengan nilai skalar
7. kaliMatriksSkalar dengan inputan nilai skalar (angka integer) untuk mengkalikan matriks dengan nilai skalar

Kemudian anda harus melakukan:

1. Buat class Main
2. Buat 3 buah matriks A dengan data 1,2,3,4,5,6,7,8,9 dari inputan pengguna, matriks B dengan data 2,2,2,3,3,3,4,4,4 dari inputan pengguna, dan matriks C (kosong)
3. hitung jumlah matriks skalar untuk matriks A dengan nilai input skalar 2 dan tampilkan. Hasilnya:

```
3 4 5
6 7 8
9 10 11
```

4. hitung kurang matriks skalar untuk matriks B dengan nilai input skalar 1 dan tampilkan. Hasilnya:

```
1 1 1
2 2 2
3 3 3
```

5. buat method static di Main untuk menjumlahkan matriks A dan B dan disimpan di C dan tampilkan. Hasilnya:

```
1 1 1
2 2 2
3 3 3
```

### 5.5.2 Pembelian

Buatlah class **Pembelian** yang merepresentasikan pembelian suatu barang dalam jumlah tertentu. Field-field yang harus ada pada class Pembelian adalah:

- Nama barang (String)
- Harga satuan (double)
- Jumlah pembelian (int)
- Apakah termasuk barang mewah (true/false)
- Besarnya pajak tetap barang tersebut (5% atau 10%)

Khusus untuk field pajak tetap, nilainya berlaku untuk **semua pembelian**. Nilai pajak tetap harus dapat diubah. Setiap pengubahan pajak tetap secara otomatis akan mengubah nilai pajak tetap untuk semua pembelian-pembelian lainnya.

Method yang harus dibuat dalam class Pembelian adalah method **bayar()** yang merupakan jumlah uang yang harus dibayar berdasarkan pembelian tersebut. Jumlah uang yang harus dibayar dihitung dengan cara:  $\text{bayar} = (\text{jumlah barang} * \text{harga satuan}) + \text{pajak}$ . Pajak terdiri dari pajak tetap dan pajak barang mewah. Jika termasuk barang mewah, maka ditambah pajak sebesar 2%. Besarnya pajak barang mewah adalah **konstan**, tidak boleh diubah.

Setelah selesai membuat class Pembelian, buatlah class Main yang menerima input pembelian dari pengguna dari console (bisa menggunakan Scanner), kemudian menampilkan jumlah uang yang harus dibayar dari pembelian tersebut.

**SELAMAT MENGERJAKAN.**



## 6. Constructor dan Overloading Method

### 6.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Memahami dan dapat menjelaskan kegunaan dari Constructor.
2. Dapat mendefinisikan dan mengimplementasikan Constructor sesuai kebutuhan.
3. Dapat membuat overloading Constructor dan overloading method.

### 6.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 8 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 17).
- IntelliJ Ultimate 64 bit (Students Key).

### 6.3 Dasar Teori

#### 6.3.1 Constructor

Pada saat proses instansiasi, constructor akan dijalankan untuk menginisialisasi object yang baru dibuat. Perhatikan source code dari class Demo berikut ini:

```
1 public class Demo {  
2     int a;  
3     public Demo(){  
4         System.out.println("Constructor Demo dipanggil");  
}
```

```

5     }
6 }

```

Constructor dari class Demo terdapat pada baris 3-5. Berikut ini adalah potongan kode program untuk menunjukkan penggunaan constructor dari class Demo tersebut:

```

1 .....
2 Demo d1 = new Demo();
3 System.out.println("Nilai a di d1 = " + d1.a);

```

Output yang dihasilkan dari contoh tersebut adalah:

```

Constructor Demo dipanggil
Nilai a di d1 = 0

```

Dari contoh tersebut, ada beberapa hal yang perlu diperhatikan:

- Constructor adalah sebuah method yang tidak memiliki return type.
- Constructor akan dijalankan saat proses instansiasi (pemanggilan dengan **new**).
- Nilai field a tidak diubah di constructor, sehingga nilainya akan diinisialisasi dengan nilai defaultnya.

Pada Java, semua field pada sebuah class akan diinisialisasi dengan nilai default-nya, dengan ketentuan sebagai berikut:

- Jika field bertipe data reference maka akan bernilai **null**.
- Jika field bertipe data primitives, akan bernilai sesuai dengan nilai default-nya. Untuk int, byte, short, long, double dan float akan bernilai 0. Sedangkan boolean akan bernilai false dan char akan bernilai karakter null ('\0').

### 6.3.2 Overloading Constructor

Sebuah class bisa saja memiliki constructor lebih dari satu, yang dinamakan sebagai **Overloading Constructor**. Misalnya pada class Waktu. Class Waktu memiliki field jam, menit dan detik. Class Waktu didefinisikan sebagai berikut:

```

1 public class Waktu {
2     private int detik;
3     private int menit;
4     private int jam;
5
6     public Waktu(int jam, int menit, int detik){
7         this.detik = detik;
8         this.menit = menit;
9         this.jam = jam;
10    }
11 }

```

Pada class Waktu tersebut terdapat constructor dengan method signature **Waktu(int, int, int)**. Jika dikembangkan, bisa saja hanya diperlukan informasi jam-nya saja, sedangkan menit dan detiknya adalah 0. Misalnya ingin membuat object dari class Waktu yang merepresentasikan pukul 11:00, maka hanya perlu jam saja, sedangkan menit dan detiknya dibuat bernilai 0. Untuk mengakomodasi kebutuhan tersebut, kita bisa menambahkan constructor baru ke dalam class Waktu sebagai berikut:

---

```
1 public class Waktu {
2     private int detik;
3     private int menit;
4     private int jam;
5
6     public Waktu(int jam, int menit, int detik){
7         this.detik = detik;
8         this.menit = menit;
9         this.jam = jam;
10    }
11
12    public Waktu(int jam){
13        this.jam = jam;
14        this.menit = 0;
15        this.detik = 0;
16    }
17 }
```

---

Sehingga untuk membuat object dari class Waktu tersebut dapat dilakukan dengan dua cara, yaitu sebagai berikut:

---

```
1 Waktu sekarang = new Waktu(11,52,0); //11:52:00
2 Waktu nanti = new Waktu(13); //13:00:00
```

---

Beberapa overloading constructor bisa dijadikan chained constructor, seperti pada contoh kode berikut ini:

---

```
1 public class Waktu {
2     private int detik;
3     private int menit;
4     private int jam;
5
6     public Waktu(int jam, int menit, int detik){
7         this.detik = detik;
8         this.menit = menit;
9         this.jam = jam;
10    }
11
12    public Waktu(int jam){
13        this(jam, 0, 0);
14    }
15 }
```

---

Baris kode program `this(jam, 0, 0);` merupakan pemanggilan constructor lain pada Class Waktu, yaitu constructor pada baris ke-6. Untuk memanggil constructor lain pada class yang sama digunakan **this**. Pemanggilan constructor lain dari suatu constructor dinamakan sebagai constructor berantai (*chained constructor*).

Jenis constructor lain yang biasa digunakan adalah **copy constructor**, yang bertujuan untuk menduplikasi suatu object. Misalnya kita sudah punya suatu object dari class Waktu yang merepresentasikan pukul 14:25:15, kemudian ingin diduplikasi, maka dapat melalui constructor seperti pada kode program berikut ini:

```
1 public class Waktu {  
2     private int detik;  
3     private int menit;  
4     private int jam;  
5  
6     public Waktu(int jam, int menit, int detik){  
7         this.detik = detik;  
8         this.menit = menit;  
9         this.jam = jam;  
10    }  
11  
12    public Waktu(int jam){  
13        this(jam, 0, 0);  
14    }  
15  
16    //copy constructor  
17    public Waktu(Waktu waktu){  
18        detik = waktu.detik;  
19        menit = waktu.menit;  
20        jam = waktu.jam;  
21    }  
22 }
```

Penggunaan copy constructor tersebut dapat dilihat pada potongan kode program berikut ini:

```
1 //buat object dari class Waktu, misalnya: 16:28:11  
2 Waktu sekarang = new Waktu(16, 28, 11);    //16:28:11  
3  
4 //buat copy dari object sekarang  
5 Waktu copySekarang = new Waktu(sekarang);  //16:28:11 juga
```

Yang perlu diperhatikan, pada copy constructor akan menghasilkan **dua object yang berbeda** tetapi **memiliki nilai property yang sama**.

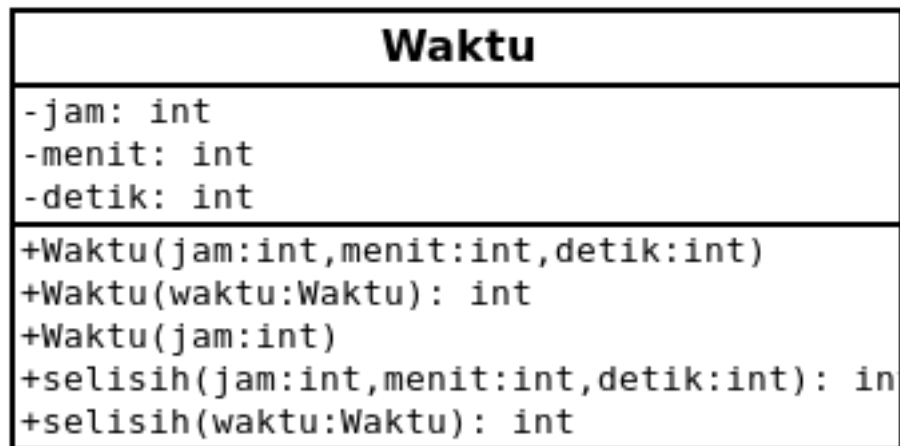
### 6.3.3 Overloading Method

Dalam suatu class dimungkinkan ada dua atau lebih method dengan nama yang sama, tetapi memiliki *method signature* yang berbeda. Sebagai contoh, perhatikan class Waktu yang memiliki dua method bernama **selisih()** seperti pada Gambar 6.1.

Pada class Waktu terdapat dua method selisih dengan method signature berbeda, yaitu:

- selisih(int jam, int menit, int detik) dengan method signature selisih(int, int, int)
- selisih(Waktu waktu) dengan method signature selisih(Waktu).





Gambar 6.1: Class Diagram untuk Class Waktu

Overloading method digunakan jika method tersebut dapat menerima lebih dari satu kemungkinan input/parameter, sehingga ditambahkan method dengan nama yang sama tetapi dengan parameter yang berbeda. Tujuan dari kedua method selisih tersebut sama, yaitu untuk menghitung selisih antara dua waktu. Hanya bedanya dua waktu tersebut bisa berupa informasi jam-menit-detik, bisa juga informasi object dari class Waktu. Dengan nama method yang sama maka pengguna class yang kita buat dapat mengetahui bahwa overloading method tersebut memiliki tujuan yang sama, tetapi membutuhkan parameter-parameter yang berbeda. Pengguna class bisa menggunakan method yang sesuai dengan parameter yang dimilikinya. Berikut ini adalah contoh potongan kode program yang memanfaatkan overloading method tersebut:

```

1  //buat object waktu 18:00:15
2  Waktu sekarang = new Waktu(18, 0, 15);
3
4  //hitung selisihnya (dalam detik) dengan pukul 19:00:00
5  //seharusnya return 3585 detik
6  int selisihDetik = sekarang.selisih(19, 0, 0);
7  System.out.println("Selisih: " + selisihDetik);
8
9  Waktu target = new Waktu(19); //pukul 19:00:00
10
11 //hitung selisih sekarang dengan target
12 //seharusnya selisihnya sama, 3585 detik
13 int selisihDetikTarget = sekarang.selisih(target);
14 System.out.println("Selisih: " + selisihDetikTarget);

```

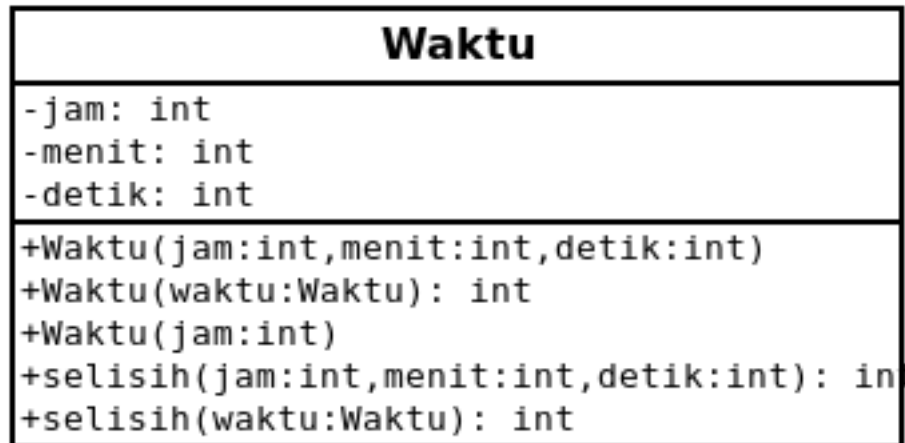
## 6.4 Kegiatan Praktikum

Pada bagian ini anda akan membuat implementasi class Waktu berdasarkan spesifikasi sebagai berikut:

- Class Waktu memiliki 3 constructor.
- Class Waktu memiliki method-method sebagai berikut:
  - Method selisih() yang dapat menghitung selisih dua waktu dalam detik dengan parameter object dari class Waktu.

- Method selisih() yang dapat menghitung selisih dua waktu dalam detik dengan parameter jam, menit dan detik dari waktu yang lain.
- Setter dan getter untuk field jam, menit dan detik.

Class Waktu tersebut dapat ditampilkan dalam bentuk Class Diagram seperti pada Gambar 6.2.



Gambar 6.2: Overloading Constructor dan Overloading Method pada Class Waktu

Buatlah sebuah project baru di Eclipse dengan nama Waktu, kemudian tambahkan sebuah class baru bernama Waktu ke dalam project tersebut. Berikut ini adalah isi dari class Waktu sesuai dengan spesifikasi yang telah diberikan sebelumnya:

```

1 public class Waktu {
2     private int detik;
3     private int menit;
4     private int jam;
5
6     public Waktu(int jam, int menit, int detik){
7         this.detik = detik;
8         this.menit = menit;
9         this.jam = jam;
10    }
11
12    public Waktu(int jam){
13        this(jam, 0, 0);
14    }
15
16    //copy constructor
17    public Waktu(Waktu waktu){
18        detik = waktu.detik;
19        menit = waktu.menit;
20        jam = waktu.jam;
21    }
22
23    //selisih dalam detik
24    public int selisih(int jam, int menit, int detik){
25        int waktuIni = this.detik + (this.menit * 60) + (this.jam * 3600);
  
```

```

26         int waktuParameter = detik + (menit * 60) + (jam * 3600);
27         int hasil = Math.abs(waktuIni - waktuParameter);
28         return hasil;
29     }
30
31     //selisih dalam detik, memanggil method selisih() sebelumnya
32     public int selisih(Waktu waktu){
33         int jamParameter = waktu.getJam();
34         int menitParameter = waktu.getMenit();
35         int detikParameter = waktu.getDetik();
36         return selisih(jamParameter, menitParameter, detikParameter);
37     }
38 }

```

Tambahkan class Main ke dalam project tersebut, kemudian cobalah untuk menggunakan ketiga constructor dan kedua method selisih() yang telah dibuat.

## 6.5 Latihan Mandiri

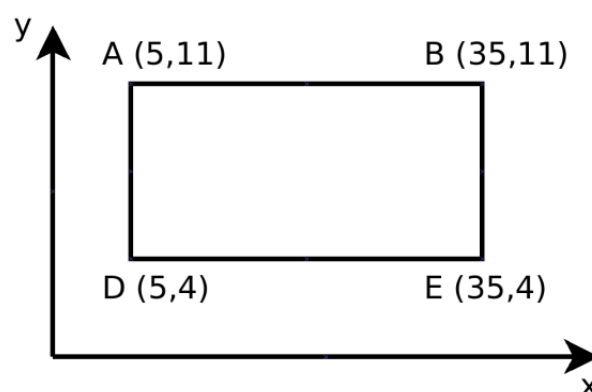
### 6.5.1 Melengkapi Class Waktu

Lengkapi class Waktu dengan method-method berikut ini:

- Getter dan setter untuk semua atributnya. Untuk setter, pastikan jam, menit dan detik tetap valid.
- tampilWaktu() menampilkan waktu dalam bentuk string dengan format "h:m:s"
- tambahDetik(), tambahMenit() dan tambahJam(). Gunakan sistem 24 jam.

### 6.5.2 Implementasi Class Rectangle

Sebuah segiempat (rectangle) dapat dibentuk dengan beberapa cara, misalnya dengan menyusunnya dari 4 titik yang berbeda. Perhatikan Gambar 6.3. Segiempat ABCD tersusun dari 4 titik, yaitu A, B, C dan D. Setiap titik memiliki koordinat (x,y).



Gambar 6.3: Segiempat ABCD pada sistem koordinat Cartesius

Pada bagian ini, anda diminta untuk membuat class Rectangle dan class Point. Berikut adalah spesifikasi dari class Point yang harus dipenuhi:

- Class Point memiliki dua atribut, yaitu x dan y yang masing-masing bertipe integer.
- Class Point memiliki tiga constructor, yaitu:

- Point(int, int), langsung mengisi nilai x dan y.
  - Point(), berarti membuat sebuah titik pada koordinat (0,0).
  - Point(Point), membuat titik berdasarkan titik yang sudah ada sebelumnya.
  - Class Point memiliki beberapa method, yaitu:
    - Method distance(int, int), untuk menghitung jarak antara titik sekarang dengan suatu titik di koordinat (x,y).
    - Method distance(Point), untuk menghitung jarak antara titik sekarang dengan suatu titik yang lain.
  - Setter dan getter untuk atribut x dan y.
- Untuk class Rectangle, berikut ini adalah spesifikasi yang harus dipenuhi:
- Class Rectangle memiliki 4 atribut, yaitu pointA, pointB, pointC dan pointD yang merupakan keempat titik sudut dari segiempat tersebut. Setiap atribut bertipe Point.
  - Class Rectangle memiliki minimal 4 Constructor.
  - Class Rectangle memiliki method area() untuk menghitung luas dan perimeter() untuk menghitung keliling dari segiempat tersebut.
  - Lengkapi dengan getter dan setter untuk seluruh atribut yang ada.

**SELAMAT MENGERJAKAN.**



# Konsep Pemrograman Berorientasi Obyek

<b>7</b>	<b>Inheritance dan Overriding</b>	<b>61</b>
7.1	Tujuan Praktikum	
7.2	Alat dan Bahan	
7.3	Dasar Teori	
7.4	Kegiatan Praktikum	
7.5	Latihan Mandiri	
<b>8</b>	<b>Abstract Class dan Interface</b>	<b>71</b>
8.1	Tujuan Praktikum	
8.2	Alat dan Bahan	
8.3	Dasar Teori	
8.4	Kegiatan Praktikum	
8.5	Latihan Mandiri	
<b>9</b>	<b>Polimorfisme</b>	<b>79</b>
9.1	Tujuan Praktikum	
9.2	Alat dan Bahan	
9.3	Dasar Teori	
9.4	Kegiatan Praktikum	
9.5	Latihan Mandiri	
<b>10</b>	<b>Penanganan Exception</b>	<b>85</b>
10.1	Tujuan Praktikum	
10.2	Alat dan Bahan	
10.3	Dasar Teori	
10.4	Kegiatan Praktikum	
10.5	Latihan Mandiri	



## 7. Inheritance dan Overriding

### 7.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Dapat memahami dan menjelaskan konsep Inheritance.
2. Dapat menerapkan Inheritance pada permasalahan yang sesuai.
3. Dapat membuat implementasi Inheritance dan Overriding.

### 7.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 8 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 8).
- IntelliJ Ultimate 64 bit (Students Key).

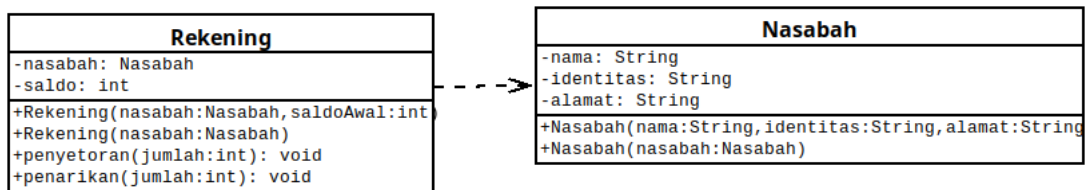
### 7.3 Dasar Teori

#### 7.3.1 Inheritance

Inheritance merupakan salah satu konsep utama dalam pemrograman berorientasi obyek. Inheritance adalah proses dimana suatu class mendapatkan atribut dan fungsionalitas (method) dari class yang sudah ada sebelumnya. Class yang baru tidak perlu mendefinisikan semuanya dari awal, class tersebut hanya perlu menambahkan fungsionalitas (extends) yang tidak ada dari class-class sebelumnya.

