

PEMROGRAMAN BERORIENTASI OBJEK (JAVA I)

Penyusun:
Agus Suharto



Jl. Surya Kencana No. 1 Pamulang
Gd. A, Ruang 212 Universitas Pamulang
Tangerang Selatan – Banten

PEMROGRAMAN BERORIENTASI OBJEK (JAVA 1)

Penulis:

Agus Suharto

ISBN: 978-623-6352-86-1

Editor:

Agus Suharto

Penyunting:

Agus Suharto

Desain Sampul:

Aden

Tata Letak:

Agus Suharto

Penerbit:

Unpam Press

Redaksi:

Jl. Surya Kecana No. 1
Pamulang – Tangerang Selatan
Telp. 021-7412566
Fax. 021 74709855
Email: unpampress@unpam.ac.id

Cetakan pertama, 11 Februari 2022

Hak cipta dilindungi undang-undang.

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun tanpa izin penerbit.

LEMBAR IDENTITAS ARSIP

DATA PUBLIKASI UNPAM PRESS

I Lembaga Pengembangan Pendidikan dan Pembelajaran Universitas Pamulang

Gedung A. R.211 Kampus 1 Universitas Pamulang
Jalan Surya Kencana No.1 Pamulang Barat, Tangerang Selatan, Banten.
Website : www.unpam.ac.id | email : unpampress@unpam.ac.id

Pemrograman Berorientasi Objek (Java I) / Agus Suharto - 1sted.
ISBN 978-623-6352-86-1

1. Pemrograman Berorientasi Objek (Java I) I. Agus Suharto
M216-11022022-01

Ketua Unpam Press : Pranoto
Koordinator Editorial : Aden
Koordinator Bidang Hak Cipta : Susanto
Koordinator Produksi : Dameis Surya Anggara
Koordinator Publikasi : Kusworo, Heri Haerudin
Koordinator Dokumentasi : Ramdani Putra, Nara Dwi Angesti
Desain Cover : Putut Said Permana

Cetakan pertama, 11 Februari 2022

Hak cipta dilindungi undang-undang.
Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun tanpa
ijin penerbit.

MODUL MATA KULIAH PEMROGRAMAN BERORIENTASI OBJEK

IDENTITAS MATA KULIAH

Program Studi	: Sistem Informasi S-1
Mata Kuliah / Kode	: Pemrograman Berorientasi Objek (Java 1)/ KB1105
Sks	: 3 Sks
Prasyarat	:
Deskripsi Mata Kuliah	: Mata kuliah ini membahas mengenai Pemrograman Berorientasi Objek menggunakan bahasa java SE Mata kuliah ini merupakan mata kuliah wajib Program Studi S-1 sistem Informasi yang membahas tentang Pengantar Java, dan Instalasi perangkat , variable , tipe data, operator , Control Flow If, If-else, Switch case, loop for, loop while, loop do while, Oop <i>Class</i> , Object, method, Constructor, Inheritance, polymorphism, Abstraction, Encapsulation, Access modifier , array, dan exception handling.
Capaian Pembelajaran	: Setelah pembelajaran perkuliahan mahasiswa diharapkan mampu mengerti java dasar, <i>keyword</i> perintah java, Konsep <i>Object Oriented Programming</i> , membuat program bahasa java dengan konsep berbasis objek /OOP dengan benar

Ketua Program Studi
Sistem Informasi S-1

Ketua Tim Penyusun

Dede Supriadi S.Kom, M.Kom
NIDN 0442760018

Ir . Agus Suharto, M.Kom.
NIDN 0329056504

KATA PENGANTAR

Dengan mengucapkan puja dan puji syukur kehadirat Allah SWT, dimana memberikan kelimpahan rahmat dan hidayahNya, sampai saat ini kami panjatkan atas kelancaran dan kemudahan untuk menyusun bahan ajar Pemrograman Berorientasi Objek. Harapan kami dengan modul ajar ini dapat memotivasi dan menjadi acuan belajar mahasiswa Fakultas Teknik Program Studi Sistem Informasi.

Bahan ajar ini disusun berdasar pengalaman mengajar dari tahun 2012 beserta beberapa referensi dari buku, tutorial, serta karya ilmiah yang mendukung materi pada setiap bab dalam 18 pertemuan. Modul ini berisi teori, uraian soal, penjelasan mendetail sehingga memberikan kemudahan untuk mahasiswa pelajari mandiri. Selain itu contoh pengaplikasian dalam praktek bahasa java serta konsep *objek oriented programming* yang di praktekkan dalam bahasa java yang memiliki bahasa yang handal, multi sistem operasi dan multi basis (desktop, *mobile*, *web*).

Besar Harapan kami dengan contoh yang diberikan secara sederhana mahasiswa/i akan dapat diterima, dipahami dan dimengerti, khususnya mahasiswa Prodi SI maupun masyarakat Umum.

Permintaan maaf kami sampaikan apabila terdapat kesalahan dalam tulisan yang kurang berkenan atas penyampaian materi atau kalimat dalam modul ini, harapan kami kritik dan saran yang membangun dari para pembaca untuk kemajuan modul ini dan memperbaiki kekurangan kedepannya.

Tangerang Selatan, Desember 2021

Penyusun

DAFTAR ISI

PEMROGRAMAN BERORIENTASI OBJEK	i
(JAVA I)	i
PEMROGRAMAN BERORIENTASI OBJEK (JAVA 1)	ii
LEMBAR IDENTITAS ARSIP	iii
KATA PENGANTAR	v
DAFTAR ISI	vi
DAFTAR GAMBAR	xiii
DAFTAR TABEL	xv
PERTEMUAN 1	1
PENGANTAR PEMROGRAMAN BAHASA JAVA	1
A. TUJUAN PEMBELAJARAN.....	1
B. URAIAN MATERI	1
1. Pengantar bahasa Java	1
2. Java Development Kit (JDK)	2
3. Java Runtime Environment(JRE).....	2
4. Fitur fitur utama Bahasa JAVA.....	3
5. Compiler dan Interpreter Java	5
6. Instalalasi JDK dengan Netbeans 8.2	7
7. Program java pertama	8
8. Aplikasi yang dijalankan dengan bahasa Java.....	9
9. Jenis jenis Aplikasi Java	10
10. Platform pada Java	10
C. LATIHAN / TUGAS	11
D. REFERENSI	11
PERTEMUAN 2	13
VARIABEL PADA JAVA.....	13
A. TUJUAN PEMBELAJARAN.....	13
B. URAIAN MATERI	13
1. Cara mendeklarasikan Variabel.....	13
2. Aturan umum penamaan variabel di java	15
3. Jenis jenis Variabel pada Java.....	15

4. Contoh contoh program dan jenis variable.....	20
C.LATIHAN /TUGAS	24
D. REFERENSI	25
PERTEMUAN 3	26
TIPE DATA PADA JAVA.....	26
A. TUJUAN PEMBELAJARAN.....	26
B. URAIAN MATERI	26
1. Jenis Jenis Tipe Data :	26
2. Tipe data primitif	27
3. Literals pada Java.....	35
4. Contoh contoh Program Tipe data	35
C. LATIHAN / TUGAS	37
D. REFERENSI	38
PERTEMUAN 4	40
OPERATOR PADA JAVA	40
A. TUJUAN PEMBELAJARAN.....	40
B. URAIAN MATERI	40
1. Operator Aritmatika Dasar	40
2. Operator Penugasan (Assignment Operator).....	42
3. Operator auto-increment dan auto-decrement	44
4. Operator Logika (Logical Operator)	45
5. Operator Pembandingan (Relasi)	46
6. Operator Bitwise	49
7. Operator Ternary	52
C. LATIHAN /TUGAS :	54
D. REFERENSI	54
PERTEMUAN 5	55
INPUT OUTPUT STREAM dan <i>KEYWORD</i> PADA JAVA	55
A. TUJUAN PEMBELAJARAN.....	55
B. URAIAN MATERI	55
1. Menggunakan Input <i>Scanner</i> melalui Keyboard.....	56
2. Tipe tipe Input (Masukan)	57
3. Menggunakan Input <i>BufferedReader</i> melalui Keyboard.....	59
4. Metode metode yang digunakan class <i>BufferedReader</i>	60

5. Sistem.err	61
6. Keyword pada Java	62
C. LATIHAN DAN TUGAS.....	67
D. REFERENSI	68
PERTEMUAN 6	69
MENGONTROL ALIRAN PROGRAM (IF,IF.. ELSE).....	69
A. TUJUAN PEMBELAJARAN.....	69
B. URAIAN MATERI	69
1. Pernyataan IF (IF Statement)	69
2. Pernyataan IF Bersarang (Nested IF Statement)	71
3. Pernyataan IF ..ELSE	73
4. Pernyataan IF ..ELSE ..IF	75
C. LATIHAN	78
D. REFERENSI	80
PERTEMUAN 7	81
MENGONTROL ALIRAN PROGRAM PERNYATAAN SWITCH ..CASE.....	81
A. TUJUAN PEMBELAJARAN.....	81
B. URAIAN MATERI	81
1. Aturan Penulisan Pernyataan Switch	81
2. Bentuk Umum penulisan pernyataan Switch.....	82
3. Contoh 1 Program dengan Pernyataan switch.....	83
4. Beberapa poin Pernyataan Switch Case.....	86
5. Pernyataan Switch Case dengan multi value	87
6. Pernyataan Switch Bersarang (Nested Switch).....	88
C. LATIHAN/TUGAS	91
D. REFERENSI	91
PERTEMUAN 8	93
MENGONTROL ALIRAN PROGRAM PERULANGAN FOR (for loop)	93
A. TUJUAN PEMBELAJARAN.....	93
B. URAIAN MATERI	93
1. Perulangan For.....	93
2. Bentuk Umum Penulisan For	94
3. Perulangan For bersarang (Nested For)	95
4. Perulangan For each	98

5. Label For Loop	99
6. Infinite For Loop.....	101
7. Enhanced for loop.....	102
C. LATIHAN /TUGAS	103
D. REFERENSI	105
PERTEMUAN 9	106
MENGONTROL ALIRAN PROGRAM PERULANGAN WHILE, DO WHILE	106
A. TUJUAN PEMBELAJARAN.....	106
B. URAIAN MATERI	106
1. Perulangan While	106
2. Gambar diagram flowchart perulangan <i>While</i>	107
3. Infinite while loop	109
4. Perulangan while dengan pernyataan Break.....	110
5. Perulangan Do While.....	112
C. LATIHAN/TUGAS	115
D. REFERENSI	116
PERTEMUAN 10	117
KONSEP PEMROGRAMAN BERORIENTASI OBJEK <i>CLASS, OBJECT, ATTRIBUTE, METHOD</i>	118
A. TUJUAN PEMBELAJARAN.....	118
B. URAIAN MATERI	118
1. <i>Object</i>	119
2. <i>Class</i>	120
3. Membuat class , object dan attribute pada program java	121
4. Membuat class, object dan method nya pada program java	123
5. jenis method accesor dan mutator	124
C. LATIHAN / TUGAS	127
D. REFERENSI	127
PERTEMUAN 11	128
KONSEP PEMROGRAMAN BERORIENTASI OBJEK <i>CONSTRUCTOR</i>	128
A. TUJUAN PEMBELAJARAN.....	128
B. URAIAN MATERI	128
1. Bagaimana constructor bekerja	129
2. Jenis jenis constructor	130

C. LATIHAN/TUGAS.....	137
D. REFERENSI.....	138
PERTEMUAN 12	140
KONSEP PEMROGRAMAN BERORIENTASI OBJEK	
<i>INHERITANCE</i> /PEWARISAN	140
A. TUJUAN PEMBELAJARAN	140
B. URAIAN MATERI.....	140
1. Bentuk Umum penulisan <i>Inheritance</i>	140
2. Jenis Jenis <i>Inheritance</i>	147
C. LATIHAN / TUGAS	151
D. REFERENSI	152
PERTEMUAN 13	153
KONSEP PEMROGRAMAN BERORIENTASI OBJEK <i>POLYMORPHISM</i>	153
A. TUJUAN PEMBELAJARAN.....	153
B. URAIAN MATERI	153
1. Jenis jenis <i>Polymorphism</i>	154
2. Method overloading	155
3. Method overriding.....	157
4. Perbedaan method overloading dan method overriding.....	159
5. Beberapa Contoh lain Method Overloading dan Method Overriding	160
C. LATIHAN / TUGAS	161
D. REFERENSI	162
PERTEMUAN 14	163
ABSTRACT CLASS INTERFACE	163
A. TUJUAN PEMBELAJARAN.....	163
B. URAIAN MATERI	163
1. Mengapa menggunakan <i>abstract class</i>	164
2. Aturan penggunaan abstract class pada bahasa java	165
3. Bentuk Umum pendeklarasian abstract class	165
4. Interface pada java	168
5. Mengapa menggunakan Interface	168
6. Bentuk Umum penulisan Interface	168
7. Perbedaan antara Abstract class dan interface pada Bahasa Java	171
1. Contoh program untuk perbedaan antara abstract class dan interface	172

C. LATIHAN/TUGAS	174
D. REFERENSI	174
PERTEMUAN 15	175
ENCAPSULATION, PACKAGE	176
A. TUJUAN PEMBELAJARAN.....	176
B. URAIAN MATERI	176
1. Cara mengimplementasikan enkapsulasi pada java :	176
2. Keuntungan menggunakan enkapsulasi:	179
3. Package pada Java	182
4. Keuntungan menggunakan Package	183
5. Jenis jenis Package	183
C. LATIHAN / TUGAS	185
D. REFERENSI	186
PERTEMUAN 16	187
ACCESS MODIFIER.....	187
A. TUJUAN PEMBELAJARAN.....	187
B. URAIAN MATERI	187
1. Jenis jenis access modifier	187
2. Default access modifier	188
3. Private access modifier.....	190
4. Protected access modifier.....	191
5. Public access modifier	193
6. Access Modifiers dengan Method Overriding.....	195
C. LATIHAN/TUGAS	196
PERTEMUAN 17	197
ARRAY / LARIK.....	198
A. TUJUAN PEMBELAJARAN.....	198
B. URAIAN MATERI	198
1. Jenis jenis Array	199
2. Deklarasi Array satu dimensi :	199
3. Instansiasi / Membuat Object dan mengakses Array	200
4. Menyalin Array (Copying)	205
5. Manipulasi Array.....	206
6. Array multidimensi	207

7. Contoh program sederhana <i>Array</i> multidimensi	209
C. LATIHAN /TUGAS	210
D. REFERENSI	211
PERTEMUAN 18	212
<i>EXCEPTION HANDLING</i>	212
A. TUJUAN PEMBELAJARAN.....	212
B. URAIAN MATERI	212
1. Kelebihan menggunakan Exception handling	212
2. Perbedaan error dan exception.....	213
3. Jenis jenis dari exception.....	214
4. Hirarki dari class exception	215
5. Keyword Java exception	215
6. Penggunaan Blok try catch	216
7. Cara kerja blok try – catch	217
8. Penggunaan Keyword Throw.....	220
9. Penggunaan Blok Finally	223
C. LATIHAN/TUGAS	224
D. REFERENSI	225
DAFTAR PUSTAKA.....	226
RENCANA PEMBELAJARAN SEMESTER	228

DAFTAR GAMBAR

Gambar 1. 1 Ilustrasi Java Run Time Environment	3
Gambar 1. 2 compiler dan interpreter Java	6
Gambar 1. 3 Class Loader	6
Gambar 1. 4 Instalasi Netbean	8
Gambar 2. 1 Jenis Jenis Variable	16
Gambar 3. 1 Hirarki Database	27
Gambar 5. 1 Input dan Output Stream	55
Gambar 5. 2 Standar I/O Stream	55
Gambar 6. 1 diagram flow chart pernyataan IF	70
Gambar 6. 2 Flow Chart Nested If.....	72
Gambar 6. 3 Flowchart diagram if-else	74
Gambar 6. 4 Flow diagram Pernyataan IF..ELSE..IF	76
Gambar 7.1 diagram flowchart Switch.....	83
Gambar 8. 1 Jenis Perulangan	93
Gambar 8. 2 diagram Flowchart Perulangan For	94
Gambar 9. 1 diagram flowchart perulangan while	107
Gambar 9. 2 diagram flowchart do-while.....	113
Gambar 10. 1 Paradigma pemrograman berorientasi objek	118
Gambar 10. 2 ilustrasi sebuah object Konsep OOP	119
Gambar 10. 3 ilustrasi sebuah class Konsep OOP	120
Gambar 10. 4 ilustrasi sebuah object dengan Value	120
Gambar 10. 5 membuat class mobil.....	121
Gambar 10. 6 memberi penamaan class mobil	121
Gambar 11. 1 Jenis Constructor	130
Gambar 11. 2 Default Constructor	131
Gambar 12. 1 ilustrasi inheritance java OOP	141
Gambar 12. 2 Single inheritance.....	147
Gambar 12. 3 Multiple inheritance	148
Gambar 12. 4 Multilevel inheritance.....	149
Gambar 12. 5 Hierarchical inheritance.....	150
Gambar 14. 1 aturan penggunaan abstract class.....	165
Gambar 15. 1 Ilustrasi Encapsulation	176
Gambar 15. 2 ilustrasi <i>package java</i>	182

Gambar 17. 1 1 Ilustrasi Objek array	199
Gambar 17. 2 Definisi Array 1 Dimensi	199
Gambar 17. 3 Array multi dimensi.....	208
Gambar 17. 4 Meng akses Array multi dimensi.....	208
Gambar 17. 5 Contoh pesan <i>Excetion</i>	213
Gambar 17. 6 Jenis jenis exception	214
Gambar 18. 1 Jenis Jenis <i>Exception</i>	214
Gambar 18. 2 Hirarki <i>Exception</i>	215
Gambar 18. 3 cara kerja blok try catch	217
Gambar 18. 4 <i>Blok Finally</i>	223

DAFTAR TABEL

Tabel 3. 1 Jenis jeni tipe data primitif	28
Tabel 4. 1 Simbol Operator Aritmatika	41
Tabel 4. 2 Simbol Operator Penugasan	42
Tabel 4. 3 Operator Increment decrement	44
Tabel 4. 4 Operator Logika	45
Tabel 4. 5 Operator Pembanding	46
Tabel 4. 6 Operator <i>Bitwise</i>	50
Tabel 5. 1 Tipe tipe Input Metode Pembacaan.....	57
Tabel 5. 2 Tipe tipe Metode class BufferedReader	60
Tabel 11. 1 Tabel Golongan.....	138
Tabel 13. 1 Perbedaan method overloading dan method overriding	159
Tabel 14. 1 Perbedaan Abstract dan Interface	171
Tabel 16. 1 cakupan access modifier	188
Tabel 18. 1 Tabel 18. 1 Keyword pada exception	216

PERTEMUAN 1

PENGANTAR PEMROGRAMAN BAHASA JAVA

A. TUJUAN PEMBELAJARAN

Mahasiswa mampu memahami sejarah bahasa java, memahami Terminologi bahasa java, mengerti JDK, JVM, JRE, dapat membuat program java pertama.

B. URAIAN MATERI

1. Pengantar bahasa Java

JAVA dikembangkan oleh Sun Microsystems Inc pada tahun 1991, kemudian Java diakuisisi oleh Oracle Corporation dan dikembangkan oleh James Gosling dan Patrick Naughton. Bahasa java adalah bahasa pemrograman yang sederhana. Menulis, mengkompilasi, dan men-debug program, sangat mudah dan bahasa java membantu untuk membuat program modular dan kode yang dapat digunakan kembali.

a. Terminologi Bahasa Java

Sebelum kita mulai belajar Java, mari kita kenali istilah-istilah umum java. *Java Virtual Machine* (JVM) atau Mesin Virtual Java Ini umumnya disebut sebagai JVM. Sebelum kita membahas tentang JVM mari kita lihat tahapan-tahapan eksekusi program. Tahapannya adalah sebagai berikut: kita menulis programnya, kemudian kita compile programnya dan terakhir kita menjalankan programnya.

- 1) Penulisan program java yang dilakukan oleh programmer .
- 2) Kompilasi program dilakukan oleh kompiler javac, javac adalah kompiler java utama yang termasuk dalam java development kit (JDK). Dibutuhkan program java sebagai input dan menghasilkan bytecode java sebagai output.
- 3) Pada fase ketiga, JVM mengeksekusi *bytecode* yang dihasilkan oleh compiler. Ini disebut fase menjalankan program (Run Program).

Sehingga fungsi utama JVM adalah untuk mengeksekusi bytecode yang dihasilkan oleh compiler. Setiap sistem operasi memiliki JVM yang berbeda, namun output yang mereka hasilkan setelah eksekusi *bytecode*

sama di semua sistem operasi. Itu sebabnya kami menyebut java sebagai bahasa platform independen.

b. Bytecode

Seperti penjelasan di atas, kompiler javac dari JDK mengkompilasi kode sumber java menjadi *bytecode* sehingga dapat dieksekusi oleh JVM. *Bytecode* disimpan dalam file .class oleh compiler.

2. Java Development Kit (JDK)

Saat menjelaskan JVM dan bytecode, istilah umumnya adalah JDK. Mari kita bahas tentang itu. Seperti namanya, ini adalah kit pengembangan java lengkap yang mencakup JRE (Java Runtime Environment), kompiler dan berbagai alat seperti JavaDoc, Java debugger dll.

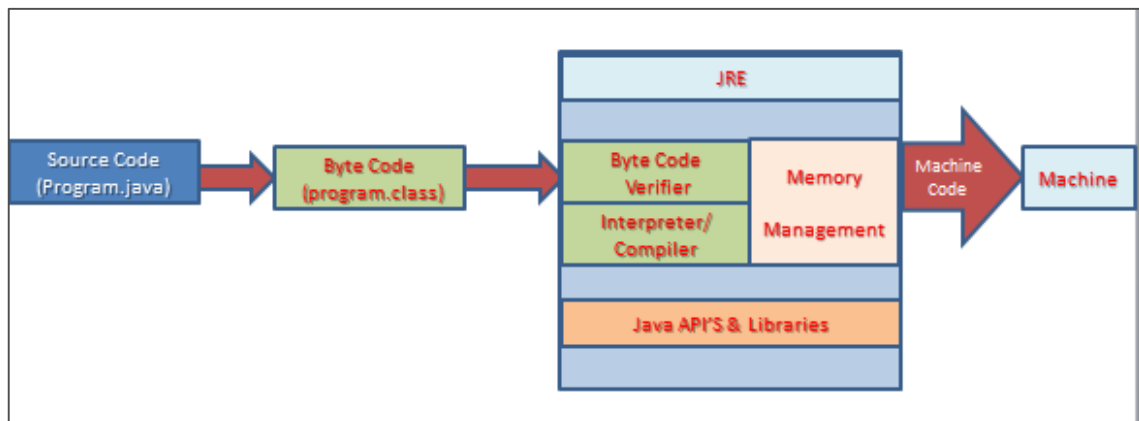
Untuk membuat, mengkompilasi, dan menjalankan program Java, Anda perlu menginstal JDK di komputer Anda.

3. Java Runtime Environment(JRE)

Java Run-time Environment (JRE) adalah bagian dari Java Development Kit (JDK). Merupakan distribusi perangkat lunak yang dapat di download JRE secara gratis yang memiliki Java Class Library, alat khusus, dan JVM yang berdiri sendiri.

Ini adalah lingkungan paling umum yang tersedia di perangkat untuk menjalankan program java. Kode sumber Java dikompilasi dan diubah menjadi bytecode Java. Jika Anda ingin menjalankan bytecode ini di platform apa pun, Anda harus download JRE.

JRE memuat kelas, memverifikasi akses ke memori, dan mengambil sumber daya sistem. JRE bertindak sebagai lapisan di atas sistem operasi.



Gambar 1. 1 Ilustrasi Java Run Time Environment

4. Fitur fitur utama Bahasa JAVA

a. Java adalah bahasa platform independen

Compiler (javac) mengonversi kode sumber (file.java) ke *bytecode* (file.class). Seperti disebutkan di atas, JVM mengeksekusi bytecode yang dihasilkan oleh compiler. *Bytecode* /Kode byte ini dapat berjalan di platform apa pun seperti Windows, Linux, Mac OS dll. Yang berarti program yang dikompilasi di windows dapat berjalan di Linux dan sebaliknya. Setiap sistem operasi memiliki JVM yang berbeda, namun output yang mereka hasilkan setelah eksekusi *bytecode* sama di semua sistem operasi. Itu sebabnya java disebut sebagai bahasa platform independen.

b. Java adalah bahasa Berorientasi Objek (OOP)

Pemrograman berorientasi objek adalah cara mengatur program sebagai kumpulan objek, yang masing-masing mewakili turunan dari kelas.

4 konsep utama pemrograman Berorientasi Objek adalah:

- 1) Abstraksi
- 2) Enkapsulasi
- 3) Pewarisan
- 4) Polimorfisme

c. Simple /Sederhana

Java dianggap sebagai salah satu bahasa yang sederhana karena tidak memiliki fitur yang kompleks seperti Operator overloading, Multiple inheritance, pointer dan alokasi memori Explicit.

d. *Robust* /Bahasa yang Kuat

Kuat berarti dapat diandalkan. Bahasa pemrograman Java dikembangkan dengan cara yang memberikan banyak penekanan pada pemeriksaan awal untuk kemungkinan kesalahan, itu sebabnya kompiler java mampu mendeteksi kesalahan yang tidak mudah dideteksi dalam bahasa pemrograman lain. Fitur utama java yang membuatnya kuat adalah pengumpulan sampah, Penanganan Pengecualian, dan alokasi memori.

e. *Secure* /Aman

Bahasa java tidak memiliki operasi pointer , pada pemrograman dengan mode operasi pointer merupakan hal yang luar biasa untuk optimalisasi dan pembuatan program yang handal serta efisien. Namun operasi pointer dapat menjadi bumerang , karena pointer merupakan sarana luar biasa untuk pengaksesan tak diotorisasi. Dengan peniadaan operasi pointer, Java dapat menjadi bahasa yang lebih aman sehingga tidak dapat mengakses array terikat di java. Itu sebabnya beberapa kelemahan keamanan seperti tumpukan atau buffer overflow tidak mungkin dieksploitasi di Java.

f. Bahasa Java terdistribusi

Dengan menggunakan bahasa pemrograman java kita dapat membuat aplikasi terdistribusi. RMI (Remote Method Invocation) dan EJB (Enterprise Java Beans) digunakan untuk membuat aplikasi terdistribusi di java. Dengan kata sederhana: Program java dapat didistribusikan di lebih dari satu sistem yang terhubung satu sama lain menggunakan koneksi internet. Objek pada satu JVM (mesin virtual java) dapat menjalankan prosedur pada JVM jarak jauh.

g. Multithreading

Bahasa Java mendukung multithreading. Multithreading adalah fitur Java yang memungkinkan eksekusi bersamaan dari dua atau lebih bagian program untuk pemanfaatan CPU secara maksimal.

h. Portabel

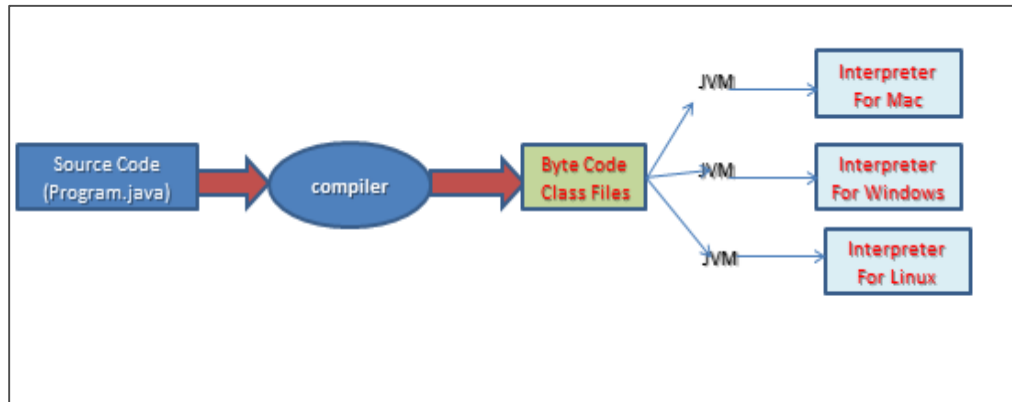
Seperti dibahas di atas, kode java yang ditulis di satu mesin dapat berjalan di mesin lain. Kode byte platform independen dapat dibawa ke platform apa pun untuk dieksekusi yang membuat kode java portabel.

5. Compiler dan Interpreter Java

Compiler adalah suatu program yang menerjemahkan bahasa program (source code) kedalam bahasa objek (obyek code) atau istilah pada java disebut *byte code*.

Sedangkan Interpreter adalah Perangkat lunak yang mampu mengeksekusi code program (yang ditulis oleh programmer) lalu menterjemahkannya ke dalam bahasa mesin atau byte code, sehingga mesin melakukan instruksi yang diminta oleh programmer pada java di dieksekusi oleh JVM

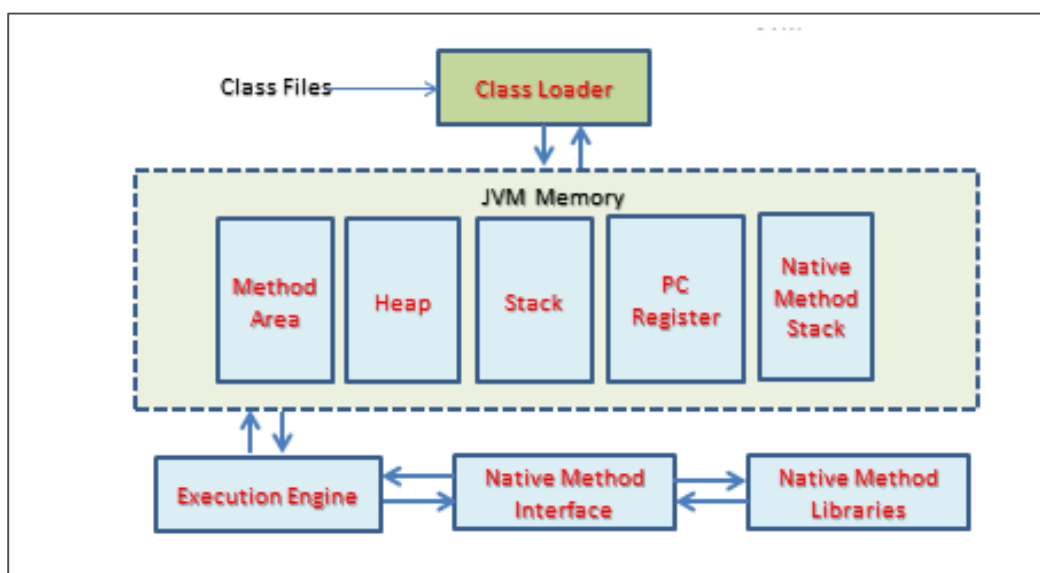
Program yang ditulis dalam bahasa tingkat tinggi (*High Level Language*) tidak dapat dijalankan di mesin mana pun secara langsung. Perlu diterjemahkan terlebih dahulu ke dalam bahasa mesin tertentu. Kompiler javac melakukan hal ini, dibutuhkan program java (file .java yang berisi kode sumber) dan menerjemahkannya ke dalam kode mesin (disebut sebagai byte code atau file .class). Java Virtual Machine (JVM) atau mesin virtual Java mengeksekusi *byte code* tersebut untuk menghasilkan output yang sama di setiap sistem operasi berbeda. Artinya, kode byte yang dihasilkan pada Windows dapat dijalankan di Mac OS dan sebaliknya. Itu sebabnya java sebagai bahasa platform independen. Hal yang sama dapat dilihat pada gambar di bawah ini:



Gambar 1. 2 compiler dan interpreter Java

Kesimpulan :

Java Virtual machine (JVM) adalah mesin virtual yang berjalan pada komputer /laptop dan mengeksekusi *byte code* . JVM tidak memahami kode sumber Java yang ditulis oleh programmer, oleh karena itu kita perlu memiliki Compiler javac yang mengkompilasi file *.java untuk mendapatkan file *.class yang berisi *Code byte* yang dipahami oleh JVM. JVM membuat java portabel (tulis sekali, jalankan di mana saja). Setiap sistem operasi memiliki JVM yang berbeda, namun output yang mereka hasilkan setelah eksekusi *byte code* sama di semua sistem operasi.