Mari kita lihat kembali class Rekening yang telah dibahas pada pertemuan sebelumnya. Class diagram untuk class Rekening dapat dilihat pada Gambar 7.1. Relasi antara class Rekening dan

class Nasabah adalah **dependency**, dimana class Nasabah merupakan tipe dari salah satu atribut di class Rekening. Jika terjadi perubahan pada class Nasabah, class Rekening juga perlu diubah.



Gambar 7.1: Class Diagram untuk Class Rekening dan Nasabah

Pada class Nasabah terdapat tiga atribut yang tidak dapat diubah, sehingga class Nasabah dapat disebut sebagai **Immutable Class**. Berikut ini adalah implementasi dari class Nasabah:

```

1 public class Nasabah {
2     private String nama;
3     private String identitas;
4     private String alamat;
5
6     public Nasabah(String nama, String identitas, String alamat){
7         this.nama = nama;
8         this.identitas = identitas;
9         this.alamat = alamat;
10    }
11
12    public String getNama() { return nama; }
13    public String getIdentitas() { return identitas; }
14    public String getAlamat(){ return alamat; }
15 }
  
```

Setelah kita membuat object dari class Nasabah, tidak ada cara untuk mengubah property nama, identitas atau alamat. Walaupun demikian, kita tetap dapat mendapatkan/membaca nilai dari property tersebut melalui getter yang telah dibuat. Getter perlu disediakan karena nantinya akan diperlukan pada class Rekening. Berikut ini adalah implementasi dari class Rekening:

```

1 public class Rekening {
2     private Nasabah nasabah;
3     private int saldo;
4
5     public Rekening(Nasabah nasabah, int saldoAwal){
6         this.nasabah = new Nasabah(nasabah.getNama(),
7             ↳ nasabah.getIdentitas(), nasabah.getAlamat());
8         this.saldo = saldoAwal;
9     }
10
11    public Rekening(Nasabah nasabah){
12        this(nasabah, 0);
13    }
14 }
  
```



```

13
14     public void penyetoran(int jumlah){
15         if(jumlah > 0)
16             saldo = saldo + jumlah;
17     }
18
19     public void penarikan(int jumlah){
20         int saldoBaru = saldo + jumlah;
21         if(saldoBaru > 0 && jumlah > 0)
22             saldo = saldoBaru;
23     }
24 }

```

Class Rekening tersebut telah berjalan dengan baik sesuai dengan spesifikasi yang diberikan. Dalam pengembangannya ditambahkan fitur rekening keluarga (RekeningKeluarga) dan rekening bisnis (RekeningBisnis). Perbedaan dari Rekening, RekeningBisnis dan RekeningKeluarga dapat dilihat pada Tabel 7.1.

Tabel 7.1: Perbedaan class Rekening, RekeningBisnis dan RekeningKeluarga

Operasi	Rekening	RekeningBisnis	RekeningKeluarga
penyetoran()	Tidak ada bunga	0.1%	0.5%
penarikan()	Tidak ada biaya administrasi	Tidak ada biaya administrasi	biaya administrasi 5000

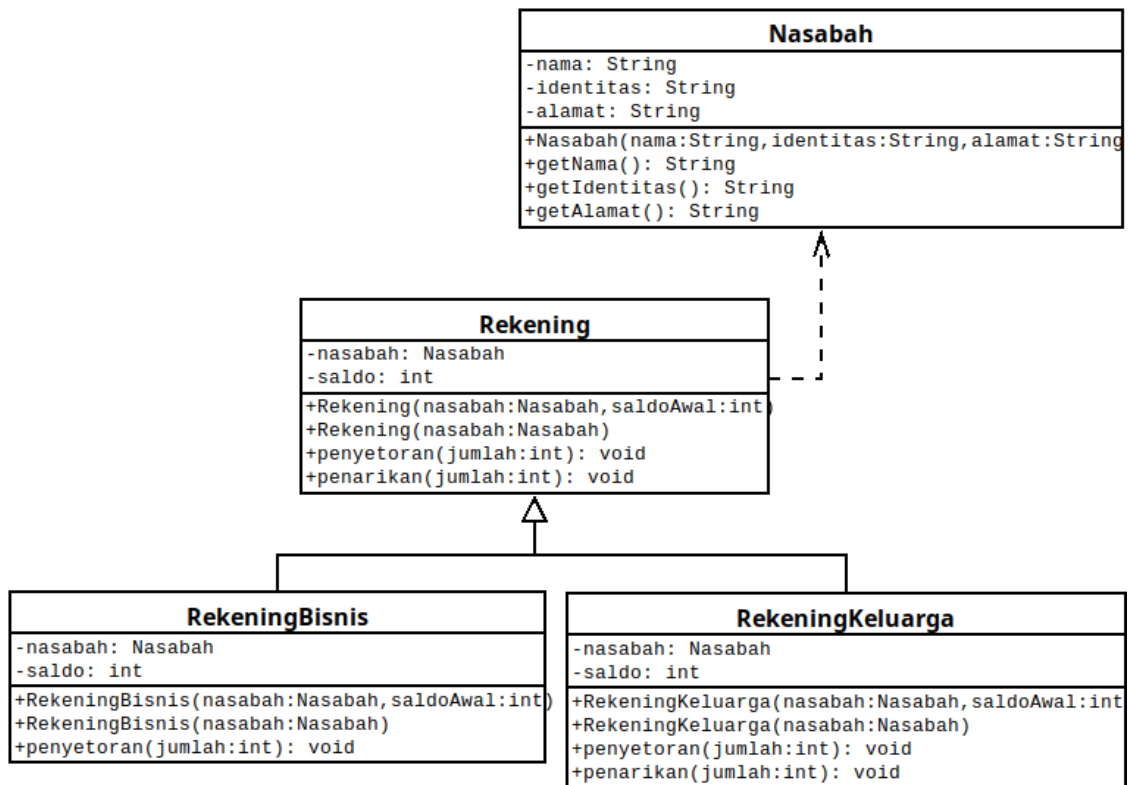
Secara umum RekeningBisnis adalah Rekening, tetapi memiliki aturan yang lebih spesifik, yaitu pada method penyetoran yang ditambahkan implementasi penambahan bunga sebesar 0.1%. Method penarikan() tidak mengalami perubahan. Kasus seperti ini dapat dikatakan bahwa RekeningBisnis **is a** Rekening. Artinya RekeningBisnis adalah Rekening. Relasi seperti ini menandakan adanya hubungan Inheritance antara RekeningBisnis dan Rekening. Yang perlu diperhatikan relasi ini tidak berlaku kebalikannya, artinya jika kita mengatakan Rekening **is a** RekeningBisnis, maka pernyataan tersebut tidak valid. Sama seperti **burung adalah hewan** itu valid, tetapi jika kita mengatakan **hewan adalah burung** maka pernyataan tersebut tidak valid.



Is-a adalah relasi Inheritance, yang merupakan relasi satu arah, tidak berlaku kebalikannya.

Kita akan membuat implementasi class RekeningBisnis dan RekeningKeluarga berdasarkan class Rekening yang sudah dibuat sebelumnya. Yang akan dilakukan adalah menggunakan konsep Inheritance untuk class RekeningBisnis dan RekeningKeluarga berdasarkan class Rekening. Class diagram setelah ditambahkan class RekeningBisnis dan class RekeningKeluarga dapat dilihat pada Gambar 7.2.

Hubungan Inheritance terjadi antara super-class dan sub-class. Super-class juga biasa disebut sebagai base-class, yang di-extends (diperluas) oleh sub-class. Pada Java hanya bisa melakukan single Inheritance, sehingga suatu sub-class hanya bisa memiliki 1 super-class saja. Tetapi sebuah class dapat di-extends oleh beberapa class lainnya. Pada implementasi yang akan kita buat, class RekeningBisnis dan RekeningKeluarga merupakan sub-class dari class Rekening. Untuk mendefinisikan hubungan Inheritance, digunakan keyword extends pada Java, seperti pada contoh-contoh berikut ini:



Gambar 7.2: Relasi antara class Rekening, RekeningBisnis dan RekeningKeluarga

```

1 public class RekeningBisnis extends Rekening {
2
3 }

```

```

1 public class RekeningKeluarga extends Rekening {
2
3 }

```

Pada Gambar 7.2 dapat dilihat bahwa pada class `RekeningBisnis` tidak terdapat method `penarikan()` karena implementasinya sama dengan super-classnya, yaitu class `Rekening`. Karena sama, sub-class tidak perlu mendefinisikan ulang method tersebut dan cukup menggunakan method yang ada di super-classnya. Sub-class dapat mengakses method-method yang ada di super-class, kecuali jika access specifier diatur menjadi `private`. Selain itu, method dengan access specifier default juga tidak dapat diakses jika super-class dan sub-class berada di package yang berbeda.

### 7.3.2 Constructor pada Inheritance

Pada Inheritance, constructor di sub-class akan memanggil constructor yang ada di super-classnya. Perhatikan contoh berikut ini:

```

1 public class Demo {
2     public Demo(){
3         System.out.println("Constructor Demo dipanggil");

```

```

4     }
5 }
6
7 public class AnakDemo extends Demo {
8     public AnakDemo(){
9         System.out.println("Constructor AnakDemo dipanggil");
10    }
11 }
12
13 public class CucuDemo extends Demo {
14     public CucuDemo(){
15         System.out.println("Constructor CucuDemo dipanggil");
16    }
17 }

```

Pada kode program tersebut didefinisikan tiga class yang memiliki hubungan Inheritance. Jika kita membuat object baru dari class CucuDemo seperti berikut:

```

1 public class Main {
2     public static void main(String[] args){
3         CucuDemo cd = new CucuDemo();
4     }
5 }

```

Saat dijalankan, output yang dihasilkan adalah:

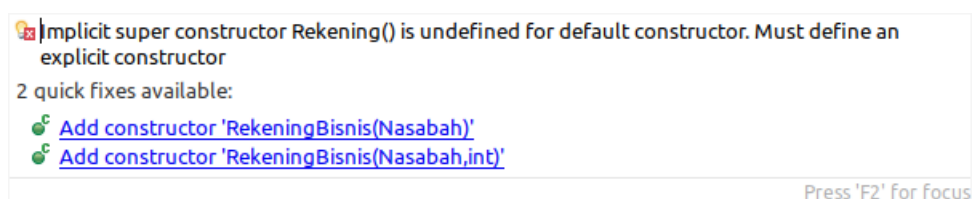
```

Constructor Demo dipanggil
Constructor AnakDemo dipanggil
Constructor CucuDemo dipanggil

```

Dari contoh program tersebut, dapat dilihat bahwa pemanggilan constructor dari class CucuDemo akan memanggil constructor seluruh super-classnya. Urutan pemanggilan dimulai dari super-class yang tertinggi, yaitu class Demo, kemudian class AnakDemo dan yang terakhir adalah class Demo. Pemanggilan constructor dari super-class menggunakan **super()**. Secara default, setiap constructor dari sub-class akan memanggil constructor pada super-classnya.

Pada implementasi class RekeningBisnis, perlu diperhatikan bahwa class Rekening yang merupakan super-classnya tidak memiliki constructor default, karena telah ditambahkan dua constructor yaitu Rekening(Nasabah, int) dan Rekening(Nasabah). Pada class RekeningBisnis perlu ditambahkan pemanggilan constructor yang ada di class Rekening. Pada class RekeningBisnis akan muncul pesan kesalahan seperti pada Gambar 7.3.



Gambar 7.3: Pesan kesalahan pada class RekeningBisnis


Berikut ini adalah implementasi constructor pada class RekeningBisnis untuk menghindari kesalahan tersebut:

```

1 public class RekeningBisnis extends Rekening {
2     public RekeningBisnis(Nasabah nasabah, int saldoAwal){
3         super(nasabah, saldoAwal);
4     }
5
6     public RekeningBisnis(Nasabah nasabah){
7         super(nasabah);
8     }
9     ..... // dipotong di sini
10 }

```

Pemanggilan `super(nasabah, saldoAwal)` pada class `RekeningBisnis` berarti pemanggilan constructor `Rekening(Nasabah, int)`. Demikian juga dengan `super(nasabah)` yang merupakan pemanggilan dari constructor `Rekening(nasabah)`.

 Perlu diingat, pemanggilan `super()` harus dilakukan di baris pertama pada constructor.

### 7.3.3 Overriding

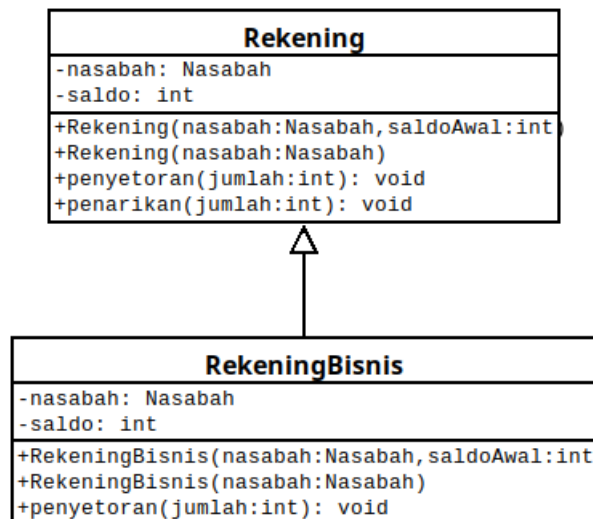
Overriding adalah mendefinisikan ulang suatu method pada sub-class walaupun method tersebut sudah ada di super-class. Overriding dilakukan saat proses bisnis/algorithm pada sub-class berbeda dengan implementasi di super-class. Perbedaan antara overloading dan overriding dapat dilihat pada Tabel 7.2.

Tabel 7.2: Perbedaan Overloading dan Overriding

Aspek	Overloading	Overriding
Nama method	Harus sama	Harus sama
Method signature	Harus berbeda	Harus sama
Posisi	Berada pada class yang sama	Berada pada class yang berbeda, harus memiliki hubungan inheritance
Dilakukan terhadap	Method dan constructor	Hanya method, constructor tidak bisa di-overriding

Perhatikan relasi antara class `Rekening` dan `RekeningBisnis` pada Gambar 7.4. Method `penyetoran()` didefinisikan ulang di class `RekeningBisnis` walaupun method tersebut sudah ada di super-classnya, yaitu class `Rekening`.

Pendefinisian ulang tersebut dinamakan overriding. Overriding dilakukan pada saat method di sub-class memiliki algoritma/proses bisnis yang berbeda dengan method di super-class. Penamaan method tetap dibuat sama karena sebenarnya apa yang ingin dicapai sama, tetapi dengan cara yang berbeda.



Gambar 7.4: Relasi antara class Rekening dan RekeningBisnis

## 7.4 Kegiatan Praktikum

Pada bagian ini kita akan melengkapi class Rekening, RekeningBisnis dan class RekeningKeluarga sesuai dengan spesifikasi yang diberikan. Tambahkan implementasi getter pada class Rekening sehingga menjadi seperti berikut:

```

1 public class Rekening {
2     private Nasabah nasabah;
3     private int saldo;
4
5     public Rekening(Nasabah nasabah, int saldo) {
6         nasabah = new Nasabah(nasabah.getNama(),
7             ↳ nasabah.getIdentitas(), nasabah.getAlamat());
8         this.saldo = saldo;
9     }
10
11     public Rekening(Nasabah nasabah) {
12         this(nasabah, 0);
13     }
14
15     public void penyetoran(int jumlah) {
16         if(jumlah > 0)
17             saldo = saldo + jumlah;
18     }
19
20     public void penarikan(int jumlah) {
21         int saldoBaru = saldo - jumlah;
22         if(saldoBaru > 0 && jumlah > 0)
23             saldo = saldoBaru;
24     }
25
26     public Nasabah getNasabah() {

```

```

26         return nasabah;
27     }
28
29     public int getSaldo() {
30         return saldo;
31     }
32 }

```

Berikut ini adalah implementasi dari class RekeningBisnis:

```

1  public class RekeningBisnis extends Rekening {
2      public final double BUNGA_SETOR = 0.001;
3
4      public RekeningBisnis(Nasabah nasabah) {
5          super(nasabah);
6      }
7
8      public RekeningBisnis(Nasabah nasabah, int saldo) {
9          super(nasabah, saldo);
10     }
11
12     public void penyetoran(int jumlah) {
13         if(jumlah > 0) {
14             int totalSetoran = jumlah + bunga(jumlah);
15             super.penyetoran(totalSetoran);
16         }
17     }
18
19     private int bunga(int jumlah) {
20         return (int) BUNGA_SETOR * jumlah;
21     }
22 }

```

Sebagai latihan, buatlah implementasi dari class RekeningKeluarga sesuai dengan spesifikasi yang telah diberikan. Kemudian ujikan setiap class yang telah anda buat, apakah sudah memenuhi spesifikasi yang telah diberikan atau belum.

## 7.5 Latihan Mandiri

Setelah anda menyelesaikan class Rekening, RekeningBisnis dan RekeningKeluarga dengan baik, pada bagian ini anda diminta untuk membuat class-class baru untuk beberapa jenis rekening baru sebagai berikut:

1. Class RekeningSyariah dengan spesifikasi sebagai berikut:

- Setiap penarikan uang di atas 100000 dikenakan biaya administrasi 1000. Penarikan uang 100000 ke bawah tidak dikenakan biaya administrasi.
- Penyetoran tidak mendapatkan bunga.
- Nasabah dapat memberikan sumbangan dalam jumlah tertentu yang dipotong langsung dari saldo pada rekening syariahnya. Sumbangan diimplementasikan dalam method

bernama sedekah(). Perhatikan, jumlah saldo harus mencukupi untuk melakukan sedekah sesuai dengan nominal yang diberikan.

2. Class RekeningUtama dan RekeningTambahan yang ditujukan untuk orang tua dan anaknya.

Spesifikasinya sebagai berikut:

- Rekening utama memiliki batas penarikan sejumlah 3000000, sedangkan rekening tambahan memiliki batas penarikan sejumlah 500000 saja.
- Untuk setiap penarikan dikenakan biaya administrasi 5000, baik untuk rekening utama maupun rekening tambahan.
- Setiap kali penyetoran mendapatkan bunga 0.5%, sama seperti rekening keluarga.

**SELAMAT MENGERJAKAN.**





## 8. Abstract Class dan Interface

### 8.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Dapat mendefinisikan abstract class dan interface.
2. Dapat mendefinisikan dan menggunakan class yang dibuat berdasarkan abstract class yang telah didefinisikan sebelumnya.
3. Dapat mendefinisikan dan menggunakan class yang mengimplementasikan suatu interface yang telah didefinisikan sebelumnya.

### 8.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 8 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

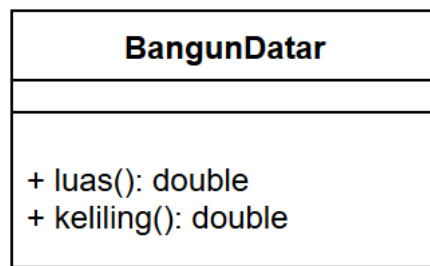
- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 8).
- IntelliJ Ultimate 64 bit (Students Key).

### 8.3 Dasar Teori

#### 8.3.1 Abstract Class

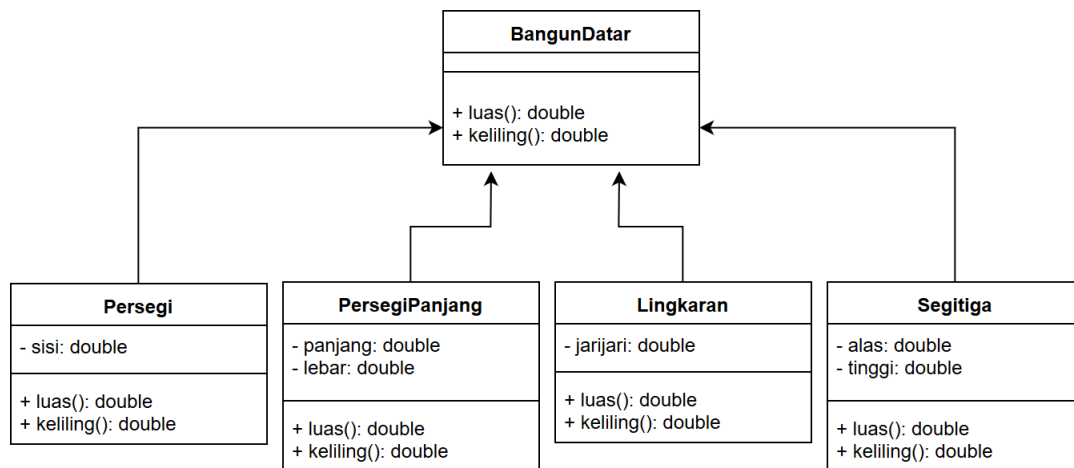
Abstract class merupakan suatu class yang salah satu (atau lebih) method yang dimilikinya belum memiliki implementasi. Misalnya saja pada class BangunDatar pada Gambar 8.1.

Setiap bangun datar (segitiga, persegi panjang, persegi, segilima, lingkaran, dsb) dapat dihitung luas dan kelilingnya. Tetapi akan sulit untuk mendefinisikan bagaimana cara menghitung luas/kelilingnya karena setiap bangun datar tersebut memiliki cara perhitungan luas dan keliling



Gambar 8.1: Class BangunDatar dengan method luas() dan keliling()

yang berbeda-beda. Misalnya pada bangun datar segitiga, luas didapatkan dengan perhitungan  $luas = \frac{1}{2} * alas * tinggi$ , sedangkan luas lingkaran adalah  $luas = \pi * r^2$ . Hubungan antara bangun datar dengan segitiga, persegi, persegi panjang dan lingkaran adalah hubungan inheritance seperti yang ditunjukkan pada Gambar 8.2.



Gambar 8.2: Relasi inheritance class BangunDatar dengan subclassnya

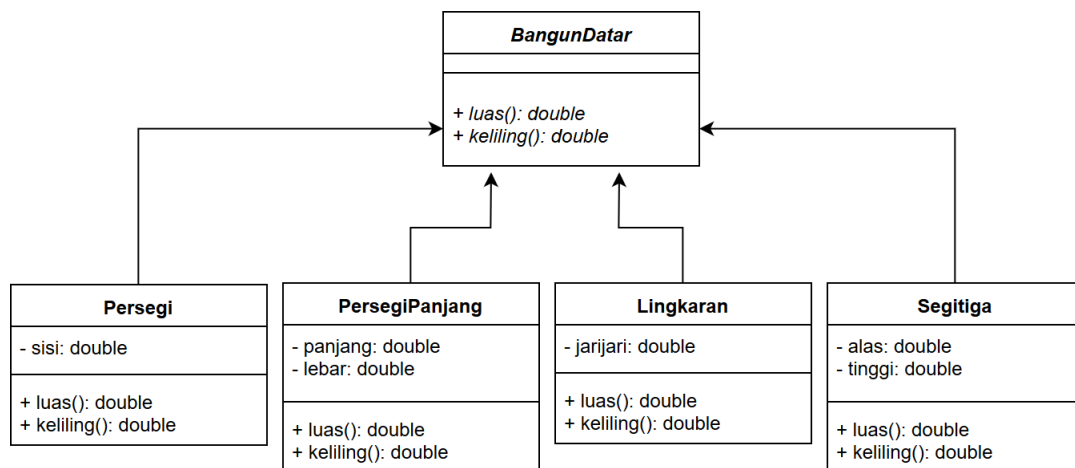
BangunDatar sebagai superclass memberikan spesifikasi bahwa setiap bangun datar harus dapat dihitung luas dan kelilingnya, tetapi pada BangunDatar belum dapat didefinisikan bagaimana cara menghitung luas dan keliling. Untuk mengatasi permasalahan tersebut, maka method luas() dan keliling() pada class BangunDatar perlu didefinisikan sebagai **abstract method**. Abstract method adalah method yang belum memiliki implementasi. Abstract method didefinisikan dengan cara berikut ini:

```

1  ...
2  public abstract double luas();
3  public abstract double keliling();
4  ...
  
```

Abstract method tersebut tentunya tidak dapat dipanggil karena belum memiliki implementasi. Karena itu, class BangunDatar perlu dijadikan sebagai Abstract class. Class diagram pada Gambar 8.2 perlu diperbaiki menjadi class diagram seperti pada Gambar 8.3.

Implementasi class BangunDatar adalah sebagai berikut:



Gambar 8.3: Class BangunDatar sebagai abstract class

```

1 public abstract class BangunDatar {
2     public abstract double luas();
3     public abstract double keliling();
4 }

```

Sebagai contoh class PersegiPanjang. Persegi panjang merupakan bangun datar, sehingga sangat tepat jika dikatakan class PersegiPanjang adalah subclass dari class BangunDatar, seperti pada implementasi berikut:

```

1 public class PersegiPanjang extends BangunDatar {
2     private double panjang;
3     private double lebar;
4
5     //Constructor (tidak dituliskan di sini, untuk menghemat tempat
6
7     //Getter dan setter (tidak dituliskan di sini, untuk menghemat tempat
8
9     //implementasi konkrit dari method luas yang diwarisi dari class BangunDatar
10    public double luas(){
11        return panjang * lebar;
12    }
13
14    //implementasi konkrit dari method keliling
15    public double keliling(){
16        return 2 * (panjang + lebar);
17    }
18 }

```

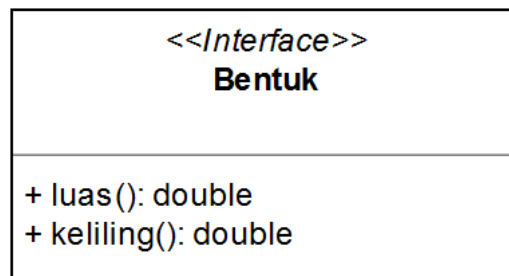
Beberapa hal yang perlu diperhatikan terkait abstract class:

- Abstract class tidak dapat diinstansiasi, karena mengandung satu atau lebih abstract method yang tidak memiliki implementasi.

- Abstract class tidak dapat dideklarasikan sebagai class private, karena abstract class didefinisikan dengan tujuan supaya di-extends dan di-override oleh class-class turunannya.

### 8.3.2 Interface

Interface mirip dengan abstract class, tetapi pada interface hanya dimungkinkan diisi oleh abstract method dan konstanta. Seluruh method pada interface adalah abstract method, yang harus diimplementasikan oleh class-class yang mengimplementasikan interface tersebut. Pada Java 8 ditambahkan kemampuan untuk mendeklarasikan static method pada interface dan pada Java 9 dimungkinkan untuk mendeklarasikan private method pada interface. Sebagai contoh, perhatikan diagram UML untuk interface Bentuk pada Gambar 8.4.



Gambar 8.4: Interface Bentuk

Berikut ini adalah cara untuk mendeklarasikan interface Bentuk tersebut:

```

1 public interface Bentuk{
2     public double luas();
3     public double keliling();
4 }
  
```

Berbeda dengan abstract class, pada interface tidak diperlukan menggunakan keyword *abstract* karena seluruh method pada interface adalah abstract method. Hubungan antara class dengan interface adalah implementasi, misalnya pada class Segitiga yang mengimplementasikan interface Bentuk seperti berikut:

```

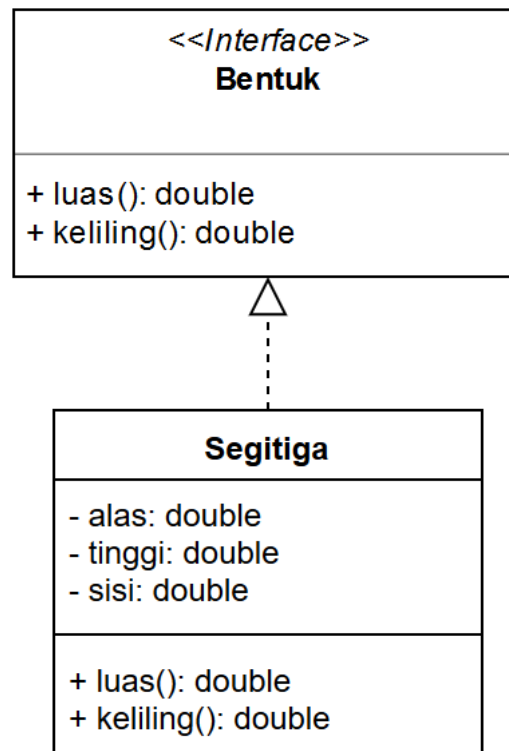
1 public class Segitiga implements Bentuk {
2     private double alas;
3     private double tinggi;
4     private double sisi;
5
6     //Constructor (tidak ditulis di sini)
7
8     //Getter-setter (tidak ditulis di sini)
9
10    //implementasi luas() dan keliling()
11    public double luas(){
12        return 0.5 * alas * tinggi;
13    }
14 }
  
```

```

15     public double keliling(){
16         return 3 * sisi; //diasumsikan diketahui panjang sisinya sama semua
17     }
18 }

```

Class diagram yang menggambarkan hubungan antara interface Bentuk dan class Segitiga dapat dilihat pada Gambar 8.5.



Gambar 8.5: Relasi antara interface Bentuk dengan class Segitiga

Berbeda dengan abstract class, suatu class dapat meng-implements lebih dari satu interface sekaligus, misalnya:

```

1 public class BelahKetupat extends BangunDatar implements Movable, Adjustable {
2     ...
3 }

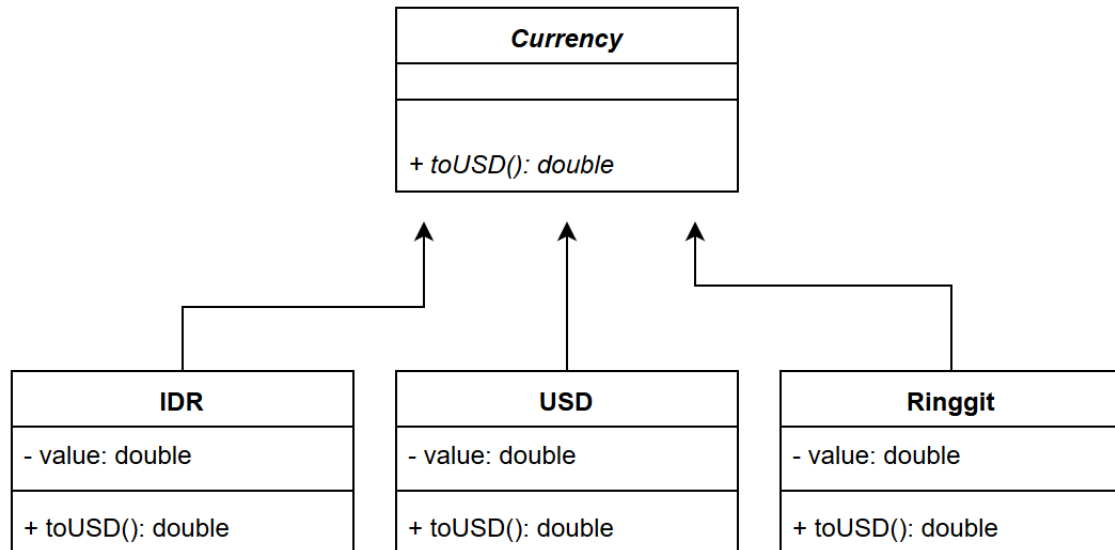
```

Class **BelahKetupat** meng-extends class **BangunDatar** dan mengimplementasi dua interface sekaligus, yaitu **Movable** dan **Adjustable**. Java menganut prinsip **Single Inheritance**, dimana suatu class hanya dapat meng-extends satu class lainnya. Multiple inheritance pada Java dapat dilakukan dengan menggunakan interface, dimana satu class meng-implements lebih dari satu interface sekaligus.

## 8.4 Kegiatan Praktikum

Pada bagian ini anda akan mengimplementasikan program untuk menghitung konversi mata uang. Mata uang USD (US Dollar) dapat dikatakan sebagai salah satu mata uang terkuat di

dunia, sehingga seluruh mata uang dapat dikonversikan/ditukar ke dalam USD. Mata uang akan diimplementasikan dalam bentuk abstract class `Currency`. Sedangkan untuk mata uang yang lebih khusus, misalnya IDR (Indonesian Rupiah), Ringgit ataupun USD akan diimplementasikan menggunakan class-class IDR, Ringgit dan USD seperti yang ditunjukkan oleh class diagram pada Gambar 8.6.



Gambar 8.6: Relasi antara class `Currency` dan class `IDR`, `USD` dan `Ringgit`

Buatlah class-class tersebut seperti berikut ini:

```

1 public abstract class Currency {
2     public abstract double toUSD();
3 }
4
5 public class IDR extends Currency {
6     private double value;
7     public IDR(double value){
8         this.value = value;
9     }
10
11     public double toUSD(){
12         return value / 14000; //diasumsikan 1 USD = IDR 14000
13     }
14 }
15
16 public class Ringgit extends Currency {
17     private double value;
18     public Ringgit(double value){
19         this.value = value;
20     }
21
22     public double toUSD(){
23         return value / 4; //diasumsikan 1 USD = 4 Ringgit
  
```

```

24     }
25 }
26
27 public class USD extends Currency {
28     private double value;
29     public USD(double value){
30         this.value = value;
31     }
32
33     public double toUSD{
34         return value; //tidak berubah, karena 1 USD = 1 USD
35     }
36 }

```

Untuk mencoba class-class tersebut, buatlah class Main seperti berikut:

```

1 public class Main {
2     public static void main(String[] args){
3         IDR rupiah = new IDR(70000); //IDR 70000
4         System.out.println("IDR 70000 = USD" + rupiah.toUSD());
5
6         USD usd = new USD(5.5);
7         System.out.println("USD 5.5 = USD " + usd.toUSD());
8
9         Ringgit ringgit = new Ringgit(20);
10        System.out.println("20 Ringgit = USD " + ringgit.toUSD());
11    }
12 }

```

Perhatikan output yang dihasilkan dari program tersebut. Menurut anda, mengapa class Currency dideklarasikan sebagai abstract class?

## 8.5 Latihan Mandiri

Definisikan sebuah class abstract, yaitu class Wallet yang memiliki spesifikasi sebagai berikut:

- Wallet adalah implementasi sebuah dompet virtual, yang memiliki atribut totalUang bertipe double.
- Kita dapat memasukkan berbagai macam mata uang (USD, Ringgit, Yen, Peso, dsb) ke dalam suatu dompet.
- Secara otomatis, setiap uang yang dimasukkan ke dalam dompet tersebut akan dikonversi terlebih dahulu menjadi mata uang tertentu tergantung jenis Walletnya. Misal DollarWallet berarti dikoversi ke USD, RinggitWallet berarti dikonversi ke Riggit, RupiahWallet berarti dikonversi ke Rupiah, dst.
- Kita dapat mengambil/membelanjakan uang dari dompet tersebut, dan akan dikonversi ke mata uang tertentu tergantung jenis Walletnya.
- pada class Wallet terdapat method abstract insertMoney(), expenseMoney(), dan printMoney() dengan ketentuan sebagai berikut:
  - Method insertMoney(Currency c) adalah method yang dipanggil jika ingin memasukkan suatu mata uang ke dalam dompet tersebut. Pada method ini setiap mata uang yang

masuk akan dikonversi ke dalam suatu nilai mata uang tertentu tergantung jenis Walletnya.

- Method `expenseMoney(Currency c)` adalah method yang dipanggil jika ingin mengambil suatu mata uang ke dalam dompet tersebut. Pada method ini setiap mata uang yang keluar akan dikonversi ke dalam suatu nilai mata uang tertentu tergantung jenis Walletnya.
- Method `printMoney()` akan menampilkan informasi nilai keseluruhan dari isi dompet sekarang dalam mata uang tertentu tergantung jenis Walletnya.

- Buatlah setter dan getter untuk `totalUang`.

Uji coba class `Wallet` dengan diturunkan ke class `DollarWallet` dan override method `insertCurrency(Currency c)`, `expenseCurrency(Currency c)`, dan method `printMoney()`. `DollarWallet` bisa juga dikembangkan ke `RinggitWallet`, `YenWallet`, dan sebagainya. Untuk mengerjakan soal ini, bisa dengan mengubah interface `Currency` yang baru memiliki method `toUSD()` di atas.

**SELAMAT MENGERJAKAN.**



## 9. Polimorfisme

### 9.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Dapat memahami dan menerapkan prinsip polimorfisme dalam bahasa pemrograman Java.
2. Dapat merancang dan mengimplementasikan penerapan polimorfisme dalam memecahkan suatu masalah tertentu.

### 9.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 8 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 8).
- IntelliJ Ultimate 64 bit (Students Key).

### 9.3 Dasar Teori

#### 9.3.1 Definisi, Jenis Polimorfisme dan Kegunaannya

Polimorfisme (polymorphism) berasal dari kata *poly* (bermakna banyak) dan *morphos* (bermakna bentuk) dalam bahasa Yunani. Polimorfisme adalah kemampuan obyek-obyek yang berbeda untuk memberi respon terhadap permintaan yang sama, sesuai dengan cara dan bentuk masing-masing obyek tersebut. Ada dua jenis polimorfisme, yaitu:

1. Satu nama, banyak bentuk (jenis 1). Contoh: mobil adalah kendaraan, sepeda adalah kendaraan. Kendaraan bisa berupa mobil, juga bisa berupa sepeda. Mobil dan sepeda adalah

kendaraan. Polimorfisme jenis ini dicapai dengan menerapkan inheritance, overriding dan substitution.

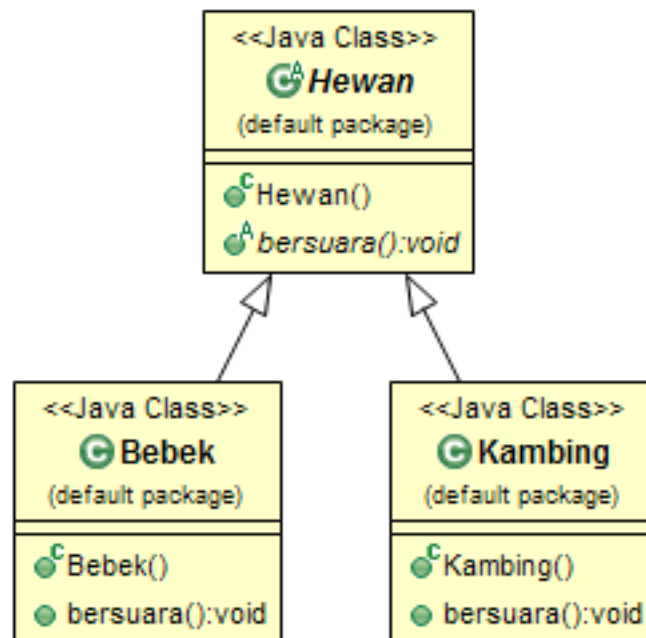
2. Satu nama, banyak cara untuk melakukannya (jenis 2). Misalnya bersuara. Anjing, ular dan jangkrik dapat bersuara. Suara yang dihasilkan berbeda, dan cara untuk menghasilkan suara tersebut juga berbeda. Polimorfisme jenis ini dicapai dengan menerapkan overloading.

Berikut ini adalah beberapa kegunaan dari polimorfisme yang dapat dimanfaatkan:

- Menghasilkan substitutability, dimana sebuah superclass dapat menggantikan seluruh subclassnya. Hal ini akan menyebabkan program lebih fleksibel dan lebih mudah dalam mengimplementasikan perubahan/penambahan kemampuan jika diperlukan.
- Mengurangi redudansi dan menyederhanakan implementasi.
- Mudah dikembangkan jika ada kebutuhan tambahan tanpa perlu mengubah yang sudah berjalan dengan baik.

### 9.3.2 Penerapan Polimorfisme

Contoh penerapan polimorfisme dapat dilihat dari relasi class Hewan, Bebek dan Kambing seperti yang ditunjukkan pada Gambar 9.1.



Gambar 9.1: Penerapan polimorfisme sederhana

Class Hewan didefinisikan sebagai abstract class karena method bersuara() belum dapat didefinisikan. Class Bebek dan Kambing merupakan subclass dari class Hewan, sehingga kedua class tersebut harus mengimplementasikan method bersuara(). Pada class Bebek dan Kambing kita dapat membuat implementasi dari method bersuara(). Implementasi untuk ketiga class tersebut dapat dilihat pada potongan kode program berikut ini:

```

1 //class Hewan (abstract class)
2 public abstract class Hewan {
3     public abstract void bersuara();
4 }
5

```

---

```

6  //class Bebek
7  public class Bebek extends Hewan {
8      @Override
9      public void bersuara(){
10         System.out.println("Kwek.. kwek .. kwek ..");
11     }
12 }
13
14 //class Kambing
15 public class Kambing extends Hewan {
16     @Override
17     public void bersuara(){
18         System.out.println("Mbeeeekk... Mbeeeekk...");
19     }
20 }

```

---

Secara umum dari diagram dan potongan kode program tersebut, ada beberapa hal yang perlu diperhatikan sebagai berikut:

- Class Hewan merupakan superclass dari class Bebek dan Kambing. Hubungan tersebut dapat menimbulkan sifat *substitutability*, dalam artian class Bebek dan class Kambing dapat digantikan dengan class Hewan.
- Relasi ketiga class tersebut menunjukkan bahwa Bebek adalah Hewan, Kambing adalah Hewan. Setiap hewan dapat bersuara, seperti halnya Bebek dan Kambing yang dapat bersuara.
- Bebek dan Kambing dapat bersuara, tetapi keduanya memiliki cara dan hasil yang berbeda saat bersuara.