Gambar 1. 3 Class Loader

Cara kerja JVM:

Class Loader: Class loader membaca file .class dan menyimpan kode byte di area metode.

Area Metode: Hanya ada satu area metode dalam JVM yang dibagikan di antara semua kelas. Ini menyimpan informasi tingkat kelas dari setiap file .class.

Heap: Heap adalah bagian dari memori JVM tempat objek dialokasikan. JVM membuat objek Kelas untuk setiap file .class.

Stack: Stack juga merupakan bagian dari memori JVM tetapi tidak seperti Heap, ini digunakan untuk menyimpan variabel sementara.

Register PC: Ini melacak instruksi mana yang telah dieksekusi dan mana yang akan dieksekusi. Karena instruksi dieksekusi oleh utas, setiap utas memiliki register PC yang terpisah.

Native Method stack : Metode asli dapat mengakses area data runtime dari mesin virtual.

Native Method interface: Ini memungkinkan kode java untuk memanggil atau dipanggil oleh aplikasi asli. Aplikasi asli adalah program yang khusus untuk perangkat keras dan OS suatu sistem.

Garbage collection: Sebuah instance kelas secara eksplisit dibuat oleh kode java dan setelah digunakan secara otomatis dihancurkan oleh pengumpulan sampah untuk manajemen memori.

6. Instalalasi JDK dengan Netbeans 8.2

Sebelum kita membuat aplikasi dengan java maka anda harus menginstal software Bahasa pemrograman java. java dapat anda dapatkan dengan cara download di alamat link berikut :

<https://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-3413139-esa.html> . JDK ini mencakup bundel Java SE dari NetBeans IDE, yang merupakan lingkungan pengembangan terintegrasi untuk mengembangkan aplikasi pada platform Java.

Setelah masuk ke halaman web resminya kita harus memilih *accept License Agreement* [1], Kemudian pilih JDK sesuai dengan sistem operasi yang digunakan [2].

The screenshot shows the NetBeans installation window. At the top, there are two radio buttons: "Accept License Agreement" (selected) and "Decline License Agreement". Below this is a table titled "Java SE and NetBeans Cobundle (JDK 8u111 and NB 8.2)". The table has three columns: "Product / File Description", "File Size", and "Download". The rows list the download links for Linux x86, Linux x64, Mac OS X x64, Windows x86, and Windows x64. A red box highlights the "Accept License Agreement" button and the "Download" column, with a red '1' and '2' respectively.

Product / File Description	File Size	Download
Linux x86	286.73 MB	jdk-8u111-nb-8_2-linux-i586.sh
Linux x64	282.57 MB	jdk-8u111-nb-8_2-linux-x64.sh
Mac OS X x64	342.99 MB	jdk-8u111-nb-8_2-macosx-x64.dmg
Windows x86	317.21 MB	jdk-8u111-nb-8_2-windows-i586.exe
Windows x64	326.03 MB	jdk-8u111-nb-8_2-windows-x64.exe

Below the table, there is a link for "License".

Gambar 1. 4 Instalasi Netbean

Selanjutnya ikut langkah langkah instalasi, hingga selesai.

7. Program java pertama

Dalam pertemuan ini, kita akan melihat bagaimana menulis, mengkompilasi dan menjalankan program java dengan Netbeans . kita akan bahas sintaks java, dan beberapa cara untuk menjalankan program java.

```
public class FirstJavaProgram {
    public static void main(String[] args) {
        System.out.println("Ini adalah program java pertama saya");
    } //End of main
} //End of FirstJavaProgram Class
```

Penjelasan program yang telah kita tulis di atas adalah :

```
public class FirstJavaProgram {
```

Ini adalah baris pertama dari program java . Setiap aplikasi java harus memiliki setidaknya satu definisi kelas yang terdiri dari kata kunci kelas diikuti dengan nama kelas. kata kunci code , itu berarti tidak boleh diubah, kita harus menggunakannya apa adanya. Untuk nama kelas bisa apa saja.

```
public static void main(String[] args) {
```

baris berikutnya dalam program ini seperti diatas adalah

public: Ini membuat metode utama menjadi publik yang berarti bahwa kita dapat memanggil metode dari luar kelas.

static: Kita tidak perlu membuat objek untuk menjalankan metode statis. Mereka bisa berjalan sendiri.

void: Tidak mengembalikan nilai apa pun.

main: Ini adalah nama metode. metode titik awal masuk dari mana JVM dapat menjalankan program kita.

(String[] args): Digunakan untuk argumen baris perintah yang diteruskan sebagai string. akan dibahas pada pertemuan terpisah.

```
System.out.println("This is my first program in java");
```

Metode ini mencetak konten di dalam tanda kutip ganda ke dalam layar dan menyisipkan baris baru setelahnya.

8. Aplikasi yang dijalankan dengan bahasa Java

Aplikasi yang digunakan perangkat di mana Java saat dijalankan di antaranya adalah sebagai berikut:

1. Aplikasi Desktop seperti acrobat reader, media player, antivirus, dll.
2. Aplikasi Web seperti irttc.co.in, javatpoint.com, dll.
3. Aplikasi untuk Perusahaan seperti aplikasi perbankan.
4. Aplikasi Mobile
5. *Embedded System*
6. *Smart Card*
7. Robotika
8. Game , dll.

9. Jenis jenis Aplikasi Java

Berdasarkan klasifikasi jenis ada 4 jenis aplikasi yang dapat dibuat menggunakan pemrograman Java:

a. Aplikasi *Stand Alone*

Aplikasi *Stand Alone* juga dikenal sebagai aplikasi desktop atau aplikasi berbasis *windows*. Ini adalah perangkat lunak tradisional yang perlu kita instal di setiap mesin. Contoh aplikasi standalone adalah Media player, antivirus, dll. AWT dan Swing digunakan di Java untuk membuat aplikasi standalone.

b. Aplikasi berbasis Web

Aplikasi yang berjalan di sisi server dan membuat halaman dinamis atau disebut aplikasi berbasis web. Saat ini, teknologi Servlet, JSP, Struts, Spring, Hibernate, JSF, dll. digunakan untuk membuat aplikasi berbasis web.

c. Aplikasi *Enterprise*

Aplikasi yang sifatnya terdistribusi, seperti aplikasi perbankan, dll disebut aplikasi *enterprise*. Ini memiliki keunggulan seperti keamanan tingkat tinggi, penyeimbangan beban, dan pengelompokan. pada Java, EJB digunakan untuk membuat aplikasi *Enterprise*

d. Aplikasi Mobile

Aplikasi yang dibuat untuk perangkat *Mobile* disebut aplikasi seluler. Saat ini, seperti Android dan Java ME digunakan untuk membuat aplikasi seluler.

10. Platform pada Java

Berdasarkan platform ada 4 platform pada Bahasa Java:

a. Java SE (Java Standard Edition)

Platform ini adalah platform inti pemrograman Java. termasuk pemrograman API Java seperti `java.lang`, `java.io`, `java.net`, `java.util`, `java.sql`, `java.math` dll. termasuk topik inti seperti OOP, *String*, *Regex*, *Exception*, *Class*, *Multithreading*, I/O Stream, Jaringan, AWT, *Swing* dll.

b. Java EE (Edisi Java Enterprise)

Platform Ini adalah platform *enterprise* terutama digunakan untuk mengembangkan aplikasi berbasis web dan Aplikasi *Enterprise* . dibangun di atas platform Java SE. mencakup topik seperti Servlet, JSP, *Web Service*, EJB, JPA, dll.

c. Java ME (Java Micro Edition)

Platform ini adalah platform micro yang didedikasikan untuk aplikasi berbasis mobile seperti android.

d. JavaFX

Platform ini adalah open source untuk aplikasi desktop, mobil , dan *embedded* system yang dibangun di atas Java. platform ini adalah upaya kolaboratif oleh banyak individu dan perusahaan dengan tujuan menghasilkan toolkit yang modern, efisien, dan berfitur lengkap

C. LATIHAN / TUGAS

1. Jelaskan apa itu Jre ? Apa kegunaannya!
2. Jelaskan apa itu JDK?
3. Jelaskan apa itu Java Virtual Machine?
4. Jelaskan tahap membuat proyek aplikasi baru dan class baru di netbeans!
5. Berdasarkan jenis , aplikasi apa saja yang dapat dibuat dengan menggunakan pemrograman Java?
6. Berdasarkan platform , platform apa saja pada Java?

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

PERTEMUAN 2

VARIABEL PADA JAVA

A. TUJUAN PEMBELAJARAN

Setelah mempelajari pertemuan 2 Mahasiswa dapat memahami apa itu variabel, Mahasiswa dapat membedakan jenis jenis variable , Mahasiswa dapat aturan penulisan variable , serta mempraktekan penggunaan variabel pada bahasa program java

B. URAIAN MATERI

Variabel adalah nama yang dikaitkan sebagai lokasi penyimpanan sementara untuk menyimpan informasi atau Nilai , nilai variable dapat diubah di pernyataan dimanapun didalam program . Misalnya ketika kita menulis `int i=10;` di sini nama variabel adalah `i` yang diberikan dengan nilai 10, `int` adalah tipe data yang menyatakan bahwa variabel ini dapat menyimpan nilai integer. Kita akan membahas tipe data di tutorial berikutnya. Dalam pertemuan ini, kita akan membahas tentang variabel terlebih dahulu .

1. Cara mendeklarasikan Variabel

Untuk mendeklarasikan variabel berikut sintaks dibawah ini:

```
Tipe Data NamaVariabel = nilai;
```

nilai di sini adalah opsional dan dapat berubah ubah , kita dapat mendeklarasikan variabel terlebih dahulu kemudian baru menetapkan nilainya.

Sebagai contoh: **var1** adalah variabel dan **int** adalah tipe data(akan dibahas pada pertemuan berikutnya) . Variabel **var1** ini akan menyimpan nilai integer,

Variabel **var2** ini akan menyimpan nilai type data float, Variabel **var3** ini akan menyimpan nilai type data *double*, Variabel **var4** ini akan menyimpan nilai type data *double*, Variabel **var5** ini akan menyimpan nilai type data string .

```
int var1;

float var2;

double var3;

boolean var4

String var5
```

Demikian juga kita dapat menetapkan nilai ke variabel langsung sambil mendeklarasikannya, seperti dibawah ini:

```
char char = 'X';

int nomor = 100;

float var2=100.5

double var3=100.55551111

boolean var 4= true
```

atau kita dapat juga menulis variabel dahulu dan memberikan nilai seperti dibawah ini :

```
char char;

int number;

float var2;

double var3;

boolean var4;
```

```
char='X'  
  
number=10  
  
var2=0.5  
  
var3=100.555555  
  
var4=true
```

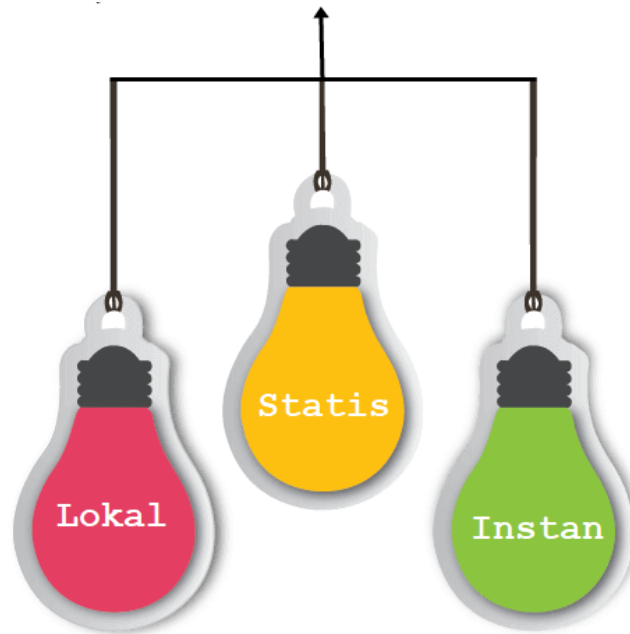
2. Aturan umum penamaan variabel di java

- a. Penamaan variabel tidak boleh mengandung spasi, contoh: int num ber = 100; tidak valid karena nama variabel memiliki spasi di dalamnya.
- b. Hindari penggunaan nama variabel berupa keyword java
- c. Tidak didahului oleh angka dan tidak mengandung tanda baca (",?;! dst)
- d. Nama variabel dapat dimulai dengan karakter khusus seperti \$ dan _
- e. Sesuai standar pengkodean pada bahasa java, nama variabel harus dimulai dengan huruf kecil, misalnya int nomor; Untuk nama variabel yang memiliki lebih dari satu kata bisa digabungkan seperti ini
int bilangan_Kedua ; int bilangan_Kedua;
- f. Nama Variabel pada java huruf kecil dan besar dibedakan (case sensitive)

3. Jenis jenis Variabel pada Java

Terdapat tiga jenis variabel pada bahasa Java yaitu : .

- a. Variabel lokal
- b. Variabel statis (atau class variable)
- c. Variabel instan



Gambar 2. 1 Jenis Jenis Variable

a. Variable Lokal

Variabel-variabel ini dideklarasikan di dalam *method class* . Cakupannya terbatas pada metode itu sendiri yang berarti kita tidak dapat mengubah nilainya dan mengaksesnya di luar metode.

Pada contoh ini, saya kita mendeklarasikan variabel instan dengan nama yang sama dengan variabel lokal, ini untuk menunjukkan ruang lingkup variabel lokal.

```
public class VariableExample {  
    // instan variabel  
    public String Variabelku="instance variable";  
  
    public void Methodku() {  
        // lokal variabel  
        String Variabelku = "Inside Method";  
        System.out.println(Variabelku);  
    }  
}
```

```
public static void main(String args[]){  
    // membuat object  
    VariableExample objek = new VariableExample();  
  
    /* Kita memanggil method dan mengubah nilai  
    * dari variabelku. Kita akan tampilkan variableku  
    kembali  
    * method ini memanggil untuk mendemonstrasikan batasan  
    local * variable  
    */  
    System.out.println("Calling Method");  
    objek.Methodku();  
    System.out.println(obj.variabelku);  
}  
}
```

Output:

```
Calling Method  
Inside Method  
instance variable
```

b. Variabel Statis (*Class Variable*)

Variabel statis merupakan jenis modifier di Java yang berfungsi agar suatu atribut atau method dapat diakses oleh kelas atau objek tanpa harus melakukan instansiasi terhadap kelas tersebut yang dideklarasikan menggunakan keyword *static*.

Sebagai contoh, Jika kita membuat tiga objek kelas dan mengakses variabel statis ini, itu akan menjadi umum untuk semua, perubahan yang dilakukan pada variabel menggunakan salah satu objek akan tercermin saat kita mengaksesnya melalui objek lain.

Contoh Variabel Statis :

```
public class StaticVarExample {  
    public static String myClassVar="class or static  
variable";  
  
    public static void main(String args[]){  
        StaticVarExample objek = new StaticVarExample();  
        StaticVarExample objek2 = new StaticVarExample();  
        StaticVarExample objjek3 = new StaticVarExample();  
  
        //All three will display "class or static  
variable"  
        System.out.println(objek.myClassVar);  
        System.out.println(objek2.myClassVar);  
        System.out.println(objek3.myClassVar);  
  
        //changing the value of static variable using obj2  
        objek2.myClassVar = "Changed Text";  
  
        //All three will display "Changed Text"  
        System.out.println(objek.myClassVar);  
        System.out.println(objek2.myClassVar);  
        System.out.println(objek3.myClassVar);  
    }  
}
```

Output

```
class or static variable
```

```
class or static variable  
class or static variable  
Changed Text  
Changed Text  
Changed Text
```

Seperti yang kita lihat, hasil output ketiga pernyataan menampilkan output yang sama terlepas dari instance yang digunakan untuk mengaksesnya. Itu sebabnya kita dapat mengakses variabel statis tanpa menggunakan objek seperti ini:

```
System.out.println(myClassVar);
```

Perhatikan bahwa hanya variabel statis yang dapat diakses seperti ini. Ini tidak berlaku untuk instance dan variabel lokal.

c. Variabel *instance*

Setiap instance (objek) kelas memiliki salinan variabel instance sendiri. Tidak seperti variabel statis, variabel instan memiliki salinan terpisah dari variabel instan. Kami telah mengubah nilai variabel instan menggunakan objek obj2 dalam program berikut dan ketika kita menampilkan variabel menggunakan ketiga objek, hanya nilai obj2 yang diubah, yang lain tetap tidak berubah. Ini menunjukkan bahwa mereka memiliki salinan variabel instan mereka sendiri.

Contoh Variabel Instan

```
public class InstanceVarExample {  
    String myInstanceVar="instance variable";  
  
    public static void main(String args[]){  
        InstanceVarExample obj = new InstanceVarExample();  
        InstanceVarExample obj2 = new  
InstanceVarExample();
```

```
InstanceVarExample obj3 = new
InstanceVarExample();

System.out.println(obj.myInstanceVar);
System.out.println(obj2.myInstanceVar);
System.out.println(obj3.myInstanceVar);

obj2.myInstanceVar = "Changed Text";

System.out.println(obj.myInstanceVar);
System.out.println(obj2.myInstanceVar);
System.out.println(obj3.myInstanceVar);
}
}
```

Output:

```
instance variable
instance variable
instance variable
instance variable
Changed Text
instance variable
```

4. Contoh contoh program dan jenis variable

a. Contoh Program Variabel menambahkan Dua Angka

```
public class sederhana{  
    public static void main(String[] args){  
        int a=5;  
        int b=10;  
        int c=a+b;  
        System.out.println(c);  
    }  
}
```

Output :

```
15
```

b. Contoh Program Variabel dengan 2 jenis data

```
public class Simple{  
    public static void main(String[] args){  
        int a=40;  
        float b=a;  
        System.out.println(a);  
        System.out.println(b);  
    }  
}
```

Output :

```
40  
40.0
```

c. Contoh Program Variabel *Overflow*

```
class Simple{  
    public static void main(String[] args){
```

```
//Overflow
int a=130;
byte b=(byte)a;
System.out.println(a);
System.out.println(b);
}}
```

Output :

```
130
-126
```

d. Contoh Program Variabel byte

```
class Simple{
public static void main(String[] args){
byte a=10;
byte b=10;
//byte c=a+b;//Compile Time Error: because a+b=20 will
be int
byte c=(byte) (a+b);
System.out.println(c);
}}
```

Output

```
20
```

e. Contoh Program Variabel *lokal dan Method*

```
public class lokalvariableMethod {
    //Membuat method yang bernama Makanan
```

```
//didalam class VariableLokal

public void makanan(){

    String MakananFavorit = "Ayam Geprek ";

    int Harga = 15000;

    System.out.println(MakananFavorit + Harga);

}

public static void main(String[] args) {

    // TODO code application logic here

    //Membuat sebuah Objek dari class variableLokal

    //dan memanggil method game pada method main

    lokalvariableMethod    latihan    =    new

    lokalvariableMethod();

    latihan.makanan();

}

}
```

Output

Ayam Geprek 15000

Catatan : Method akan dibahas pada pertemuan selanjutnya

f. Contoh Program Variabel *Static*

```
public class staticvariable {

    public static String Hobi;

    //Membuat variable tanpa Value/Nilai pada class

    staticvariable

    public static final String Hewan = "Ikan Hias";

    public static void main(String[] args) {
```

```

        // TODO code application logic here

        Hobi = "Sepak Bola "; //Mengisi Value pada variable
Hobi

        System.out.println("Hobi Saya " + Hobi); //Memanggil
Variable Hobi

        System.out.println("Peliharaan Saya " +
Hewan); //Memanggil Variable Hewan

    }
}

```

Output :

```

Hobi Saya Sepak Bola
Peliharaan Saya Ikan Hias

```

C.LATIHAN /TUGAS

1. Modifikasi Program B5 no 3 dengan merubah variabel dan nilai variabel

Hasil Output yang diharapkan :

```

50
50.0

```

2. Modifikasi Program B5 no 6 dengan menambah variabel Makanan , variable static dengan menambahkan variable Minuman, berikan nilai variabel sesuai output yang diharapkan :

```

public class staticvariable {

    public static String Hobi;

    //Membuat variable tanpa Value/Nilai pada class
staticvariable

    public static final String Hewan = "Ikan Hias";
}

```

```
public static void main(String[] args) {  
    // TODO code application logic here  
  
    Hobi = "Sepak Bola ";//Mengisi Value pada variable Hobi  
  
    System.out.println("Hobi Saya " + Hobi);//Memanggil  
Variable Hobi  
  
    System.out.println("Peliharaan Saya " +  
Hewan);//Memanggil Variable Hewan  
  
    }  
}
```

Hasil Output yang diharapkan :

Makanan Favorit saya : Ayam Geprek

Minuman Favorit saya : Jus Jambu

D. REFERENSI

- Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.
- Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, *publishing as prentice hall*.
- Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.
- Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press
- Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.
- Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

PERTEMUAN 3

TIPE DATA PADA JAVA

A. TUJUAN PEMBELAJARAN

Mahasiswa dapat memahami Tipe tipe Data pada Bahasa java, Mahasiswa dapat membedakan Tipe Data Primitf dan Non Primitif, Mahasiswa dapat mempraktekan penggunaan tipe data bahasa java.

B. URAIAN MATERI

Tipe data adalah pendefinisian jenis data untuk nilai yang dapat diambil oleh suatu variabel, misalnya jika suatu variabel memiliki tipe data int, variabel tersebut hanya dapat mengambil nilai integer. Di java kita memiliki dua kategori tipe data:

Java adalah bahasa yang diketik secara statis. Suatu bahasa diketik secara statis, jika tipe data suatu variabel diketahui pada waktu kompilasi. Ini berarti bahwa kita harus menentukan jenis variabel (Deklarasikan variabel) sebelum kita dapat menggunakannya.

Dalam pertemuan sebelumnya tentang Variabel Java, kita belajar cara mendeklarasikan variabel, mari kita ingat kembali:

```
int angka ;
```

Jadi untuk menggunakan variabel angka dalam program kita, kita harus mendeklarasikannya terlebih dahulu seperti contoh penulisan di atas. Ini adalah praktek pemrograman yang baik untuk mendeklarasikan semua variabel (yang akan kita gunakan) di awal program.

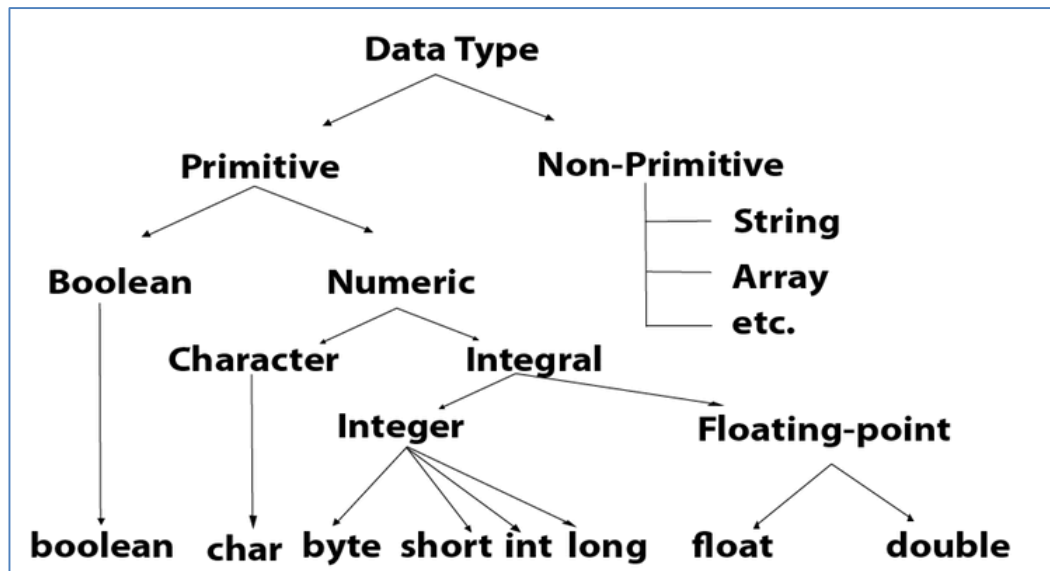
1. Jenis Jenis Tipe Data :

Tipe data menentukan berbagai ukuran dan nilai yang dapat disimpan dalam variabel.

Jenis jenis tipe data dibagi menjadi 2 bagian besar yaitu

- a. Tipe data primitif
- b. Tipe data non-primitif

Array dan String adalah tipe data non-primitif, kita akan membahasnya nanti pada pertemuan selanjutnya. pertemuan 3 ini akan membahas tipe data primitif dan literal di Java.



Gambar 3. 1 Hirarki Database

2. Tipe data primitif

Pada Java memiliki delapan tipe data primitif yaitu : ***boolean, char, byte, short, int, long, float dan double***. Pengembang Java memasukkan tipe data ini untuk menjaga portabilitas java karena ukuran tipe data primitif ini tidak berubah dari satu sistem operasi ke sistem operasi lainnya.

- Tipe data byte, short, int dan long digunakan untuk menyimpan bilangan bulat.
- float dan double digunakan untuk bilangan pecahan.
- char digunakan untuk menyimpan karakter (huruf).
- tipe data boolean digunakan untuk variabel yang memiliki nilai benar/*True* atau salah/*False*.

Tabel 3. 1 Jenis jenis tipe data primitif

Tipe Data Primitif	Keterangan	Ukuran	Jangkauan
byte	Bilangan bulat	8 bit	-128 → 127
short	Bilangan bulat	16 bit	-32.768 → 32.767
int	Bilangan bulat	32 bit	-2.147.483.648 → 2.147.483.647
long	Bilangan bulat	64 bit	-9.223.372.036.854.775,808 → 9.223...807
float	Bilangan pecahan	32 bit (presisi 6-7 bit)	-3.4E38 → 3.4E38
double	Bilangan pecahan	64 bit (presisi 14-15 bit)	-1.7E308 → 1.7E308
char	Karakter (unicode)	16 bit	\u0000 → \uffff
boolean	Logika (true/false)		

a. Tipe Data *byte*:

Tipe data byte adalah contoh tipe data primitif. Tipe data ini adalah menampung bilangan bulat komplement dua bertanda 8-bit. Nilai -range-nya terletak antara -128 hingga 127 (inklusif). Nilai minimumnya adalah -128 dan nilai maksimumnya adalah 127. Nilai defaultnya adalah 0.

Tipe data byte digunakan untuk menyimpan memori dalam array besar di mana penghematan memori paling dibutuhkan. Menghemat ruang karena byte 4 kali lebih kecil dari integer. Tipe ini juga dapat digunakan sebagai pengganti tipe data "int".

Contoh penulisan :

```
byte a = 10, byte b = -20
```

Contoh program sederhana:

```
class JavaExample {  
    public static void main(String[] args) {  
  
        byte angka;  
  
        num = 150;  
  
        System.out.println(angka);  
    }  
}
```

Output:

150

b. Tipe Data *short*:

Tipe data bilangan bulat dengan jangkauan pendek 2 kali lebih kecil dari integer. mempunyai komplement dua bertanda 16-bit. Rentang nilainya terletak antara -32,768 hingga 32,767 (inklusif). Nilai minimumnya adalah -32,768 dan nilai maksimumnya adalah 32,767. Ukuran default dari tipe data ini: 2 byte

Tipe data short lebih besar dari byte dalam hal ukuran dari integer

Contoh penulisan :

```
Short a = 10, short b = -20
```

Contoh program tipe data short

```
class Javashort {  
    public static void main(String[] args) {  
  
        short angka;  
  
    }  
}
```

```
        num = 150;

        System.out.println(angka);

    }

}
```

Output:

150

c. Type data int

Tipe data int adalah tipe data bilangan bulat integer komplement dua bertanda 32-bit. Rentang nilainya terletak antara -2.147.483.648 (-2^{31}) hingga 2.147.483.647 ($2^{31} - 1$) (inklusif). Nilai minimumnya adalah -2.147.483.648 dan nilai maksimumnya adalah 2.147.483.647. Nilai defaultnya adalah 0.

Tipe data int umumnya digunakan sebagai tipe data default untuk nilai integral kecuali tidak ada masalah dengan memori.

Contoh penulisan tipe data int

```
int a = 100000, int b = -200000 ;
```

Contoh program tipe data int :

```
class Javaint{

    public static void main(String[] args) {

        int a=100000, int b=-200000;

        System.out.println("variable a dan b bertipe data
int" );

        System.out.println("nilai a=" +a);

        System.out.println("nilai b =" +b);

    }

}
```

```
    }  
}
```

Output:

```
variable a dan b bertipe data int  
Nilai a=100000  
Nilai b=200000
```

d. Tipe Data *Long*

Type Data *Long* digunakan ketika tipe *int* tidak cukup besar untuk menampung nilai, ia memiliki jangkauan yang lebih luas daripada tipe data *int*, mulai dari -9.223.372.036.854.775.808 hingga 9.223.372.036.854.775.807. ukuran: 8 byte Nilai default: 0

Contoh penulisan :

```
long a = 100000L, long b = -200000L ;
```

Contoh program tipe data long :

```
class JavaExample {  
    public static void main(String[] args) {  
  
        long num = -12332252626L;  
        System.out.println(num);  
    }  
}
```

Output:

-12332252626

e. Tipe Data *Double*

Tipe data **double** dipakai untuk menampung angka pecahan seperti 3.14, 44.53 atau -0.09876. Sama seperti bahasa pemrograman pada umumnya, kita menggunakan tanda titik sebagai pemisah angka bulat dan pecahan, bukan tanda koma seperti yang kita pakai sehari-hari di Indonesia.

Tipe data ini dapat menampung 15 angka desimal ukuran: 8 byte

contoh penulisan :

```
double d1 = 12.3;
```

Contoh program tipe data double :

```
class Javadouble {  
    public static void main(String[] args) {  
  
        double angka = -42937737.9d;  
  
        System.out.println(angka );  
    }  
}
```

Output:

-4.29377379E7

f. Tipe Data *float*

Tipe data **float** sama seperti **double** dipakai untuk menampung angka pecahan. perbedaan antara **float** dan **double** terletak dari jangkauan angka serta tingkat ketelitian.

Tipe data ini dapat menampung 6 sampai 7 angka desimal ukuran: 4 byte.