Penggunaan sifat-sifat polimorfisme dapat dilihat pada potongan kode program berikut ini:

---

```

1  public class Main {
2      public static void main(String[] args){
3          Hewan donaldDuck = new Bebek();
4          donaldDuck.bersuara();           //output: Kwek.. kwek .. kwek ..
5
6          Hewan goatie = new Kambing();
7          goatie.bersuara();              //output: Mbeeeekk... Mbeeeekk...
8      }
9  }

```

---

Pada potongan kode program tersebut, kita mendefinisikan object bernama donaldDuck dari class Hewan, tetapi menggunakan constructor dari class Bebek. Hal ini dapat dilakukan karena class Hewan merupakan superclass dari class Bebek (sesuai sifat substitutability). Demikian juga halnya dengan object goatie yang didefinisikan bertipe Hewan tetapi diinstansiasi dengan menggunakan constructor dari class Kambing. Contoh ini menggambarkan penerapan polimorfisme jenis 1, yaitu satu nama banyak bentuk. Class Hewan dapat berupa Bebek atau Kambing.

Object donaldDuck dan goatie bertipe Hewan sehingga method-method yang dapat dipanggil hanya yang berasal dari class Hewan. Pada pemanggilan donaldDuck.bersuara(), output yang dihasilkan adalah **Kwek.. kwek .. kwek ..** karena object donaldDuck diinstansiasi menggunakan constructor dari class Bebek. Contoh ini menggambarkan penerapan polimorfisme jenis 2, yaitu satu nama banyak cara melakukannya. Pemanggilan method bersuara() pada object bertipe Hewan

dapat menghasilkan output yang berbeda-beda karena object tersebut walaupun bertipe sama, tetapi diinstansiasi dengan menggunakan constructor dari class yang berbeda.

Tambahkan satu method static pada class Main bernama tampilSuara, sebagai berikut:

---

```

1 public class Main {
2
3     public static void main(String[] args) {
4         // TODO Auto-generated method stub
5         Hewan donaldDuck = new Bebek();
6         donaldDuck.bersuara();           //output: Kwek.. kwek .. kwek ..
7
8         Hewan goatie = new Kambing();
9         goatie.bersuara();               //output: Mbeeeekk... Mbeeeekk...
10
11         tampilSuara(donaldDuck);
12         tampilSuara(goatie);
13     }
14
15     static void tampilSuara(Hewan h) {
16         h.bersuara();
17     }
18
19 }

```

---

Dilihat dari program di atas, void tampilSuara menerima input bertipe Hewan. Dengan menerima input bertipe Hewan, maka semua obyek yang bertipe Hewan dapat diterima oleh method tersebut, dan di dalam method tersebut hanya perlu memanggil method bersuara() sesuai dengan method bersuara pada class Hewan yang bersifat abstract dan telah dioverride pada class anak-anaknya.

## 9.4 Kegiatan Praktikum

Silahkan dibuat semua class di atas. Tambahkan pada class Kambing method berjalan() sebagai berikut:

---

```

1 public class Kambing extends Hewan {
2     //method bersuara tidak ditulis ..
3     //....
4     public void berjalan(){
5         System.out.println("Kambing berjalan!");
6     }
7 }

```

---

Tambahkan juga pada class Bebek method berenang() sebagai berikut:

---

```

1 public class Bebek extends Hewan {
2     //method bersuara tidak ditulis
3     //...
4     public void berenang() {

```

---

```

5         System.out.println("Bebek berenang!");
6     }
7 }

```

Jika dilihat maka method tambahan class Bebek berbeda dari class Kambing karena memang menjadi ciri khas masing-masing class. Tambahkan juga method static void tampilMethodTambahkan(Hewan h) pada class Main untuk menjalankan method berenang dan berjalan pada class Bebek dan Kambing. Apakah bisa dilakukan? Jika tidak bisa, bagaimana agar bisa dilakukan?

```

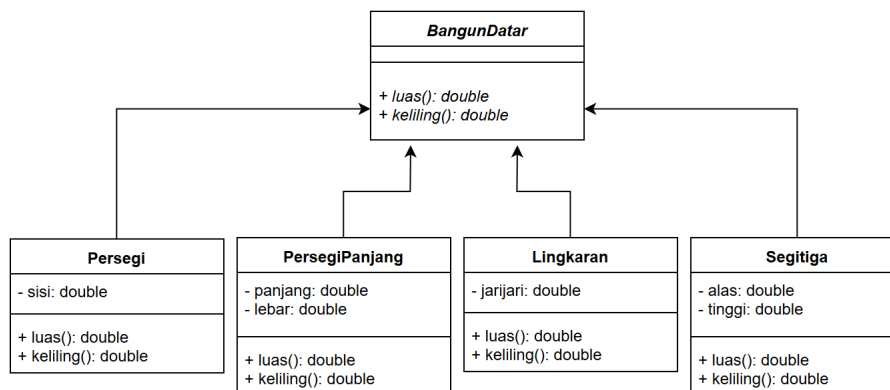
1 static void tampilMethodTambahkan(Hewan h) {
2     if(h instanceof Bebek) {
3         Bebek b = (Bebek) h;
4         b.berenang();
5     } else if(h instanceof Kambing) {
6         Kambing k = (Kambing) h;
7         k.berjalan();
8     }
9 }

```

## 9.5 Latihan Mandiri

### 9.5.1 Soal 1

Buatlah program Java untuk menghitung luas dan keliling BangunDatar seperti pada latihan-latihan sebelumnya. Class diagram yang harus dibuat dapat dilihat pada gambar 9.2.



Gambar 9.2: Soal Mandiri

Kemudian buatlah method static void hitungLuas(BangunDatar b) di class Main. Buat juga method static void hitungKeliling(BangunDatar b) di class Main. Masukkan sebagai input semua class turunan bangun datar untuk dihitung luas dan kelilingnya melalui kedua method tersebut di Main.

### 9.5.2 Soal 2

Buatlah program lengkap untuk kasus Math Expression Evaluator yang sudah dibahas di kelas. Buatlah dalam program utuh dan tambahkan minimal 1 operator unary dan 1 operator binary lainnya dengan bebas. Misalnya: operator sin, tan, pangkat, akar, dan lain-lain yang belum ada.



## 10. Penanganan Exception

### 10.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Dapat menjelaskan konsep exception pada Java.
2. Dapat merancang dan mengimplementasikan penerapan exception pada suatu kasus tertentu.

### 10.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 8 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 8).
- IntelliJ Ultimate 64 bit (Students Key).

### 10.3 Dasar Teori

#### 10.3.1 Definisi, Jenis Exception dan Kegunaannya

Exception adalah suatu kejadian yang tidak seharusnya terjadi pada suatu jalannya alur program. Exception menandakan ada masalah dalam alur program yang sedang berjalan. Exception perlu ditangani dengan benar agar program tetap dapat berjalan dengan baik walaupun ada masalah. Dengan penanganan yang tepat program akan terbebas dari crash. Exception terjadi saat program sedang berjalan (running). Contoh exception adalah:

1. File tidak ditemukan pada suatu path yang diberikan
2. Pembagian dengan nilai 0
3. Koneksi internet terputus

4. Pointer exception
5. Hak akses (permission) error
6. dan masih banyak lagi ...

Exception dapat digunakan untuk menangani kasus-kasus yang bersifat synchronous error , yaitu error yang terjadi pada program, bukan error yang berasal dari luar program. Contoh synchronous error adalah: division by zero, file not found, pointer exception, permission denied, number format, invalid method parameter, dan lain-lain. Sedangkan contoh untuk asynchronous error adalah power outage, network connection failed, mouse click error, dan lain sebagainya. Contoh synchronous error pada program Java:

---

```

1 //Class pembagian
2 public class Bagi{
3     public static void main(String args[]){
4         int a=5;
5         int b=0;
6         float c=a/b;
7         System.out.println("Hasil bagi adalah: " + c);
8         System.out.println("Terima kasih.");
9     }
10 }
```

---

Kode di atas digunakan untuk membagi 2 buah bilangan. Jika program tersebut dieksekusi maka akan terjadi synchronous error yaitu pembagian dengan 0 (division by zero) dengan output error sebagai berikut: Bagian output Terima kasih tidak keluar karena program sudah crash terlebih

```

Exception in thread "main" java.lang.ArithmeticException: / by zero
    at Bagi.main(Bagi.java:5)

Process completed.
```

Gambar 10.1: Contoh terjadi synchronous error

dahulu.

Agar tidak crash, error handling bisa dilakukan dengan dua cara, yaitu menggunakan inline error handling maupun dengan menggunakan exception. Contoh inline error handling:

---

```

1 public class Bagi{
2     public static void main(String args[]){
3         int a=5;
4         int b=0;
5         if(b!=0){
6             float c=a/b;
7             System.out.println("Hasil bagi adalah: " + c);
8         }
9         else
10             System.out.println ("Pembagi tidak boleh 0!");
11         System.out.println ("Terima kasih");
12     }
13 }
```

---



```
Pembagi tidak boleh 0!
Terima kasih
```

Gambar 10.2: Contoh inline error handling

Output: Dengan menggunakan inline error handling, pekerjaan menjadi lebih sulit karena semua kemungkinan error harus dihandle, kode program menjadi tercampur antara handle error dan program sebenarnya. Dengan menggunakan inline error handling juga harus tahu pasti apa yang harus dihandle. Bagian output Terima kasih tetap keluar karena error sudah dihandle.

### 10.3.2 Penerapan Exception

Untuk menggunakan exception pada Java dilakukan dengan cara sebagai berikut:

1. *try-catch*
2. *try-catch-finally*

Contoh **try-catch** adalah:

```
1 public class Bagi{
2     public static void main(String args[]){
3         int a=5;
4         int b=0;
5         try {
6             float c=a/b;
7             System.out.println("Hasil bagi adalah: " + c);
8         }
9         catch (Exception ex) {
10            System.out.println ("Error: Pembagi tidak boleh 0!");
11        }
12        System.out.println("Terima kasih");
13    }
14 }
```

Output:

```
Error: Pembagi tidak boleh 0!
Terima kasih
```

Gambar 10.3: Contoh try catch

Dengan menggunakan try catch, maka program akan menjalankan try dengan percobaan, jika ternyata error, maka error akan dicatch di bagian catch sesuai dengan errornya. Jika errornya spesifik dan dapat diprediksi, penyebutan Exception bisa ditulis dengan jelas. Salah satu contoh exception adalah `DivisionByZeroException`. Namun jika tidak jelas exceptionnya, maka bisa disebutkan class `Exception` sebagai class hirarki exception tertinggi. Tulisan terima kasih tetap ditampilkan karena error sudah dihandle dengan baik.

Contoh **try-catch-finally** adalah:

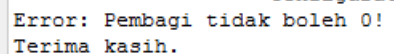
```
1 public class Bagi{
2     public static void main(String args[]){
```

```

3         int a=5;
4         int b=0;
5         try {
6             float c=a/b;
7             System.out.println("Hasil bagi adalah: " + c);
8         }
9         catch (Exception ex) {
10            System.out.println ("Error: Pembagi tidak boleh 0!");
11        }
12        finally{
13            System.out.println ("Terima kasih.");
14        }
15    }
16 }

```

Output: Pada contoh menggunakan try catch finally, hampir sama dengan yang sebelumnya, namun



```

Error: Pembagi tidak boleh 0!
Terima kasih.

```

Gambar 10.4: Contoh try catch finally

ada perbedaan yaitu bagian finally akan selalu dikerjakan dan ditampilkan.

Pada Exception, terdapat keyword lain yaitu:

1. *throw*
2. *throws*

Keyword throw digunakan untuk melempar error pada method body secara eksplisit. Throws digunakan untuk melempar error pada bagian method title. Contoh:

```

1  ...
2  void myMethod() {
3      try {
4          //melempar arithmetic exception menggunakan throw
5          throw new ArithmeticException("Ada kesalahan aritmatika");
6      }
7      catch (Exception exp) {
8          System.out.println("Error: "+exp.getMessage());
9      }
10 }
11 ...
12
13 ...
14 //Melempar error arithmetic dan null pointer exception menggunakan throws
15 void sample() throws ArithmeticException, NullPointerException{
16     //Statements
17 }
18 ...

```

Contoh kode program **throw**:

```
1 public class ContohThrow{
2     void checkAge(int age){
3         if(age<18)
4             throw new ArithmeticException("Tidak bisa memilih Pemilu");
5         else
6             System.out.println("Bisa memilih Pemilu");
7     }
8
9     public static void main(String args[]){
10        ContohThrow obj = new ContohThrow();
11        obj.checkAge(13);
12        System.out.println("Selesai.");
13    }
14 }
```

Output:

Exception in thread "main" java.lang.ArithmeticException: Tidak bisa memilih Pemilu  
at ContohThrow.checkAge(ContohThrow.java:4)  
at ContohThrow.main(ContohThrow.java:11)

Contoh kode program **throws**:

```
1 public class ContohThrows{
2     public int division(int a, int b) throws ArithmeticException{
3         int t = a/b;
4         return t;
5     }
6
7     public static void main(String args[]){
8        ContohThrows obj = new ContohThrows();
9        try{
10            System.out.println(obj.division(15,0));
11        }
12        catch(ArithmeticException e){
13            System.out.println("tidak bisa pembagian dengan 0");
14        }
15    }
16 }
```

Output:

tidak bisa pembagian dengan 0

## 10.4 Kegiatan Praktikum

### 10.4.1 Exception Password

Buatlah sebuah class Exception yang custom untuk menangani error pada sebuah inputan password. Terdapat beberapa hal yang menyebabkan password salah:

- Password tidak boleh kosong

- Password minimal 8 karakter, maksimal 50 karakter
- Password harus mengandung angka, huruf besar, dan huruf kecil
- Password tidak boleh sama dengan nama pengguna

Program meminta pengguna memasukkan username dan password untuk disimpan dan kemudian program akan memvalidasi dengan menggunakan custom exception untuk passwordnya.

Cara pengerjaan (model 1):

1. Buatlah class **PasswordException** untuk penanganan custom Exception Buatlah kelas PasswordException yang extends ke Exception

---

```

1  public class PasswordException extends Exception{
2      private int errCode;
3      private String errMessage;
4      public PasswordException(int errCode) {
5          super();
6          this.errCode = errCode;
7          if(errCode==1){
8              this.errMessage = "Password tidak boleh kosong!";
9          }
10         else if(errCode==2){
11             this.errMessage = "Password harus antara 8-50 karakter!";
12         }
13         else if(errCode==3){
14             this.errMessage = "Password harus mengandung huruf besar,
15             kecil, dan angka!";
16         }
17         else if(errCode==4){
18             this.errMessage = "Password tidak boleh sama
19             dengan nama pengguna!";
20         }
21     }
22
23     public PasswordException(String errorMessage){
24         super(errorMessage);
25     }
26
27     public int getErrorCode(){
28         return this.errCode;
29     }
30
31     public String getErrorMessage(){
32         return this.errMessage;
33     }
34 }

```

---

2. Buatlah class **PasswordTest** sebagai program utama Buatlah kelas PasswordTest sebagai berikut.

---

```

1  public class PasswordTest {
2
3      public static void main (String[] args) throws PasswordException {

```

```

4      String username = System.console().readLine("Nama pengguna: ");
5      char[] pass = System.console().readPassword("Password: ");
6      String password = new String(pass);
7      try{
8          if(password.isEmpty()){
9              throw new PasswordException(1);
10     } else if(password.length() < 8 || password.length() > 50){
11         throw new PasswordException(2);
12     } else if(password.equals(username)){
13         throw new PasswordException(4);
14     } else {
15         boolean flagKecil = false;
16         boolean flagBesar = false;
17         boolean flagAngka = false;
18         for(int i=0;i<password.length();i++){
19             char c = password.charAt(i);
20             if(Character.isLetter(c) &&
21                 Character.isLowerCase(c))
22                 flagKecil = true;
23             if(Character.isLetter(c) &&
24                 Character.isUpperCase(c))
25                 flagBesar = true;
26             if(Character.isDigit(c))
27                 flagAngka = true;
28             }
29             if(!(flagKecil &&
30                 flagAngka && flagBesar)){
31                 throw new PasswordException(3);
32             }
33         }
34     } catch(PasswordException e){
35         System.out.println ("Error : " + e.getErrorMessage());
36     }
37 }
38 }

```

Cara pengerjaan (model 2):

1. Buatlah class PasswordEmptyException, PasswordLengthException, PasswordSameException, PasswordValidationException untuk masing-masing exception sesuai soal.

```

1      public class PasswordEmptyException extends Exception {
2          public PasswordEmptyException(){
3              super("Password tidak boleh kosong!");
4          }
5      }
6
7      public class PasswordLengthException extends Exception {
8          public PasswordLengthException() {
9              super("Password harus antara 8-50 karakter!");
10         }

```

```

11     }
12
13     public class PasswordSameException extends Exception {
14         public PasswordSameException() {
15             super("Password tidak boleh sama dengan nama pengguna!");
16         }
17     }
18
19     public class PasswordValidationException extends Exception {
20         public PasswordValidationException() {
21             super("Password harus mengandung huruf besar, kecil, dan angka");
22         }
23     }

```

## 2. Buatlah class Login.

```

1     public class Login {
2         private String username;
3         private String password;
4         public Login(String username, String password) throws PasswordEmptyException {
5             boolean ok = true;
6             this.username = username;
7             if(password.isEmpty()){
8                 ok = false;
9                 throw new PasswordEmptyException();
10            } else if(password.length() < 8 || password.length() > 50){
11                ok = false;
12                throw new PasswordLengthException();
13            } else if(password.equals(username)){
14                ok = false;
15                throw new PasswordSameException();
16            } else {
17                boolean flagKecil = false;
18                boolean flagBesar = false;
19                boolean flagAngka = false;
20                for(int i=0; i<password.length(); i++){
21                    char c = password.charAt(i);
22                    if(Character.isLetter(c) && Character.isLowerCase(c)){
23                        flagKecil = true;
24                    }
25                    if(Character.isLetter(c) && Character.isUpperCase(c)){
26                        flagBesar = true;
27                    }
28                    if(Character.isDigit(c)){
29                        flagAngka = true;
30                    }
31                }
32                if(!(flagKecil && flagAngka && flagBesar)){
33                    ok = false;
34                    throw new PasswordValidationException();
35                }
36            }
37            if(ok) this.password = password;
38        }
39    }

```

```

35         else this.password = "";
36     }
37
38     public void show() {
39         System.out.println(this.username);
40         System.out.println(this.password);
41     }
42
43     public static void main(String[] args) {
44         try {
45             Login l = new Login("anton", "anton");
46             l.show();
47         } catch (PasswordEmptyException e) {
48             e.printStackTrace();
49         } catch (PasswordLengthException e) {
50             e.printStackTrace();
51         } catch (PasswordSameException e) {
52             e.printStackTrace();
53         } catch (PasswordValidationException e) {
54             e.printStackTrace();
55         } finally {
56             System.out.println("Terima kasih");
57         }
58     }
59
60 }

```

## 10.5 Latihan Mandiri

### 10.5.1 IncorrectFileName Exception

Pada bagian ini Anda diminta membuat sebuah class custom Exception untuk handle input nama file yang salah pada sebuah aplikasi dengan ketentuan nama file sebagai berikut:

1. Input nama tidak boleh kosong, jika kosong ada exception: "Nama file tidak boleh kosong".
2. Input nama file huruf pertama harus huruf besar, jika tidak sesuai ada exception "Huruf pertama harus huruf besar".
3. Input nama file harus tidak boleh mengandung spasi, jika tidak sesuai ada exception "Tidak boleh mengandung spasi".
4. Input nama file harus mengandung ekstensi file misalnya .txt, .exe dsb. Jika tidak mengandung ekstensi yang benar ditampilkan exception "Harus memiliki ekstensi file".
5. Input ekstensi file harus memiliki ekstensi berjumlah minimal 3 karakter yang dipisahkan dengan titik, misalnya .txt, .text, .program dsb. Jika tidak mengandung ekstensi yang benar ditampilkan exception "Ekstensi file harus minimal 3 karakter".

Silahkan anda membuat class sebagai berikut:

1. **IncorrectFileNameException.java** yang digunakan untuk membuat custom exception seperti pada keterangan sebelumnya.
2. **IncorrectFileNameTest.java** yang berisi main program untuk testing aplikasi. Program ini berisi langkah sebagai berikut:
  - Program utama meminta pengguna memasukkan nama file dalam bentuk String, kemudian program utama akan melemparkan exception ke IncorrectFileNameException jika

memang ada kesalahan.

- Jika tidak ada kesalahan maka pengguna diminta memasukkan ukuran file dalam byte.
- Setelah itu program akan menampilkan nama file dan ukuran yang telah diinputkan sebelumnya tersebut.