Contoh penulisan

```
float f1 = 234.5f
```

Contoh penulisan program tipe data float

```
class Javafloat{  
    public static void main(String[] args) {  
  
        float angka= 19.98f;  
        System.out.println(angka);  
    }  
}
```

Output:

```
19.98
```

g. Tipe Data *Boolean*

Tipe data boolean ini adalah tipe data yang mempunyai nilai: true atau false. Tipe data boolean banyak dipakai untuk percabangan kode program atau untuk memutuskan apa yang mesti dijalankan ketika sebuah kondisi terjadi.

Contoh penulisan :

```
Boolean one = false
```

Contoh program tipe data boolean

```
class JavaExample {  
    public static void main(String[] args) {  
  
        boolean b = false;  
        System.out.println(b);  
    }  
}
```



```
}  
  
}
```

Output:

```
false
```

h. Tipe Data *char*

Tipe data char dalam bahasa Java dipakai untuk menampung 1 digit karakter, baik itu berupa huruf, angka maupun karakter lain seperti ^, %, dan #. Variabel yang didefinisikan untuk menampung tipe data char butuh 2 byte

Tipe data char adalah karakter Unicode 16-bit tunggal. Rentang nilainya terletak antara '\u0000' (atau 0) hingga '\uffff' (atau 65.535 inklusif).

Contoh penulisan :

```
char hurufA = 'A'
```

contoh program

```
class Javachar{  
    public static void main(String[] args) {  
  
        char char = 'A';  
        System.out.println(char);  
    }  
}
```

Output:

```
A
```

3. Literals pada Java

Literal adalah suatu nilai konstan yang langsung muncul baik pada bahasa java ataupun pada bahasa pemrograman lain .

```
int num=10;
```

Nilai 10 is adalah Integer literal.

```
char ch = 'A';
```

Nilai A adalah sebuah char literal

a. Literal Integer

Literal integer adalah pemberian nilai ke variabel yang mempunyai tipe data byte, short, int dan long.

b. Char and String Literal

Digunakan untuk tipe data *char* dan *String* .

```
char ch = 'Z';
```

```
String string = "Program studi Sistem Informasi ";
```

4. Contoh contoh Program Tipe data

a. Contoh Program dengan dengan berbagai tipe data

```
public class DatatypeEx {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
  
        int a = 15;  
        short s = 20;  
        byte b = 60;  
        long l = 32536213322311;  
    }  
}
```

```
float f = 65.20298f;

double d = 876.765d;

System.out.println("The integer variable is " + a);
System.out.println("The short variable is " + s);
System.out.println("The byte variable is " + b);
System.out.println("The long variable is " + l);
System.out.println("The float variable is " + f);
System.out.println("The double variable is " + d);

}

}
```

Output :

```
The integer variable is 15
The short variable is 20
The byte variable is 60
The long variable is 3253621332231
The float variable is 65.20298
The double variable is 876.765
```

b. Contoh program dengan tipe data dan konversi tipe data

```
public class dataTypeKonversi {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        byte x;
        int a = 270;
        double b = 128.128;
        System.out.println("int konversi ke byte");
    }
}
```

```
x = (byte) a;
System.out.println("a and x " + a + " " + x);
System.out.println("double Konversi ke int");
a = (int) b;
System.out.println("b dan a " + b + " " + a);
System.out.println("\ndouble konversi ke byte");
x = (byte)b;
System.out.println("b and x " + b + " " + x);
    }
}
```

Output :

```
int konversi ke byte
a and x 270 14
double Konversi ke int
b dan a 128.128 128
double konversi ke byte
b and x 128.128 -128
```

C. LATIHAN / TUGAS

1. Lengkapi titik titik potongan kode dibawah ini dengan tipe data sesuai nilainya

```
..... Var1 = 9;
.... Var2= 8.99f;
.... Var3= 'A';
.... Var4= false;
.... Var5= "Hello World";
```

2. Buat program biodata diri dimana terdapat variabel dan tipe datanya sebagai berikut

Variabel	Tipe data
NIM	Int
Nama Mahasiswa	String
Kelas	String
Usia	Int
Tinggi badan	Double

Hasil Output yang diharapkan :

```
NIM           : 2110117000
Nama Mahasiswa : Andika Jaya Utama
Kelas        : 01SIFE020
Usia          : 19,5
Tinggi        : 1.72
```

D. REFERENSI

- Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.
- Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.
- Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.
- Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press
- Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training ,
<https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember
2021

PERTEMUAN 4

OPERATOR PADA JAVA

A. TUJUAN PEMBELAJARAN

1. Mahasiswa dapat memahami jenis jenis operator pada java.
2. Mahasiswa dapat membedakan Jenis operator pada bahasa java.
3. Mahasiswa dapat mempraktekan penggunaan Jenis operator pada bahasa program java.

B. URAIAN MATERI

Operator adalah simbol yang melakukan operasi pada variabel dan nilai. Misalnya, simbol + adalah operator yang digunakan untuk penjumlahan, sedangkan simbol * adalah operator yang digunakan untuk perkalian.

Jenis jenis operator pada java yaitu terdiri dari

1. Operator Aritmatika Dasar
2. Operator Penugasan
3. Operator Auto-increment dan Auto-decrement
4. Operator Logika
5. Operator perbandingan (relasional)
6. Operator Bitwise
7. Operator Ternary

1. Operator Aritmatika Dasar

Simbol dari Operator aritmatika dasar adalah : +, -, *, /, %

Tabel 4. 1 Simbol Operator Aritmatika

Simbol Operator Aritmatika	Fungsi
+	untuk penambahan.
-	untuk pengurangan.
*	untuk perkalian.
/	untuk pembagian.
%	Untuk Modulo

Catatan: Operator Modulo berfungsi untuk mengembalikan sisa hasil bagi, misalnya 10%5 akan mengembalikan 0.

Contoh Code java untuk Operator Aritmatika Dasar.

```
public class OperatorAritmatika{  
    public static void main(String args[]) {  
        int angka1 = 100;  
        int Angka2 = 20;  
  
        System.out.println("bilangan1 + bilangan2: " +  
            (angka1 + angka2) );  
        System.out.println("bilangan1 - bilangan2: " +  
            (angka1 - angka2) );  
        System.out.println("bilangan1 * bilangan2: " +  
            (angka1 * angka2) );  
        System.out.println("bilangan1 / bilangan2: " +  
            (angka1 / angka2) );  
        System.out.println("bilangan1 % bilangan2: " +  
            (angka1 % angka2) );  
  
    }  
}
```


Output:

```

Bilangan 1 + Bilangan2 : 120
Bilangan 1 - Bilangan2 : 80
Bilangan 1 * Bilangan2 : 2000
Bilangan 1 / Bilangan2 : 5
Bilangan 1 % Bilangan2 : 0

```

2. Operator Penugasan (Assignment Operator)

Simbol operator Penugasan pada java adalah ; =, +=, -=, *=, /=, %=

Tabel 4. 2 Simbol Operator Penugasan

Simbol Operator Penugasan	Fungsi
angka2 = angka1	akan memberikan nilai variabel angka1 ke variabel angka2
angka2+=angka1	sama dengan angka2 = angka2+angka1
angka2-=angka1	sama dengan angka2 = angka2-angka1
angka2*=angka1	sama dengan angka2 = angka2*angka1
angka2/=angka1	sama dengan angka2 = angka2/angka1
angka2%=angka1	sama dengan angka2 = angka2%angka1

Contoh Code java untuk Operator Penugasan.

```

public class OperatorPenugasan {

    public static void main(String args[]) {

```

```
int angka1 = 10;

int angka2 = 20;

angka2 = angka1;

System.out.println("= Output: "+angka2);

angka2 += angka1;

System.out.println("+= Output: "+angka2);

angka2 -= angka1;

System.out.println("-= Output: "+angka2);

angka2 *= angka1;

System.out.println("*= Output: "+angka2);

angka2 /= angka1;

System.out.println("/= Output: "+angka2);

angka2 %= angka1;

System.out.println("%= Output: "+angka2);

}

}
```

Output

```
= Output: 10
+= Output: 20
-= Output: 10
```

```
*= Output: 100
/= Output: 10
%= Output: 0
```

3. Operator auto-increment dan auto-decrement

Operator *auto increment* (++) dan operator *auto decrement* (--) adalah operasi yang menambahkan 1 ke dan mengurangi 1 ke dari variabel.

Tabel 4. 3 Operator Increment decrement

Simbol Operator	Fungsi
angka++	sama dengan angka=angka+1;
Angka--	sama dengan angka=angka-1;

Contoh Code java untuk Operator *Auto-increment* dan *Auto-decrement*

```
public class AutoOperatorDemo {
    public static void main(String args[]){
        int angka1=100;
        int angka2=200;

        angka1++;
        angka2--;

        System.out.println("angka1++ adalah : "+angka1);
        System.out.println("angka2--adalah : "+angka2);
    }
}
```

Output

```
angka1++ adalah : 101
angka2-  adalah : 199
```

4. Operator Logika (Logical Operator)

Operator Logika digunakan dengan variabel biner atau menghasilkan boolean *true* atau *false*. terutama digunakan dalam pernyataan bersyarat (*Condition*) dan loop untuk mengevaluasi suatu kondisi. Operator logika pada bahasa java adalah: `&&`, `||`, `!`

Katakanlah kita memiliki dua variabel boolean b1 dan b2.

Tabel 4. 4 Operator Logika

Operator Logika	Nilai <i>Boolean</i>
<code>b1&&b2</code>	akan mengembalikan true jika b1 dan b2 benar jika tidak maka akan mengembalikan false .
<code>b1 b2</code>	akan mengembalikan false jika b1 dan b2 salah jika tidak maka akan mengembalikan true .

Contoh Code Operator Logika

```
public class OperatorLogika{
    public static void main(String args[]) {
        boolean b1 = true;
        boolean b2 = false;

        System.out.println("b1 && b2: " + (b1&&b2));
        System.out.println("b1 || b2: " + (b1||b2));
        System.out.println("!(b1 && b2): " + !(b1&&b2));
    }
}
```

```

    }
}

```

Output :

```

b1 && b2: false
b1 || b2: true
!(b1 && b2): true

```

5. Operator Pembandingan (Relasi)

Operator Relasional atau Operator Pembandingan memeriksa hubungan satu ekspresi dengan ekspresi lain seperti lebih banyak, lebih sedikit atau sama, dan lain sebagainya. Output dari Operasi Perbandingan atau Relasional adalah nilai tipe data boolean.

Simbol yang dimiliki enam operator pembandingan relasional di Java adalah : ==, !=, >, <, >=, <=

Tabel 4. 5 Operator Pembandingan

Operator Pembandingan/Relasi	Nilai
==	mengembalikan nilai true jika ruas kiri dan ruas kanan sama
!=	mengembalikan nilai true jika sisi kiri tidak sama dengan sisi kanan operator.
>	mengembalikan nilai true jika ruas kiri lebih besar dari kanan.
<	mengembalikan nilai true jika sisi kiri lebih kecil dari sisi kanan.
>=	mengembalikan nilai true jika ruas

Operator Pembanding/Relasi	Nilai
	kiri lebih besar dari atau sama dengan ruas kanan.
<=	mengembalikan nilai true jika ruas kiri lebih kecil atau sama dengan ruas kanan.

Contoh code java Operator Pebanding Relasional

Catatan: Contoh ini menggunakan pernyataan if-else yang merupakan pembahasan pada pertemuan selanjutnya.

```
public class RelationalOperatorDemo {
    public static void main(String args[]) {
        int angka1 = 10;
        int angka2 = 50;
        if (angka1==angka2) {
            System.out.println("angka1 dan angka2 adalah
sama");
        }
        else{
            System.out.println("angka1 dan angka2 adalah
tidak sama");
        }

        if( angka1 != angka2 ){
            System.out.println("angka1 dan angka2 adalah
tidak sama");
        }
        else{
```

```
        System.out.println("angka1 dan angka2 adalah sama
");
    }

    if( angka1 > angka2 ){
        System.out.println("angka1 lebih besar dari
angka2");
    }
    else{
        System.out.println("angka1 tidak lebih besar dari
angka2");
    }

    if( angka1 >= angka2 ){
        System.out.println("angka1 lebih besar dari atau
sama untuk angka2");
    }
    else{
        System.out.println("angka1 lebih kecil dari
angka2");
    }

    if( angka1 < angka2 ){
        System.out.println("angka1 lebih kecil dari
angka2");
    }
    else{
        System.out.println("angka1 tidak lebih kecil dari
angka2");
    }
}
```

```
    }

    if( angka1 <= angka2){

        System.out.println("angka1 lebih kecil dari atau
sama untuk angka2");

    }

    else{

        System.out.println("angka1 lebih besar dari
angka2");

    }

}

}
```

Output

```
angka1 dan angka2 adalah tidak sama
angka1 dan angka2 adalah tidak sama
angka1 tidak lebih besar dari angka2
angka1 lebih kecil dari angka2
angka1 lebih kecil dari angka2
angka1 lebih kecil dari atau sama untuk angka2
```

6. Operator Bitwise

Operator bitwise digunakan untuk menangani bilangan biner dan operasi logika. Operator ini dapat diterapkan untuk tipe data integer – long, int, short, char, dan byte.

Terdapat 6 Simbol pada operator Bitwise yaitu :

Tabel 4. 6 Operator *Bitwise*

Simbol	Operator
&	Bitwise AND
^	Bitwise exclusive OR
	Bitwise inclusive OR
~	Bitwise Compliment
<<	Bit Left Shift Operators
>>	Bit Right Shift Operators

Contoh kasus

angka1 = 11; /*konversi bilangan biner 00001011*/

angka2 = 22; /* konversi bilangan biner 00010110 */

Operator bitwise melakukan pemrosesan bit demi bit.

angka1 & angka2 membandingkan bit yang sesuai dari angka1 dan angka2 dan menghasilkan 1 jika kedua bit sama, selain itu mengembalikan 0. Dalam kasus ini akan mengembalikan desimal 2 yaitu 00000010 karena dalam bentuk biner angka1 dan angka2 hanya bit terakhir kedua yang cocok .

angka1 | angka2 membandingkan bit yang sesuai dari angka1 dan angka2 dan menghasilkan 1 jika salah satu bit adalah 1, selain itu mengembalikan 0. Dalam kasus kami ini akan mengembalikan 31 yaitu **00011111**

angka1 ^ angka2 membandingkan bit yang sesuai dari angka1 dan angka2 dan menghasilkan 1 jika mereka tidak sama, selain itu mengembalikan 0. Dalam contoh ini akan mengembalikan 29 yang setara dengan **00011101**

~angka1 adalah operator pelengkap yang hanya mengubah bit dari 0 menjadi 1 dan 1 menjadi 0. Dalam contoh kita akan mengembalikan -12 8 bit setara dengan **11110100**

angka1 << 2 adalah operator shift kiri yang memindahkan bit ke kiri, membuang bit paling kiri, dan memberikan bit paling kanan nilai 0. Dalam kasus kami, outputnya adalah 44 yang setara dengan 00101100.

angka1 >> 2 adalah operator shift kanan yang memindahkan bit ke kanan, membuang bit paling kanan, dan memberikan bit paling kiri nilai 0. Dalam kasus ini, outputnya adalah 2 yang setara dengan 00000010

Contoh Code Java Operator Bitwise

```
public class OperatorBitwise{  
    public static void main(String args[]) {  
  
        int angka1 = 11; /* 11 = 00001011 */  
        int angka2 = 22; /* 22 = 00010110 */  
        int result = 0;  
  
        result = angka1 & angka2;  
        System.out.println("angka1 & angka2: "+result);  
  
        result = angka1 | angka2;  
        System.out.println("angka1 | angka2: "+result);  
  
        result = angka1 ^ angka2;  
        System.out.println("angka1 ^ angka2: "+result);  
    }  
}
```

```
        result = ~ angka1;

        System.out.println("~angka1: "+result);

        result = angka1 << 2;

        System.out.println("angka1<< 2: "+result); result =
angka1>>2;

        System.out.println("angka1 >> 2: "+result);

    }
}
```

Output :

```
angka1 | angk2: 31
angka1 ^ angka2: 29
~angka1: -12
angka1<< 2: 44
angka1 >> 2: 2
```

7. Operator Ternary

Operator ternary terdiri dari suatu kondisi yang mengevaluasi benar atau salah, operator ini mengevaluasi ekspresi boolean dan menetapkan nilai berdasarkan hasil.

Contoh sintaks :

```
variable angka1 = (expression) ? value if true : value if
false
```

Jika hasil ekspresi benar maka nilai pertama sebelum titik dua (:) ditetapkan ke variabel angka1 jika tidak, nilai kedua ditetapkan ke angka1.

Contoh Code Java Operator Ternary

```
public class OperatorTernary{

    public static void main(String args[]) {

        int angka1, angka2;

        angka1= 25;

        /* angka1 tidak sama dengan 10 itu sebabnya
        * nilai kedua setelah titik dua ditetapkan
        * ke variabel angka2
        */

        angka2= (angka1== 10) ? 100 : 200;
        System.out.println( " angka2: "+ angka2);

        /* angka1 sama dengan 25 itu sebabnya
        * nilai pertama diberikan
        * ke variabel angka2
        */

        angka2= (angka1== 25) ? 100 : 200;
        System.out.println( " angka2: "+ angka2);

    }

}
```

Output:

```
angka2: 200
```

```
angka2: 100
```

C. LATIHAN /TUGAS :

1. Buat program aritmatika untuk menghitung penambahan, pengurangan, perkalian (*) dan pembagian (/) dari 2 bilangan dimana terdapat variable bilangan :

Angka1=10, angka 2=3

Output :

- a. Hasil Pertambahan $A+B = 13$
 - b. Hasil Pengurangan $A- B = 7$
 - c. Hasil pengurangan $AXB = 30$
 - d. Hasil Pembagian $A/B = 3,333333$
2. Tampilkan Nilai w,x,y,z

```
int w, x, y, z;
```

```
x = 6; w =5 ;
```

```
y = 8 - x++;
```

```
z = 8 - ++w
```

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

PERTEMUAN 5

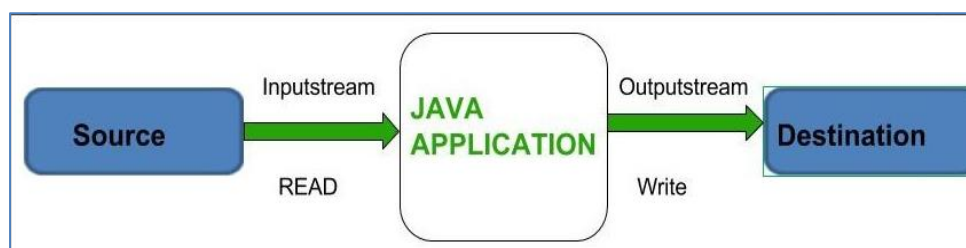
INPUT OUTPUT STREAM dan *KEYWORD* PADA JAVA

A. TUJUAN PEMBELAJARAN

1. Mahasiswa dapat memahami penggunaan input output pada java
2. Mahasiswa dapat mempraktekan menggunakan input Scanner
3. Mahasiswa dapat mempraktekan menggunakan input BufferedReader
4. Mahasiswa dapat memahami *Keyword* pada java

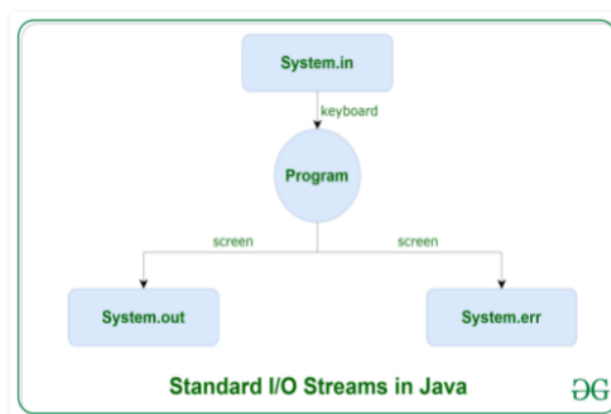
B. URAIAN MATERI

Java menghadirkan berbagai penggunaan I/O *Stream* yang membantu pengguna melakukan semua operasi input-output. I/O *Stream* ini mendukung semua jenis objek, tipe data, karakter, file dll untuk sepenuhnya menjalankan operasi I/O.



Gambar 5. 1 Input dan Output Stream

Sebelum menjelajahi berbagai aliran input dan output mari kita lihat 3 aliran standar atau default yang harus disediakan Java yang juga paling umum digunakan:



Gambar 5. 2 Standar I/O Stream

1. *System.in* : adalah aliran input standar yang digunakan untuk membaca karakter dari keyboard atau perangkat input standar lainnya.
 2. *System.out* : adalah aliran keluaran standar yang digunakan untuk menghasilkan hasil program pada perangkat keluaran seperti layar komputer.
 3. *System.Err* : adalah aliran untuk mengirimkan kesalahan hasil dari program
- print()* : Metode ini digunakan untuk menampilkan teks di konsol. Teks ini dilewatkan sebagai parameter ke metode ini dalam bentuk String. Metode ini mencetak teks di konsol dan kursor tetap berada di akhir teks di konsol.

1. Menggunakan Input *Scanner* melalui Keyboard

Class *Scanner* digunakan untuk mendapatkan masukan pengguna melalui keyboard , dan membutuhkan library *java.util*. sehingga harus di import terlebih dahulu, contoh sintaks penulisan nya :

```
import java.util.Scanner; // meng impor class Scanner
```

selanjutnya membuat objek *scanner* pada badan program perintah defaultnya adalah :

```
Scanner namaObjek = new Scanner (System.in) ;
```

Selanjutnya gunakan salah satu metode yang tersedia dalam dokumentasi kelas *Class Scanner* . Dalam contoh , kita akan menggunakan metode **nextLine()**, yang digunakan untuk membaca Strings:

Contoh Program :

```
import java.util.Scanner; // Import library class
Scanner

public class menggunakanScanner{
    public static void main(String[] args) {
        Scanner inputnya= new Scanner(System.in); // membuat
        a Scanner object
    }
}
```

```

        System.out.print("Masukan Nama Anda ? ");

        String namanya = inputnya.nextLine(); // membaca
input dari keyboard

        System.out.println("Nama Anda: " + namanya); //
Output user input

    }

}

```

Output :

Masukan Nama Anda ? Agus

Nama Anda: Agus

2. Tipe tipe Input (Masukan)

Pada contoh di atas, menggunakan metode `nextLine()`, yang digunakan untuk membaca Strings. Untuk membaca tipe lainnya, lihat pada tabel di bawah ini:

Tabel 5. 1 Tipe tipe Input Metode Pembacaan

Metode	Keterangan
<code>nextBoolean()</code>	Membaca nilai <i>boolean</i> dari pengguna
<code>nextByte()</code>	Membaca nilai <i>byte</i> dari pengguna
<code>nextDouble()</code>	Membaca nilai <i>double</i> dari pengguna
<code>nextFloat()</code>	Membaca nilai <i>float</i> dari pengguna
<code>nextInt()</code>	Membaca nilai <i>int</i> dari pengguna
<code>nextLine()</code>	Membaca nilai <i>String</i> dari pengguna

`nextLong()` Membaca nilai *Long* dari pengguna

`nextShort()` Membaca nilai *short* dari pengguna

Pada contoh di bawah ini, menggunakan metode yang berbeda untuk membaca data dari berbagai jenis:

```
import java.util.Scanner;
* @author agustav
*/
public class Scanner2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Scanner myInput = new Scanner(System.in);
        System.out.println("Masukan Nama Anda:");
        // String input
        String name = myInput.nextLine();
        // Numerical input
        System.out.println("Masukan Usia Anda:");
        int age = myInput.nextInt();
        System.out.println("Masukan Gaji Anda:");
        double salary = myInput.nextDouble();

        // Output input by user
        System.out.println(" Nama   : " + name);
        System.out.println("Usia: " + age);
        System.out.println("Gaji: " + salary);
    }
}
```

Output :

```
Masukan Nama Anda:
Agustav
Masukan Usia Anda:
25
Masukan Gaji Anda:
50000
Nama : Agustav
Usia: 25
Gaji: 50000.0
```

3. Menggunakan Input BufferedReader melalui Keyboard.

Class `BufferedReader` digunakan untuk membaca teks dari aliran input berbasis karakter. Class ini dapat digunakan untuk membaca data baris demi baris dengan metode `readLine()`.

Basic input `BufferedReader` menggunakan class `System` pada package `java.io`, package ini harus di load oleh java, dan harus di import secara manual (di tulisi) di awal program.

contoh sintaks penulisan nya :

```
import java.io.*;
```

Selanjutnya membuat objek `BufferedReader` contoh penulisannya :

```
BufferedReader namaObjek=new BufferedReader (new
InputStreamReader(System.in));
```

untuk menahan kursor/menerima input keyboard penulisan :

```
Tipedata variableInput=
[typedata].parse[typedata](namaObjek.readLine())
```

4. Metode metode yang digunakan class Buffered Reader

Tabel 5. 2 Tipe tipe Metode class Buffered Reader

Metode	Keterangan
int read()	Digunakan untuk membaca satu karakter.
int read(char[] cbuf, int off, int len)	Digunakan untuk membaca karakter ke dalam bagian array.
boolean markSupported()	Digunakan untuk menguji dukungan aliran input untuk metode mark dan reset.
String readLine()	Digunakan untuk membaca satu baris teks.
boolean ready()	Digunakan untuk menguji apakah input stream siap dibaca.
long skip(long n)	Digunakan untuk melewati karakter.
void reset()	memposisikan ulang aliran pada posisi metode tanda dipanggil terakhir kali pada aliran input .
void mark(int readAheadLimit)	Digunakan untuk menandai posisi sekarang dalam aliran.
void close()	Menutup aliran input dan melepaskan semua sumber daya sistem yang terkait dengan aliran.

Contoh code java menggunakan Input *BufferedReader*

```
import java.io.*;

/**
 *
 * @author feiruz
 */
```

```
public class BufferedReader1 {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String args[]) throws Exception {  
        InputStreamReader r=new InputStreamReader(System.in);  
        BufferedReader br=new BufferedReader(r);  
        System.out.println("Masukan Nama Anda");  
        String name=br.readLine();  
        System.out.println("Selamat Datang "+name);  
    }  
}
```

Output :

```
Masukan Nama Anda :  
Agustav  
  
    Selamat Datang Agustav
```

5. Sistem.err

System.err adalah PrintStream. System.err bekerja seperti System.out, biasanya hanya digunakan untuk menampilkan teks kesalahan. Beberapa program akan menampilkan output ke System.err dalam teks merah, untuk membuatnya lebih jelas bahwa itu adalah kesalahan teks.

Contoh *Code* penggunaan System.err.

```
try {
```

```
InputStream input = new
FileInputStream("c:\\data\\...");

System.out.println("File opened...");

} catch (IOException e){

System.err.println("File opening failed:");

e.printStackTrace();

}
```

6. Keyword pada Java

Keyword / Kata kunci pada Java juga dikenal sebagai *reserve word* adalah kata-kata atau perintah tertentu yang bertindak sebagai kunci dari sebuah kode. Kata kunci ini telah ditentukan sebelumnya pada bahasa Java sehingga tidak dapat digunakan sebagai variabel atau nama objek ataupun nama kelas.

Berikut daftar Kata kunci/Keyword pada java

a. ***abstract***

Kata kunci *abstract* Java digunakan untuk mendeklarasikan kelas abstrak. Kelas abstrak yang dapat menyediakan implementasi antarmuka. memiliki metode abstrak dan non-abstrak.

b. ***boolean***

Kata kunci *boolean* Java digunakan untuk mendeklarasikan variabel sebagai tipe data boolean. Tipe data ini hanya dapat menampung nilai True atau False.

c. ***break***

Kata kunci *break* digunakan untuk memutus pernyataan loop atau switch pada aliran program yang berlangsung di kondisi tertentu.

d. ***byte***

Kata kunci *byte* Java digunakan untuk mendeklarasikan variabel yang dapat menampung nilai data 8-bit.

e. case

Kata kunci *Case* Java digunakan dengan pernyataan *Switch* untuk menandai blok teks.

f. catch

Kata kunci *catch* Java digunakan untuk menangkap pengecualian yang dihasilkan oleh pernyataan *try*

g. char

Kata kunci *char* digunakan untuk mendeklarasikan variabel yang dapat menampung karakter Unicode 16-bit

h. class

Kata kunci *Class* Java digunakan untuk mendeklarasikan sebuah *Class*.

i. continue

Kata kunci Java *continue* digunakan untuk melanjutkan perulangan. melanjutkan aliran program yang berlangsung dan melewati kode yang tersisa pada kondisi yang ditentukan.

j. default

Kata kunci *default* Java digunakan untuk menentukan blok kode default dalam pernyataan switch.

k. do

Kata kunci Java *do* digunakan dalam pernyataan kontrol untuk mendeklarasikan *loop*. *do* dapat mengulangi bagian dari program beberapa kali.

l. double

Kata kunci *double* Java digunakan untuk mendeklarasikan variabel yang dapat menampung angka floating-point 64-bit.

m. else

Kata kunci Java *else* digunakan untuk menunjukkan cabang alternatif dalam pernyataan *if*.

n. Enum

Kata kunci *enum* Java digunakan untuk mendefinisikan satu set fix konstanta.

o. extend

Kata kunci *extend* Java digunakan untuk menunjukkan bahwa suatu kelas diturunkan dari kelas atau antarmuka lain.

p. *final*

Kata kunci *final* Java digunakan untuk menunjukkan bahwa suatu variabel memiliki nilai konstan. Ini digunakan untuk membatasi pengguna memperbarui nilai variabel.

q. *finally*

Kata kunci *finally* Java menunjukkan blok kode dalam struktur try-catch. Blok ini selalu dieksekusi

r. *float*

Kata kunci *float* Java digunakan untuk mendeklarasikan variabel yang dapat menampung angka floating-point 32-bit.

s. *for*

kata kunci *for* untuk mengeksekusi serangkaian instruksi/fungsi berulang kali ketika beberapa kondisi menjadi true. Jika jumlah iterasi tetap, direkomendasikan untuk menggunakan perulangan for.

t. *if*

kata kunci *if* untuk menguji kondisi. If mengeksekusi blok jika kondisinya benar.

u. *implement*

Kata kunci *implement* Java digunakan untuk mengimplementasikan antarmuka.

v. *Impor*

Kata kunci *import* Java membuat kelas dan antarmuka dan dapat diakses oleh kode sumber yang berlangsung .

w. *instanceof*

Kata kunci *instanceof* Java digunakan untuk menguji apakah objek adalah turunan dari kelas yang ditentukan atau mengimplementasikan antarmuka.

x. *int*

Kata kunci *int* Java digunakan untuk mendeklarasikan variabel yang dapat menampung bilangan bulat bertanda 32-bit.

y. *Interface*

Kata kunci *interface* Java digunakan untuk mendeklarasikan interface yang hanya dapat memiliki metode abstrak.

z. *long*

Kata kunci *long* digunakan untuk mendeklarasikan variabel yang dapat menampung bilangan bulat 64-bit.

aa. *Native*

Kata kunci *native* Java digunakan untuk menentukan bahwa suatu metode diimplementasikan dalam kode asli menggunakan JNI (Java Native Interface).

bb. *new*

Kata kunci *new* Java digunakan untuk membuat objek baru.

cc. *null*

Kata kunci Java *null* digunakan untuk menunjukkan bahwa referensi tidak merujuk ke apa pun.

dd. *package*

Kata kunci *package* Java digunakan untuk mendeklarasikan package Java yang menyertakan kelas.

ee. *private*

Kata kunci *private* Java adalah kata kunci pengubah akses. digunakan untuk menunjukkan bahwa metode atau variabel hanya dapat diakses di kelas di mana yang dideklarasikan.

ff. *protected*

Kata kunci *protected* Java adalah pengubah akses. dapat diakses di dalam package dan di luar package tetapi hanya melalui pewarisan

gg. *publik*

Kata kunci publik Java adalah pengubah akses juga. Ini digunakan untuk menunjukkan bahwa suatu item dapat diakses di mana saja dan memiliki cakupan terluas di antara semua pengubah lainnya.

hh. *return*

Kata kunci *return* Java digunakan untuk mengembalikan suatu metode ketika eksekusinya selesai.

ii. *short*

Kata kunci *short* Java digunakan untuk mendeklarasikan variabel yang dapat menampung tipe bilangan bulat integer 16-bit.

jj. *statics*

Kata kunci *statics* Java digunakan untuk menunjukkan bahwa variabel atau metode adalah suatu metode kelas. Kata kunci *static* di Java terutama digunakan untuk manajemen memori.

kk. Super

Kata kunci *super* Java adalah variabel referensi yang digunakan untuk merujuk ke objek kelas induk. dapat digunakan untuk memanggil metode kelas induk langsung.

ll. switch

Kata kunci Java *switch* berisi pernyataan switch yang mengeksekusi kode berdasarkan nilai pengujian. Pernyataan switch menguji kesetaraan variabel terhadap beberapa nilai.

mm. synchronized

Kata kunci *synchronized* Java digunakan untuk menentukan bagian atau metode penting dalam kode multithread.

nn. This

kata kunci *this* dapat digunakan untuk merujuk objek saat ini dalam metode atau konstruktor.

oo. throw

Kata kunci *throw* Java digunakan untuk melempar pengecualian secara eksplisit. Kata kunci *throw* terutama digunakan untuk melempar pengecualian khusus. Hal ini diikuti oleh sebuah contoh.

pp. throws

Kata kunci Java *throws* digunakan untuk mendeklarasikan eksepsi. Pengecualian yang diperiksa dapat disebarkan dengan lemparan.

qq. transient

Kata kunci *transient* Java digunakan dalam serialisasi.

rr. try

Kata kunci Java *try* digunakan untuk memulai blok kode yang akan diuji untuk pengecualian. Blok *try* harus diikuti oleh blok *catch* atau *finally*.

ss. void

Kata kunci Java *void* digunakan untuk menentukan bahwa suatu metode tidak memiliki nilai kembalian.

tt. volatile

Kata kunci volatil Java digunakan untuk menunjukkan bahwa suatu variabel dapat berubah secara tidak sinkron.

uu. while

kata kunci Java *while* digunakan untuk memulai perulangan *while*. Loop ini mengulangi bagian dari program beberapa kali. Jika jumlah iterasi tidak tetap, disarankan untuk menggunakan perulangan *while*.

C. LATIHAN DAN TUGAS

1. Sebuah perusahaan ingin Mendata semua karyawan dimana hasil outputnya seperti gambar dibawah ini :

Output pada program Pendataan Karyawan adalah sebagai berikut :

```
-----
Nama Karyawan: iyan
Alamat: Ampenan
Usia: 20 tahun
Gaji: Rp 10000000
BUILD SUCCESSFUL (total time: 40 seconds)
|
```

Input pada program Pendataan Karyawan adalah sebagai berikut :

```
### Pendataan Karyawan PT. Petani Kode ###
Nama karyawan:
Alamat:
Usia:
Gaji:
-----
```

Buat Program sederhana nya dengan menggunakan Input Scanner

2. Ubah class LuasSegitiga dengan nilai alas dan tinggi dimasukkan oleh user

```
public class LuasSegitiga {
    public static void main(String[] args) {
        double alas= 17; double tinggi = 11;
        double luas = (alas*tinggi)/2;
        System.out.println("Luas Segitiga : " + luas);
    }
}
```

D. REFERENSI

- Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.
- Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.
- Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.
- Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press
- Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.
- Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

PERTEMUAN 6

MENGONTROL ALIRAN PROGRAM (IF,IF.. ELSE)

A. TUJUAN PEMBELAJARAN

1. Mahasiswa dapat memahami penggunaan Kondisi aliran program IF
2. Mahasiswa dapat memahami penggunaan Kondisi aliran program IF ..ELSE
3. Mahasiswa dapat mempraktekan menggunakan IF
4. Mahasiswa dapat mempraktekan menggunakan IF..ELSE

B. URAIAN MATERI

Ketika kita perlu mengeksekusi satu set pernyataan berdasarkan suatu kondisi maka kita perlu menggunakan pernyataan dengan mengontrol aliran program. Misalnya jika suatu bilangan lebih besar dari nol maka kita ingin mencetak "Bilangan Positif" tetapi jika lebih kecil dari nol maka kita ingin mencetak "Bilangan Negatif". Dalam hal ini kami memiliki dua pernyataan untuk mencetak dalam program, tetapi hanya satu pernyataan cetak yang dieksekusi pada satu waktu berdasarkan nilai input. Kita akan melihat bagaimana menulis kondisi seperti itu dalam program java menggunakan pernyataan kontrol.

Dalam pertemuan ini, kita akan melihat empat jenis pernyataan kontrol yang dapat digunakan dalam program java berdasarkan kebutuhan: pernyataan bersyarat sebagai berikut:

1. if statement
2. nested if statement
3. if-else statement
4. if-else-if statement

1. Pernyataan IF (IF Statement)

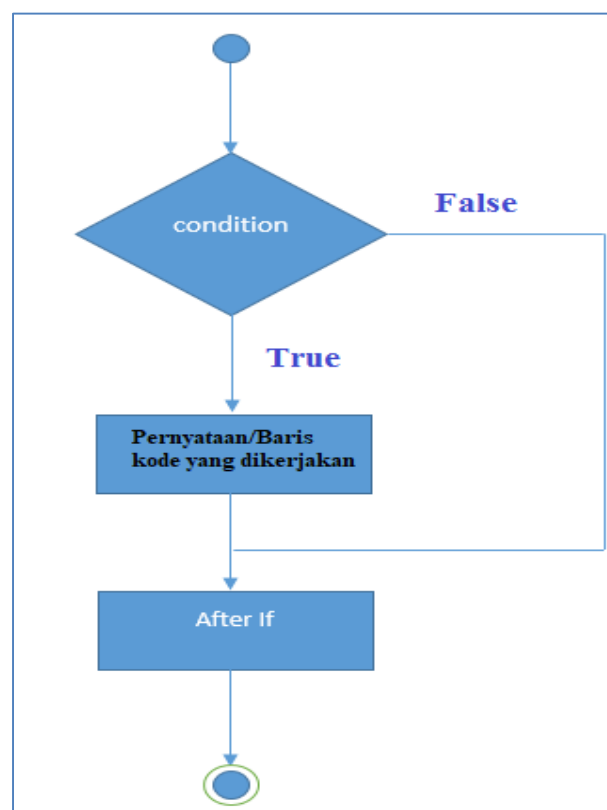
Pernyataan IF adalah pernyataan pengambilan keputusan yang paling sederhana. Pernyataan ini digunakan untuk memutuskan apakah pernyataan

atau blok pernyataan tertentu akan dieksekusi atau tidak, jika kondisi tertentu **True** maka blok pernyataan akan dieksekusi jika **False** , maka diabaikan.

Bentuk Umum penulisan pernyataan IF :

```
if(condition) {  
    Statement(s); //Block True  
}
```

Secara alur diagram flowchart gambarnya seperti dibawah ini :



Gambar 6. 1 diagram flow chart pernyataan IF

Contoh program sederhana : pernyataan IF :

```
// Program Java penggunaan if statement.  
  
public class ContohIF1 {  
  
    public static void main(String[] args) {  
  
        //deklarasi variabel 'usia'    }  
}
```

```
int usia=20;

//cek kondisi pernyataan If

if(usia>18){

    System.out.print("Usia anda lebih besar dari 18
Tahun");

}

}

}
```

Output :

```
Usia anda lebih besar dari 18 Tahun
```

2. Pernyataan IF Bersarang (Nested IF Statement)

Pernyataan if bersarang adalah pernyataan if di dalam pernyataan if yang lain.

Bentuk Umum penulisan nya adalah sebagai berikut :

```
if(condition_1) {

    Statement1(s);

    if(condition_2) {

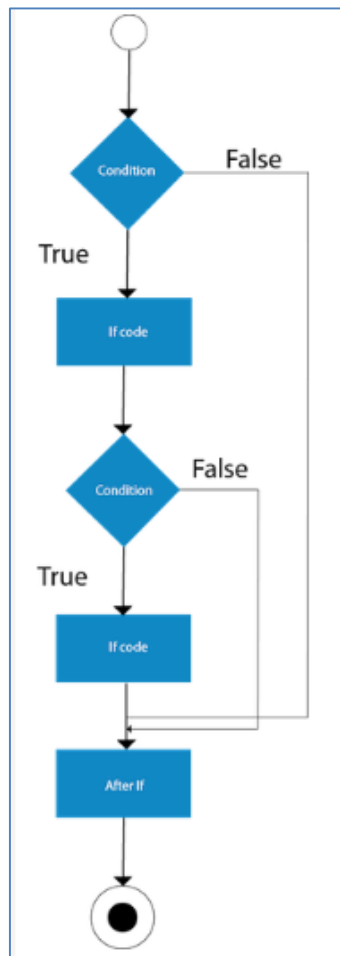
        Statement2(s);

    }

}
```

Pernyataan1 akan dijalankan jika kondisi_1 benar. Pernyataan2 hanya akan dijalankan jika kedua kondisi (kondisi_1 dan kondisi_2) benar.

Secara alur diagram flowchart gambarnya seerti dibawah ini :



Gambar 6. 2 Flow Chart Nested If

Contoh code sederhana penggunaan Nested IF

```
public class nestedIF {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        int angka=70;  
        if( angka < 100 ){
```

```
        System.out.println("angka adalah lebih kecil
100");

        if(angka > 50){

            System.out.println("angka adalah lebih besar
dari 50");

        }

    }

}
```

Output :

```
angka adalah lebih kecil 100
angka adalah lebih besar dari 50
```

3. Pernyataan IF ..ELSE

Pada Pernyataan if sebelumnya jika suatu kondisi **true**, itu akan mengeksekusi blok pernyataan dan jika kondisinya salah, itu tidak akan dilakukan atau diabaikan. Tetapi bagaimana jika kita ingin melakukan sesuatu jika kondisinya **false**. Kita dapat menggunakan pernyataan *else* dengan pernyataan if untuk mengeksekusi blok kode ketika kondisinya **false**

Bentuk Umum pernyataan **IF..Else**

```
. if(condition) {

    Statement(s);

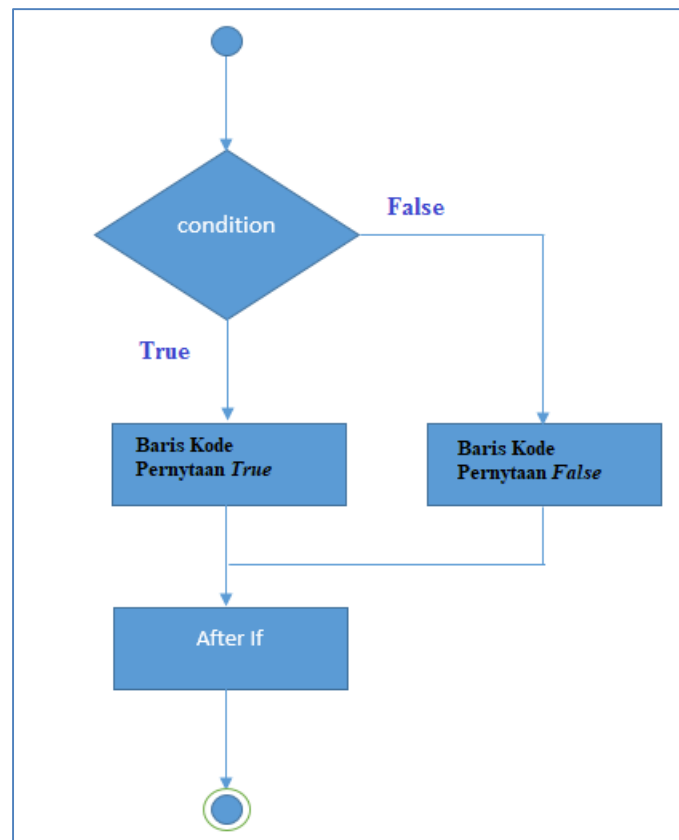
}

else {

    Statement(s);

}
```


Secara alur diagram flowchart gambarnya seperti dibawah ini :



Gambar 6. 3 Flowchart diagram if-else

```
public class IFElseStatement {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        int angka=120;  
        if( angka < 50 ){  
            System.out.println("angka adalah lebih kecil dari  
50");  
        }  
    }  
}
```

```
        else {  
            System.out.println("angka adalah lebih besar atau  
sama dengan 50");  
        }  
    }  
}
```

Output

```
angka adalah lebih besar atau sama dengan 50
```

4. Pernyataan IF ..ELSE ..IF

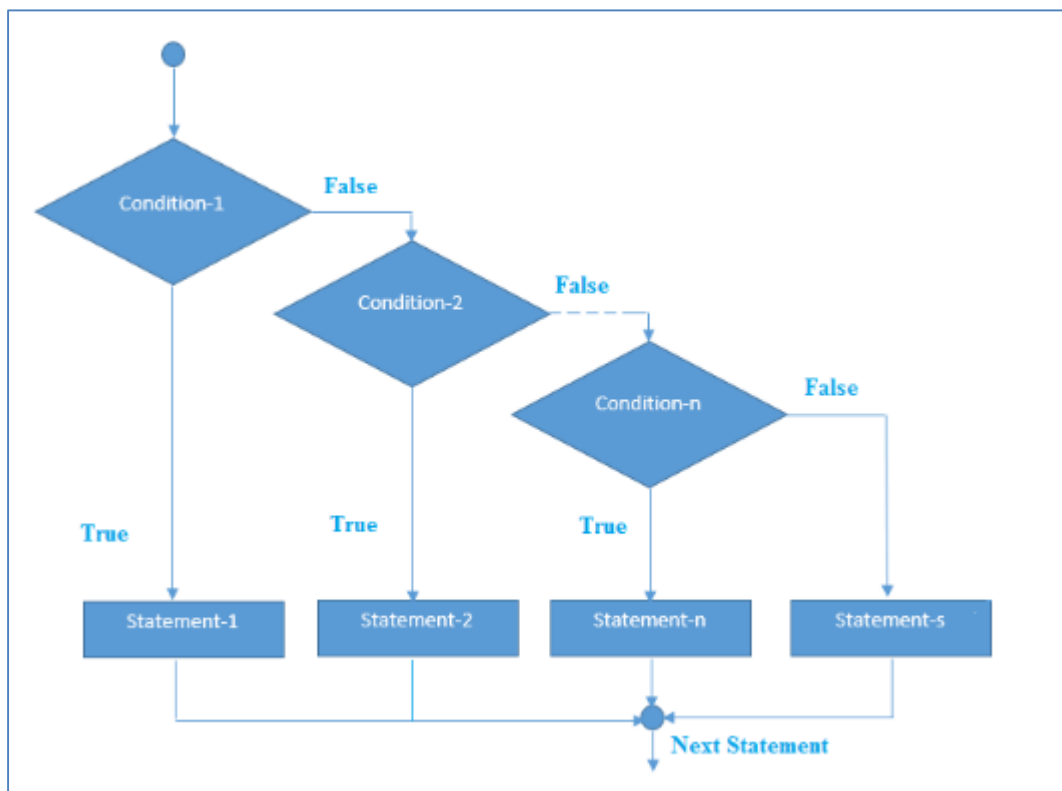
Pernyataan if-else-if digunakan untuk memutuskan di antara beberapa opsi. Pernyataan if dijalankan dari atas ke bawah. Setelah salah satu kondisi yang mengendalikan if **True**, pernyataan tersebut akan dieksekusi. Jika tidak ada kondisi yang benar, maka pernyataan else terakhir akan dieksekusi.

Bentuk Umum Penulisan IF..ELSE..IF

```
if(condition_1) {  
    /*if condition_1 is true execute this*/  
    statement(s);  
}  
  
else if(condition_2) {  
    /* execute this if condition_1 is not met and  
    * condition_2 is met  
    */  
    statement(s);  
}  
  
else if(condition_3) {  
    /* execute this if condition_1 & condition_2 are  
    * not met and condition_3 is met
```

```
*/  
  
statement(s);  
}  
  
.  
.  
.  
  
else {  
  
    /* if none of the condition is true  
    * then these statements gets executed  
    */  
  
    statement(s);  
}
```

Secara alur diagram flowchart gambarnya seperti dibawah ini :



Gambar 6. 4 Flow diagram Pernyataan IF..ELSE..IF

Program ini akan menentukan Grade nilai Akhir mahasiswa dimana ketentuannya adalah sebagai berikut :

- a. Jika Nilai <50 , Grade = Tidak lulus
- b. Jika Nilai >=50 dan Nilai <60 , Grade = D
- c. Jika Nilai >=60 dan nilai <70 , Grade = C
- d. Jika Nilai >=70 dan Nilai <80 , Grade = B
- e. Jika Nilai >=80 dan <90 , Grade = A
- f. Jika Nilai >=90 dan <100 , Grade = A+
- g. Selain itu data nilai Salah

```
public class IfElseIF {  
  
    /**  
     * @param args the command line arguments  
     */  
  
    public static void main(String[] args) {  
        // TODO code application logic here  
  
        int Nilai=65;  
  
        if(Nilai<50){  
            System.out.println("Tidak Lulus ");  
        }  
  
        else if(Nilai>=50 && Nilai<60){  
            System.out.println("grade D ");  
        }  
  
        else if(Nilai>=60 && Nilai<70){  
            System.out.println("grade C ");  
        }  
  
        else if(Nilai>=70 && Nilai<80){  
            System.out.println("grade B ");  
        }  
  
        else if(Nilai>=80 && Nilai<90){
```

```
        System.out.println("grade A ");
    }else if(Nilai>=90 && Nilai<100){
        System.out.println("grade A+");
    }else{
        System.out.println("Data Nilai Salah !");
    }
}
}
```

Output :

grade C

C. LATIHAN

1. Modifikasi program untuk menentukan Grade nilai diatas dimana untuk Input Nilainya menggunakan class Scanner :

Contoh : Output yang ditampilkan :

Contoh Input Scanner 1

```
NIM ? : 1111
NAMA ? : Agustav
NILAI ? : 85
```

Output :

grade A

contoh Input Scanner 2

```
NIM ? : 1112
NAMA ? : Indah
NILAI ? : 75
```

Output :

```
grade B
```

contoh Input Scanner 3

```
NIM ? : 1113
NAMA ? : Rendi
NILAI ? : 65
```

Output :

```
grade C
```

2. Modifikasi program selanjutnya untuk menentukan Keterangan diatas dimana untuk Input Nilainya menggunakan class Scanner
Input : NIM, NAMA, NILAI
Output : Grade, Keterangan

Contoh Input Scanner 1

```
NIM ? : 1111
NAMA ? : Agustav
NILAI ? : 85
```

Contoh output :

Output :

```
grade A
Keterangan : Lulus
```

contoh Input Scanner 2

```
NIM ? : 1112
NAMA ? : Indah
NILAI ? : 75
```

Output :

```
grade B
Keterangan : Lulus
```

D. REFERENSI

- Horstmann Cay S., (2011). *Big Java 4th Edition*, san jose university , united state Of America. RRD jefferson city publishing.
- Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.
- Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.
- Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press
- Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.
- Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

PERTEMUAN 7

MENGONTROL ALIRAN PROGRAM PERNYATAAN SWITCH ..CASE

A. TUJUAN PEMBELAJARAN

1. Mahasiswa dapat memahami penggunaan Kondisi aliran SWITCH..CASE
2. Mahasiswa dapat mempraktekan penggunaan SWITCH..CASE
3. Mahasiswa dapat memilih penggunaan opsi struktur kendali /*control flow*

B. URAIAN MATERI

Pernyataan **switch** mengeksekusi satu pernyataan dari beberapa kondisi. Pernyataan ini seperti pernyataan **if-else-if**. Pernyataan switch bekerja dengan tipe byte, short, int, long, enum, String dan beberapa tipe wrapper seperti Byte, Short, Int, dan Long. Kita juga dapat menggunakan string dalam pernyataan switch.

Dengan kata lain, pernyataan switch menguji kesetaraan variabel terhadap beberapa nilai.

1. Aturan Penulisan Pernyataan Switch

Pernyataan switch memiliki aturan sebagai berikut :

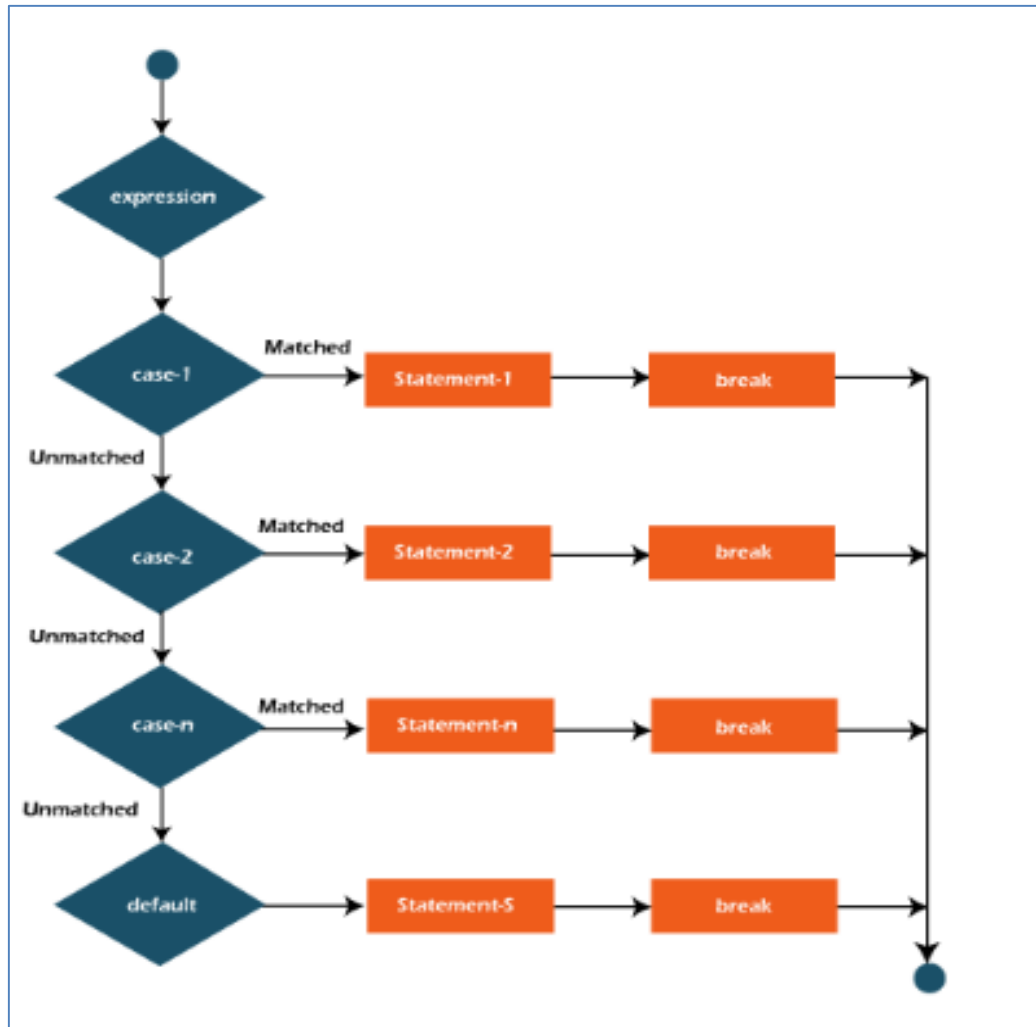
- a. Nilai switch dihasilkan dari tipe char, byte, short, int, atau String dan diapit menggunakan tanda kurung.
- b. Nilai x mempunyai tipe data yang dimilikinya sama seperti nilai dari ekspresi switch. Nilai x adalah ekspresi konstan, artinya tidak dapat mengandung variabel yang berubah.
- c. Jika nilai pada pernyataan **case** sama sesuai dengan nilai dari ekspresi switch, pernyataan case ini akan dieksekusi sampai ditemukannya pernyataan **break** atau akhir dari pernyataan switch terpenuhi.
- d. Pernyataan **default**, sifatnya opsional, jika tidak ada satupun case yang sudah ditetapkan sesuai dengan ekspresi switch.
- e. Pernyataan break juga opsional. Pernyataan ketika pernyataan switch akan diakhiri.

2. Bentuk Umum penulisan pernyataan Switch

```
switch(expression){  
    case value1:  
        //code to be executed;  
        break; //optional  
    case value2:  
        //code to be executed;  
        break; //optional  
    .....  
    default:  
        code to be executed if all cases are not matched;  
}
```

Pertama, variabel, nilai, atau ekspresi yang disediakan dalam tanda kurung *switch* dievaluasi dan kemudian berdasarkan hasilnya, blok case yang sesuai dieksekusi yang cocok dengan hasilnya jika blok case selesai di eksekusi keluar dari case tersebut dengan *keyword break*, dilanjutkan ke case berikutnya:

Gambaran diagram Flowchart pernyataan Switch seperti dibawah ini :



Gambar 7.1 1 diagram flowchart Switch

3. Contoh 1 Program dengan Pernyataan switch

```
public class Switcase {  
  
    /**  
     * @param args the command line arguments  
     */  
  
    public static void main(String[] args) {  
        // TODO code application logic here  
        int Menu=2;
```

```
//Switch expression

System.out.println("Pilihan Anda = ");

switch(Menu){

//Case statements

case 1: System.out.println(" Ayam Geprek");
break;

case 2 : System.out.println("Ayam Penyet");
break;

case 3: System.out.println("Ayam Bakar");
break;

//Default case statement

default: System.out.println("Aneka Minuman");

}

}
```

Output :

```
Pilihan Anda =

Ayam Penyet
```

Contoh 2 pernyataan Switch

```
// Program java mendemonstrasikan contoh dari
pernyataan Switch

//where we are printing month name for the given
number

public class Switchbulan{

public static void main(String[] args) {
```

```
//Specifying month number
int bulan=7;
String bulanHuruf="";
//Switch statement
switch(bulan){
//case statements within the switch block
case 1: bulanHuruf ="1 - Januari";
break;
case 2: bulanHuruf ="2 - Februari";
break;
case 3: bulanHuruf ="3 - Maret";
break;
case 4: bulanHuruf="4 - April";
break;
case 5: bulanHuruf ="5 - Mei";
break;
case 6: bulanHuruf ="6 - Juni";
break;
case 7: bulanHuruf ="7 - Juli";
break;
case 8: bulanHuruf ="8 - Agustus";
break;
case 9: bulanHuruf ="9 - September";
break;
case 10: bulanHuruf ="10 - Oktober";
break;
case 11: bulanHuruf ="11 - November";
break;
case 12: bulanHuruf ="12 - Desember";
break;
default: System.out.println("Pilihan bulan
salah!");
}
```

```
//Tampilkan bulan yang diberikan angka pada variabel bulan
    System.out.println(bulanHuruf);
}
}
```

Output :

```
7 - Juli
```

4. Beberapa poin Pernyataan Switch Case

- Case tidak selalu harus memiliki urutan 1, 2, 3 dan seterusnya. Setelah kata kunci case. Tidak harus selalu dalam urutan menaik, kita dapat menentukannya dalam urutan apa pun berdasarkan persyaratan.
- Kita juga dapat menggunakan karakter dalam Pernyataan Switch.

Contoh Switch dengan karakter:

```
public class SwitchString {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        char ch='b';
        switch(ch)
        {
            case 'd':
                System.out.println("Case1 ");
                break;
            case 'b':
                System.out.println("Case2 ");
                break;
            case 'x':
                System.out.println("Case3 ");
        }
    }
}
```

```
        break;
    case 'y':
        System.out.println("Case4 ");
        break;
    default:
        System.out.println("Default ");
    }
}
```

Output :

Case2

5. Pernyataan Switch Case dengan multi value

Jika kita ingin memeriksa beberapa nilai untuk satu pernyataan? Berikut adalah contoh beberapa nilai Switch Case pada Java.

Bentuk Umum penulisannya :

```
case text1: case text4:
    do stuff;
    break;
```

Contoh code Switch case multivalue :

```
public class Swtichmultivalue {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        //Declaring a variable
        int pilihan = 2;
        //Switch expression
```

```
switch (pilihan) {  
    //Case statements  
    case 1:  
    case 2:  
    case 3:  
    case 4:  
    case 5:  
        System.out.println("Hari Kerja");  
        break;  
    case 6:  
    case 7:  
        System.out.println("HAri Libur ");  
        break;  
    //Default case statement  
    default:  
        System.out.println("Silahkan pilih  
1,2,3,4,5,6,7");  
}  
}
```

6. Pernyataan Switch Bersarang (Nested Switch)

Kita dapat menggunakan pernyataan switch di dalam pernyataan switch lain di Java. atau dikenal dengan ***nested switch***

Contoh *code Nested Switch*

```
public class NestedSwitch {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        char peminatan = 'C';
```

```
int tahun = 4;
switch( tahun )
{
    case 1:
        System.out.println("Bahasa Inggris,
Matematika, Sain");
        break;
    case 2:
        switch( peminatan )
        {
            case 'C':
                System.out.println("Sistem
Operasi, Pemrograman Java, Struktur Data");
                break;
            case 'E':
                System.out.println("Algoritma,
Logika Informatika");
                break;
            case 'M':
                System.out.println("DBMS, Manajemen
Proyek ");
                break;
        }
        break;
    case 3:
        switch( peminatan )
        {
            case 'C':
                System.out.println("Organisasi
Komputer, MultiMedia");
                break;
            case 'E':
                System.out.println("Perancangan
Sistem, Pemrograman WEB");
                break;
        }
}
```



```
        case 'M':
            System.out.println("Pemrograman
Mobile, Pemrograman Java 2");
            break;
        }
        break;
    case 4:
        switch( peminatan )
        {
            case 'C':
                System.out.println("Komunikasi
Data, MultiMedia");
                break;
            case 'E':
                System.out.println(" Sistem
Terdistribusi , Image Processing");
                break;
            case 'M':
                System.out.println("SIM , Sistem
Jaringan  ");
                break;
        }
        break;
    }
}
```

Output :

```
Komunikasi Data, MultiMedia
```

C. LATIHAN/TUGAS

1. Sisipkan dan lengkapi bagian potongan program switch dibawah ini

```
int day = 2;
switch (  ) {
     1:
        System.out.println("Sabtu ");
        break;
     2:
        System.out.println("Minggu");
     ;
}
```

2. Kembangkan contoh program nested switch diatas dengan input scanner :

Contoh input :

```
NIM    : 1111111
NAMA   : Agustav

TAHUN  : 3
PILIH PEMINATAN : M
```

Output :

```
PEMROGRAMAN MOBILE, PEMROGRAMAN JAVA 2
```

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

PERTEMUAN 8

MENGONTROL ALIRAN PROGRAM PERULANGAN FOR (for loop)

A. TUJUAN PEMBELAJARAN

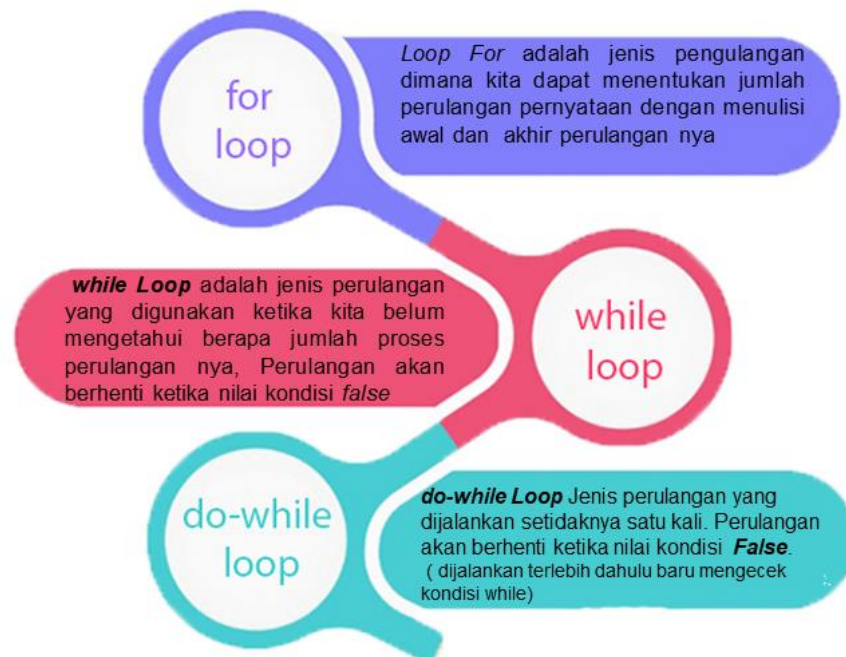
1. Mahasiswa dapat memahami Konsep Kontrol pengulangan / Looping
2. Mahasiswa dapat mengerti aturan kode Kontrol pengulangan For
3. Mahasiswa dapat mempraktekan penggunaan Kontrol pengulangan For

B. URAIAN MATERI

1. Perulangan For

Adalah jenis perulangan dimana kita dapat menentukan jumlah perulangan pernyataan dengan menulisi kondisi nilai awal, nilai akhir perulangan nya, serta kenaikan dan penurunannya.