**SELAMAT MENGERJAKAN.**





# PBO dan Java Tingkat Lanjut

<b>11</b>	<b>String Manipulation pada Java</b> .....	<b>97</b>
11.1	Tujuan Praktikum	
11.2	Alat dan Bahan	
11.3	Dasar Teori	
11.4	Kegiatan Praktikum	
11.5	Latihan Mandiri	
<b>12</b>	<b>Java Collection</b> .....	<b>103</b>
12.1	Tujuan Praktikum	
12.2	Alat dan Bahan	
12.3	Dasar Teori	
12.4	Latihan Praktikum	
<b>13</b>	<b>Java IO, Serialisasi, dan Deserialisasi</b>	<b>121</b>
13.1	Tujuan Praktikum	
13.2	Alat dan Bahan	
13.3	Dasar Teori	
<b>14</b>	<b>Package dan JDBC</b> .....	<b>131</b>
14.1	Tujuan Praktikum	
14.2	Alat dan Bahan	
14.3	Dasar Teori	



## 11. String Manipulation pada Java

### 11.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Dapat memahami karakteristik dari class String di Java.
2. Dapat menggunakan method-method di dalam class String dan class-class pendukung lainnya untuk memecahkan masalah dalam pengolahan String dan karakter.

### 11.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 8 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 8).
- IntelliJ Ultimate 64 bit (Students Key).

### 11.3 Dasar Teori

#### 11.3.1 Class String pada Java

Class String pada Java bersifat *immutable*, yaitu isi/nilainya tidak bisa diubah setelah dibuat. Seperti halnya pada bahasa pemrograman lainnya, String pada Java terdiri dari satu atau lebih karakter. String pada bahasa pemrograman C/C++ adalah array of characters, sedangkan pada Java, String adalah tipe data reference.

Pada class String, terdapat banyak method-method yang bisa digunakan, beberapa diantaranya dapat dilihat pada Tabel 11.1.

Tabel 11.1: Beberapa methods di class String

Method	Kegunaan
<code>int length()</code>	Panjang dari String
<code>boolean isEmpty()</code>	Mengecek apakah merupakan String kosong ( <code>length = 0</code> )
<code>boolean equals(String other)</code>	Mengecek kesamaan dua String (tidak bisa menggunakan <code>==</code> atau <code>!=</code> )
<code>boolean equalsIgnoreCase(String other)</code>	Mengecek kesamaan dua String secara case-insensitive
<code>int compareTo(String other)</code>	Mengecek kesamaan dua String, mengembalikan nilai -1, 0 atau 1
<code>int indexOf(String search)</code>	Mencari posisi substring dari suatu String
<code>char charAt(int index)</code>	Mengambil karakter pada posisi ke-index (mulai dari 0)
<code>String toUpperCase()</code>	Menghasilkan String baru yang sudah diubah menjadi huruf besar semua
<code>String toLowerCase()</code>	Menghasilkan String baru yang sudah diubah menjadi huruf kecil semua
<code>char[] toCharArray()</code>	Menghasilkan array of characters dari suatu String
<code>String[] split(String regex)</code>	Memisahkan substring berdasarkan aturan regex (tokenisasi)
<code>static String format(String format, objects args...)</code>	Seperti <code>printf()</code> pada C

### 11.3.2 Sifat dan Karakteristik Class String

Class String memiliki beberapa keistimewaan sebagai berikut:

- Bersifat immutable. Setiap operasi pada String akan menghasilkan String baru.
- Operator `+` dapat digunakan dalam operasi String, gunanya untuk menyambung dua String atau lebih menjadi satu String.
- String dapat dibuat dengan dua cara, yaitu literal dan instansiasi object dari class String.
  - Secara literal artinya object String baru dapat dibuat dan langsung diberi nilai, sama seperti tipe data primitive.
  - Menggunakan `new String()` untuk menginstansiasi sebuah String. Tetapi cara ini jarang dipakai dan tidak direkomendasikan.

Potongan kode program berikut ini menunjukkan bagaimana membuat sebuah object String baik secara literal maupun instansiasi object:

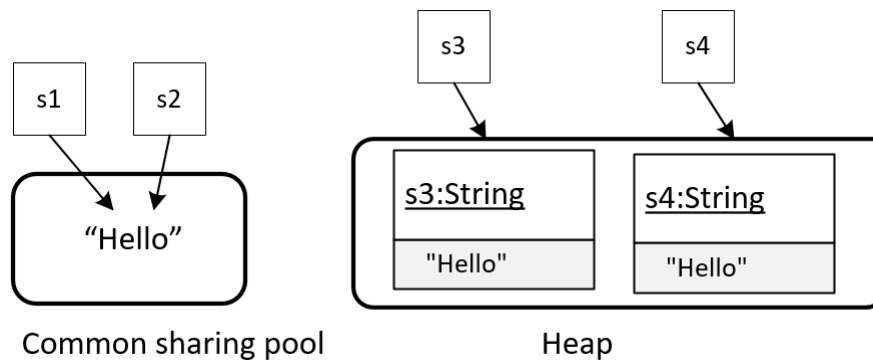
```

1      ...
2      String s1 = "Hello";
3      String s2 = "Hello";
4      String s3 = new String("Hello");
5      String s4 = new String("Hello");
6      ...

```

Dari potongan kode tersebut, dapat dilihat bahwa object `s1` dan `s2` dibuat dengan cara mengisi nilai

literal "Hello" ke dalam s1 dan s2. Sedangkan object s3 dan s4 dibuat dengan cara instansiasi dari class String. Cara pembuatan secara literal lebih direkomendasikan karena akan menghemat sumber daya (terutama memory), karena pada pembuatan secara literal, nilai dari object String disimpan pada suatu lokasi yang sama, sedangkan pada pembuatan secara instansiasi akan membutuhkan memory lebih banyak di heap. Diagram yang menggambarkan hal tersebut dapat dilihat pada Gambar 11.1.



Gambar 11.1: Perbedaan cara literal dan instansiasi object baru

## 11.4 Kegiatan Praktikum

### 11.4.1 Mengecek Kesamaan Dua String

Untuk mengecek kesamaan dua String dapat dilakukan dengan menggunakan method equals() atau equalsIgnoreCase(). Sedangkan operator == dan != digunakan untuk mengecek apakah dua String tersebut mengacu pada reference yang sama. Cobalah menjalankan contoh program berikut ini dan amati output yang dihasilkan:

```

1
2     public class Main{
3         public static void main(String[] args){
4             String s1 = "Hello";
5             String s2 = "Hello";
6             String s3 = new String("Hello");
7             String s4 = new String("Hello");
8
9             //tampilkan isi dari s1, s2, s3 dan s4
10            System.out.println(s1);
11            System.out.println(s2);
12            System.out.println(s3);
13            System.out.println(s4);
14
15            //cek kesamaan
16            System.out.println(s1==s1);
17            System.out.println(s1==s2);
18            System.out.println(s2==s3);
19            System.out.println(s1.equals(s2));
20            System.out.println(s1.equals(s3));

```

---

```

21         System.out.println(s3==s4);
22         System.out.println(s3.equals(s4));
23     }
24 }
25

```

---

### 11.4.2 Menggunakan Method-method pada Class String

Perlu anda ingat sekali lagi bahwa class String bersifat immutable, sehingga method-method yang ada di class String akan mengembalikan sebuah String baru, tidak secara otomatis mengubah isi dari String yang sudah ada. Perhatikan contoh program berikut ini yang menunjukkan sifat immutable dari class String:

---

```

1     public class Main{
2         public static void main(String[] args){
3             String s1 = "Hello World";
4             String s2 = "Hello World";
5
6             //operasi pada s1
7             s1.toUpperCase();
8             System.out.println(s1);
9             s1 = s1.toUpperCase();
10            System.out.println(s1);
11
12            //operasi pada s2
13            s2.concat(" Java");
14            System.out.println(s2);
15            s2 = s2.concat(" Java");
16            System.out.println(s2);
17        }
18    }

```

---

### 11.4.3 Penggunaan Class StringBuilder

Karena class String bersifat immutable, maka setiap operasi pada String mungkin saja memerlukan alokasi memory tambahan, seperti pada contoh potongan kode program berikut ini:

---

```

1     public class Main{
2         public static void main(String[] args){
3             //dengan operasi +
4             String tanggal = "17";
5             String bulan = "08";
6             String tahun = "1945";
7             String hariMerdeka = tanggal + "-" + bulan + "-" + tahun;
8             System.out.println(hariMerdeka);
9
10            //dengan StringBuilder
11            StringBuilder sb = new StringBuilder();
12            sb.append(tanggal);

```

```
13         sb.append("-");
14         sb.append(bulan);
15         sb.append("-");
16         sb.append(tahun);
17
18         String hariMerdeka2 = sb.toString();
19         System.out.println(hariMerdeka2);
20     }
21 }
```

## 11.5 Latihan Mandiri

Dengan menggunakan method-method yang ada di class String dan StringBuilder, buatlah program untuk masing-masing permasalahan berikut ini:

1. Buatlah program untuk mengecek apakah suatu kalimat yang dimasukkan oleh pengguna merupakan palindrome atau tidak!
  - Pengecekan palindrome harus mengabaikan karakter selain alfabet(a-z dan A-Z). Sebagai contoh kalimat "A man, a plan, a canal – Panama!" merupakan sebuah palindrome.
  - Kalimat "A man, a plan, a canoe, pasta, heros, rajahs, a coloratura, maps, snipe, percale, macaroni, a gag, a banana bag, a tan, a tag, a banana bag again (or a camel), a crepe, pins, Spam, a rut, a Rolo, cash, a jar, sore hats, a peon, a canal – Panama!" adalah palindrome.
  - Kalimat "Doc, note: I dissent. A fast never prevents a fatness. I diet on cod" adalah palindrome.
2. Buatlah program untuk mengubah setiap huruf pertama pada suatu kata menjadi huruf besar.
  - Kalimat input: "today is a beautiful day", menghasilkan output: "Today Is A Beautiful Day".
  - Kalimat input: "Aku telah menunggumu selama 99 hari, 99 malam", menghasilkan output: "Aku Telah Menunggumu Selama 99 Hari, 99 Malam".
3. Buatlah program untuk menghapus kata tertentu dari suatu kalimat yang dimasukkan oleh pengguna.
  - Kalimat input: "hari libur ini saya hanya tidur-tiduran di rumah". Kata yang dihapus: "saya". Output yang dihasilkan: "hari libur ini hanya tidur-tiduran di rumah".
4. Buatlah program untuk membalik urutan kata dari sebuah kalimat.
  - Kalimat input: "saya tidak malas belajar"
  - Kalimat output: "belajar malas tidak saya"

**SELAMAT MENGERJAKAN.**





## 12. Java Collection

### 12.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Dapat mendefinisikan array dan collection pada Java.
2. Dapat menggunakan array pada Java.
3. Dapat menggunakan Collection pada Java.

### 12.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 8 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 8).
- IntelliJ Ultimate 64 bit (Students Key).

### 12.3 Dasar Teori

#### 12.3.1 Array

Array adalah struktur data yang dapat menyimpan data yang bersifat homogen (sejenis). Array pada Java agak berbeda dengan bahasa C, karena array pada Java dapat bersifat dinamis dan merupakan tipe data references (bukan primitif).

Cara pembuatan array pada Java:

---

```
1      ...
2      //pembuatan array integer sebanyak 10 buah
```

---

```

3      int[] umur = new int[10];
4      //pembuatan array String sebanyak 5 buah
5      String[] nama = new String[5];
6      //pembuatan array data dengan jumlah dinamis
7      int[] data;
8      //inisialisasi array dengan 5 buah data bertipe integer
9      data = new int[5];
10     //sama saja dengan int[10] umur;
11     int umur[10];
12     // sama saja dengan int a; int[] b;
13     int a,b[];
14     ...

```

---

Cara pengaksesan data array pada Java adalah menggunakan indeksinya (mulai dari 0), sebagai berikut:

---

```

1      ...
2      //pembuatan array integer sebanyak 10 buah
3      int[] umur = new int[10];
4      umur[0] = 17;
5      umur[1] = 35;
6      System.out.println("Umur ke-2 adalah: " + umur[1]);
7      ...

```

---

Tipe data array dapat digunakan untuk menyimpan data dalam bentuk reference obyek seperti berikut:

---

```

1      ...
2      Dosen[5] datadosen;
3      datadosen[0] = new Dosen();
4      datadosen[1] = new Dosen();
5      ...

```

---

Untuk data multi dimensi pada Java, dapat dibuat seperti pada bahasa C sebagai berikut:

---

```

1      ...
2      //membuat array 2 dimensi integer 3 x 4
3      int[] [] a = new int[3][4];
4      //array 3 dimensi of String
5      String[] [] [] personalInfo = new String[3][4][2];
6      ...

```

---

Untuk menginisialisasi data pada array digunakan cara sebagai berikut:

---

```

1      ...
2      //inisialisasi array 1 dimensi dengan keyword new
3      int[] data = new int[]{1,2,3,4};
4      //inisialisasi array 2 dimensi secara langsung

```

---

```

5      int[] [] a = {{1, 2, 3}, {4, 5, 6, 9}, {7}};
6      int[] [] [] test = { {{1, -2, 3}, {2, 3, 4}}, {{-4, -5, 6, 9}, {1}, {2, 3}} }; //inisialisasi array 3 dimensi
7      ...

```

Karena array pada Java bertipe References (Object), maka obyek array memiliki atribut `length` untuk mengetahui panjang (ukuran) array. Cara penggunaan:

```

1      ...
2      //inisialisasi array 1 dimensi dengan keyword new
3      int[] data = new int[]{1,2,3,4};
4      System.out.println("Banyak data: " + data.length);
5      ...

```

Karena array adalah Object, maka Java juga menyediakan class `Arrays` yang merupakan static class dengan beberapa method yang berguna yaitu:

- method **List asList(T..a)**. Method ini digunakan untuk mengembalikan array sebagai tipe data List. Contoh:

```

1      ...
2      int intArr[] = { 10, 20, 15, 22, 35 };
3      System.out.println("Integer Array sebagai List: " + Arrays.asList(intArr));
4      ...

```

- **int binarySearch(elementToBeSearched)**. Method ini digunakan untuk melakukan pencarian binary search dari data sebuah array. Contoh:

```

1      ...
2      int intArr[] = { 10, 20, 15, 22, 35 };
3      Arrays.sort(intArr);
4      int intKey = 22;
5      System.out.println(intKey + " ditemukan di index = " + Arrays.binarySearch(intArr, intKey));
6      ...

```

- **Array copyOf(originalArray,newLength)**. Method ini digunakan untuk melakukan copy array ke array lain. Contoh:

```

1      ...
2      int intArr[] = { 10, 20, 15, 22, 35 };
3      int datacopy[] = Arrays.copyOf(intArr,intArr.length);
4      ...

```

- **Array copyOfRange(originalArray,fromIndex,toIndex)**. Method ini digunakan untuk melakukan copy array ke array lain dari indeks tertentu saja. Contoh:

```

1      ...
2      int intArr[] = { 10, 20, 15, 22, 35 };
3      int datacopy[] = Arrays.copyOfRange(intArr,1,3); //menghasilkan [20, 15, 22]
4      ...

```

- **boolean equals(array1,array2)**. Method ini digunakan untuk memastikan apakah dua buah array adalah identik atau tidak. Contoh:

---

```

1          ...
2          int intArr[] = { 10, 20, 15, 22, 35 };
3          int intArr2[] = { 10, 21, 15, 22, 35};
4          boolean hasil = Arrays.equals(intArr,intArr2); //return false
5          ...

```

---

- **fill(originalArray,fillValue).** Method ini digunakan untuk mengisi array dengan nilai tertentu sebanyak ukuran arraynya. Method ini akan mereplace isi array lama. Contoh:

---

```

1          ...
2          int intArr[] = { 10, 20, 15, 22, 35 };
3          Arrays.fill(intArr, 1000); //hasil = {1000, 1000, 1000, 1000, 1000}
4          ...

```

---

- **toString(array).** Untuk mengubah array menjadi String.
- **sort(array).** Untuk mengurutkan isi array.

Cara pengaksesan data array sangat mudah dengan looping biasa setiap elemennya seperti contoh berikut:

---

```

1          ...
2          String[] avengers = new String[10];
3          for(int i=0; i<avengers.length; i++) {
4              .....
5          }
6          ...

```

---

Sedangkan cara pengaksesan data array dengan menggunakan sintaks foreach adalah:

---

```

1          ...
2          String[] avengers = new String[10];
3          for(String hero: avengers) {
4              .....
5          }
6          ...

```

---

Berikut adalah contoh program untuk mendemonstrasikan penggunaan iterator:

---

```

1          import java.util.*;
2          class IteratorDemo {
3              public static void main(String args[]) {
4                  ArrayList<String> al = new ArrayList<String>();
5                  al.add("C");
6                  al.add("A");
7                  al.add("E");
8                  al.add("B");
9                  al.add("D");
10                 al.add("F");
11
12                 System.out.print("Isi data: ");

```

```

13         Iterator<String> itr = al.iterator();
14         while(itr.hasNext()) {
15             String element = itr.next();
16             System.out.print(element + " ");
17         }
18         System.out.println();
19
20         ListIterator<String> litr = al.listIterator();
21         while(litr.hasNext()) {
22             String element = litr.next();
23             litr.set(element + "+");
24         }
25         System.out.print("Hasil modifikasi: ");
26         itr = al.iterator();
27         while(itr.hasNext()) {
28             String element = itr.next();
29             System.out.print(element + " ");
30         }
31         System.out.println();
32
33         System.out.print("Tampilan terbalik: ");
34         while(litr.hasPrevious()){
35             String element = litr.previous();
36             System.out.print(element + " ");
37         }
38         System.out.println();
39     }
40 }

```

Output dari program adalah:

Isi data: C A E B D F

Hasil modifikasi: C+ A+ E+ B+ D+ F+

Tampilan terbalik: F+ D+ B+ E+ A+ C+

Contoh Penggunaan For Each adalah sebagai berikut:

```

1     import java.util.*;
2     class ForEachDemo {
3         public static void main(String args[]) {
4
5             ArrayList<Integer> vals = new ArrayList<Integer>();
6
7             vals.add(1);
8             vals.add(2);
9             vals.add(3);
10            vals.add(4);
11            vals.add(5);
12
13            System.out.print("Isi data: ");

```

```

14         for(int v : vals)
15             System.out.print(v + " ");
16         System.out.println();
17
18         int sum = 0;
19         for(int v : vals)
20             sum += v;
21         System.out.println("Jumlah: " + sum);
22     }
23 }

```

Output program adalah:

Isi data: 1 2 3 4 5

Jumlah: 15

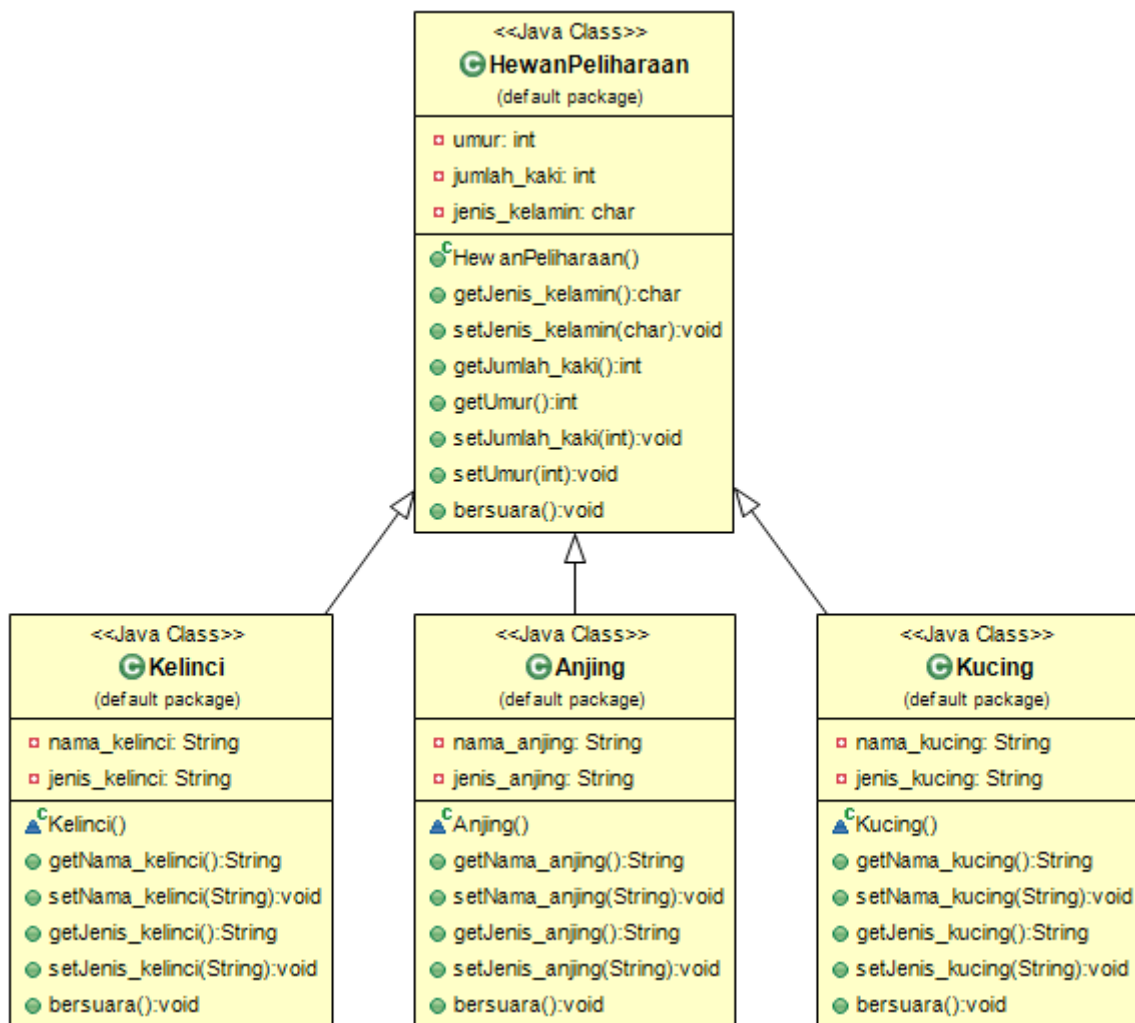
### Kegiatan Praktikum Array

Pada bagian ini terdapat sebuah latihan tentang array yang berupa Object. Terdapat sebuah class diagram HewanPeliharaan, Anjing, Kucing, dan Kelinci pada gambar 12.1 Kode program dari class diagram 12.1 adalah sebagai berikut:

```

1     public class HewanPeliharaan{
2         private int umur;
3         private int jumlah_kaki;
4         private char jenis_kelamin;
5
6         public char getJenis_kelamin() {
7             return jenis_kelamin;
8         }
9
10        public void setJenis_kelamin(char jenis_kelamin) {
11            this.jenis_kelamin = jenis_kelamin;
12        }
13
14        public int getJumlah_kaki(){
15            return this.jumlah_kaki;
16        }
17
18        public int getUmur(){
19            return this.umur;
20        }
21
22        public void setJumlah_kaki(int jml){
23            this.jumlah_kaki = jml;
24        }
25
26        public void setUmur(int u){
27            this.umur = u;
28        }
29    }