Perulangan for adalah bagian perintah perulangan dari 3 jenis perulangan seperti pada gambar dibawah ini (pada pertemuan selanjutnya kita akan membahas perulangan while, do while)



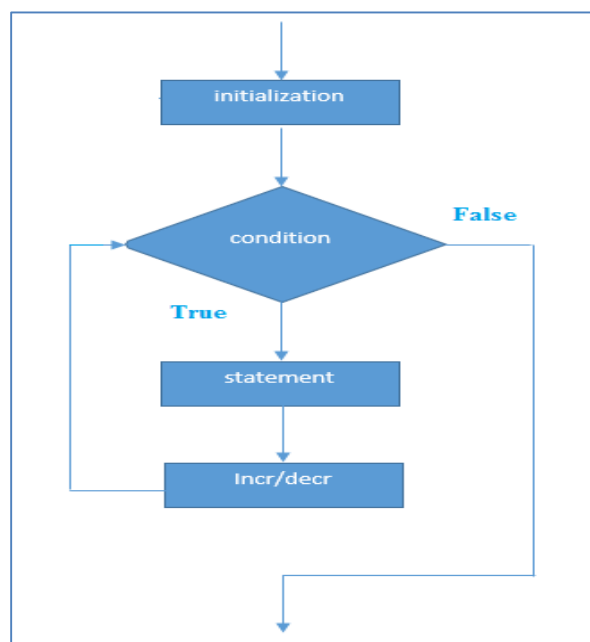
Gambar 8. 1 Jenis Perulangan

2. Bentuk Umum Penulisan For

```
for(initialization; condition ; increment/decrement)
{
    statement(s) ;
}
```

- Initialitation** : adalah kondisi awal atau nilai awal yang dieksekusi sekali ketika loop/perulangan dimulai. kita dapat menginisialisasi variabel, atau kita dapat menggunakan variabel yang sudah diinisialisasi(opsional).
- Condition** : adalah kondisi kedua yang dijalankan setiap kali untuk menguji kondisi loop. Ini melanjutkan eksekusi sampai kondisinya **false** atau batasan nilai akhir yang ditentukan.
- Increment /Decrement**: adalah menambah atau mengurangi nilai variabel yang ditentukan pada kondisi awal.
- Statement** : adalah pernyataan loop dieksekusi beberapa kali sampai kondisi berikutnya **False**

Berikut diagram flowchart dari perulangan **for**



Gambar 8. 2 diagram Flowchart Perulangan For

Contoh code sederhana perulangan for

```
public class For1 {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
  
        for(int i=1; i<=10; i++){  
            System.out.println("Nilai Variabel i adalah  
: "+i);  
        }  
    }  
}
```

Output :

```
Nilai Variabel i adalah : 1  
Nilai Variabel i adalah : 2  
Nilai Variabel i adalah : 3  
Nilai Variabel i adalah : 4  
Nilai Variabel i adalah : 5  
Nilai Variabel i adalah : 6  
Nilai Variabel i adalah : 7  
Nilai Variabel i adalah : 8  
Nilai Variabel i adalah : 9  
Nilai Variabel i adalah : 10
```

3. Perulangan For bersarang (Nested For)

Perulangan For bersarang /*nested for* adalah perulangan for terdapat perulangan for lainnya, ini merupakan pengembangan dari perulangan For. Pada perulangan For bersarang ini yang terlebih dahulu dieksekusi yaitu For yang ada di dalam terdahulu.

Bentuk Umum penulisan :

```
for(statement; condition check; increment/decrement) {  
    // Code inside outer loop  
    for(statement; condition check; increment/decrement) {  
        // Code inside inner loop  
    }  
    // Code inside outer loop }
```

Contoh program nested for

```
public class nestedFor {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        //loop of i  
        for(int i=1;i<=4;i++){  
            //loop of j  
            for(int j=1;j<=3;j++){  
                System.out.println(i+" "+j);  
            }  
        }  
    }  
}
```

Output :

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

```
2  2
2  3
3  1
3  2
3  3
4  1
4  2
4  3
```

Contoh 2 program membuat piramid nested for

```
public class piramidnested {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        for(int i=1;i<=5;i++){
            for(int j=1;j<=i;j++){
                System.out.print("* ");
            }

            System.out.println();//baris baru
        }
    }
}
```

Output :

```
*
* *
* * *
* * * *
* * * * *
```


4. Perulangan For each

Perulangan for-each digunakan untuk melintasi array atau koleksi di Java. Lebih mudah digunakan daripada loop for sederhana karena kita tidak perlu menambah nilai dan menggunakan notasi subscript.

For each bekerja atas dasar elemen dan bukan indeks. Ini akan mengembalikan elemen satu per satu dalam variabel yang ditentukan.

Bentuk Umum penulisan For each :

```
for(data_type variable : array_name){  
    //code to be executed  
}
```

Catatan : keyword *array* akan dibahas pada pertemuan selanjutnya

Contoh program perulangan *for each*.

```
public class foreachDemo {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        int arr[]={12,23,44,56,78};  
        //Printing array using for-each loop  
        for(int i:arr){  
            System.out.println(i);  
        }  
    }  
}
```

Output :

```
12
23
44
56
78
```

5. Label For Loop

Kita dapat memiliki nama masing-masing pada Java for loop dengan pelabelan. Contoh menggunakan label sebelum perulangan for. Ini berguna saat menggunakan loop for bersarang karena kita dapat memutus atau melanjutkan loop for tertentu.

Bentuk umum penulisan pelabelan ***For***

```
labelname:
for(initialization; condition; increment/decrement){
//code to be executed
}
```

Contoh program label *for*

```
public class labeledFor {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        aa:
        for(int i=1;i<=3;i++){
            bb:
            for(int j=1;j<=3;j++){
```

```
        if (i==2&& j==2) {  
            break aa;  
        }  
        System.out.println(i+" "+j);  
    }  
}  
}
```

Output :

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

Contoh lain program pelabelan for

```
public class labeledfor2 {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        aa:  
        for(int i=1;i<=3;i++){  
            bb:  
            for(int j=1;j<=3;j++){  
                if (i==2&& j==2) {  
                    break bb;  
                }  
            }  
        }  
    }  
}
```

```
                System.out.println(i+" "+j);  
            }  
        }  
    }  
}
```

Output :

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

6. Infinite For Loop

infinite loop adalah perulangan terus menerus atau kondisinya tidak akan pernah mengembalikan *false*. Contoh inisialisasi awal nilai variabel *i* ke 1, selanjutnya : *i>1*, dan menaikkan *i++* (increment) sehingga nilai *i* tidak akan pernah mengembalikan *false*. Pada akhirnya akan mengarah pada kondisi loop terus menerus. *Infinite loop* bisa dilakukan juga jika penulisan setelah perintah *for* tidak diberikan nilai *variable* awal, akhir dan kenaikan dan penurunan ,

Contoh sintaks penulisan :

```
for(;;){  
    //kode untuk di eksekusi :  
}
```

Contoh program *infinite loop*

```
public class ForExample {
```

```
public static void main(String[] args) {  
    //Gunakan tanpa kondisi pada for loop  
    for(;;){  
        System.out.println("infinitive loop");  
    }  
}  
}
```

Output :

```
infinitive loop  
infinitive loop  
infinitive loop  
infinitive loop  
infinitive loop  
ctrl+c
```

7. Enhanced for loop

Enhanced for loop berguna ketika kita ingin mengulangi perintah pada Array/larik(Array akan dibahas lebih lanjut pertemuan berikutnya), *Enhanced for loop* melakukan iterasi secara otomatis berdasarkan jumlah elemen array sehingga tidak perlu lagi menentukan batasan awal dan batasan akhir melalui variabel control control nya. Keuntungan nya ketika menggunakan *enhanced for statement* adalah menghindari proses iterasi jumlah elemen array diluar yang seharusnya karena kesalahan memberi nilai pada control variabel.

Bentuk Umum penulisan *enhanced for* :

```
for (parameter : nama-array)  
    pernyataan tunggal;
```

atau

```
for (parameter : nama-array ) {
```

```
    blok pernyataan;  
}
```

contoh :

```
class ForLoopExample3 {  
    public static void main(String args[]){  
        int arr[]={2,11,45,9};  
        for (int num : arr) {  
            System.out.println(num);  
        }  
    }  
}
```

Output :

```
2  
11  
45  
9
```

C. LATIHAN /TUGAS

1. Buat program pengulangan for untuk menampilkan bilangan pertambahan 5. Nilai awal dan nilai akhir ditentukan menggunakan input menggunakan keyboard (*Scanner*)

Contoh output :

```
Nilai Awal ? : 5  
Nilai Akhir ? : 100
```

Hasil nya :

- 1. 5**
- 2. 10**
- 3. 15**
- 4. 20**
- 5. 25**
- 6. 30**
- 7. 35**
- 8. 40**
- 9. 45**
- 10. 50**

2. Buat program pengulangan for untuk menampilkan bilangan tahun, Nilai awal dan nilai akhir ditentukan menggunakan input menggunakan keyboard (*Scanner*)

Contoh output :

Tahun Akhir ? : 2015

Tahun Awal ? : 1950

Hasil nya :

2015

2014

2013

2012

.

.

1950

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4th Edition*, san jose university , united state Of America. RRD jefferson city publishing.

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal, (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

PERTEMUAN 9

MENGONTROL ALIRAN PROGRAM PERULANGAN WHILE, DO WHILE

A. TUJUAN PEMBELAJARAN

1. Mahasiswa dapat memahami Konsep Kontrol pengulangan / Looping While Do while
2. Mahasiswa dapat mengerti aturan penulisan Kontrol pengulangan While
3. Mahasiswa dapat mengerti aturan penulisan Kontrol pengulangan Do While
4. Mahasiswa dapat mempraktekan penggunaan Kontrol pengulangan While dan Do While

B. URAIAN MATERI

1. Perulangan While

Perulangan *while* digunakan untuk mengulang bagian program sampai kondisi Boolean yang ditentukan bernilai **true**. setelah kondisi Boolean bernilai **false**, pengulangan otomatis berhenti.

Jika jumlah iterasi tidak tetap, disarankan untuk menggunakan perulangan *while*.

Bentuk umum penulisan perulangan **while** adalah :

```
while (condition)
{
    statement(s);
    Increment / decrement
}
```

a. **Condition**/kondisi:

Adalah ekspresi yang diuji. Jika kondisinya benar atau *true*, badan loop atau statement dieksekusi. Ketika kondisi bernilai *false*, maka keluar dari perulangan *while*.

contoh $i \leq 90$

b. **Statement**/Pernyataan

badan program yang dieksekusi selama kondisi *true*

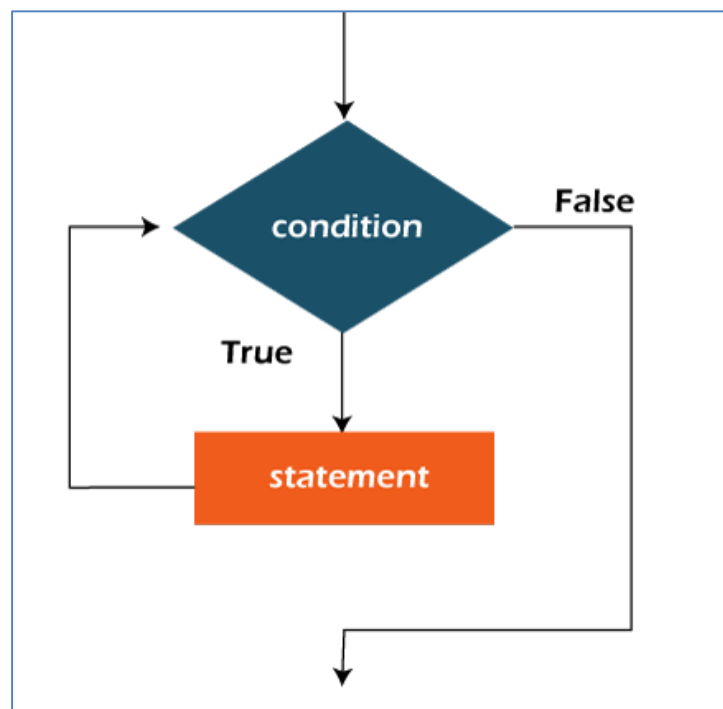
c. **Increment /decrement**

Adalah ekspresi penambahan / pengurangan suatu variable kondisi

Contoh $i++$

2. **Gambar diagram flowchart perulangan While**

Hal penting tentang while loop adalah, terkadang perulangan ini tidak dapat dieksekusi. Jika kondisi yang akan diuji menghasilkan *false*, badan perulangan akan dilewati dari pernyataan pertama setelah perulangan *while* akan dieksekusi.



Gambar 9. 1 diagram flowchart perulangan while

Contoh program perulangan while :

- a. Pada contoh di bawah ini, kita akan mencetak nilai bilangan integer dari 10 hingga 100. Jika melewati angka 100 maka perulangan akan berhenti ekspresi variabel menggunakan penambahan (increment)

```
public class while1 {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        int i=10;  
        while(i<=100){  
            System.out.println(i);  
            i+=10;  
        }  
    }  
}
```

Output :

```
10  
20  
30  
40  
50  
60  
70  
80  
90  
100
```

- b. Pada contoh berikutnya, kita akan mencetak nilai bilangan integer dari 15 hingga 5. Jika angka dibawah 5 maka perulangan akan berhenti ekspresi menggunakan pengurangan/penurunan 1 (decrement)

```
public class while2 {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        int i=10;  
        while(i>=1){  
            System.out.println(i);  
            i--;  
        }  
    }  
}
```

Output :

```
10  
9  
8  
7  
6  
5  
4  
3  
2  
1
```

3. Infinite while loop

Infinite while loop atau perulangan yang tidak akan pernah berakhir, adalah loop while yang tak terbatas. karena kondisinya selalu **true** contoh

program dibawah ini adalah *Infinite while loop* karena $i > 1$ akan selalu benar saat kita menaikkan nilai i di dalam while loop.

Contoh program :

```
public class infinitewhile {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        int i=10;  
        while(i>1)  
        {  
            System.out.println(i);  
            i++;  
        }  
    }  
}
```

Output :

```
10  
11  
12  
13  
14  
Ctrl+C
```

4. Perulangan while dengan pernyataan Break

Saat bekerja dengan while loop, terkadang diinginkan untuk melewati beberapa pernyataan di dalam loop atau segera menghentikan loop tanpa memeriksa ekspresi pengujian.

Dalam kasus seperti itu, pernyataan *break*/ Pernyataan *break* di Java akan mengakhiri loop, dan kontrol program berpindah ke pernyataan berikutnya setelah loop. biasanya digunakan dengan pernyataan pengambilan keputusan (Pernyataan *if...else*).

Berikut contoh pernyataan break pada while Java:

```
public class whileBreak {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        // for loop  
        for (int i = 1; i <= 10; ++i) {  
  
            // if the value of i is 5 the loop terminates  
            if (i == 5) {  
                break;  
            }  
            System.out.println(i);  
        }  
    }  
}
```

Output :

```
1  
2  
3  
4
```

Pada program di atas, kita menggunakan perulangan for untuk mencetak nilai i pada setiap iterasi.

Perhatikan pernyataan dibawah,

```
if (i == 5) {  
    break;  
}
```

Artinya ketika nilai *i* sama dengan 5, loop berakhir. Oleh karena hasil output nilai kurang dari 5 saja.

5. Perulangan Do While

Perulangan *do-while* digunakan untuk mengulangi bagian dari program berulang kali, sampai kondisi yang ditentukan benar atau *true*. setidaknya perulangan dieksekusi satu kali, disarankan untuk menggunakan loop *do-while*.

Perulangan *do-while* disebut *exit control loop*. Oleh karena itu, tidak seperti perulangan *for* dan *while*, *do-while* memeriksa kondisi di ujung loop body. Perulangan *do-while* loop dieksekusi setidaknya sekali karena kondisi diperiksa setelah badan perulangan.

Bentuk umum perulangan ***do-while***

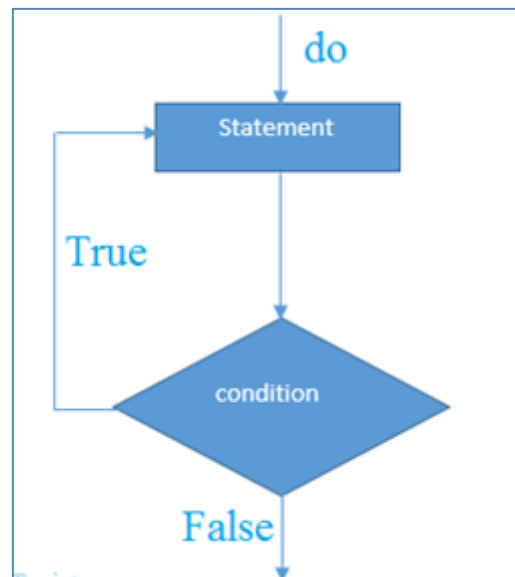
```
do
{
    statement(s);
} while(condition);
```

a. **Condition**/kondisi:

adalah ekspresi yang diuji. Jika kondisinya benar atau *true*, badan loop atau statement dieksekusi. Ketika kondisi bernilai *false*, maka keluar dari perulangan
contoh $i \leq 90$.

b. **Statement**/Pernyataan

badan program yang dieksekusi selama kondisi ***true***



Gambar 9. 2 diagram flowchart do-while

Contoh 1 program perulangan do-while (increment)

Pada contoh di bawah ini, hasilnya akan mencetak nilai integer dari 1 hingga 10. Berbeda dengan pengulangan for, kita perlu menginisialisasi variabel yang digunakan terlebih dahulu dalam suatu kondisi (contoh disini i). Jika tidak, perulangan akan mengeksekusi tanpa batas.

```
public class dowhileloop {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        int i=1;  
        do{  
            System.out.println(i);  
            i++;  
        }while(i<=10);  
    }  
}
```


Output :

```
1
2
3
4
5
6
7
8
9
10
```

Contoh 2 program perulangan do-while (decrement)

```
public class doWhilededcrement {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int i=10;
        do{
            System.out.println(i);
            i-=3;
        }while(i>=1);
    }
}
```

```
}
```

Output :

```
10
 7
 4
 1
```

C. LATIHAN/TUGAS

1. Modifikasi program pengulangan nested for dibawah ini menggunakan perulangan *while*

```
public class piramidnested {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        for(int i=1;i<=5;i++){
            for(int j=1;j<=i;j++){
                System.out.print("* ");
            }

            System.out.println();//baris baru
        }
    }
}
```

Output yang diharapkan :

```
*
```

```
* *  
* * *  
* * * *  
* * * * *
```

2. Buat Program deret bilangan bulat dengan penambahan 5, variabel nilai awal dan nilai akhir di input menggunakan *scanner* :

Output yang diharapkan :

```
Masukan bilangan awal : 3  
Masukan bilangan akhir : 15  
Hasil deret bilangan :  
3 , 8, 13
```

3. Lengkapi bidang kosong pada potongan program dibawah ini untuk menampilkan variabel i

```
int i = 1;  
  
[ ] (i < 6) {  
    System.out.println(i);  
    [ ] ;  
}
```

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

PERTEMUAN 10

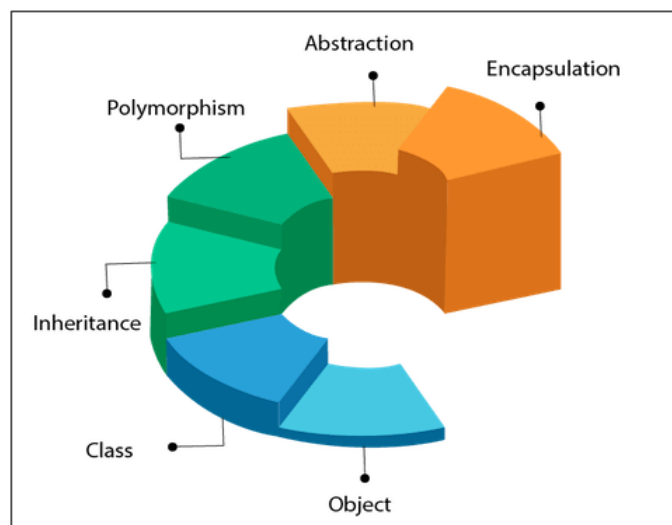
KONSEP PEMROGRAMAN BERORIENTASI OBJEK *CLASS, OBJECT, ATTRIBUTE, METHOD*

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami Konsep Pemrograman berorientasi Objek
2. Mahasiswa memahami apa itu object , *class* pada Pemrograman berorientasi Objek
3. Mahasiswa memahami apa itu *attribute, method* pada Pemrograman berorientasi Objek
4. Mahasiswa dapat mempraktekan pembuatan class,object , attribute dan method pada Pemrograman berorientasi Objek

B. URAIAN MATERI

Pemrograman berorientasi Objek atau Object-oriented programming System (OOPs) adalah paradigma pemrograman berdasarkan konsep "objek" yang berisi attribute atau data dan metode. Tujuan utama dari pemrograman berorientasi objek adalah untuk meningkatkan fleksibilitas dan pemeliharaan program. Pemrograman berorientasi objek menyatukan data dan perilakunya (metode) dalam satu lokasi (objek) sehingga pembuatannya lebih mudah untuk dipahami bagaimana sebuah program bekerja.



Gambar 10. 1 Paradigma pemrograman berorientasi objek

1. *Object*

Setiap entitas yang memiliki state/atribut dan behavior dikenal sebagai objek. Misalnya, kursi, pena, meja, keyboard, sepeda, dll. Objek bisa berupa fisik atau logikal.

Objek dapat didefinisikan sebagai turunan (instance of) dari kelas. Sebuah objek berisi alamat dan mengambil beberapa ruang di memori. Objek dapat berkomunikasi tanpa mengetahui detail data atau kode satu sama lain.

Contoh: motor adalah *object* karena memiliki state atau atribut seperti warna, tahun pembuatan, jenis, dll. serta perilaku/methode seperti cara hidupkan, cara menambahkan kecepatan, menyalakan lampu depan dll.



Gambar 10. 2 ilustrasi sebuah object Konsep OOP

Object Motor

Attribute :

Jenis, Warna, tahun produksi, no seri, no mesin

Behaviour :

Cara menghidupkan, cara menjalankan, cara berbelok, cara akselerasi.

Attribute → Variable

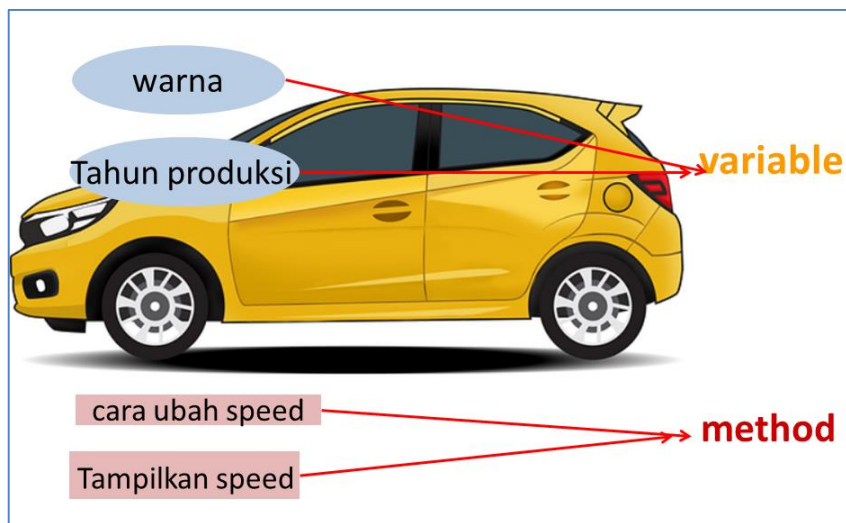
Behaviour → method/ fungsi

2. Class

Class adalah Kumpulan objek, *class* jika di instansiasi menjadi objek *Class* juga dapat didefinisikan sebagai cetak biru dari objek individual, dimana hasil cetakan itu menjadi *object*. *Class* merupakan kesatuan *Method* (Behavior) dan *variable* (*Attribute*)

Contoh ilustrai *Class* :

class mobil

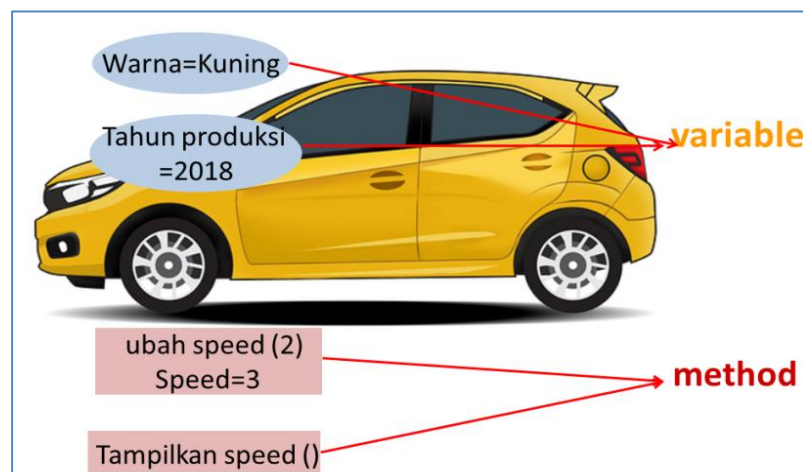


Gambar 10. 3 ilustrasi sebuah class Konse OOP

Sementara *object* adalah *Method* (Behavior) dan *variable* (*Attribute*) yang diberikan nilai.

Contoh ilustrasi *object* :

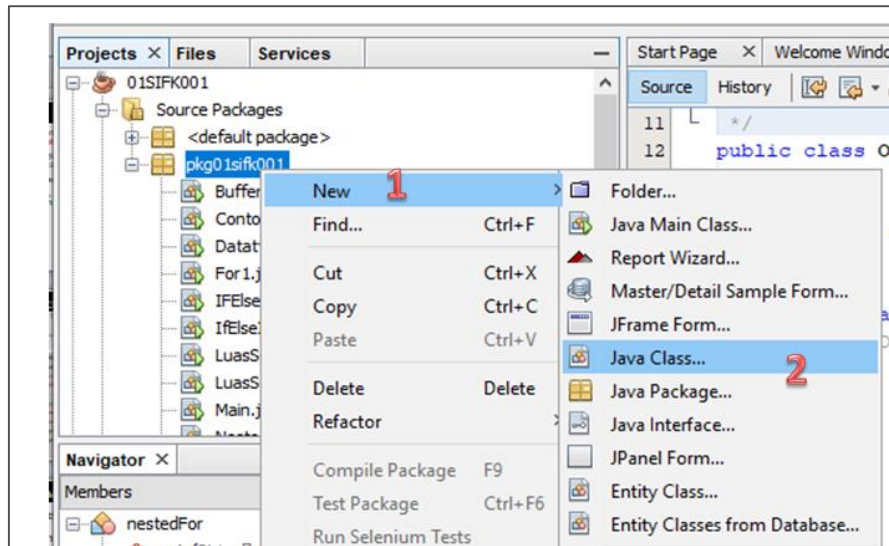
class mobilku :



Gambar 10. 4 ilustrasi sebuah object dengan Value

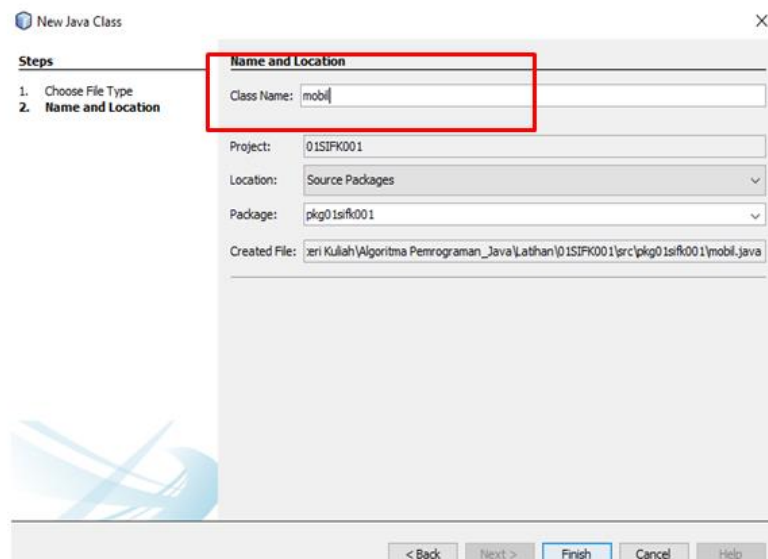
3. Membuat class , object dan attribute pada program java

- a. Buat *class mobil* pada program java langkahnya klik menu file new [1], pilih java class[2]



Gambar 10. 5 membuat class mobil

- b. Beri nama class pada kotak isian penamaan dan lokasi penyimpanan contoh nama class : mobil



Gambar 10. 6 memberi penamaan class mobil

- c. Buat attributnya (Variable) Jenis, warna, tahun produksi pada class mobil


```
- | ~/  
    public class mobil {  
        String jenis;  
        String warna;  
        int tahunproduksi;  
    }
```

- d. Buat program utama namakan OopMobil untuk membuat objek dan instansiasi dari *class*

Contoh Nama objek =mymobil()

Contoh pemberian nilai attribute jenis="SUV" " , warna="Kuning", tahun produksi=2010

```
public class OopMobil {  
    /**  
     * @param args the command line arguments  
     */  
  
    public static void main(String[] args) {  
        // TODO code application logic here  
        mobil mymobil=new mobil(); //membuat objek mymobil dari class mobil  
        mymobil.jenis="SUV";  
        mymobil.warna="Kuning";  
        mymobil.tahunproduksi=2010;  
        System.out.println("Jenis Mobil "+mymobil.jenis);  
        System.out.println("Warna Mobil "+mymobil.warna);  
        System.out.println("Tahun Produksi Mobil "+mymobil.tahunproduksi);  
    }  
}
```

Output :

```
Jenis Mobil SUV  
Warna Mobil Kuning  
Tahun Produksi Mobil 2010
```

4. Membuat class, object dan method nya pada program java

Method adalah suatu fungsi atau urutan instruksi yang mengakses data dari objek. Pada penulisan method biasanya di akhiri () , method melakukan manipulasi data, perhitungan matematika dan memonitor kejadian dari suatu event.

Contoh membuat method pada class mobil sebelumnya, kita akan membuat method tampilkan nilai nilai dari atribut.

- a. Modifikasi class mobil sebelumnya dengan menambah method printdatamobil()

```

/*
 * @author Irfan
 */
public class mobil {
    String jenis;
    String warna;
    int tahunproduksi;

    void printdatamobil(){
        System.out.println("Jenis: " + jenis);
        System.out.println("Warna: " + warna);
        System.out.println("Tahun: " + tahunproduksi);
    }
}

```




- b. Selanjutnya akses method tersebut di program utama, seperti gambar dibawah ini

```

/*
 * @param args the command line arguments
 */
public class OopMobil {

    public static void main(String[] args) {
        // TODO code application logic here
        mobil mymobil=new mobil(); //membuat objek mymobil dari class mobil
        mymobil.jenis="SUV";
        mymobil.warna="Kuning";
        mymobil.tahunproduksi=2010;
        mymobil.printdatamobil();
    }
}

```



Output :

```
Jenis: SUV
Warna: Kuning
Tahun: 2010
```

5. jenis method accesor dan mutator

a. method *accesor*

Metode ***accesor*** digunakan untuk mengembalikan nilai dari *private field*. skema penamaan diawali dengan kata "get" ke awal nama metode.

b. method *mutator*

Metode ***mutator*** digunakan untuk menetapkan nilai *private field*. skema penamaan awalan diawali dengan kata "set"

Contoh penulisan method accesor dan mutator :

```
public class aritmatika{
    int angka=10;

    // ini method (mutator) dengan parameter
    void setAngka(int pertambahan) {
        angka= angka+ pertambahan;
    }

    // ini method (accessor)
    int getAngka() {
        return angka;
    }
}
```

Contoh membuat program pertambahan nya dengan nama program artimatikaMain :

```
public class artimatikaMain {
```

```
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    aritmatika aritmatikaku=new aritmatika();
    aritmatikaku.setTambah(20);

    System.out.println("Hasil Pertambahan
"+aritmatikaku.getTambah());
}
}
```

Output

```
run:
Hasil Pertambahan 20
BUILD SUCCESSFUL (total time: 0 seconds)
```

Contoh membuat method dengan 2 paramater

a. Buat class aritmatika2 dengan *attribute* dan *method* nya

```
public class aritmatika {

    int angka;
    // ini method (mutator) dengan 2 parameter bil1 dan
    bil2
    void setTambah(int bil1, int bil2) {
        angka= bil1+bil2;
    }
    // ini method (accessor)
    int getTambah() {
        return angka;
    }
}
```

```
}
```

- b. Buat program class aritmatika_Utama2 untuk membuat objek dan mengakses class aritmatika

```
public class aritmatikaMain2 {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
  
        aritmatika2 aritmatikaku=new aritmatika2();  
        int a=10,b=20;  
        aritmatikaku.setTambah(a,b);  
        System.out.println("Bilangan 1= " +a);  
        System.out.println("Bilangan 2= " +b);  
        System.out.println("Hasil Pertambahan  
"+aritmatikaku.getTambah());  
  
    }  
}
```

Output :

```
run:  
Bilangan 1= 10  
Bilangan 2= 20  
Hasil Pertambahan 30  
BUILD SUCCESSFUL (total time: 0 seconds)
```

C. LATIHAN / TUGAS

1. Buat Class aritmatika3 modifikasi dari class aritmatika2, yang berisi method dengan dua parameter tambahkan method :
 - a. pengurangan(int a, int b)
 - b. perkalian(int a, int b)
 - c. pembagian(int a, int b)
 - d. pangka(int a, int b)
2. Buat Class aritmatika3Main untuk membuat object serta meng akses method pada class aritmatika 1 nilai nilai objeknya sebagai berikut ;
 - a. Pengurangan: bilangan1 – bilangan2
 - b. Perkalian: bilangan1*bilangan2
 - c. Pembagian: bilangan1/bilangan2
 - d. Pangkat: bilangan1^bilangan2

D. REFERENSI

- Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.
- Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.
- Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.
- Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press
- Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.
- Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

PERTEMUAN 11

KONSEP PEMROGRAMAN BERORIENTASI OBJEK *CONSTRUCTOR*

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami Konsep OOP *constructor*
2. Mahasiswa memahami cara kerja dan jenis jenis *constructor*
3. Mahasiswa dapat mempraktekan penggunaan *constructor* dalam bahasa java

B. URAIAN MATERI

Constructor adalah blok kode yang menginisialisasi objek yang baru dibuat. *Constructor* menyerupai *method* pada java tetapi tidak memiliki tipe pengembalian. *Constructor* dan metode berbeda. Orang sering menyebut *Constructor* sebagai jenis *method* khusus pada java.

Constructor memiliki nama yang sama dengan *class* dan terlihat seperti ini dalam kode bahasa java.

```
public class MyClass{  
    //ini adalah constructor  
    MyClass() {  
    }  
    ..  
}
```

Catatan : Perhatikan bahwa nama constructor cocok dengan nama class dan tidak memiliki tipe pengembalian.

Konstruktor di Java adalah metode khusus yang digunakan untuk menginisialisasi objek. Konstruktor dipanggil ketika objek kelas dibuat. Ini dapat digunakan untuk menetapkan nilai awal untuk atribut objek.

1. Bagaimana constructor bekerja

Untuk memahami cara kerja konstruktor, mari kita ambil contoh. katakanlah kita memiliki kelas MyClass.

Selanjutnya buat objek MyClass seperti contoh dibawah ini :

```
MyClass objekku = new MyClass()
```

Pada saat membuat *object* bernama objekku dari *class* MyClass dan memanggil constructor untuk instansisi, mungkin sedikit bingung, mari kita lanjutkan untuk melengkapi kode sederhananya.

a. Buat class MyClass ketikan kode dibawah ini :

```
public class myClass {  
    String nama;  
    //Constructor  
    myClass() {  
        this.nama = "agustav.web.id";  
    }  
}
```

b. Buat program namakan myConstructor1 untuk menginstansiasi class MyClass menjadi *object* ketikan kode dibawah ini :

```
public class myConstructor1 {  
    public static void main(String[] args) {  
        // TODO code application logic here  
        myClass objekku = new myClass();  
        System.out.println(objekku.nama);  
    }  
}
```

Output :

```
run:  
agustav.web.id
```



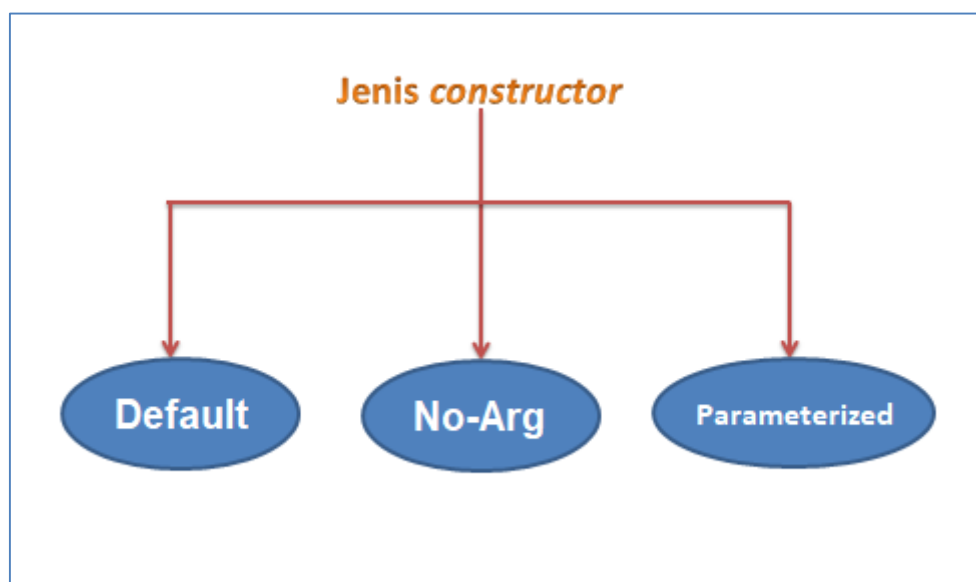
```
BUILD SUCCESSFUL (total time: 1 second)
```

Seperti yang kita lihat bahwa outputnya adalah `agustav.web.id` yang telah diberikan ke nama selama inisialisasi di konstruktor. Ini menunjukkan bahwa ketika kita membuat objek `objekku`, *constructor* dipanggil. Dalam contoh ini kita telah menggunakan kata kunci `ini`.

2. Jenis jenis constructor

Terdapat tiga jenis *constructor* yaitu :

- Default
- No-arg
- Parameterized



Gambar 11. 1 Jenis Constructor

a. Default Constructor

Jika kita tidak mengimplementasikan *constructor* apa pun pada *class*. Compiler Java menyisipkan *constructor* default ke dalam kode sumber. *constructor* ini dikenal sebagai *constructor* default. *Constructor* disebut "**Default**" ketika tidak memiliki parameter apa pun.

Bentuk umum penulisan **Default Constructor**

```
<class_name>() {  
  
}
```

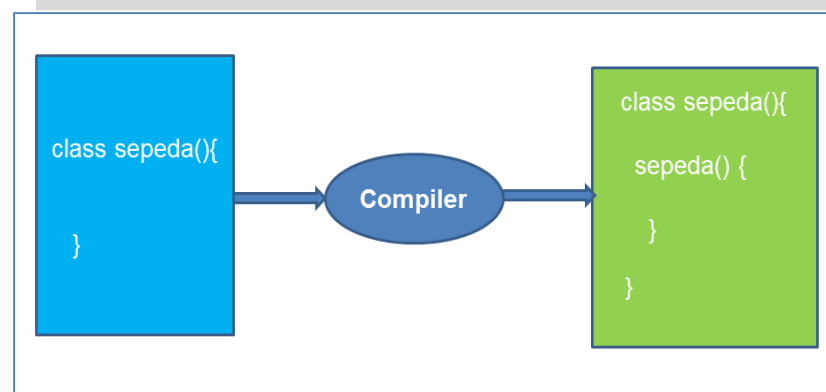
Contoh program sederhana penulisan *default constructor* :

```
// Program untuk membuat dan memanggil sebuah default  
constructor  
class sepeda{  
    //membuat default constructor  
    sepeda()  
    {System.out.println("sepeda sedang dibuat");  
    }  
}
```

```
//main method  
public static void main(String args[]){  
    //calling a default constructor  
    Sepedal b=new sepeda();  
    }  
}
```

Output :

```
sepeda sedang dibuat
```



Gambar 11. 2 Default Constructor

b. *No-Arg Constructor*

Constructor tanpa argumen atau lebih dikenal *No-Arg Constructor* ini sama dengan constructor *default*, namun ada perbedaan badan konstruktor memiliki kode.

Meskipun mungkin beberapa orang mengklaim bahwa konstruktor default dan no-arg itu sama tetapi kenyataannya tidak, jika kita menulis `Demo publik() { } class Demo`, itu bukan disebut konstruktor default karena kita telah menulis kode tersebut.

Contoh : constructor *class demoNoArg*

```
public class demoNoArg {  
    public demoNoArg(){  
        System.out.println("Ini constructor tanpa argumen ");  
    }  
}
```

```
public class NoArgMain {  
  
    public static void main(String[] args) {  
        // TODO code application logic here  
        new demoNoArg();  
    }  
}
```

Output :

```
run:  
  
Ini constructor tanpa argumen  
  
BUILD SUCCESSFUL (total time: 1 second)
```

c. Paramterized Constructor (Konstruktor ber parameter)

Constructor dapat diberikan nilai input dengan mendefinisikan variabel Parameter nya setelah menulisi konstruktor nya diantara kurung buka dan kurung tutup. Atau bisa disebut dengan konstruktor dengan argumen.

Contoh program konstruktor berparameter

```
public class Karyawan {

    int IdKaryawan;

    String NamaKaryawan;

    // constructor dengan 2 parameter
    Karyawan (int id, String nama){

        this.IdKaryawan = id;

        this.NamaKaryawan = nama;

    }

    void info(){

        System.out.println("Id Karyawan : "+IdKaryawan+"
Nama: "+NamaKaryawan);

    }

}
```

```
public class KaryawanMain {

    /**
     * @param args the command line arguments
     */

    public static void main(String[] args) {

        // TODO code application logic here

    }

}
```

```
Karyawan objek1=new Karyawan(1111, "Agus");  
Karyawan objek2=new Karyawan(1112,  
"Suharto");  
objek1.info();  
objek2.info();  
  
}  
  
}
```

Output :

```
run:  
Id Karyawan : 1111 Nama: Agus  
Id Karyawan : 1112 Nama: Suharto  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Pada contoh diatas kita memiliki konstruktor berparameter dengan dua parameter id dan nama. Saat membuat objek objek1 dan objek2 setelah melewati dua argumen, lalu konstruktor itu dipanggil maka objek1 dan objek2.mengisikan nilai nilai paramater diantara kurung buka dan kurung tutup objek tersebut.

Contoh 2 program konstruktor berparameter

```
public class contoh2 {  
    private int var;  
    //default constructor  
    public contoh2()  
    {  
        this.var = 10;  
    }  
}
```

```
    }

    //parameterized constructor
    public contoh2(int angka)
    {
        this.var = angka;
    }

    public int getValue()
    {
        return var;
    }
}
```

```
public class CContoh2Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        contoh2 obj = new contoh2();
        contoh2 obj2 = new contoh2(150);
    }
}
```

```
System.out.println("Nilai      paramater      :  
"+obj.getValue());  
  
System.out.println("Nilai      paramater      :  
"+obj2.getValue());  
  
    }  
  
}
```

Output :

```
run:  
Nilai paramater : 10  
Nilai paramater : 150  
BUILD SUCCESSFUL (total time: 1 second
```

Pada contoh diatas, kita memiliki dua konstruktor, konstruktor default dan konstruktor berparameter. Ketika tidak melewati parameter apa pun saat membuat objek menggunakan kata kunci baru maka konstruktor default dipanggil, namun ketika melewati parameter, konstruktor berparameter yang cocok dengan daftar parameter yang diteruskan nilai parameter dipanggil.

d. Keyword Super()

Ketika Setiap kali konstruktor kelas anak dipanggil, secara implisit memanggil konstruktor kelas induk. Ini dapat dikatakan bahwa kompiler menyisipkan *keyword* `super()`; pernyataan `super()` di awal konstruktor kelas anak atau *sub class*.

Contoh program keyword Super ()

```
public class IndukClass {  
  
    IndukClass () {  
  
        System.out.println("Induk Class Constructor");  
  
    }  
  
}
```

```
}  
  
class AnakClass extends IndukClass{  
  
    AnakClass() {  
  
        System.out.println("Anak Class Constructor");  
  
    }  
  
}
```

```
public class IndukSuper {  
  
    public static void main(String[] args) {  
  
        // TODO code application logic here  
  
        new AnakClass();  
  
    }  
  
}
```

Output :

```
run:  
  
Induk Class Constructor  
  
Anak Class Constructor  
  
BUILD SUCCESSFUL (total time: 0 seconds)
```

C. LATIHAN/TUGAS

1. Buat program untuk input data karyawan menggunakan constructor paramater :
ID Karyawan , Nama Karyawan, Gol, Jabatan, GajiPokok.
2. Buat Main program nya untuk mengisi nilai nilai paramater yang didefinisikan oleh *constructor parameter*, dengan fungsi scanner untuk menginput nilai nya :
isian Jabatan, gaji pokok otomatis (IF..Else, Switch) berdasarkan gol, acuan tabelnya sebagai berikut :

Tabel 11. 1 Tabel Golongan

Gol	Jabatan	Gaji Pokok
1	Assisten Staff	3.000.0000
2	Staff	3.500.000
3.	Supervisor	4.000.000
4.	Assisten Manager	5.000.000
5.	Manager	6.000.000

Contoh Output program :

```
ID Karyawan      : 0001
Nama Karyawan    : Agus Suharto
Gol              : 2
```

```
                ID Karyawan      : 0001
Gol              : 3
Jabatan          : Supervisor
Gaji Pokok       : 4.000.000
```

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training ,
<https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember
2021

PERTEMUAN 12

KONSEP PEMROGRAMAN BERORIENTASI OBJEK

INHERITANCE/PEWARISAN

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami Konsep OOP *constructor Inheritance*
2. Mahasiswa memahami cara kerja dan jenis jenis *Inheritance*
3. Mahasiswa dapat mempraktekan penggunaan *Inheritance* dalam bahasa java
4. Mahasiswa dapat mempraktekan dan membedakan jenis jenis *Inheritance* dalam bahasa java

B. URAIAN MATERI

Inheritance atau pewarisan pada Java adalah mekanisme di mana satu objek *child class* memperoleh semua properti dan perilaku atau method dari *parent class*. Ini adalah bagian penting dari OOP (sistem pemrograman Berorientasi Objek).

Inheritance memungkinkan kita untuk menggunakan kembali kode, serta meningkatkan reusability dalam aplikasi java. Keuntungan terbesar dari *Inheritance* adalah bahwa kode yang sudah ada di *base* atau *parent class* tidak perlu ditulis ulang di *child class*.

Child class :

Adalah class yang memperluas fitur dari class lain dikenal sebagai kelas *Child class*, subclass atau class turunan.

Parent Class :

Class dimana properti dan fungsinya digunakan (diwarisi) oleh class lain dikenal sebagai kelas Parent Class super, atau Base class.

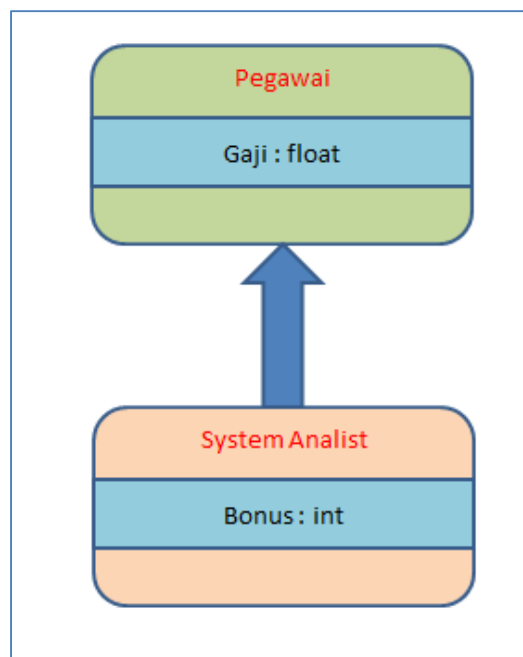
1. Bentuk Umum penulisan *Inheritance*

```
class XYZ extends ABC
{
}
}
```

Untuk mewarisi class, kita menggunakan kata kunci **extends**. Pada penulisan diatas class **XYZ** adalah **child class** dan class **ABC** adalah **parent class**. class XYZ mewarisi properti dan metode class ABC.

Kata kunci **extends** menunjukkan bahwa kita membuat *class* baru yang diturunkan dari class yang ada. Arti dari "extends" adalah untuk meningkatkan fungsionalitas.

Dalam terminologi Bahasa Java, kelas yang diwarisi disebut parent atau superclass, dan class baru disebut child class atau subclass.



Gambar 12. 1 ilustrasi inheritance java OOP

Seperti yang ditampilkan pada gambar di atas, **System Analyst** adalah subclass atau *child class* dan **Pegawai** adalah superclass atau *parent class*. Hubungan antara kedua *class* adalah **System Analyst** IS-A **Pegawai**.

Contoh 1 penulisan program *inheritance* :

```
class pegawai{  
    float gaji=4000000;  
}
```

```
class SystemAnalyst extends pegawai{  
    int bonus=1000000;  
    public static void main(String args[]){  
        SystemAnalyst SA=new SystemAnalyst ();  
        System.out.println("Gaji          System          Analyst  
:" +SA.salary);  
        System.out.println("Bonus              System  
Analyst:" +SA.bonus);  
    }  
}
```

Output :

```
Gaji System Analyst : 4000000  
Bonus System Analyst : 1000000
```

Contoh 2 penulisan program *inheritance* :

```
public class dosen {  
    String prodi = "Sistem Informasi";  
    String universitas = "Universitas Pamulang";  
    void semester(){  
        System.out.println("Semester 1");  
    }  
}
```

```
public class ChildInherit extends dosen {  
    String matakuliah1 = "PBO";  
    String matakuliah2 = "Bahasa Inggris";  
    String matakuliah3 = "Matematika";  
}
```

```
String matakuliah4 = "Agama";

/**
 * @param args the command line arguments
 */

public static void main(String[] args) {
    // TODO code application logic here
    ChildInherit matkul=new ChildInherit();

    System.out.println(matkul.universitas);
    System.out.println(matkul.prodi);
    matkul.semester();
    System.out.println(matkul.matakuliah1);
    System.out.println(matkul.matakuliah2);
    System.out.println(matkul.matakuliah3);
    System.out.println(matkul.matakuliah4);

}
}
```

Output :

```
Universitas Pamulang
Sistem Informasi
Semester 1
PBO
Bahasa Inggris
Matematika
Agama
```

Pada contoh diatas, kita memiliki class `dosen()` sebagai *parent class*, dan `ChildInherit` sebagai *child class*. Kemudian class `ChildInherit` di `extend` sebagai *parent class*, sehingga properti dan method di warisi oleh *parent class*, kita tidak perlu mendeklarasikan properti dan *method* ini di `ChildInherit()` lagi.

Di sini kita memiliki metode dan property untuk semua yang ada di *parent class* dengan cara ini *child class* seperti diatas tidak perlu menulis kode lagi (*reuseable*) dan dapat digunakan langsung dari *child class*.

Berdasarkan contoh di atas kita dapat mengatakan bahwa `ChildInherit()` IS-A. `dosen()` atau *child class* memiliki *relationship* IS-A dengan *parent class*. Inilah yang disebut *inheritance* atau pewarisan.

Catatan:

Class turunan mewarisi semua anggota dan metode yang dideklarasikan sebagai **public** atau **protect**. Jika anggota atau metode class super dideklarasikan sebagai **private** maka class turunan tidak dapat menggunakannya secara langsung. Anggota *private* hanya dapat diakses di class nya sendiri. Anggota **private** tersebut hanya dapat diakses menggunakan metode **setter** dan **getter** yang dilindungi seperti yang ditunjukkan pada contoh di bawah ini.

Contoh 3 Inheritance :

Parent Class : Dosen2

```
public class dosen2 {  
  
    private String prodi = "Sistem Informasi";  
  
    private String universitas = "Universitas  
Pamulang";  
  
    private String getprodi() {  
        return prodi;  
    }  
  
    protected void setprodi(String prodi) {  
        this.prodi = prodi;  
    }  
}
```

```
}

    private String getuniversitas() {
        return universitas;
    }

    protected void setuniversitas(String universitas)
{
        this.universitas = universitas;
    }

    void semester(){
        System.out.println("Semester 1");
    }

}
```

Child Class : Inheritchild2

```
public class Inheritchild2 extends dosen {

    String matakuliah1 = "PBO";

    String matakuliah2 = "Bahasa Inggris";

    String matakuliah3 = "Matematika";

    String matakuliah4 = "Agama";

    /**
     * @param args the command line arguments
     */
}
```



```
public static void main(String[] args) {  
    // TODO code application logic here  
    InheritMain matkul=new InheritMain();  
    System.out.println(matkul.universitas);  
    System.out.println(matkul.prodi);  
    matkul.semester();  
    System.out.println(matkul.matakuliah1);  
    System.out.println(matkul.matakuliah2);  
    System.out.println(matkul.matakuliah3);  
    System.out.println(matkul.matakuliah4);  
  
}  
}
```

Output :

```
Universitas Pamulang  
  
Sistem Informasi  
  
Semester 1  
  
PBO  
  
Bahasa Inggris  
  
Matematika  
  
Agama
```

Poin penting yang perlu diperhatikan dalam contoh di atas adalah bahwa *child class* dapat mengakses anggota *private parent class* melalui metode *parent class* yang diprivate . Ketika kita membuat variabel instan atau metode yang *protect* , ini berarti bahwa *parent class* hanya dapat diakses di kelas itu sendiri dan di *child class* .*public* , *private* , *protect* dll semua penentu akses akan dibahas pada pertemuan mendatang.

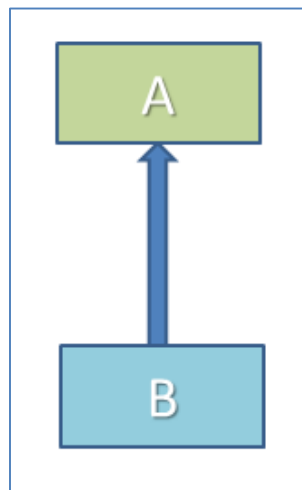
2. Jenis Jenis Inheritance

Pada *Inheritance* terdapat 4 jenis *inheritance* yaitu :

- a. *Single Inheritance*
- b. *Multiple Inheritance*
- c. *Multilevel Inheritance*
- d. *Hierarchical Inheritance*

a. Single Inheritance

Mengacu pada hubungan *child class* dan *parent class* di mana sebuah class meng extends class lain.



Gambar 12. 2 Single inheritance

Contoh program Single Inheritance

```
class hewan{  
    void makan(){System.out.println("memakan...");}  
}  
  
class kucing extends hewan{  
    void meong(){  
        System.out.println("mengeong...");  
    }  
}
```

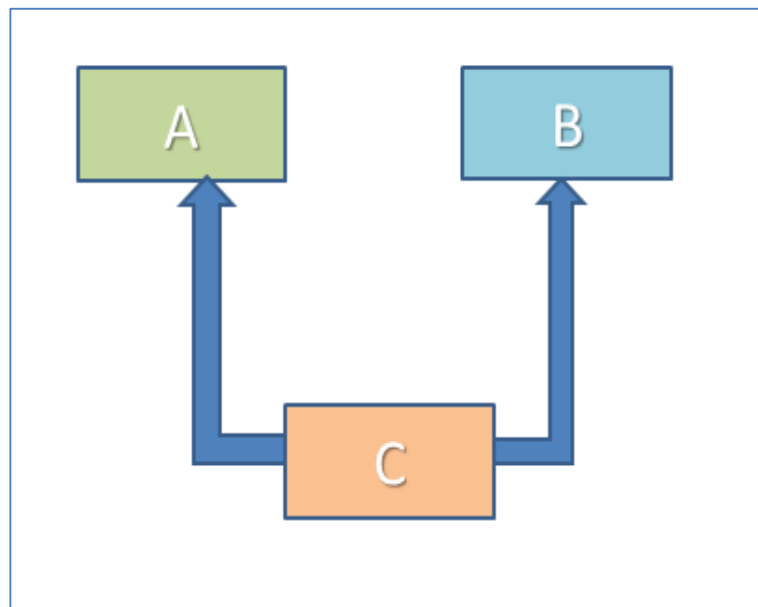
```
class testinheritance1{  
    public static void main(String args[]){  
        kucing k=new kucing();  
        d.meong();  
        d.makan();  
    }  
}
```

Output

```
mengeong...  
memakan...
```

b. Multiple *Inheritance*

Mengacu pada konsep satu *class* yang meng *extend* lebih dari satu *class*, yang berarti *child class* memiliki dua *Parent class*. Misalnya class C meng *extend* class A dan B. Pada Java tidak mendukung *Multiple Inheritance*

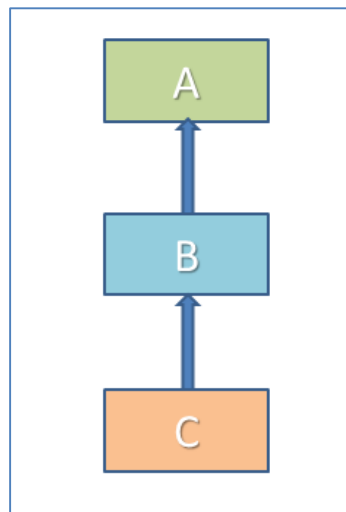


Gambar 12. 3 Multiple inheritance

Kenapa *Multiple inheritance* tidak didukung java? Pertimbangannya skenario di mana A, B, dan C adalah tiga class. Class C mewarisi kelas A dan B. Jika kelas A dan B memiliki metode yang sama dan Anda memanggilnya dari objek *child class*, akan ada ambiguitas untuk memanggil metode kelas A atau B.

c. Multilevel *Inheritance*

Mengacu pada hubungan *child class* dan *parent class* berdasarkan level di mana *parent class* meng extends *child class*. Misalnya kelas C meng-extend kelas B dan kelas B meng-extend kelas A



Gambar 12. 4 Multilevel inheritance

Contoh program *Multilevel inheritance*

```
class hewan{
    void makan(){System.out.println("menyusui...");}
}
class kucing extends hewan{
    void meong(){System.out.println("mengeong...");}
}
class bayikucing extends kucing{
    void tangis(){System.out.println("menangis...");}
}
```

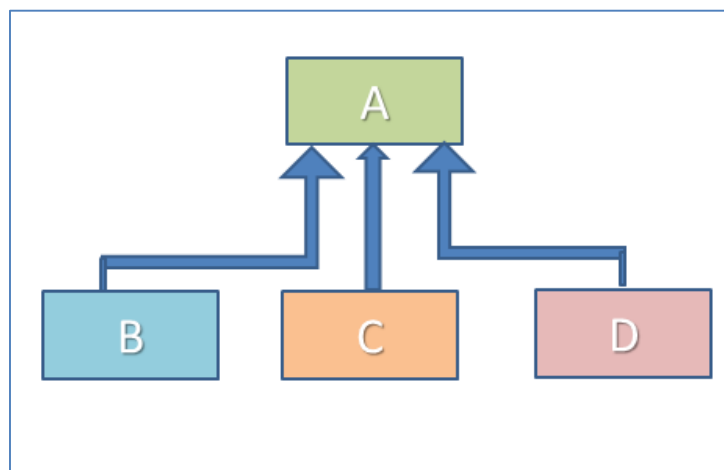
```
class TestInheritance2{  
    public static void main(String args[]){  
        baikkucing d=new baikkucing();  
        d.tangis();  
        d.meong();  
        d.makan();  
    }  
}
```

Output