```



Gambar 12.1: Class Diagram HewanPeliharaan dan Anjing, Kucing, Kelinci

```

30         public void bersuara() {
31             System.out.println("...");
32         }
33     }
34 }

```

Kemudian kode untuk 3 jenis HewanPeliharaan yaitu Anjing, Kucing, dan Kelinci adalah sebagai berikut:

```

1     class Anjing extends HewanPeliharaan{
2         private String nama_anjing;
3         private String jenis_anjing;
4
5         public String getNama_anjing() {
6             return nama_anjing;
7         }
8         public void setNama_anjing(String nama_anjing) {
9             this.nama_anjing = nama_anjing;

```

```
10         }
11         public String getJenis_anjing() {
12             return jenis_anjing;
13         }
14         public void setJenis_anjing(String jenis_anjing) {
15             this.jenis_anjing = jenis_anjing;
16         }
17         public void bersuara() {
18             System.out.println("Guguguk");
19         }
20     }
21
22     class Kucing extends HewanPeliharaan{
23         private String nama_kucing;
24         private String jenis_kucing;
25
26         public String getNama_kucing() {
27             return nama_kucing;
28         }
29         public void setNama_kucing(String nama_kucing) {
30             this.nama_kucing = nama_kucing;
31         }
32         public String getJenis_kucing() {
33             return jenis_kucing;
34         }
35         public void setJenis_kucing(String jenis_kucing) {
36             this.jenis_kucing = jenis_kucing;
37         }
38         public void bersuara() {
39             System.out.println("Meooong");
40         }
41     }
42
43     class Kelinci extends HewanPeliharaan{
44         private String nama_kelinci;
45         private String jenis_kelinci;
46
47         public String getNama_kelinci() {
48             return nama_kelinci;
49         }
50         public void setNama_kelinci(String nama_kelinci) {
51             this.nama_kelinci = nama_kelinci;
52         }
53         public String getJenis_kelinci() {
54             return jenis_kelinci;
55         }
56         public void setJenis_kelinci(String jenis_kelinci) {
57             this.jenis_kelinci = jenis_kelinci;
58         }
59     }
```



```

59         public void bersuara() {
60             System.out.println("kriuk-kriuk");
61         }
62     }

```

Dan terakhir adalah dibuat kode program untuk test, dengan nama class PeliharaanTest sebagai berikut:

```

1     public class PeliharaanTest {
2         public static void main(String[] args) {
3             HewanPeliharaan[] hp = new HewanPeliharaan[2];
4             //HewanPeliharaan biasa
5             hp[0] = new HewanPeliharaan();
6             hp[0].setJenis_kelamin('L');
7             hp[0].setJumlah_kaki(4);
8             hp[0].setUmur(2);
9
10            //HewanPeliharaan yang dinyatakan dlm Anjing
11            hp[1] = new Anjing();
12            hp[1].setJenis_kelamin('P');
13            hp[1].setJumlah_kaki(4);
14            hp[1].setUmur(1);
15
16            //Anjing yang berisi HewanPeliharaan (Anjing)
17            Anjing[] anjing = new Anjing[2];
18            //Casting agar bisa diisi
19            //dengan HewanPeliharaan (Anjing)
20            anjing[0] = (Anjing) hp[1];
21            anjing[0].setJenis_anjing("puddle");
22            anjing[0].setNama_anjing("riki");
23            anjing[0].bersuara();
24
25            //Anjing biasa
26            anjing[1] = new Anjing();
27            anjing[1].setJenis_anjing("herder");
28            anjing[1].setNama_anjing("herboy");
29            anjing[1].bersuara();
30
31            //Polimorfisme, HewanPeliharaan
32            //yang dinyatakan dalam bentuk Kucing
33            HewanPeliharaan kucing = new Kucing();
34            kucing = hp[0];
35            kucing.bersuara();
36
37            //Kucing biasa
38            Kelinci kelinci = new Kelinci();
39            kelinci.setJenis_kelamin('P');
40            kelinci.setJumlah_kaki(2);
41            kelinci.setUmur(2);

```

```

42         kelinci.setJenis_kelinci("pendek");
43         kelinci.setNama_kelinci("rabit");
44         kelinci.bersuara();
45
46         //Polimorfisme, HewanPeliharaan
47         //yang dinyatakan dalam bentuk Kelinci dan diisi kelinci sebelumnya
48         HewanPeliharaan kelinci2 = new Kelinci();
49         kelinci2 = kelinci;
50         //Agar bisa menggunakan method dari kelinci
51         ((Kelinci) kelinci2).setNama_kelinci("bunny");
52         kelinci2.bersuara();
53     }
54
55 }
```

Keterangan dari program di atas dapat dibaca pada bagian komentar di dalam program. Output dari program di atas adalah:

```

Guguguk
Guguguk
....
kriuk-kriuk
kriuk-kriuk
```

### 12.3.2 Java Collection Framework

Java Collection Framework adalah kumpulan library dalam bentuk class yang siap digunakan berkaitan dengan struktur data yang mampu memanajemen data lebih dari satu (sering disebut dengan collection). Beberapa contoh class yang sudah umum pada Java Collection Framework adalah: **Stack**, **Vector**, **Dictionary**, dan **HashTable**. Class-class tersebut sudah disediakan oleh Java sehingga kita tinggal menggunakannya saja dengan lebih mudah. Java Collection Framework ini mirip dengan Standard Template Library (STL) pada bahasa C++. Beberapa kelebihan yang dimiliki JCF adalah:

- collection adalah object dinamis
- ukuran bersifat dinamis
- thread safe

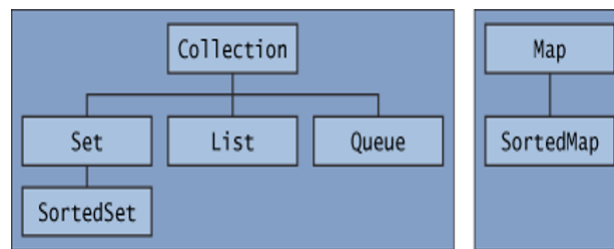
Beberapa hal yang perlu diperhatikan terkait abstract class:

- Abstract class tidak dapat diinstansiasi, karena mengandung satu atau lebih abstract method yang tidak memiliki implementasi.
- Abstract class tidak dapat dideklarasikan sebagai class private, karena abstract class didefinisikan dengan tujuan supaya di-extends dan di-override oleh class-class turunannya.

Terdapat 3 bagian utama dari collection pada Java:

- Interfaces: Spesifikasi kemampuan yang harus dimiliki oleh sebuah Collection (digunakan juga untuk polymorphism).
- Implementation (Classes): Implementasi konkret dari Interfaces.
- Algorithms: Metode untuk operasi-operasi pada Collection seperti pencarian, penambahan data, penghapusan data, dan sebagainya.

Gambar 12.2 adalah menunjukkan Interface dari Collection pada Java. Pada interface Collection terdapat beberapa method yang dimiliki dan diimplementasikan oleh semua class yang terdapat pada Collection. Berikut ada interface Collection:



Gambar 12.2: Interface Collection

```

1      public interface Collection<E> extends Iterable<E> {
2          // Basic operations
3          int size();
4          boolean isEmpty();
5          boolean contains(Object element);
6          boolean add(E element);           //optional
7          boolean remove(Object element);   //optional
8          Iterator<E> iterator();
9
10         // Bulk operations
11         boolean containsAll(Collection<?> c);
12         boolean addAll(Collection<? extends E> c); //optional
13         boolean removeAll(Collection<?> c);       //optional
14         boolean retainAll(Collection<?> c);        //optional
15         void clear();                             //optional
16
17         // Array operations
18         Object[] toArray();
19         <T> T[] toArray(T[] a);
20     }
  
```

Setiap Collection menggunakan interface Iterator untuk proses pergerakan (looping) datanya, sebagai berikut:

```

1      public interface Iterator<E>
2      {
3          E next();
4          boolean hasNext();
5          void remove();
6          default void forEachRemaining(Consumer<? super E> action);
7      }
  
```

Daftar class Collection yang konkrit pada Java yang sering digunakan dapat dilihat pada Gambar 12.3. Berikut akan dibahas beberapa class penting yang sering digunakan.

**Interface List** Interface List memiliki sifat berikut:

- Elemen berada dalam urutan tertentu
- Mengizinkan duplikasi
- Elemen diakses menggunakan index

Collection Type	Description	See Page
ArrayList	An indexed sequence that grows and shrinks dynamically	484
LinkedList	An ordered sequence that allows efficient insertion and removal at any location	474
ArrayDeque	A double-ended queue that is implemented as a circular array	494
HashSet	An unordered collection that rejects duplicates	485
TreeSet	A sorted set	489
EnumSet	A set of enumerated type values	506
LinkedHashSet	A set that remembers the order in which elements were inserted	504
PriorityQueue	A collection that allows efficient removal of the smallest element	495
HashMap	A data structure that stores key/value associations	504
TreeMap	A map in which the keys are sorted	497
EnumMap	A map in which the keys belong to an enumerated type	506
LinkedHashMap	A map that remembers the order in which entries were added	504
WeakHashMap	A map with values that can be reclaimed by the garbage collector if they are not used elsewhere	504
IdentityHashMap	A map with keys that are compared by ==, not equals	507

Gambar 12.3: Class konkrit Collection

- Penambahan dilakukan di posisi akhir, penghapusan akan menghapus elemen pada posisi awal

Beberapa implementasi List adalah:

- LinkedList (java.util.LinkedList)
- ArrayList (java.util.ArrayList)

### LinkedList

Salah satu struktur data yang terkenal adalah linked-list. Jika array menyimpan referensi obyek dalam memory secara berurutan, linked list menyimpan setiap obyek pada link yang terpisah. Setiap link juga menyimpan referensi ke link berikutnya pada urutan yang benar. Pada Java semua linked list merupakan double linked list, sehingga setiap link juga menyimpan referensi ke data sebelumnya. Contoh berikut adalah contoh untuk menambah 3 elemen data dan kemudian menghapus data kedua:

```

1      List<String> staff = new LinkedList<>(); // LinkedList implements List
2      staff.add("Amy");
3      staff.add("Bob");
4      staff.add("Carl");
5      Iterator iter = staff.iterator();
6      String first = iter.next(); // visit first element
7      String second = iter.next(); // visit second element
8      iter.remove(); // remove last visited element

```

Berikut adalah contoh untuk menskip elemen pertama pada linked list dan menambah data baru "Juliet" sebelum data kedua:

---

```

1      List<String> staff = new LinkedList<>();
2      staff.add("Amy");
3      staff.add("Bob");
4      staff.add("Carl");
5      ListIterator<String> iter = staff.listIterator();
6      iter.next(); // skip past first element
7      iter.add("Juliet");

```

---

Contoh berikut adalah mengganti elemen pertama pada list dengan suatu nilai baru:

---

```

1      ListIterator<String> iter = list.listIterator();
2      String oldValue = iter.next(); // returns first element
3      iter.set(newValue); // sets first element to newValue

```

---

Berbeda dengan abstract class, pada interface tidak diperlukan menggunakan keyword *abstract* karena seluruh method pada interface adalah abstract method. Hubungan antara class dengan interface adalah implementasi, misalnya pada class Segitiga yang mengimplementasikan interface Bentuk seperti berikut:

#### Kegiatan Praktikum LinkedList

Berikut adalah contoh manipulasi linked list untuk menambah data, menggabungkan data, menampilkan data, dan menghapus data.

---

```

1      import java.util.*;
2      public class LinkedListTest {
3          public static void main(String[] args)
4          {
5              List<String> a = new LinkedList<>();
6              a.add("Amy");
7              a.add("Carl");
8              a.add("Erica");
9
10             List<String> b = new LinkedList<>();
11             b.add("Argo");
12             b.add("Kuncoro");
13             b.add("Yuan");
14             b.add("Anton");
15
16             ListIterator<String> aIter = a.listIterator();
17             Iterator<String> bIter = b.iterator();
18
19             while (bIter.hasNext())
20             {
21                 if (aIter.hasNext()) aIter.next();
22                 aIter.add(bIter.next());
23             }
24             System.out.println(a);
25

```

```

26         bIter = b.iterator();
27         while (bIter.hasNext())
28         {
29             bIter.next(); // skip one element
30             if (bIter.hasNext())
31             {
32                 bIter.next(); // skip next element
33                 bIter.remove(); // remove that element
34             }
35         }
36
37         System.out.println(b);
38
39         a.removeAll(b);
40         System.out.println(a);
41     }
42 }

```

Jika program dijalankan, berikut adalah outputnya:

```

[Amy, Argo, Carl, Kuncoro, Erica, Yuan, Anton]
[Argo, Yuan]
[Amy, Carl, Kuncoro, Erica, Anton]

```

Silahkan analisis program di atas!

### ArrayList

ArrayList menyediakan array berukuran dinamis yang dapat bertambah sesuai kebutuhan. Pada Java, array biasa bersifat fixed size, setelah dibuat pertama kali array biasa tidak bisa diubah lagi, sedangkan array list bisa diubah ukurannya secara dinamis.

### Kegiatan Praktikum ArrayList

```

1     import java.util.*;
2     class ArrayListDemo {
3         public static void main(String args[]) {
4             // Create an array list.
5             ArrayList<String> al = new ArrayList<String>();
6             System.out.println("Initial size of al: " +
7                 al.size());
8             // Add elements to the array list.
9             al.add("C");
10            al.add("A");
11            al.add("E");
12            al.add("B");
13            al.add("D");
14            al.add("F");
15            al.add(1, "A2");
16            System.out.println("Size of al after additions: " +
17                al.size());
18            // Display the array list.

```

```

19         System.out.println("Contents of al: " + al);
20         // Remove elements from the array list.
21         al.remove("F");
22         al.remove(2);
23         System.out.println("Size of al after deletions: " +
24             al.size());
25         System.out.println("Contents of al: " + al);
26     }
27 }

```

Output dari program tersebut adalah:

```

Initial size of al: 0
Size of al after additions: 7
Contents of al: [C, A2, A, E, B, D, F]
Size of al after deletions: 5
Contents of al: [C, A2, E, B, D]

```

**Interface Set** Interface Set memiliki sifat berikut:

- Elemen hanya boleh 1 kali muncul di dalam Set
- Tidak mengijinkan adanya duplikasi
- Elemen diakses menggunakan index

Beberapa implementasi Set adalah:

- HashSet (java.util.HashSet)
- LinkedHashSet (java.util.LinkedHashSet)
- TreeSet (java.util.TreeSet)

### HashSet

HashSet extends AbstractSet dan mengimplementasi interface Set. Struktur data ini menggunakan hash table untuk penyimpanannya. HashSet tidak mendefinisikan method tambahan selain yang dimiliki superclass dan interface yang diimplementasikannya. HashSet tidak menggaransi urutan elemen-elemennya, sebab proses hashing tidak memperhatikan urutan set. Jika membutuhkan data set yang terurut, bisa menggunakan TreeSet yang akan dibahas berikutnya.

Berikut adalah contoh program untuk demonstrasi HashSet:

### Kegiatan Praktikum HashSet

```

1     import java.util.*;
2     class HashSetDemo {
3         public static void main(String args[]) {
4             // Create a hash set.
5             HashSet<String> hs = new HashSet<String>();
6             // Add elements to the hash set.
7             hs.add("Beta");
8             hs.add("Alpha");
9             hs.add("Eta");
10            hs.add("Gamma");
11            hs.add("Epsilon");
12            hs.add("Omega");
13            System.out.println(hs);

```

```

14         }
15     }

```

Output dari program tersebut adalah:

[Gamma, Eta, Alpha, Epsilon, Omega, Beta]

Class `LinkedHashSet` memelihara linked list pada set, ketika data ditambahkan. Ketika data diiterasi maka datanya akan berada di dalam urutan tertentu. Untuk melihat perbedaan hasilnya dengan `HashSet`, silahkan ganti program `HashSet` di atas menjadi `LinkedHashSet`. Output akan menjadi: Output dari program tersebut adalah:

[Beta, Alpha, Eta, Gamma, Epsilon, Omega]

### TreeSet

`TreeSet` extends `AbstractSet` dan mengimplementasi `NavigableSet` interface. `TreeSet` menyimpan data menggunakan tree. Datanya akan terurut secara ascending order. Waktu access dan retrieval cukup cepat, hal ini membuat `TreeSet` pilihan yang tepat untuk menyimpan data dalam jumlah besar yang diurutkan sehingga dapat ditemukan dengan cepat.

Berikut adalah contoh program untuk demonstrasi `TreeSet`:

### Kegiatan Praktikum TreeSet

```

1      import java.util.*;
2      class TreeSetDemo {
3          public static void main(String args[]) {
4              // Create a tree set.
5              TreeSet<String> ts = new TreeSet<String>();
6              // Add elements to the tree set.
7              ts.add("C");
8              ts.add("A");
9              ts.add("B");
10             ts.add("E");
11             ts.add("F");
12             ts.add("D");
13             System.out.println(ts);
14             System.out.println(ts.subSet("C", "F"));
15         }
16     }

```

Output dari program tersebut adalah:

[A, B, C, D, E, F]

[C, D, E]

**Interface Queue** Interface `Queue` memiliki sifat berikut:

- Elemen bersifat mirip dengan List, terurut
- Bersifat FIFO
- `add()`: menambah data di belakang dan `remove()`: mengambil data di depan

Beberapa implementasi `Queue` adalah:

- `LinkedList` (`java.util.LinkedList`)
- `PriorityQueue` (`java.util.PriorityQueue`)



### PriorityQueue

PriorityQueue extends AbstractQueue dan mengimplementasikan Queue interface. Class ini menciptakan queue yang memiliki prioritas. Cara pembuatannya adalah: Queue q2 = new PriorityQueue(); Elemen disimpan dalam kondisi terurut berdasarkan kriteria tertentu (Comparator). Data yang keluar adalah data yang memiliki tingkat "prioritas" paling tinggi.

Berikut adalah contoh program untuk demonstrasi TreeSet:

### Kegiatan Praktikum PriorityQueue

---

```

1      import java.util.*;
2      class PriorityQueueDemo {
3          public static void main(String args[]) {
4              // Create a PriorityQueue set.
5              Queue<String> q = new PriorityQueue<String>();
6              // Add elements to the queue.
7              q.add("A");
8              q.add("B");
9              q.add("C");
10             System.out.println(q);
11             q.poll();
12             System.out.println(q);
13         }
14     }

```

---

Output dari program tersebut adalah:

```

[A, B, C]
[B, C]

```

**Interface Map** Interface Map memiliki sifat berikut:

- Pemetaan dan penyimpanan pasangan key dan value
- Bukan extends dari java.util.Collection

Beberapa implementasi Map adalah:

- HashMap (java.util.HashMap) Key dan value tersimpan dalam kondisi tidak terurut
- TreeMap (java.util.TreeMap) Key dan value tersimpan dalam kondisi terurut (dipastikan selalu terurut) Urut berdasarkan key atau value

## 12.4 Latihan Praktikum

### 12.4.1 INFIX TO POSTFIX

Buatlah sebuah program Java menggunakan stack untuk mengubah dari notasi infix ke postfix.