```
Menangis...  
Mengeong...  
Menyusui...
```

d. Hierarchical *Inheritance*

Mengacu pada hubungan child class dan parent class di mana lebih dari satu *class* mengextends *class* yang sama. Misalnya, *class* B, C & D meng extend *class* A yang sama.



Gambar 12. 5 Hierarchical inheritance

```
class hewan{  
    void makan(){System.out.println("menyusui...");}  
}  
class kucing extends hewan{
```

```
void meong() {System.out.println("mengeong...");}
}
class burung extends hewan{
void cicit() {System.out.println("mencicit...");}
}
```

```
class TestInheritance3{
public static void main(String args[]){
burung b=new burung ();
b.cicit();
c.makan();
//c.meong (); //C.T.Error
}
}
```

Output :

```
mencicit ...
menyusui ...
```

C. LATIHAN / TUGAS

1. Kembangkan contoh 1 program diatas dengan menambahkan attribut Tunjangan Makan, Transport pada parent class, lalu pada child class kalkulasi gaji total dengan membuat mehod nya,
Rumus **gaji total** = Gaji + Bonus+ Tunjangan makan + Transport

Output yang diharapkan :

```
Gaji System Analyst      : 4000000
Bonus                    : 1000000
Tunjangan Makan          : 400000
```

Transport :	: 500000
Gaji Total	: 5900000

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4th Edition*, san jose university , united state Of America. RRD jefferson city publishing.

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 5Desember 2021

PERTEMUAN 13

KONSEP PEMROGRAMAN BERORIENTASI OBJEK *POLYMORPHISM*

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami Konsep OOP *Polymorphism*
2. Mahasiswa memahami jenis jenis *Polymorphism*
3. Mahasiswa dapat membedakan jenis *method overloading* dan *method override*
4. Mahasiswa dapat mempraktekan penggunaan *Polymorphism* dan jenis jenisnya

B. URAIAN MATERI

Polymorphism adalah konsep di mana kita dapat melakukan satu tindakan dengan cara yang berbeda. Polimorfisme berasal dari 2 kata Yunani: poli dan morf. Kata "poly" berarti banyak dan "morphs" berarti bentuk. Jadi polimorfisme berarti banyak bentuk *Polymorphism* / polimorfisme adalah salah satu fitur OOP. Sebagai contoh, katakanlah kita memiliki *class* Hewan yang memiliki *method* suara(). Karena *class* hewan adalah *class* yang lebih general *class* tersebut dapat memiliki *class* lebih spesifik atau diturunkan ke *class* sapi, *class* kuda, *class* kucing, *class* burung dan sebagainya.

Contoh sederhana implementasi *Polymorphism*

```
public class Hewan{  
    ...  
    public void suara(){  
        System.out.println("Hewan mengeluarkan suara");  
    }  
}
```

Sekarang katakanlah kita mempunyai 2 *subclass* dari *class* Hewan : yaitu *class* Kuda dan *class* Kucing yang di extend (inheritance) *class* Hewan. Kita implementasikan ke *method* yang sama seperti ini:


```
public class kuda extends hewan{  
    ...  
    @Override  
    public void suara(){  
        System.out.println("meringkik");  
    }  
}
```

```
public class kucing extends hewan{  
    ...  
    @Override  
    public void suara(){  
        System.out.println("mengeong");  
    }  
}
```

Seperti yang kita lihat bahwa meskipun memiliki method suara () untuk semua subkelas tetapi karena setiap hewan berbeda suara maka ada cara yang berbeda untuk melakukan tindakan yang sama. Inilah contoh dari polimorfisme (fitur yang memungkinkan kita untuk melakukan satu tindakan dengan cara yang berbeda). Tidak masuk akal hanya memanggil method suara () secara general karena setiap Hewan memiliki suara yang berbeda. Dengan demikian kita dapat mengatakan bahwa tindakan yang dilakukan method ini berdasarkan pada jenis objek.

1. Jenis jenis *Polymorphism*

Terdapat 2 macam jenis *Polymorphism* yaitu :

- a. *Static polymorphism* (menggunakan method *overloading*)
- b. *Dynamic polymorphism* (menggunakan method *overriding*)

a. *Static polymorphism*

Java, seperti banyak bahasa OOP lainnya, memungkinkan Anda untuk mengimplementasikan beberapa method dalam class yang sama yang menggunakan nama yang sama. Tetapi menggunakan seperangkat parameter yang berbeda atau disebut menggunakan metode overloading dan mewakili bentuk statis polimorfisme.

b. *Dynamic polymorphism*

Adalah suatu keputusan metode mana yang dipilih yang akan dijalankan atau penetapan selama masa run-time. Contoh *Dynamic polymorphism* adalah Metode Overriding. Polimorfisme ini mengacu pada kemampuan suatu objek berperilaku berbeda untuk *method* yang sama.

2. Method overloading

Method Overloading adalah fitur yang memungkinkan class memiliki lebih dari satu method yang memiliki nama yang sama, dengan parameter dan daftar argumennya atau jenis data nya berbeda.

Tujuan dari *Method Overloading* adalah ketika ada pemanggilan *method* akan lebih mudah.

Penggunaan *Method overloading* aturannya adalah sebagai berikut :

- a. Parameter dan nilai parameter harus berbeda
- b. Nilai kembalian boleh sama
- c. Nama Method harus berbeda

Contoh program *method Overloading*

```
class Overload
{
    void demoOverload (int a)
    {
        System.out.println ("a: " + a);
    }
    void demoOverload (int a, int b)
    {
        System.out.println ("a and b: " + a + "," +
b);
    }
}
```

```
    }  
    double demoOverload (double a) {  
        System.out.println("double a: " + a);  
        return a*a;  
    }  
}
```

```
class MethodOverloading  
{  
    public static void main (String args [])  
    {  
        Overload Objek = new Overload();  
        double hasil;  
        Objek.demoOverload(10);  
        Objek.demoOverload (10, 20);  
        hasil= Objek. demoOverload (5.5);  
        System.out.println("O/P : " + hasil);  
    }  
}
```

Output :

```
a: 10  
a and b: 10,20  
double a: 5.5  
O/P : 30.25
```

Pada contoh code diatas yaitu *method demoverloading ()* kelebihan beban 3 kali: metode pertama memiliki 1 parameter int, metode kedua memiliki 2 parameter int dan yang ketiga memiliki parameter ganda. Metode mana yang akan dipanggil ditentukan oleh argumen yang kita berikan saat memanggil method. Ini terjadi pada waktu kompilasi runtime sehingga jenis polimorfisme ini dikenal sebagai polimorfisme

3. Method overriding

Jika subkelas (*child class*) memiliki method yang sama dengan yang dideklarasikan di kelas induk (*Parent class*), ini dikenal sebagai metode overriding di Java.

Dengan kata lain, sebuah subclass hanya boleh mengimplementasi dari *method* yang telah dideklarasikan satu saja dari kelas induknya (parent class).

Penggunaan *Method overloading* aturannya adalah sebagai berikut :

- a. Method harus memiliki nama yang sama seperti di kelas induk (*Parent class*)
- b. Method harus memiliki parameter yang sama seperti di kelas induk.
- c. Harus ada hubungan IS-A (pewarisan).

Contoh program *method overriding* :

Hewan.java (parent class)

```
public class hewan {  
    public void suara(){  
        System.out.println("Hewan dapat mengeluarkan  
suara");  
    }  
}
```

Kucing.java (*child class*)

```
public class kucing extends hewan {  
    @Override  
    public void suara(){  
        System.out.println ("Suara kucing Meong");  
    }  
    public static void main(String[] args) {  
        // TODO code application logic here  
        hewan objek=new kucing();  
        objek.suara();  
    }  
}
```

```
}
```

Output

```
Suara Kucing Meong
```

Kuda.java (*child class*)

```
Kuda.java (child class )
public class kuda extends hewan{
@Override
    public void suara(){

        System.out.println ("Suara Kuda Meringkik");
    }

    public static void main(String[] args) {
        // TODO code application logic here

        hewan objek2=new kuda();
        objek2.suara();
    }
}
```

Output :

```
Suara Kuda Meringkik
```

4. Perbedaan method overloading dan method overriding

Tabel 13. 1 Perbedaan method overloading dan method overriding

No	<i>Method Overloading</i>	<i>Method Overriding</i>
1	Metode overloading digunakan untuk meningkatkan kemudahan dalam membaca program	Metode overriding digunakan untuk menyediakan implementasi spesifik dari method yang sudah disediakan oleh <i>super class</i> atau parent class
2	Metode overloading dilakukan di dalam class.	Method overriding terjadi pada dua <i>class</i> yang memiliki hubungan IS-A (<i>inheritance</i>).
3	Dalam kasus metode overloading, parameter nya harus berbeda.	Dalam kasus penggantian method, parameter harus sama.
4	Metode overloading adalah contoh polimorfisme pada saat di kompilasi.	Metode overriding adalah contoh polimorfisme saat run time.
5	Metode overloading tidak dapat dilakukan dengan mengubah tipe pengembalian method saja. Jenis pengembalian bisa sama atau berbeda tetapi kita harus mengubah parameternya.	Jenis pengembalian harus sama atau kovarian dalam penggantian method.

5. Beberapa Contoh lain Method Overloading dan Method Overriding

a. Method Overloading

```
class CalculatorSederhana
{
    int tambah(int a, int b)
    {
        return a+b;
    }
    int  tambah(int a, int b, int c)
    {
        return a+b+c;
    }
}
```

```
public class calculatorDemo
{
    public static void main(String args[])
    {
        CalculatorSederhana objek = new
        CalculatorSederhana();
        System.out.println(objek.add(10, 20));
        System.out.println(objek.add(10, 20, 30));
    }
}
```

Output :

30

60

b. Method Overriding

```
class BCD{  
    public void MethodKu(){  
        System.out.println("Ini Overriding  
Method");  
    }  
}
```

```
public class XYZ extends BCD{  
  
    public void MethodKu(){  
        System.out.println("Ini Overriding  
Method");  
    }  
  
    public static void main(String args[]){  
        BCD objek = new XYZ();  
        objek.MethodKu();  
    }  
}
```

Output

```
Ini Overriden Method
```

C. LATIHAN / TUGAS

1. Modifikasi Program Kalkulator sederhana pada B.4.1 *Method Overloading*
Tambahkan *method* pengurangan, pembagian, perkalian dengan 2 parameter
(20,10) pada class induk nya :

Hasil yang diharapkan :

Output :

Hasil Pertambahan : 30
Hasil Pengurangan : -10
Hasil Perkalian : 200
Hasil Pembagian : 2

2. Buat program mencari luas segitiga dimana attributnya alas, tinggi, luas
Rumus luas segitiga adalah $\text{luas} = (\text{alas} \times \text{tinggi}) / 2$
Gunakan konsep dengan method overloading atau overriding yang kalian ketahui

D. REFERENSI

- Horstmann Cay S., (2011). *Big Java 4th Edition*, san jose university , united state Of America. RRD jefferson city publishing.
- Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.
- Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.
- Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press
- Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.
- Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

PERTEMUAN 14

ABSTRACT CLASS INTERFACE

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami apa itu *abstract class* pada bahasa java oop
2. Mahasiswa memahami apa itu interface pada bahasa java oop
3. Mahasiswa dapat mengetahui aturan yang harus digunakan abstract class dan Interface
4. Mahasiswa dapat mempraktekan kapan penggunaan abstract class dan Interface

B. URAIAN MATERI

Abstract class adalah Sebuah class yang dideklarasikan menggunakan keyword "abstract". Dimana setidaknya mempunyai satu *abstract method* yang tidak mempunyai body (badan program) dan boleh mempunyai method yang non abstract. Dalam pertemuan ini kita akan belajar apa itu abstract class, mengapa kita menggunakannya dan aturan yang harus kita gunakan saat bekerja dengan bahasa program Java.

Contoh dibawah ini adalah abstract class memiliki abstract method yang tidak memiliki body

```
abstract class hewan{  
  
    //abstract method no body  
  
    public abstract void suara();  
  
}
```

Contoh dibawah ini adalah abstract class memiliki method non abstract atau memiliki *body*

```
Abstract class hewan{  
  
    // method  
  
    abstract void sound();  
  
    void eat(){  
  
        System.out.println ("Kucing ")  
  
    }  
  
}
```

1. Mengapa menggunakan *abstract class*

Katakanlah kita memiliki class Hewan yang memiliki method suara() dan subclassnya (lihat pertemuan inheritance) seperti Kuda, Kucing, burung dll. Karena suara hewan berbeda dari satu hewan ke hewan lainnya, tidak ada gunanya kalau kita menerapkan metode *abstract* ini di parent class. karena setiap *child class* harus mengganti method suara untuk memberikan detail implementasinya sendiri, seperti kelas kuda akan bersuara "meringik" , kelas kucing akan bersuara "meong".

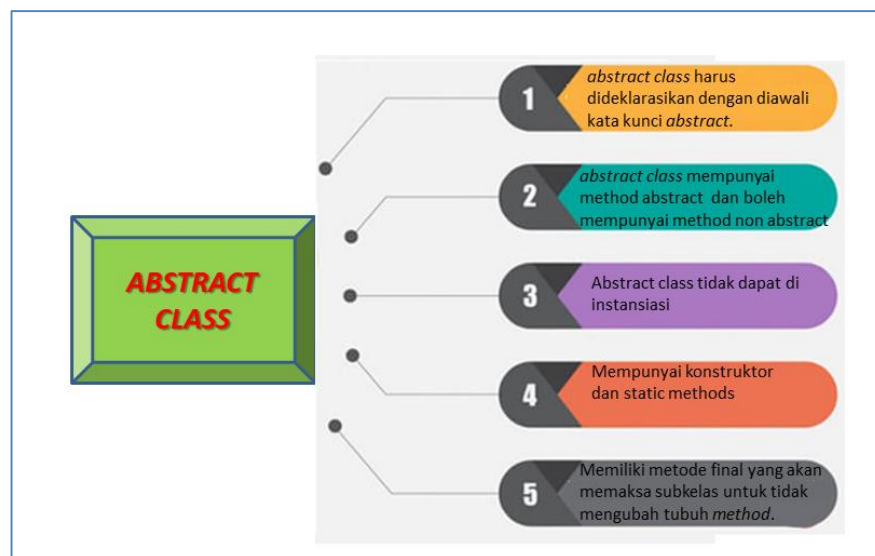
Jadi ketika kita tahu bahwa semua child class hewan harus menimpa *method* ini, method *abstract* akan menjadi pilihan yang baik karena dengan membuat metode *abstract*, memaksa semua sub class untuk mengimplementasikan metode ini (jika tidak, kita akan mendapatkan kesalahan kompilasi),

Karena class Hewan memiliki method *abstract* suara(), kita harus mendeklarasikan dengan keyword *abstract* pada class ini.

Sekarang setiap hewan harus memiliki suara, dengan membuat *method abstract*, mewajibkan child class untuk memberikan detail implementasi metode ini. Dengan cara ini kita pastikan bahwa setiap hewan memiliki suara.

2. Aturan penggunaan abstract class pada bahasa java

- abstract class* harus dideklarasikan dengan diawali kata kunci *abstract*.
- abstract class* setidaknya harus mempunyai method abstract (tidak mempunyai body) dan boleh mempunyai method non abstract (mempunyai body atau method konkrit)
- Abstract class tidak dapat di instansiasi
- Dapat mempunyai konstruktor dan *_static methods* .
- Memiliki metode final yang akan memaksa subkelas / *child class* untuk tidak mengubah tubuh *method*.



Gambar 14. 1 aturan penggunaan abstract class

3. Bentuk Umum pendeklarasian abstract class

```
//Deklarasi menggunakan keyword abstract  
abstract class A{  
  
    //dibawah ini abstract method  
    abstract void MethodAbstract ();  
  
    //dibawah ini method konkrit dengan body  
    void MethodNonAbstract(){
```

```
        //isi body  
    }  
}
```

Kelas abstrak menguraikan metode tetapi tidak harus mengimplementasikan semua metode.

Contoh program sederhana Abstract class dengan method abstract

```
abstract class abstrakhewan {  
    // method abstract  
    public abstract void suara();  
}
```

```
public class kucingabstrak extends hewan {  
  
    public void suara(){  
        System.out.println("Suara kucing mengeong");  
    }  
  
    public static void main(String[] args) {  
        // TODO code application logic here  
        hewan objek=new kucingabstrak();  
        objek.suara();  
    }  
  
}
```

Output :

Suara kucing mengeong

Contoh program sederhana Abstract class dengan method abstract dan method non abstract

```
abstract class abstrakhewan {  
  
    // method abstract  
  
    public abstract void suara();  
  
    // method non abstract  
  
    public void suara2(){  
  
        System.out.println(" ini method konkrit dari  
parent class");  
  
    }  
  
}
```

```
public class kucingabstrak extends abstrakhewan {  
  
    public void suara(){  
  
        System.out.println(" Suara kucing mengeong");  
  
    }  
  
    public static void main(String[] args) {  
  
        // TODO code application logic here  
  
        abstrakhewan objek=new kucingabstrak();  
  
        objek.suara();  
  
        objek.suara2();  
  
    }  
  
}
```

Output :

```
Suara kucing mengeong  
ini method konkrit dari parent class
```

4. Interface pada java

Interface mirip seperti class tetapi itu bukan class. Interface memiliki method dan attribute atau variabel seperti class tetapi *method* yang dideklarasikan dalam *Interface* secara default *abstract* (hanya method tanpa isi atau body, seperti method *abstract*). variabel yang dideklarasikan dalam *interface* bersifat publik, statis & final secara default.

5. Mengapa menggunakan Interface

Seperti disebutkan di atas *interface* digunakan untuk abstraksi secara penuh. Karena method tidak memiliki isi atau body, metode tersebut harus diimplementasikan oleh class sebelum kita dapat mengaksesnya. Class yang mengimplementasikan *interface* harus mengimplementasikan semua metode *interface* itu. Juga, bahasa pemrograman java tidak memungkinkan kita untuk memperluas lebih dari satu class, Namun kita dapat mengimplementasikan lebih dari satu interface di class kita.

Ada 3 alasan mengapa menggunakan interface

- Dapat digunakan untuk mencapai abstraksi.
- Dengan interface , kita dapat mendukung fungsi pewarisan berganda.
- Dapat digunakan untuk mencapai Objek independen.

6. Bentuk Umum penulisan Interface

```
interface InterfaceKu  
{  
    /* Semua methods adalah public abstract by  
    default  
    * Seperti yang kita lihat method tidak mempunyai  
    body
```

```
*/  
  
public void method1();  
  
public void method2();  
  
}
```

Contoh program menggunakan *Interface*

Ini adalah bagaimana sebuah class mengimplementasikan sebuah *Interface*. class harus menyediakan body dari semua method yang dideklarasikan dalam interface atau dengan kata lain bahwa class harus mengimplementasikan semua method *interface*.

```
public interface printable {  
  
    void method1();  
  
}
```

```
public class demo1 implements printable{  
public void method1(){  
  
    System.out.println("Hello apa kabar ini interface  
");  
}  
  
/**  
 * @param args the command line arguments  
 */  
  
public static void main(String[] args) {  
    // TODO code application logic here  
    printable objek=new demo1();  
    objek.method1();  
}  
  
}
```


Output :

```
Hello apa kabar ini interface
```

Contoh Interface dengan 2 *method*

```
public interface Interfaceku {  
    public void method1();  
    public void method2();  
}  
  
. public class demoInterface implements Interfaceku  
{  
    public void method1(){  
        System.out.println("implementasi      dari  
method1");  
    }  
  
    public void method2(){  
        System.out.println("implementasi      dari  
method2");  
    }  
  
    /**  
     * @param args the command line arguments  
     */  
  
    public static void main(String[] args) {  
        // TODO code application logic here  
        Interfaceku objeknya=new demoInterface();  
        objeknya.method1();  
    }  
}
```

```

        objeknya.method2();
    }
}

```

Output :

```

implementasi dari method1
implementasi dari method2

```

7. Perbedaan antara Abstract class dan interface pada Bahasa Java

Kita telah membahas abstract class dan interface. Berikut ini adalah tabel perbedaan antara *abstract class* dan *interface* :

Tabel 14. 1 Perbedaan Abstract dan Interface

No	<i>Abstract class</i>	<i>Interface</i>
1	abstract class hanya dapat meng extend satu class atau satu class abstract pada satu waktu	<i>Interface</i> dapat meng <i>extend</i> sejumlah interface pada satu waktu
2	abstract class dapat meng extend method abstract dan method non abstract	<i>Interface</i> hanya dapat mengextend <i>abstract method</i>
3.	Pada <i>abstract class</i> wajib menggunakan keyword "abstract" pada saat dideklarasikan method sebagai abstrak	Pada interface penggunaan keyword "abstract" adalah opsional saat dideklarasikan method tersebut sebagai abstrak
5	<i>Abstract class</i> dapat memiliki method abstract sebagai protect dan public	Interface hanya dapat memiliki method abstrak public
6	<i>abstract class</i> dapat mempunyai variable static, final atau <i>static final</i>	interface hanya dapat mempunyai variable public static final (constant)

1. Contoh program untuk perbedaan antara abstract class dan interface

abstract class hanya dapat meng extend satu class atau satu class abstract pada satu waktu

```
class Example1{
    public void display1(){
        System.out.println("display1 method");
    }
}
abstract class Example2{
    public void display2(){
        System.out.println("display2 method");
    }
}
abstract class Example3 extends Example1{
    abstract void display3();
}
```

```
class Example4 extends Example3{
    public void display3(){
        System.out.println("display3 method");
    }
}
class Demo{
    public static void main(String args[]){
        Example4 obj=new Example4();
        obj.display3();
    }
}
```

Output:

```
display3 method
```

Interface dapat meng *extend* sejumlah interface pada satu waktu

```
//first interface
interface Example1{
    public void display1();
}

//second interface
interface Example2 {
    public void display2();
}

//This interface is extending both the above interfaces
interface Example3 extends Example1,Example2{
}
```

```
class Example4 implements Example3{
    public void display1(){
        System.out.println("display2 method");
    }
    public void display2(){
        System.out.println("display3 method");
    }
}
```

```
class Demo{
    public static void main(String args[]){
        Example4 obj=new Example4();
        obj.display1();
    }
}
```

Output:

```
display2 method
```

C. LATIHAN/TUGAS

1. Kembangkan contoh program Abstract Class dibawah ini, *method abstract* suara(), method non abstract suara2(). Pada *parent class*

```
abstract class abstrakhewan {  
    // method abstract  
    public abstract void suara();  
    public void suara2(){  
        System.out.println(" ini method konkrit dari  
parent class");  
    }  
}
```

2. Buat child class **burung.java** meng extend dari abstrakhewan()

Buat objek untuk memanggil method abstract nya.

Output yang diharapkan adalah :

```
Suara Burung mencicit
```

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training ,
<https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember
2021

PERTEMUAN 15

ENCAPSULATION, PACKAGE

A. TUJUAN PEMBELAJARAN

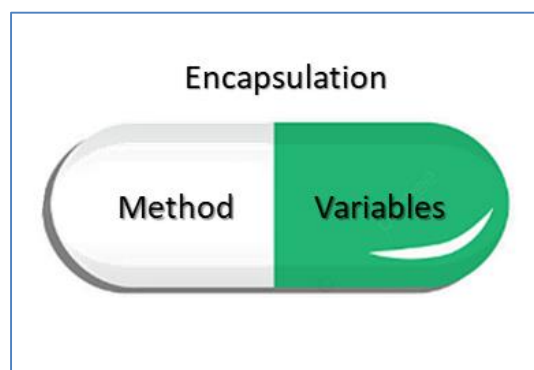
1. Mahasiswa memahami apa itu *encapsulation*, *package*
2. Mahasiswa memahami kelebihan dari *encapsulation*
3. Mahasiswa dapat mengetahui cara membuat *package*
4. Mahasiswa dapat mempraktekan *encapsulation* dan *package*

B. URAIAN MATERI

Encapsulation atau pengkapsulan adalah bagian dari konsep OOP selain dari *Inheritance*, *polymorphism* dan *abstract*, konsep ini mengikat atau membungkus sebuah objek state (*field*) dan *behaviour* (*method*) bersama-sama. Semua state (*field*) kita set atau dibungkus dengan enkapsulasi dan dibuat *private* agar tidak bisa di akses oleh *class* lain.

Pada Bahasa Java, *encapsulation* di implementasi melalui *access modifiers* - *public*, *private*, and *protected*.

Enkapsulasi pada contoh dunia nyata seperti Kapsul, yang dibungkus dengan obat-obatan yang berbeda. semua obat dikemas di dalam sebuah kapsul.



Gambar 15. 1 Ilustrasi Encapsulation

1. Cara mengimplementasikan enkapsulasi pada java :

- a. Jadikan *instance* variabel *private* sehingga tidak dapat diakses langsung dari luar *class*. Kita hanya dapat mengatur dan mendapatkan nilai dari variabel-variabel ini melalui *method class*.

- b. Memiliki method *getter* dan *setter* di class untuk mengatur dan mendapatkan nilai *fields*.

Contoh program encapsulation pada java

```
public class enkapsulasiDemo {  
  
    private int IDPegawai;  
    private String NamaPegawai;  
    private int Usia;  
  
    //Getter and Setter methods  
    public int getIdPegawai(){  
        return IDPegawai;  
    }  
    public String getNamaPegawai(){  
        return NamaPegawai;  
    }  
  
    public int  GetUsia(){  
        return Usia;  
    }  
  
    public void setIdPegawai(int newValue){  
        IDPegawai=newValue;  
    }  
  
    public void setNamaPegawai(String newValue){  
        NamaPegawai=newValue;  
    }  
}
```



```
public void setUsia(int newValue){  
    Usia=newValue;  
}  
}
```

```
public class EncapsulasiTesting {  
    public static void main(String[] args) {  
        // TODO code application logic here  
        enkapsulasiDemo objek=new enkapsulasiDemo();  
        objek.setIdPegawai(11111);  
        objek.setNamaPegawai("Agustav");  
        objek.setUsia(30);  
  
        System.out.println("Id    Pegawai    :    "    +  
objek.getIdPegawai());  
        System.out.println("Nama    Pegawai    :    "    +  
objek.getNamaPegawai());  
        System.out.println("Usia        :        "        +  
objek.GetUsia());  
  
    }  
}
```

Output :

```
Id Pegawai : 11111  
Nama Pegawai : Agustav  
Usia : 30
```

Pada contoh di atas, ketiga anggota data (atau field) bersifat *private* (lihat: *Access modifier* di java) yang tidak dapat diakses secara langsung. Field ini hanya dapat diakses melalui metode *public*. Fields `NamaPegawai`, `Idpegawai` dan `Usia` dibuat menjadi *hidden data field* menggunakan teknik enkapsulasi OOP.

2. Keuntungan menggunakan enkapsulasi:

Encapsulation meningkatkan pemeliharaan, fleksibilitas dan re-usability (penggunaan kembali) contoh implementasi dalam kode di atas dari `void setNamePegawai(String NamaPegawai)` dan `String getNamePegawai()` dapat diubah kapan saja. Karena implementasinya murni tersembunyi untuk diluar *class*, mereka masih akan mengakses field *private* `NamaPegawai` menggunakan metode yang sama `setNamePegawai(String NamaPegawai)` dan `getNamePegawai()`. Oleh karena itu kode dapat dipertahankan kapan saja tanpa merusak *class* yang menggunakan kode tersebut. Ini meningkatkan kegunaan kembali *class* yang mendasarinya.

Field dibuat hanya dapat dibaca (Jika kita tidak mendefinisikan metode *setter* di *class*) atau hanya tulis (Jika kita tidak mendefinisikan metode *getter* di *class*). Untuk itu Jika kami memiliki field (atau variabel) yang tidak ingin di ubah, jadi cukup mendefinisikan variabel sebagai *private* dan alih-alih men *setter* dan *getter* keduanya, kita hanya perlu mendefinisikan metode *get* untuk variabel itu. Karena metode *set* tidak ada, tidak mungkin *class* luar dapat mengubah nilai field tersebut.

Enkapsulasi juga dikenal sebagai "Menyembunyikan data"

Dengan hanya menyediakan method *setter* atau *getter*, kita dapat membuat *class* hanya *read(baca)* atau *write(tulis)* saja. Dengan kata lain, kita dapat melewati method *Get* atau *Set*.

Misalkan kita ingin menetapkan nilai `idPegawai` 11111 saja, kita dapat menulis logika di dalam method *setter*.

Ini adalah cara untuk menyembunyikan data di Java karena *class* lain tidak akan dapat mengakses data melalui anggota *private*.

Contoh 2 program encapsulation pada java

Mari kita lihat contoh enkapsulasi lain yang hanya memiliki empat field dengan method setter dan getter.