Contoh Input:

```

a+b*(c^d-e)^(f+g*h)-i
A*(B+C)/D

```

Contoh Output:

```

abcd^e-fgh*+^*+i-
ABC+*D/

```

Algoritma:

1. Scan the infix expression from left to right.
2. If the scanned character is an operand, output it.
3. Else,
4. If the precedence of the scanned operator is greater than the precedence of the operator in the stack (or the stack is empty or the stack contains a '(', push it.
5. Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. After doing that Push the scanned operator to the stack. (If you encounter parenthesis while popping then stop there and push the scanned operator in the stack.)
6. If the scanned character is an '(', push it to the stack.
7. If the scanned character is an ')', pop the stack and output it until a '(' is encountered, and discard both the parenthesis.
8. Repeat steps 2-6 until infix expression is scanned.
9. Print the output
10. Pop and output from the stack until it is not empty.

#### 12.4.2 POSTFIX EVALUATOR

Notasi Postfix digunakan untuk mewakili ekspresi aljabar. Ekspresi yang ditulis dalam bentuk postfix dievaluasi lebih cepat dibandingkan dengan notasi infiks karena tanda kurung tidak diperlukan dalam postfix. Setelah ekspresi sudah berada di dalam postfix, evaluasilah hasilnya menggunakan program Java.

Contoh input:

231\*+9-

123+\*8-

Contoh output:

-4

-3

Algoritma:

1. Create a stack to store operands (or values).
2. Scan the given expression and do following for every scanned element.
3. If the element is a number, push it into the stack
4. If the element is a operator, pop operands for the operator from stack.
5. Evaluate the operator and push the result back to the stack
6. When the expression is ended, the number in the stack is the final answer

**SELAMAT MENGERJAKAN.**

## 13. Java IO, Serialisasi, dan Deserialisasi

### 13.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Dapat menjelaskan dan menggunakan input output (I/O) pada Java.
2. Dapat membaca data dan mengeluarkan output dari dan ke console pada Java.
3. Dapat menjelaskan dan menggunakan Serialisasi pada Java.

### 13.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 8 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 8).
- IntelliJ Ultimate 64 bit (Students Key).

### 13.3 Dasar Teori

#### 13.3.1 Java IO

Java I/O adalah library pada Java yang digunakan untuk memproses input dan mengeluarkan output. Java menggunakan konsep stream sehingga proses IO menjadi lebih cepat. Secara umum terdapat 3 buah standar I/O:

- Standard input (System.in).
- Standard output (System.out).
- Standard error (System.err).

Pada Java, library IO terdapat pada package `java.io.*` dan `java.nio.file`. Beberapa hal yang terkait dengan IO adalah membaca dan menulis ke file, handling operasi I/O, dan menutup file. Peristiwa terkait lainnya misalnya mencari file, dan mengelola permission file. Termasuk transfer data / file melalui jaringan atau Internet.

Perbedaan antara package `java.io.*` dan `java.nio.file` adalah:

1. package `java.io` lebih kuno daripada `java.nio.file`
2. package `java.io` memiliki lebih banyak method yang mengalami exception
3. Beberapa operasi file tidak didukung (belum ada)
4. Belum mendukung ukuran file besar
5. Belum mendukung symbolic link

Sedangkan untuk `java.nio.file` memiliki ciri:

1. Bekerja dengan konsisten multiplatform
2. Sudah mendukung atribut file
3. Sudah mendukung exception yang lebih canggih
4. Sudah mendukung lebih banyak file system

Beberapa class penting yang terdapat di `java.io`:

FileWriter	BufferedInputStream	BufferedWriter
BufferedOutputStream	BufferedReader	ByteArrayInputStream
InputStream	InputStreamReader	LineNumberReader
Console	DataInputStream	DataOutputStream
File	FileInputStream	FileOutputStream
OutputStream	FileReader	StringReader
StringWriter	StringReader	dan seterusnya

Pada sistem console, Java menyediakan beberapa cara untuk membaca input, yaitu:

- Menggunakan `BufferedReader` `BufferedReader` merupakan class wrapper, dan dia membungkus `InputStreamReader`, sedangkan `InputStreamReader` sendiri membungkus `System.in`. `BufferedReader` tidak bisa menerima data dalam tipe data selain String. Programmer perlu mengkonversi sendiri tipe datanya sesuai yang dibutuhkan. Contoh penggunaan `BufferedReader`:

---

```

1  import java.io.*;
2
3  public class DemoBR {
4      public static void main(String[] args) throws Exception {
5          System.out.println("Masukkan bilangan pecahan: ");
6          BufferedReader reader = new BufferedReader
7                                  (new InputStreamReader(System.in));
8          String input = reader.readLine();
9          double number = Double.parseDouble(input);
10         System.out.println(input + ":" + Math.sqrt(number));
11         reader.close();
12     }
13 }
```

---

- Menggunakan `Scanner` `Scanner` merupakan library dari package `java.util.Scanner`. `Scanner` dapat membaca data input dengan lebih mudah dengan cara membungkus `System.in`. `Scanner` juga memiliki beberapa method untuk membaca data dalam beberapa tipe data

secara langsung, seperti integer, byte, boolean, float, double, dan bahkan BigInteger. Contoh penggunaan Scanner:

---

```

1  import java.util.*;
2
3  public class DemoSc {
4      public static void main(String[] args) throws Exception {
5          System.out.println("Masukkan bilangan pecahan: ");
6          Scanner scan = new Scanner(s);
7          Double number = scan.nextDouble();
8          System.out.println(number + ":" + Math.sqrt(number));
9      }
10 }
```

---

- Menggunakan console console merupakan method dari System, sehingga tidak perlu import package apapun. Beberapa method dari console adalah readLine, printf, format, dan readPassword, dan lain sebagainya. Perlu diingat bahwa console tidak bisa digunakan untuk IDE yang tidak memiliki console. Jadi console hanya bisa dilakukan pada lingkungan penggunaan console seperti DOS Command Prompt atau Linux Console yang sebenarnya. Contoh penggunaan console:

---

```

1  public class DemoCon {
2      public static void main(String[] args) throws Exception {
3          System.out.println("Masukkan bilangan pecahan: ");
4          String input = System.console().readLine();
5          Double number = Double.parseDouble(input);
6          System.out.println(number + ":" + Math.sqrt(number));
7      }
8  }
```

---

### File dan Directory

File merupakan bagian IO yang sangat penting. File pada Java dapat dibaca melalui beberapa cara:

- Menggunakan class File dengan path nama file yang benar dan lengkap, yang mengandung path dan nama file itu sendiri.
- Menggunakan FileReader dengan File di dalamnya. Cara ini digunakan untuk mengambil data berupa data stream karakter.
- Menggunakan BufferedReader dengan FileReader di dalamnya. BufferedReader membaca text dari karakter input stream, membuffer karakter-karakter sehingga proses membaca data karakter, array dan barisnya efisien.

Berikut adalah contoh membaca file dengan Java menggunakan BufferedReader:

---

```

1  import java.io.BufferedReader;
2  import java.io.EOFException;
3  import java.io.File;
4  import java.io.FileReader;
5  import java.io.IOException;
6
7  public class BacaFile {
```

```

8
9 public static void main(String[] args) {
10     // TODO Auto-generated method stub
11     String line = "", fileContent = "";
12     try {
13         BufferedReader fileInput = new BufferedReader(
14             new FileReader(new File("D:/anton.txt")));
15         line = fileInput.readLine();
16         fileContent = line + "\n";
17         while (line != null) {
18             line = fileInput.readLine();
19             if (line != null)
20                 fileContent += line + "\n";
21             //endif
22         } //endwhile
23         fileInput.close();
24     }
25     catch (EOFException eofe) {
26         System.out.println("No more lines to read.");
27         System.exit(0);
28     } //end catch
29     catch (IOException ioe) {
30         System.out.println("Error reading file.");
31         System.exit(0);
32     } //end catch
33     System.out.println(fileContent);
34 }
35 }

```

Class File adalah class yang memiliki banyak atribut dan method yang penting. Beberapa method dan atribut penting pada File dapat dilihat pada source code berikut. Class File tidak menggambarkan file sesungguhnya, namun hanya menggambarkan metadata filenya saja.

```

1 import java.io.File;
2 class DemoFile {
3     static void p(String s) {
4         System.out.println(s);
5     }
6     public static void main(String args[]) {
7         File f1 = new File("/java/COPYRIGHT");
8         p("File Name: " + f1.getName());
9         p("Path: " + f1.getPath());
10        p("Abs Path: " + f1.getAbsolutePath());
11        p("Parent: " + f1.getParent());
12        p(f1.exists() ? "exists" : "does not exist");
13        p(f1.canWrite() ? "is writeable" : "is not writeable");
14        p(f1.canRead() ? "is readable" : "is not readable");
15        p("is " + (f1.isDirectory() ? "" : "not" + " a directory"));
16        p(f1.isFile() ? "is normal file" : "might be a named pipe");

```

---

```

17         p(f1.isAbsolute() ? "is absolute" : "is not absolute");
18         p("File last modified: " + f1.lastModified());
19         p("File size: " + f1.length() + " Bytes");
20     }
21 }

```

---

Outputnya adalah:

```

File Name: COPYRIGHT
Path: \java\COPYRIGHT
Abs Path: C:\java\COPYRIGHT
Parent: \java
exists
is writeable
is readable
is not a directory
is normal file
is not absolute
File last modified: 1282832030047
File size: 695 Bytes

```

Selain File, Directory juga merupakan class penting yang sering digunakan pada Java IO. Directory sebenarnya adalah sebuah file yang berisi file-file lainnya. Pada class File terdapat method `isDirectory()` yang akan bernilai true jika ternyata adalah directory. Berikut adalah contoh penggunaan class Directory untuk menampilkan konten dari sebuah directory:

---

```

1  // Using directories.
2  import java.io.File;
3  class DirList {
4      public static void main(String args[]) {
5          String dirname = "/java";
6          File f1 = new File(dirname);
7          if (f1.isDirectory()) {
8              System.out.println("Directory of " + dirname);
9              String s[] = f1.list();
10             for (int i=0; i < s.length; i++) {
11                 File f = new File(dirname + "/" + s[i]);
12                 if (f.isDirectory()) {
13                     System.out.println(s[i] + " is a directory");
14                 } else {
15                     System.out.println(s[i] + " is a file");
16                 }
17             }
18         } else {
19             System.out.println(dirname + " is not a directory");
20         }
21     }
22 }

```

---

Outputnya adalah:

```

Directory of /java
bin is a directory
lib is a directory
demo is a directory
COPYRIGHT is a file
README is a file
index.html is a file
include is a directory
src.zip is a file
src is a directory

```

Untuk dapat menampilkan isi directory hanya dengan suatu ekstensi tertentu saja, kita harus menggunakan interface `FilenameFilter`. Untuk bisa menggunakannya, kita harus mendefinisikan class tertentu misalnya untuk memfilter ekstensi html saja, atau exe saja dengan mendefinisikan class tersendiri misalnya sebagai berikut:

---

```

1  import java.io.*;
2  public class OnlyExt implements FilenameFilter {
3      String ext;
4      public OnlyExt(String ext) {
5          this.ext = "." + ext;
6      }
7      public boolean accept(File dir, String name) {
8          return name.endsWith(ext);
9      }
10 }

```

---

Class `OnlyText` berarti bisa menerima string ekstensi apapun yang diterima. Penggunaannya dapat dilihat pada filter file html berikut:

---

```

1  // Directory of .HTML files.
2  import java.io.*;
3  class DirListOnly {
4      public static void main(String args[]) {
5          String dirname = "/java";
6          File f1 = new File(dirname);
7          FilenameFilter only = new OnlyExt("html");
8          String s[] = f1.list(only);
9          for (int i=0; i < s.length; i++) {
10             System.out.println(s[i]);
11         }
12     }
13 }

```

---

### Latihan Praktikum

#### PERTAMA

Buatlah program Java yang dapat menerima membaca suatu direktori tertentu dan kemudian membaca isi filenya. Untuk setiap file yang ada, program harus bisa menampilkan atribut filenya:

- Tanggal Modified



- Ukuran file
- Apakah readonly, hidden, writeable, readable, atau executable

**KEDUA**

Buatlah program Java yang dapat membaca dua buah file pada folder yang berbeda dan kemudian mendeteksi apakah kedua file tersebut identik atau tidak dari sisi

- Tanggal modifikasi
- Ukuran file
- Nama file

**KETIGA**

Buatlah program Java yang dapat menghitung apakah suatu deret angka berikut adalah triple pythagoras atau bukan. Inputan program berasal dari standard input (gunakan console atau Scanner atau BufferedReader). Baris input harus memenuhi kriteria berikut:

- Baris pertama adalah jumlah data angka
- Baris kedua dan seterusnya adalah deret angka yang akan diperiksa apakah deret tersebut adalah bilangan triple pythagoras atau bukan

Output program berupa baris "ya" jika adalah triple pythagoras, atau baris "tidak" jika bukan.

Contoh input data:

```
4
3 4 5
1 2 3
5 12 13
2 3 4
```

Dan output yang diharapkan:

```
ya
tidak
ya
tidak
```

**KEEMPAT**

Problem Statement:

Linear Search: Given an integer array and an element x, find if element is present in array or not. If element is present, then print index of its first occurrence. Else print -1.

Input: First line contains an integer, the number of test cases 'T'. Each test case should be an integer. Size of the array 'N' in the second line. In the third line, input the integer elements of the array in a single line separated by space. Element X should be inputted in the fourth line, i.e., after entering the elements of array. Repeat the above steps second line onwards for multiple test cases.

Output: Print the output in a separate line returning the index of the element X. If the element is not present, then print -1.

Constraints:  $1 \leq T \leq 100$   $1 \leq N \leq 100$   $1 \leq \text{Arr}[i] \leq 100$

Example Input and Output for Your Program:

Input:

```
2
4
1 2 3 4
3
5
10 90 20 30 40
40
```

Output:

2  
4

### 13.3.2 Serialisasi dan Deserialisasi

Obyek pada Java dapat disimpan dalam sebuah stream untuk dapat dikirimkan melalui jaringan dan kemudian diterima oleh pihak penerima untuk kemudian dibaca kembali. Serialization adalah proses menulis state sebuah object ke dalam byte stream. Hal ini sangat berguna ketika kita butuh menyimpan data program ke dalam persistent storage area, seperti misalnya file. Di akhir proses kita akan dapat mengembalikan obyek obyek yang sudah disimpan sebelumnya melalui proses deserialization.

Untuk dapat melakukan serialisasi, setiap obyek yang akan diserialisasi harus implements interface Serializable. Interface Serializable tidak memiliki method apapun, hanya digunakan untuk menandakan bahwa class tertentu dapat diserialkan atau tidak. Ketika suatu class ingin diserialkan maka semua atribut class tersebut pasti ikut diserialkan. Untuk mencegahnya dapat digunakan keyword transient.

#### Kegiatan Praktikum

Berikut adalah demo serialisasi:

```

1  import java.io.*;
2  public class SerializationDemo {
3      public static void main(String args[]) {
4          // Object serialization
5          try ( ObjectOutputStream objOStrm =
6              new ObjectOutputStream(new FileOutputStream("serial")) )
7          {
8              MyClass object1 = new MyClass("Hello", -7, 2.7e10);
9              System.out.println("object1: " + object1);
10             objOStrm.writeObject(object1);
11         }
12         catch(IOException e) {
13             System.out.println("Exception during serialization: " + e);
14         }
15         // Object deserialization
16         try ( ObjectInputStream objIStrm =
17             new ObjectInputStream(new FileInputStream("serial")) )
18         {
19             MyClass object2 = (MyClass)objIStrm.readObject();
20             System.out.println("object2: " + object2);
21         }
22         catch(Exception e) {
23             System.out.println("Exception during deserialization: " + e);
24         }
25     }
26 }
27
28 class MyClass implements Serializable {
29     String s;

```

```

30     int i;
31     double d;
32     public MyClass(String s, int i, double d) {
33         this.s = s;
34         this.i = i;
35         this.d = d;
36     }
37     public String toString() {
38         return "s=" + s + "; i=" + i + "; d=" + d;
39     }
40 }

```

Output dari program di atas adalah bahwa object1 dan object2 adalah identik. Berikut adalah output programnya.

object1: s=Hello; i=-7; d=2.7E10

object2: s=Hello; i=-7; d=2.7E10

## 13.4 Latihan Praktikum

Buatlah program Java untuk menserialkan class GameCharacter sebagai berikut, setelah diserialkan kemudian dibaca kembali dengan cara deserialisasi.

Class GameCharacter:

```

1  import java.io.*;
2  public class GameCharacter implements Serializable
3  //merupakan kelas yang akan diserialisasi
4  {
5      int power;
6      //membuat variabel power dengan tipe data integer
7      String type;
8      //membuat variabel type dengan tipe data string
9      String[] weapons;
10     //membuat array weapons dengan tipe data string
11
12     public GameCharacter(int p, String t, String[] w)
13     {
14         power = p;
15         type = t;
16         weapons = w;
17     }
18
19     public int getPower(){
20         return power;
21     }
22
23     public String getType(){
24         return type;
25     }
26

```

```
27     public String getWeapons(){
28         String weaponList = "";
29         for(int i = 0; i < weapons.length; i++)
30         {
31             weaponList += weapons[i] + " ";
32         }
33         return weaponList;
34     }
35 }
```

---

**SELAMAT MENGERJAKAN.**

## 14. Package dan JDBC

### 14.1 Tujuan Praktikum

Setelah mempelajari modul ini, mahasiswa diharapkan:

1. Dapat menjelaskan, membuat dan menggunakan package pada Java
2. Dapat menjelaskan, membuat, dan menggunakan JDBC

### 14.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat keras (hardware) komputer dengan spesifikasi minimum sebagai berikut:

- Processor Intel Core (64 bit).
- Memory RAM 8 GB.
- Ruang kosong di harddisk sebanyak 5 GB.

Berikut ini adalah perangkat lunak yang diperlukan dalam kegiatan praktikum ini:

- Sistem operasi 64 bit (Windows/Linux/Mac).
- Java Development Kit (JDK 8).
- IntelliJ Ultimate 64 bit (Students Key).

### 14.3 Dasar Teori

#### 14.3.1 Java Package

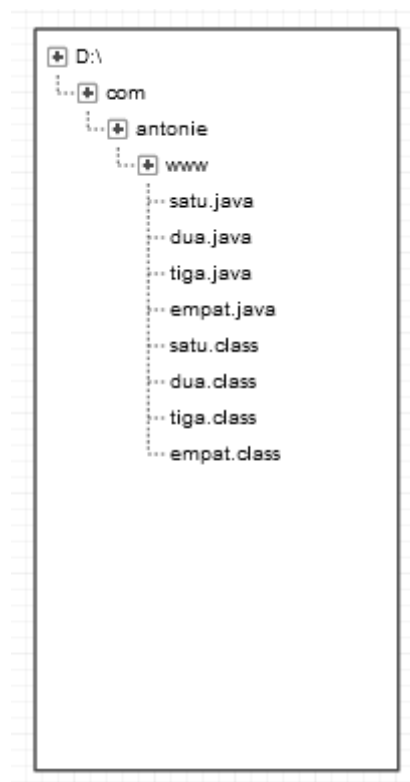
Package dalam Java adalah sebuah folder, tempat yang berisi kumpulan class-class yang disatukan menjadi satu buah program / kelompok program, bahkan berupa library dalam satu kesatuan yang memiliki nama tertentu. Package digunakan untuk mencegah terjadinya naming conflicts, untuk mengontrol hak akses, untuk mempermudah pencarian dan penggunaan class, interface, enumeration agar lebih mudah.

Package merupakan mekanisme dari encapsulation suatu kelompok atau grup yang terdiri dari class - class, sub packages dan juga interfaces. Ini akan memberi kemudahan, karena banyak penerapan dari program java menggunakan konsep hirarki untuk mengatur file-file dari class dan

source. Ketika aplikasi semakin bertambah kompleks, maka package ini dapat membantu kita dalam mengatur komponen-komponen di dalamnya.

Pemberian nama package biasanya tergantung dari untuk apa kita menggunakan class yang ada di dalamnya. Mungkin saja package itu memiliki nama seperti nama anda sendiri, atau berdasarkan bagian dari sistem pada Java seperti `graphics` atau `interface`. Namun, jika package yang dibuat itu akan didistribusikan pada skala yang lebih luas ataupun bersifat komersil, maka lebih baik anda menggunakan nama package yang benar-benar unik dan bisa merepresentasikan tentang anda dan juga organisasi. Cara penamaan yang direkomendasikan oleh Java adalah dengan menggunakan domain internet anda namun dengan urutan terbalik.

Penamaan package pada java biasanya dibuat dengan model bertingkat dan pada implementasinya berupa nama sebuah folder. Sebuah nama package biasanya memiliki suatu identitas pada packagenya mirip seperti sebuah web. Contoh package `com.antonie.www` berarti kita memiliki folder dengan hirarki sebagai berikut:



Gambar 14.1: Struktur Package

Beberapa package sudah dibuat pada saat kita menginstall Java, misalnya package: `java.lang`, `java.io`, `java.text`, `java.nio`, dsb. Kita sebagai programmer juga dapat membuat package sendiri dan digunakan pada aplikasi yang kita buat ataupun aplikasi orang lain. Karena setiap package berarti sebuah namespace baru, maka tidak akan ada konflik penamaan pada package.

Cara menggunakan package adalah dengan mengimport nya dari program Java yang kita buat. Perintah `import` harus diletakkan di baris teratas, misalnya dengan sintaks berikut: **`import java.util.Scanner;`** perintah tersebut berarti aplikasi kita mengimport package `java.util` yang disediakan java dan menggunakan class `Scanner`. Kita bisa saja mengimport class `Scanner` lain jika ada asal nama packagenya berbeda, misalnya kita membuat library `Scanner` sendiri. Misalnya: **`import com.antonie.utilty.Scanner;`**

Sebelum suatu class dapat digunakan ulang melalui pernyataan `import`, maka class tersebut

harus di simpan di dalam package. Langkah-langkah membuat class yang dapat digunakan ulang adalah:

- Mendeklarasikan class sebagai public. Jika class tidak dideklarasikan demikian, maka class tersebut hanya dapat digunakan oleh class lainnya yang berada dalam package yang sama.
- Memilih nama yang unik untuk package tersebut dan menambahkan deklarasi package pada file source code untuk deklarasi class yang digunakan ulang. Dalam setiap source code file hanya ada satu deklarasi package. Jika tidak ada pernyataan package yang disediakan maka class tersebut berada dalam default package dan hanya dapat diakses oleh class lainnya yang berada dalam default package dan berada dalam direktori yang sama.
- Melakukan compile class
- Meng-import class yang akan digunakan ulang di dalam program dilanjutkan dengan penggunaan class.

Deklarasi package harus diletakkan pada bagian paling awal (sebelum deklarasi import) dari source code setiap kelas yang dibungkus package tersebut. Bentuk umum deklarasi package : **package namaPackage;** Bentuk umum pernyataan package multilevel : **package namaPackage1[.namaPackage2[.namaPackage3]];**

Deklarasi tersebut akan memberitahukan kompilator, ke library manakah suatu kelas dikompilasi dan dirujuk. Syarat nama package :

- Diawali huruf kecil.
- Menggambarkan kelas-kelas yang dibungkusnya.
- Harus unik (berbeda dengan nama package standard).
- Merepresentasikan path dari package tersebut.
- Harus sama dengan nama direktorinya.

Ada dua cara menggunakan suatu package yaitu :

- Kelas yang menggunakan berada dalam direktori (package) yang sama dengan kelas-kelas yang digunakan. Maka tidak diperlukan import.
- Kelas yang menggunakan berada dalam direktori (package) yang berbeda dengan kelas-kelas yang digunakan. Maka pada awal source code di kelas pengguna harus mencantumkan : **import namaPackage>NamaKelas;** atau **import namaPackage.\*;**

### Kegiatan Praktikum

Sebagai contoh kita bisa mengimport `java.util.*` dan `java.sql.*`; Semua package di atas memiliki class `Date`, nah ketika kita akan membuat program yang menggunakan class `Date` maka akan terjadi conflict, karena kita memanggil class `Date` hanya dengan nama `Date` saja. Perhatikan contoh berikut:

---

```

1  import java.util.*;
2  import java.sql.*;
3
4  public class CobaDate{
5      public static void main(String[] args){
6          Date hariini = new Date(); //akan error,
7          //import dari java.sql.Date atau java.util.Date?
8      }
9  }
```

---

Untuk memperbaikinya gunakan import berikut:

---

```

1  import java.util.*;
2  import java.sql.*;
```

---

---

```

3  import java.util.Date;
4
5  public class CobaDate{
6      public static void main(String[] args){
7          Date hariini = new Date(); //tidak error!
8      }
9  }

```

---

Jika kita benar-benar akan menggunakan kedua class Date, maka gunakan cara berikut:

---

```

1  import java.util.*;
2  import java.sql.*;
3
4  public class CobaDate{
5      public static void main(String[] args){
6          java.util.Date deadline = new java.util.Date();
7          java.sql.Date today = new java.sql.Date();
8      }
9  }

```

---

Berikut adalah latihan untuk membuat package: Kita akan membuat package bernama id.ac.ukdw.www; Maka buatlah struktur folder berikut:

D:\package\id\ac\ukdw\www

Di dalam folder terakhir, www, buatlah class berikut:

---

```

1  package id.ac.ukdw.www;
2  import java.time.LocalDate;
3  public class Employee {
4      private String name;
5      private double salary;
6      private LocalDate hireDay;
7      public Employee(String name, double salary, int year, int month, int day)
8      {
9          this.name = name;
10         this.salary = salary;
11         hireDay = LocalDate.of(year, month, day);
12     }
13
14     public String getName()
15     {
16         return name;
17     }
18     public double getSalary()
19     {
20         return salary;
21     }
22
23     public LocalDate getHireDay()

```



```

24     {
25         return hireDay;
26     }
27
28     public void raiseSalary(double byPercent)
29     {
30         double raise = salary * byPercent / 100;
31         salary += raise;
32     }
33 }

```

Sedangkan di bawah folder package tuliskan kode berikut:

```

1  import id.ac.ukdw.www.*;
2  import static java.lang.System.*;
3  public class PackageTest
4  {
5      public static void main(String[] args)
6      {
7          Employee harry = new Employee("Harry Hacker", 50000, 1989, 10, 1);
8          harry.raiseSalary(5);
9          out.println("name=" + harry.getName() + ",salary=" + harry.getSalary());
10     }
11 }

```

Jika dijalankan akan menghasilkan output:

```
name=Harry Hacker,salary=52500.0
```

### Latihan Praktikum

Buatlah program untuk menghitung operasi matematika dasar, kali, bagi, tambah, kurang dengan nama class Matematika. Buatlah folder pbo13, setelah itu buatlah dalam package com.latihan.www. Buatlah program untuk mencoba operasi matematika tersebut dari folder pbo13.

### 14.3.2 Java Database Connectivity (JDBC)

JDBC adalah library dari Java untuk mengakses basis data. JDBC adalah singkatan dari Java Database Connectivity yang berusaha menyaingi ODBC milik Microsoft. Dengan menggunakan JDBC maka kita bisa membuat program yang berkomunikasi dengan basis data relasional dengan mudah baik berbasis GUI ataupun console. JDBC terdiri dari dua bagian:

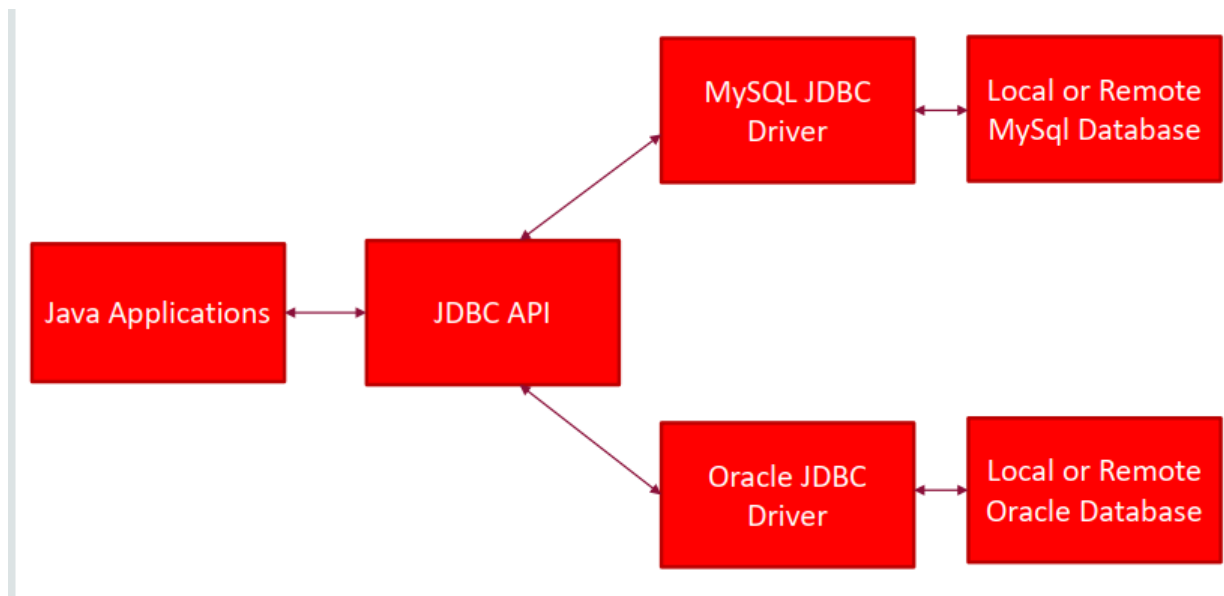
- Java application developer interface. Kita akan berfokus pada yang pertama, sebagai developer.
- Java driver developer implementation interface.

JDBC API terdiri dari dua library:

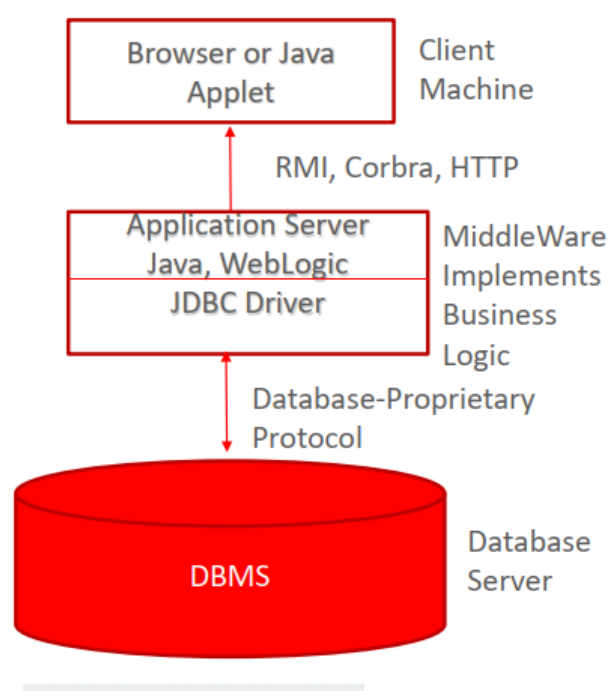
- java.sql.\*
- javax.sql.\*

Gambar relasi antara aplikasi Java dengan JDBC Driver (MySQL dan Oracle) dapat dilihat pada gambar 14.2. Sedangkan arsitektur JDBC dapat dilihat pada gambar 14.3. 3 hal dasar yang dapat dilakukan oleh JDBC adalah:

- Membuka koneksi dengan basis data.
- Mengirimkan perintah kepada basis data, termasuk SQL query dan lain-lain.



Gambar 14.2: Relasi aplikasi dan JDBC



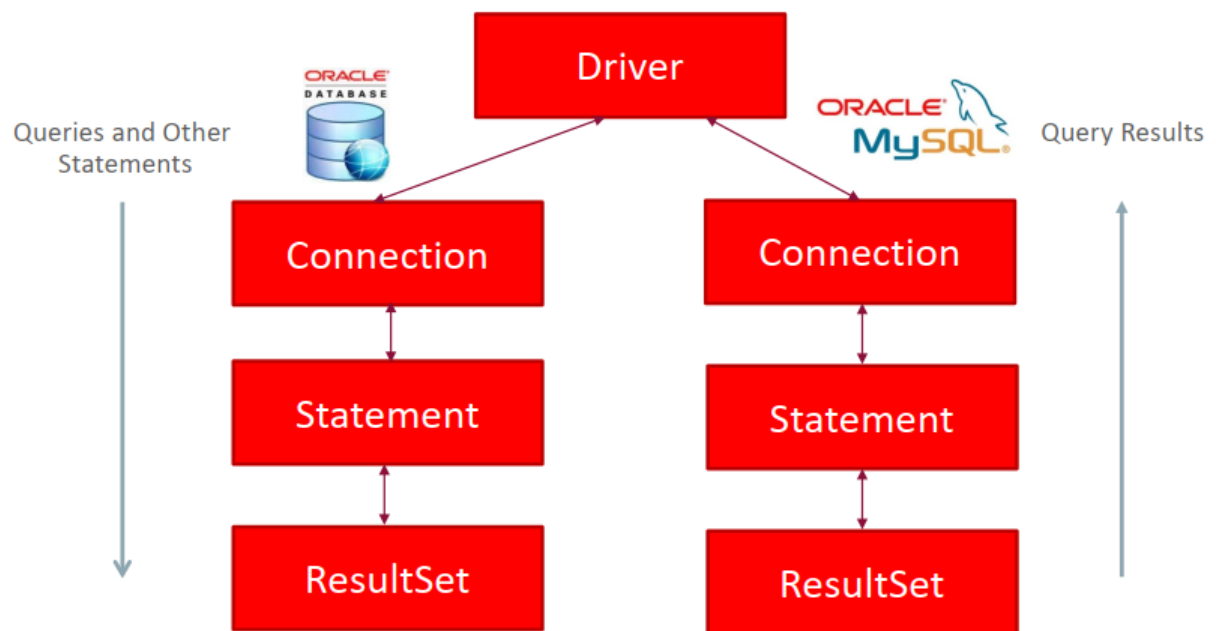
Gambar 14.3: Arsitektur JDBC

- Mendapatkan hasil dari perintah yang sudah dikirimkan.
- Menutup koneksi dengan basis data.

Secara implementatif dapat didetailkan sebagai berikut:

- Importing Packages
- Establishing a Connection
- Creating a Statement
- Executing the Statement
- Retrieving and processing ResultSet object

- Closing ResultsSets, Statements, and Connections
- Secara gambar dapat dilihat pada 14.4.



Gambar 14.4: Implementasi JDBC

### Kegiatan Praktikum

Kita akan membuat class untuk mencoba mengakses basis data SQLite dengan database test dan nama tabel mahasiswa (field: nim dan nama yang bertipe varchar).

Buatlah class ManipulasiTabelMahasiswa.java sebagai berikut:

```

1  import java.sql.Connection;
2  import java.sql.DatabaseMetaData;
3  import java.sql.DriverManager;
4  import java.sql.ResultSet;
5  import java.sql.SQLException;
6  import java.sql.Statement;
7
8  public class ManipulasiTabelMahasiswa {
9      private String url = "jdbc:sqlite:mahasiswa.db";
10
11      public Connection connect() {
12          Connection conn = null;
13          try {
14              // create a connection to the database
15              conn = DriverManager.getConnection(url);
16
17              System.out.println("Koneksi berhasil.");
18
19          } catch (SQLException e) {

```

```

20         System.out.println(e.getMessage());
21     } finally {
22         try {
23             if (conn != null) {
24                 conn.close();
25             }
26         } catch (SQLException ex) {
27             System.out.println(ex.getMessage());
28         }
29     }
30     return conn;
31 }
32
33 public void createNewDatabase() {
34
35     try (Connection conn = DriverManager.getConnection(url)) {
36         if (conn != null) {
37             DatabaseMetaData meta = conn.getMetaData();
38             System.out.println("Nama: " + meta.getDriverName());
39             System.out.println("Database tercipta");
40         }
41
42     } catch (SQLException e) {
43         System.out.println(e.getMessage());
44     }
45 }
46
47 public void createNewTable() {
48
49     // SQL statement for creating a new table
50     String sql = "CREATE TABLE IF NOT EXISTS mahasiswa (\n"
51 + "        nim varchar(8) PRIMARY KEY,\n"
52 + "        nama text NOT NULL,\n"
53 + "        ipk real\n"
54 + "    );";
55     String sql2 = "INSERT INTO mahasiswa values" +
56 + "('12345670', 'Anton', 3.5), ('12345679', 'Adi', 3.2)";
57
58     try (Connection conn = DriverManager.getConnection(url);
59         Statement stmt = conn.createStatement()) {
60         // create a new table
61         stmt.execute(sql);
62         System.out.println("Tabel tercipta!");
63         stmt.execute(sql2);
64         System.out.println("Tabel terisi!");
65     } catch (SQLException e) {
66         System.out.println(e.getMessage());
67     }
68 }

```

```

69
70     public void selectAll(){
71         String sql = "SELECT nim,nama,ipk FROM mahasiswa";
72
73         try (Connection conn = DriverManager.getConnection(url);
74             Statement stmt = conn.createStatement();
75             ResultSet rs = stmt.executeQuery(sql)){
76
77             // loop through the result set
78             while (rs.next()) {
79                 System.out.println(rs.getString("nim") + "\t" +
80                 rs.getString("nama") + "\t" +
81                 rs.getDouble("ipk"));
82             }
83         } catch (SQLException e) {
84             System.out.println(e.getMessage());
85         }
86     }
87
88
89     public static void main(String[] args) {
90         // TODO Auto-generated method stub
91         ManipulasiTabelMahasiswa data = new ManipulasiTabelMahasiswa();
92         data.createNewDatabase();
93         data.connect();
94         data.createNewTable();
95         data.selectAll();
96     }
97
98 }

```

#### Penjelasan:

- Class tersebut melakukan beberapa kegiatan: membuat database mahasiswa, melakukan koneksi ke database tsb, membuat tabel mahasiswa, mengisi tabel mahasiswa dengan 2 data, dan menampilkannya.
- Semua library harus diimport pada program
- Download library SQLite dari <https://bitbucket.org/xerial/sqlite-jdbc/downloads>, pilih yang terbaru.
- Jika menggunakan Eclipse, buatlah folder lib di dalam folder aplikasi dan kopikan file JDBC SQLite driver ke dalamnya.
- Program masih membuat semua method jadi satu dan dipanggil satu persatu.

## 14.4 Latihan Praktikum

Lanjutkan aplikasi di atas dengan melengkapinya dengan:

1. Dibuat dalam bentuk menu: menu create database, create table, insert table, view data, edit data, dan delete data. Program ini dibuat dalam main program terpisah.
2. Program yang berisi manipulasi data untuk database dijadikan satu kesatuan, sedangkan main nya dibuat dalam program terpisah.

**SELAMAT MENGERJAKAN.**







## Bibliography

- [1] Horstmann, C. (2016) *Core Java Volume 1 (10th Ed)*. Prentice Hall
- [2] Martin, C.,E. (2003) *UML for Java Programmers*. Prentice Hall
- [3] Oracle Academy (2017) *Oracle Academy: Java Programming*. Oracle
- [4] Schildt, H.,(2014) *Java The Complete Reference (9th ed)*. McGraw-Hill
- [5] Sharan, K. (2014) *Beginning Java 8 Fundamentals*. Apress



# Index

## A

abstract ..... 49  
abstract method ..... 50  
access specifier ..... 23  
Array ..... 81  
atribut ..... 6

## B

BufferedReader ..... 106  
bytecode ..... 4

## C

class ..... 5  
class diagram ..... 34  
Comparable ..... 101  
Comparator ..... 102  
console ..... 107  
constructor ..... 29

## D

Directory ..... 109

## E

Eclipse ..... 5

encapsulation ..... 21  
Exception ..... 69

## F

File ..... 107  
for ..... 8  
ForEach ..... 99

## G

getter ..... 23

## I

if-else ..... 8  
immutable ..... 57  
inheritance ..... 39  
inline error handling ..... 71  
Iterator ..... 97

## J

Java Collection Framework ..... 88  
Java I/O ..... 105  
JDBC ..... 119

## K

key ..... 12

kompilasi ..... 4

## L

list ..... 89

## M

main() ..... 5

method signature ..... 30

multiple inheritance ..... 53

multithreading ..... 125

## O

object ..... 5

OOP ..... 5

overloading ..... 31, 33

## P

Package ..... 115

Polimorfisme ..... 63

private ..... 23

protected ..... 23

public ..... 23

## R

Runnable ..... 127

## S

Scanner ..... 106

Serializable ..... 112

setter ..... 23

String ..... 57

struktur kontrol percabangan ..... 8

struktur kontrol perulangan ..... 8

synchronous error ..... 70

## T

Thread ..... 126

throw ..... 72

throws ..... 72

tipe data ..... 7

transient ..... 112

try-catch ..... 71

try-catch-finally ..... 71