```
public class account {  
    private long no_acc;  
    private String nama,email;  
    private float jumlah;  
    //public getter and setter methods  
    public long getAcc_no() {  
        return no_acc;  
    }  
    public void setAcc_no(long no_acc) {  
        this.no_acc = no_acc;  
    }  
    public String getName() {  
        return nama;  
    }  
    public void setName(String nama) {  
        this.nama = nama;  
    }  
    public String getEmail() {  
        return email;  
    }  
    public void setEmail(String email) {  
        this.email = email;  
    }  
}
```

```
public float getAmount() {  
    return jumlah;  
}  
  
public void setJumlah(float jumlah) {  
    this.jumlah = jumlah;  
}  
  
}
```

```
public class EnkapsulasiTest2 {  
  
    /**  
     * @param args the command line arguments  
     */  
  
    public static void main(String[] args) {  
        // TODO code application logic here  
        account objek=new account();  
        objek.setAcc_no(070123);  
        objek.setName("Agustav");  
        objek.setEmail("agustav@gmail.com");  
        objek.setJumlah(50000);  
  
        System.out.print(objek.getAcc_no()+"  
"+objek.getName()+" "+objek.getEmail()+"  
"+objek.getAmount());  
  
    }  
  
}
```

Output :

```
1070123   Agustav   agustav@gmail.com   50000.0
```

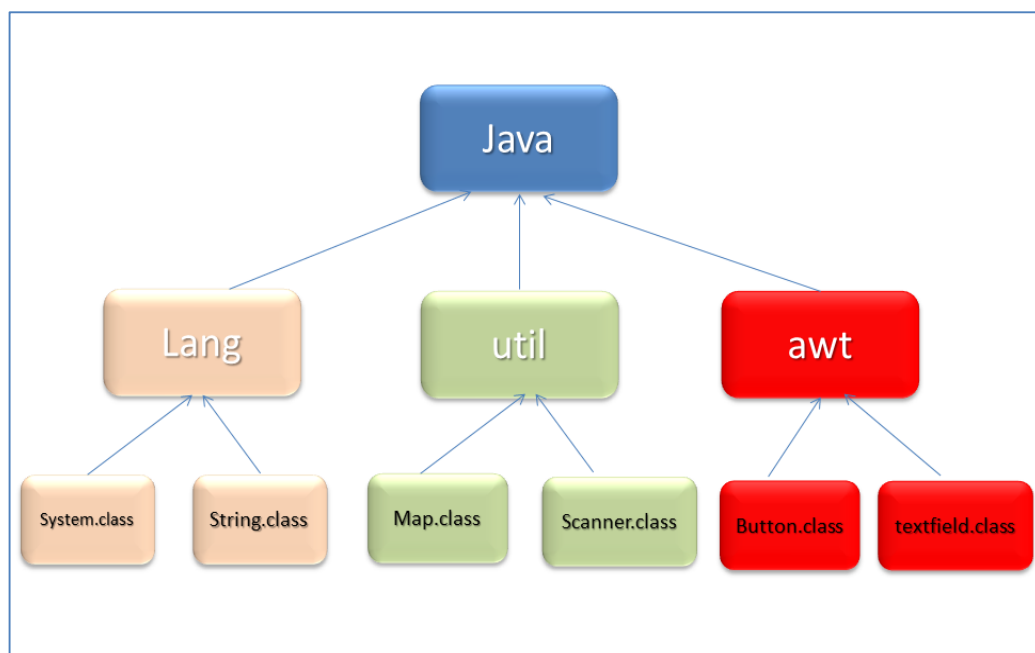
3. Package pada Java

Package seperti namanya adalah paket (grup) dari sebuah *class*, *interface*, dan paket lainnya. Pada java kita menggunakan *Package* untuk mengatur *class* dan *interface*. *Package* memiliki dua jenis yaitu *Built-int Package* atau bawaan dan *Package* dapat kita buat (juga dikenal sebagai *package user define*). Dalam pertemuan ini kita akan mempelajari apa itu *package*, apa itu *package* yang ditentukan pengguna dan bagaimana menggunakannya.

Pada java kita memiliki beberapa *package* bawaan, misalnya ketika kita membutuhkan input pengguna, kita mengimpor *package* seperti ini:

```
import java.util.Scanner
```

- **java** adalah *top level package*
- **util** adalah *sub package*
- dan **Scanner** adalah *class* yang ada pada *sub package util*.



Gambar 15. 2 ilustrasi *package java*

Sebelum kita mempelajari cara membuat package yang ditentukan pengguna */user define*, mari kita lihat keuntungan menggunakan package.

4. Keuntungan menggunakan Package

Ini adalah alasan mengapa kita harus menggunakan paket di Java:

- a. Re-usability (Dapat digunakan kembali): Saat mengembangkan proyek di java, kita sering merasa bahwa ada kode program yang kita tulis berulang kali. Dengan menggunakan package, kita dapat membuat hal-hal seperti itu dalam bentuk class di dalam package dan kapan pun kita perlu melakukan tugas yang sama, cukup mengimpor package dan menggunakan class tersebut.
- b. Organization (Pengorganisasian yang lebih baik): Dalam project java yang besar di mana kita memiliki beberapa ratus class, selalu diperlukan untuk mengelompokkan jenis kelas yang serupa dalam nama package yang bermakna sehingga kita dapat mengatur project dengan lebih baik dan ketika kita membutuhkannya salah satu class, kita dapat dengan cepat menemukan dan menggunakannya, sehingga meningkatkan efisiensi.
- c. Menghindari Konflik Nama: Kita dapat mendefinisikan dua class dengan nama yang sama dalam paket yang berbeda sehingga untuk menghindari tabrakan nama, kita dapat menggunakan package.

5. Jenis jenis Package

Seperti yang disebutkan di awal pembahasan yaitu package memiliki dua jenis pada java.

- a. Package yang ditentukan pengguna: Package yang kita buat disebut package yang ditentukan pengguna (User Defined).
- b. Package bawaan: Package yang sudah ditentukan seperti `java.io.*`, `java.lang.*` dll dikenal sebagai Package bawaan.

Contoh 1. program menggunakan package:

```
. package kalkulatorpack;

/**
 *
 * @author feiruz
 */
public class kalkulator {

    public int pertambahan(int a, int b){

        return a+b;

    }

    public static void main(String args[]){

        kalkulator obj = new kalkulator();

        System.out.println(obj.tambah(10, 20));

    }

}
```

Kita telah membuat *class* Kalkulator di dalam nama package **kalkulatorpack**. Untuk membuat kelas di dalam sebuah package, nyatakan nama paket dalam pernyataan pertama dalam program Anda. Sebuah kelas hanya dapat memiliki satu deklarasi package.

File kalkulator.java dibuat di dalam package **kalkulatorpack**

Sekarang mari kita lihat bagaimana menggunakan package ini di program lain.

```
import kalkulatorpack.kalkulator;

public class kalkulatorTambah {

    /**
     * @param args the command line arguments
     */
}
```

```
*/  
  
public static void main(String[] args) {  
    // TODO code application logic here  
    kalkulator objek=new kalkulator();  
    System.out.println(objek.pertambahan(10,30));  
  
}  
  
}
```

Output :

40

Dalam contoh program di atas terlihat Untuk menggunakan *class* Kalkulator, kita impor package **kalkulatorpack**. import kalkulatorpack.kalkulator, ini hanya mengimpor kelas kalkulator. Namun jika kita memiliki beberapa class di dalam package **kalkulatorpack** maka kita dapat mengimpor package menggunakan semua class dari package dengan menulisi seperti dibawah ini.

```
import kalkulatorpack.*
```

C. LATIHAN / TUGAS

1. Modifikasi contoh 1 *package* **kalkulatorpack** dengan menambahkan pada class kalkulator method methodnya yaitu :
 - a. Method pengurangan (int a,int b)
 - b. Method pembagian (float a,float b)
 - c. Method perkalian (float a,float b)
 - d. Berikan masing masing parameternya
2. Buat class untuk masing masing nya :dan import package **kalkulatorpack** yang telah dibuat

- a. KalkulatorKurang.java
 - b. KalkulatorBagi.java
 - c. KalkulatorKali.java
3. Tampilkan hasil output nya

D. REFERENSI

- Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. *RRD jefferson city publishing*.
- Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, *publishing as prentice hall*.
- Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.
- Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press
- Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.
- Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

PERTEMUAN 16

ACCESS MODIFIER

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami apa itu *access modifier*
2. Mahasiswa dapat membedakan jenis jenis *access modifier*
3. Mahasiswa dapat mengetahui aturan membuat *access modifier*
4. Mahasiswa dapat mempraktekan penggunaan *access modifier*

B. URAIAN MATERI

Access Modifier di Java adalah menentukan aksesibilitas atau cakupan *field*, *method*, *constructor*, atau *class*. Kita dapat mengubah tingkat hak akses *field*, *method*, *constructor*, dan *class* dengan menerapkan pengubah akses di atasnya. Dengan adanya *Access Modifier* kita bisa membatasi hak akses objek, *method*, *field* ataupun turunannya.

1. Jenis jenis *access modifier*

Terdapat 4 jenis *access modifier* yaitu :

- a. *Default*
- b. *Private*
- c. *Protected*
- d. *public*

Untuk memahami cakupan *access modifier*, marilah kita lihat tabel dibawah ini :

Tabel 16. 1 cakupan access modifier

Access Modifier	Pada Class	Pada Package	Subclass (package yang sama)	Subclass (package yang berbeda)	Diluar class
Default	Y	Y	Y	N	N
Private	Y	N	N	N	N
Protected	Y	Y	Y	Y	N
Public	Y	Y	Y	Y	Y

Keterangan : Y = Dapat diakses , N= Tidak dapat di akses

2. Default access modifier

Ketika kita tidak menentukan *access modifier* apa pun, maka *access modifier* disebut *default*. Cakupan modifier ini terbatas pada package saja. Ini berarti bahwa jika kita memiliki *class* akses default dalam sebuah package, hanya *class* yang ada dalam package ini yang dapat mengakses *class* nya. Tidak ada *class* lain di luar package yang dapat mengakses nya. Demikian pula, jika kita memiliki method *default* atau data di *class*, tidak akan terlihat di *class* lain.

Contoh **default access modifier** dalam program java

Dalam contoh ini kita memiliki dua kelas, class **Test** mencoba untuk mengakses metode default class **Penambahan**, karena kelas **Test** berbeda package, program ini akan menimbulkan kesalahan kompilasi, karena ruang lingkup modifier default terbatas pada package yang sama di yang dideklarasikan.

pertambahan.java

```
package package1;

public class pertambahan{

    /* Ketika tidak menentukan akses apapun
```

```
    * ini menjadi akses default.  
    */  
  
    int tambah2angka(int a, int b){  
        return a+b;  
    }  
}
```

Test.java

```
package package2;  
  
/* Kita akan meng import package1  
 * tapi akan terlihat error karena  
 * class mempunyai akses default  
 * modifier.  
 */  
  
import package2.*;  
  
public class Test {  
    public static void main(String args[]){  
        Pertambahan obj = new Pertambahan();  
        /* It will throw error because we are trying  
to access  
        * the default method in another package  
        */  
        obj.tambah2angka(10, 21);  
    }  
}
```

Output :

```
Exception in thread "main" java.lang.Error:
Unresolved compilation problem: The method
addTwoNumbers(int, int) from the type Addition is not
visible at xyzpackage.Test.main(Test.java:12)
```

3. Private access modifier

Ruang lingkup modifier **Private** terbatas pada class saja. data dan method private hanya dapat diakses di dalam class. Class dan interface tidak dapat dideklarasikan sebagai private. Jika suatu class memiliki constructor private maka kita tidak dapat membuat objek class itu dari luar class.

Untuk memahaminya mari kita lihat contoh berikut dibawah ini :

Contoh ini memunculkan kesalahan kompilasi karena kita mencoba mengakses data private dan method class **ABC** di kelas **Contoh**. Data dan *method* private hanya dapat diakses di dalam class.

Contoh *Private access modifier* dalam program java

```
class ABC{

    private double angka= 100;

    private int persegipanjang(int a){

        return a*a;

    }

}

public class Contoh{

    public static void main(String args[]){

        ABC obj = new ABC();

        System.out.println(obj.angka);

        System.out.println(obj.persegipanjang(10));

    }

}
```

Output:

```
Compile - time error
```

4. Protected access modifier

Data dan method yang di Protected hanya dapat diakses oleh class class dari pacakage yang sama dan sub class yang ada dalam pacakage apa pun. Kita juga dapat mengatakan bahwa pengubah akses yang di protected mirip dengan *access modifier* default dengan satu pengecualian bahwa ia memiliki visibilitas di sub class.

Class yang tidak dapat dinyatakan terlindungi, modifier access ini umumnya digunakan dalam hubungan *parent & child*.

Contoh 1 program Protected access modifier

Dalam contoh ini, class **Test** yang ada dalam package lain dapat memanggil method `tambah2angka ()`, yang dideklarasikan *protected*. Ini karena kelas Test meng extend class **Penambahan.java** dan modifier yang di protected memungkinkan akses anggota yang di protected dalam sub class (package apa pun).

Pertambahan.java

```
package package1;

public class pertambahan{

    protected int tambah2angka (int a, int b){

        return a+b;

    }

}
```

Test.java

```
package package2;

import package1.*;
```

```
class Test extends pertambahan{

    public static void main(String args[]){

        Test obj = new Test();

        System.out.println(obj.tambah2angka(11, 22));

    }

}
```

Output :

33

Contoh 2 program *Protected access modifier*

Contoh 2 Protect.java

```
package pack;

public class Contoh2Protect{

    protected void msg(){

        System.out.println("Hello ini contoh protected");

    }

}

package mypack;

import pack.*;

class Test2 extends Contoh2Protect{

    public static void main(String args[]){

        Test2 obj = new Test2();

        obj.msg();

    }

}
```

Output :

```
Hello ini contoh protected
```

Contoh 2 private Access modifier**A.java**

```
class A{  
  
private A() {} //private constructor  
  
void msg() {System.out.println("Hello java");}  
}
```

Simple.java

```
public class Simple{  
  
public static void main(String args[]){  
  
A obj=new A(); //Compile Time Error  
  
}  
}
```

Catatan : class A tidak dapat menjadi private atau diprotect kecuali nested class.

5. Public access modifier

Anggota metode dan class yang dideklarasikan publik dapat diakses dari mana saja. Modifier ini tidak membatasi akses.

contoh program modifier access public pada java

Mari kita ambil contoh yang sama yang telah kita lihat di atas tetapi kali ini metode tambah2angka() memiliki modifier public dan class **Test** dapat mengakses method ini bahkan tanpa meng extend class Pertambahan. Ini karena modifier public memiliki visibilitas di mana-mana.

pertambahan.java


```
package package1;

public class Pertambahan{

    public int tambah2angka(int a, int b){

        return a+b;

    }

}
```

Test.java

```
package package2;

import package1.*;

class Test{

    public static void main(String args[]){

        Addition obj = new Addition();

        System.out.println(obj.addTwoNumbers(100, 1));

    }

}
```

Output :

```
101
```

contoh 2 program *modifier access public* pada java

public2.java

```
package pack;

public class public2{

    public void msg(){
```

```
        System.out.println("Hello ini contoh 2 public  
");  
    }  
}
```

```
package mypack;  
  
import pack.*;  
  
class Test2{  
    public static void main(String args[]){  
        public2 obj = new public2();  
        obj.msg();  
    }  
}
```

Output :

```
Hello ini contoh 2 public
```

6. Access Modifiers dengan Method Overriding

Jika kita mengganti method apa pun, method yang diganti (yaitu yang dideklarasikan dalam sub class) tidak boleh melebihi batas.

contoh program *modifier access* dengan method overriding

```
class A{  
    protected void msg(){  
        System.out.println("Hello java");  
    }  
}
```

```
public class Simple extends A{  
    void msg(){  
        System.out.println("Hello java");  
    }//C.T.Error  
  
    public static void main(String args[]){  
        Simple obj=new Simple();  
        obj.msg();  
    }  
}
```

C. LATIHAN/TUGAS

1. Buatlah aplikasi operasi aritmatika penjumlahan, pengurangan, perkalian dan pembagian dari dua bilangan A dan B, menggunakan konsep abstract data dan method akses nya adalah di tentukan *protected* Nilai parameter bilangan A = 9.5 dan parameter bilangan B = 2.5
2. Buatlah aplikasi operasi aritmatika penjumlahan, pengurangan, perkalian dan pembagian dari dua bilangan A dan B, menggunakan konsep *polymorphism*, data dan method akses nya adalah di tentukan *protected* Nilai parameter bilangan A = 9.5 dan parameter bilangan B = 2.5

D. REFERENSI

- Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.
- Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.
- Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.
- Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

PERTEMUAN 17

ARRAY / LARIK

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami konsep dari Array
2. Mahasiswa dapat membedakan jenis array satu dimensi dan dua dimensi
3. Mahasiswa dapat mengerti fitur pada array dan mendeklarasikan array
4. Mahasiswa dapat mempraktekkan penggunaan *array* satu dimensi dan multi dimensi

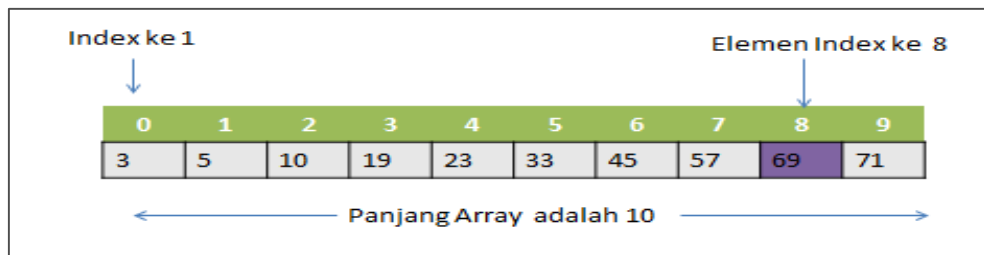
B. URAIAN MATERI

Array adalah objek yang menyimpan banyak variabel dengan tipe data yang sama. Data dapat menampung tipe data primitif serta *reference* objek. Array pada Java disimpan menggunakan indeks, elemen pertama array disimpan pada indeks ke-0, elemen ke-2 disimpan pada indeks ke-1 dan seterusnya.

Sebenarnya sebagian besar tipe koleksi di Java yang merupakan bagian dari paket `java.util` menggunakan array secara internal dalam fungsinya. Karena Array adalah objek, mereka dibuat selama runtime. Panjang array adalah tetap.

Fitur Array

1. Array adalah objek
2. Array bahkan dapat menyimpan variabel *reference* dari objek lain
3. Array dibuat selama runtime
4. Array dinamis, dibuat di heap
5. Panjang Array adalah tetap



Gambar 17. 1 1 Ilustrasi Objek array

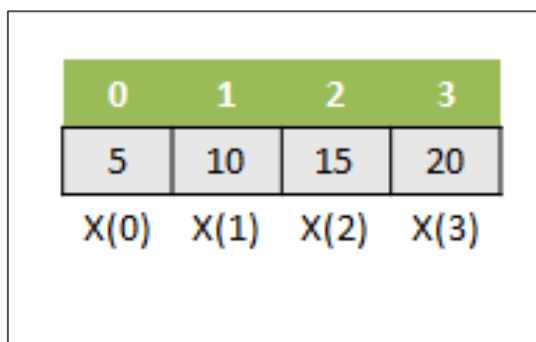
1. Jenis jenis Array

Berdasarkan jenis *Array* dapat dibedakan terdiri dari :

- Array satu dimensi
- Array dua dimensi

Array satu dimensi (atau array dimensi tunggal) adalah jenis array linier. Untuk mengakses elemen-elemennya melibatkan satu subskrip dapat mewakili indeks baris atau kolom.

- Array dapat berupa variabel atau berupa konstanta.
- Array dapat berupa array dari tipe data primitif int, long, float, String atau bahkan objek
- Definisi array 1 dimensi misalnya :
`int [] X = { 5,10,15,20 } ;`
 adalah mendefinisikan array 1 dimensi dari int yang berelemen data 5,10,15,20, ilustrasi gambar array nya adalah sebagai berikut :



Gambar 17. 2 Definisi Array 1 Dimensi

2. Deklarasi Array satu dimensi :

Untuk mendeklarasikan Variabel array seperti kita mendeklarasikan variabel dengan tipe yang diinginkan, hanya untuk deklarasi variabel *array*

menambahkan [] setelah menentukan tipe datanya. Berikut adalah contoh deklarasi array Java sederhana:

```
int[] NamaArray;  
  
String[] stringArray;  
  
MyClass[] myClassArray;  
  
int    intArray[];  
  
String    stringArray[];  
  
MyClass    myClassArray[];
```

Pada Umumnya deklarasi array menempatkan tanda kurung siku [] setelah tipe data (mis. String []) ditentukan dan bukan setelah nama variabel. Karena lebih mudah untuk membaca kode ketika tanda kurung siku ditempatkan tepat setelah tipe data dalam deklarasi array.

3. Instansiasi / Membuat Object dan mengakses Array

Saat kita mendeklarasikan variabel array, kita hanya mendeklarasikan variabel (reference) ke array itu sendiri. Deklarasi sebenarnya tidak membuat array. Untuk membuat array penulisan nya seperti dibawah ini:

```
int[] NamaArray;  
  
NamaArray = new int[10];
```

Contoh diatas adalah membuat array dengan nama `NamaArray` bertipe data primitif `int` dengan ruang atau jumlah elemen 10 variabel di dalamnya.

Kita juga dapat membuat larik reference objek String. seperti dibawah ini :

```
String[] stringNamaArray = new String[10];
```

Jika kita ingin memberikan nilai variabel yang akan dimasukkan ke dalam array, kita dapat menggunakan literal array. Berikut adalah bagaimana tampilan literal array dalam kode Java:

```
int[] angka= { 1,2,3,4,5,6,7,8,9,10 };  
  
String[] huruf= {"satu", "dua", "tiga"};
```

Contoh sederhana program *array* pada java

KotaArray.java

```
public class KotaArray{  
    public static void main(String[] args){  
        String[] kota; //deklarasi variabel array  
  
        kota = new String[4];          // membuat objek  
        array  
  
        // mengisi elemen array  
        kota[0] = " Bandung";  
        kota[1] = " Semarang ";  
        kota[2] = " Surabaya ";  
        kota[3] = " Medan";  
  
        // menampilkan elemen array  
        System.out.println(kota[0]);  
        System.out.println(kota[1]);  
        System.out.println(kota[2]);  
        System.out.println(kota[3]);  
  
    }  
}
```

Output :

```
Bandung  
Semarang  
Surabaya
```


Medan

Contoh sederhana program dengan memberikan nilai literal *array* pada java

```
public class ArrayKota2{  
    public static void main(String[] args){  
        String[] kota={"Bandung","Semarang","Surabaya","Medan"  
" };  
  
        // menampilkan elemen array  
        System.out.println(kota[0]);  
        System.out.println(kota[3]);  
        System.out.println(kota[2]);  
        System.out.println(kota[1]);  
    }  
}
```

Output

```
Bandung  
Medan  
Surabaya  
Semarang
```

Jumlah Elemen *array* dapat diketahui menggunakan variable *instance length*

ArraykotaLength.java

```
public class arraykotalength {  
    public static void main(String[] args) {  
        // TODO code application logic here  
String[]    kota    =    {"Medan","Jakarta",    "Surabaya",  
"Semarang"};
```

```
// menampilkan elemen array
for(int i=0; i<kota.length; i++)
    System.out.println(kota[i]);
}
```

Output :

```
Medan
Jakarta
Surabaya
Semarang
```

Contoh sederhana program *array* dengan type data Integer

ArrayInt.java

```
class ArrayInt{
    public static void main(String[] args) {
        // declares an array of integers
        int[] intArray;

        // alokasi memori untuk 10 element type
integer
        intArray = new int[10];

        // initialize elemen pertama
        intArray[0] = 100;
        // initialize elemen Kedua
        intArray[1] = 200;
        // dan seterusnya
    }
}
```

```
intArray[2] = 300;
intArray[3] = 400;
intArray[4] = 500;
intArray[5] = 600;
anArray[6] = 700;
intArray[7] = 800;
intArray[8] = 900;
intArray[9] = 1000;

System.out.println("Elemen index 0: "
                    + intArray[0]);
System.out.println("Element index 1: "
                    + intArray[1]);
System.out.println("Element index 2: "
                    + intArray[2]);
System.out.println("Element index 3: "
                    + intArray[3]);
System.out.println("Element index 4: "
                    + intArray[4]);
System.out.println("Element index 5: "
                    + intArray[5]);
System.out.println("Element index 6: "
                    + intArray[6]);
System.out.println("Element index 7: "
                    + intArray[7]);
System.out.println("Element index 8: "
                    + intArray[8]);
```

```
        System.out.println("Element index 9: "
                            + intArray[ 9]);
    }
}
```

Output :

```
Element index 0: 100
Element index 1: 200
Element index 2: 300
Element index 3: 400
Element index 4: 500
Element index 5: 600
Element index 6: 700
Element index 7: 800
Element index 8: 900
Element index 9: 1000
```

4. Menyalin Array (Copying)

Class Sistem memiliki method *arraycopy* yang dapat kita gunakan untuk menyalin data secara efisien dari satu array ke array lain

Bentuk umum penulisan *copying array*

```
public static void arraycopy(Object src, int srcPos,
                             Object dest, int destPos, int length)
```

Dua argumen Objek menentukan array sumber yang akan disalin dan array sebagai target tujuan. Tiga argumen int menentukan posisi awal dalam array sumber, posisi awal dalam array tujuan, dan jumlah elemen array yang akan disalin.

Program berikut, mendeklarasikan array elemen String. Array ini menggunakan metode `System.arraycopy` untuk menyalin urutan komponen array ke dalam array kedua:

Arraycopy.java

```
class ArrayCopy{  
    public static void main(String[] args) {  
        String[] copyFrom = {  
            "Corretto", "Cortado", "Doppio",  
            "Affogato", "Americano", "Cappuccino", "Espresso",  
            "Frappuccino", "Freddo", "Lungo", "Macchiato", "Marocchino",  
            "Ristretto" };  
  
        String[] copyTo = new String[7];  
        System.arraycopy(copyFrom, 2, copyTo, 0, 7);  
        for (String coffee : copyTo) {  
            System.out.print(coffee + " ");  
        }  
    }  
}
```

Output :

```
Doppio Affogato Americano Cappuccino Espresso Frappuccino  
Freddo
```

5. Manipulasi Array

Array adalah merupakan konsep yang *powerfull* dan berguna dalam pemrograman. Java SE menyediakan metode untuk melakukan beberapa manipulasi yang paling umum terkait dengan *array*. Misalnya pada, contoh *ArrayCopy* menggunakan metode *arraycopy* dari kelas *System* alih-alih melakukan iterasi secara manual melalui elemen *array* sumber dan menempatkan masing-masing elemen ke dalam *array* tujuan. Ini dilakukan di belakang layar, sehingga memungkinkan pengembang hanya menggunakan hanya satu baris kode untuk memanggil metode.

Java SE menyediakan beberapa metode untuk melakukan manipulasi array (tugas umum, seperti menyalin, menyortir, dan mencari array) di kelas `java.util.Arrays`. Misalnya, contoh sebelumnya dapat dimodifikasi untuk menggunakan metode `copyOfRange` dari kelas `java.util.Arrays`, seperti yang Anda lihat dalam contoh `ArrayCopy`. Perbedaannya adalah menggunakan metode `copyOfRange` tidak mengharuskan kita membuat *array* tujuan sebelum memanggil metode tersebut, karena *array* tujuan dikembalikan oleh metode:

```
class ArrayCopyOfDemo {  
    public static void main(String[] args) {  
        String[] copyFrom = {  
            "Affogato",    "Americano",    "Cappuccino",    "Corretto",  
            "Cortado",     "Doppio",     "Espresso",     "Frappuccino",  
            "Freddo",     "Lungo",     "Macchiato",     "Marocchino",  
            "Ristretto" };  
  
        String[] copyTo = java.util.Arrays.copyOfRange(copyFrom,  
            2, 9);  
  
        for (String coffee : copyTo) {  
            System.out.print(coffee + " ");  
        }  
    }  
}
```

6. Array multidimensi

Array multidimensi atau *array* dua dimensi pada Java direpresentasikan sebagai array dari array satu dimensi dengan tipe data yang sama. Penggunaan Array multidimensi sebagian besar digunakan untuk mewakili nilai dari tabel dimana terdapat nilai baris dan kolom.

Array multidimensi adalah array dari array , dengan konsep pengaksesan `[noBaris][noKolom]`

Deklarasi array multidimensi :

```
dataType[][] arrayRefVariable; (or) dataType arrayRefVariable[][];
```

Pendeklarasian Terdapat 2 kurung buka dan kurung tutup bracket berpasangan setelah menentukan tipe datanya :

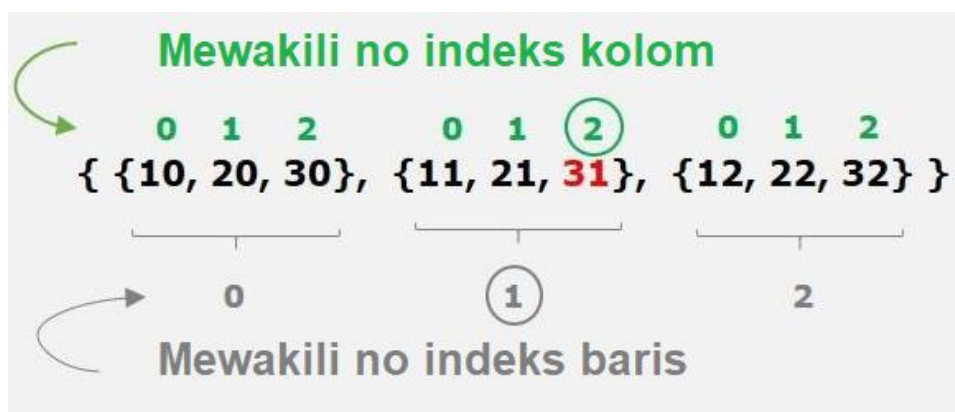
Contoh :

```
int[][] VarArray=new int[3][3]; //3 baris dan 3 column
int[][] VarArray= {{10, 20, 30},{11, 21, 31},{12, 22, 32}} }
```

Singkatnya, array dua dimensi berisi elemen array satu dimensi. Ini diwakili oleh dua indeks di mana indeks pertama menunjukkan posisi array baris dan indeks kedua mewakili posisi elemen kolom dalam array tersebut.

	Kolom 0	Kolom 1	Kolom 2
Baris 0	10	20	30
Baris 1	11	21	31
Baris 2	12	22	32

Gambar 17. 3 Array multi dimensi



Gambar 17. 4 Meng akses Array multi dimensi

VarArray [1][2] akan memberikan nilai **31**

7. Contoh program sederhana *Array* multidimensi

Contoh 1

```
public class Main
{
    public static void main(String[] args) {
        //2-d array initialised with values
        int[][] intArray = { { 1, 2 }, { 3, 4 }, {5,6}};
        //print the array
        System.out.println("Inisialisasi array multi
dimensi :");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print(intArray [i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Output

```
Inisialisasi array multi dimensi

1    2

3    4

5    6
```


Contoh 2

```
class ArrayMultidimensi {  
    public static void main(String[] args) {  
        // TODO code application logic here  
        String[][] nama = {"Mas ", "Pak ", "Bu " ,  
                            "Mbak "}, {"Agus ", "Iyus ", "Nabila "};  
        System.out.println(nama[0][0] + nama[1][0]);  
        System.out.println(nama[0][1] + nama[1][1]);  
        System.out.println(nama[0][2] + nama[1][2]);  
  
    }  
}
```

Output :

Mas Agus

Pak Iyus

Bu Nabila

C. LATIHAN /TUGAS

1. Modifikasi program Arraylength.java diatas , gunakan class scanner untuk menerima inputan dari keyboard

Contoh hasil run

```
Masukan Jumlah Kota : 4  
Kota ke 1 : Jakarta  
Kota ke 2 : Bandung  
Kota ke 3 : Surabaya  
Kota ke 4 : Makasar  
Kota-kota yang dimasukan  
Jakarta
```

```
Bandung
Surabaya
Makasar
```

2. Buat array multidimensi untuk nama negara dan ibukotanya, Akses array dan tampilkan di layar outputnya adalah sebagai berikut :

```
Ibukota Indonesia adalah Jakarta
Ibukota Jepang adalah Tokyo
Ibukota Iran adalah Teheran
Ibukota Malaysia adalah Kuala Lumpur
Ibukota Thailand adalah Bangkok
```

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

Wahono Satrio Romi, Java Fundamental, <https://romisatriawahono.net/java/>, diakses pada tanggal 5 Desember 2022

PERTEMUAN 18

EXCEPTION HANDLING

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami konsep dari *Exception Handling*
2. Mahasiswa dapat membedakan *Exception Handling* dan *Error*,
3. Mahasiswa dapat mengetahui Jenis jenis *Exception Handling*
4. Mahasiswa mengetahui *keyword Exception Handling*
5. Mahasiswa dapat mempraktekan penggunaan *Exception*

B. URAIAN MATERI

Exception Handling salah satu fitur terpenting dari pemrograman java yang memungkinkan kita untuk menangani kesalahan runtime yang disebabkan oleh *exception*. *Exception Handling* di Java adalah salah satu mekanisme yang ampuh sehingga program aplikasi dapat di normalkan dan dipertahankan.

Apa itu exception? adalah peristiwa yang tidak diinginkan yang mengganggu aliran normal program. Ketika exception terjadi, eksekusi program dihentikan. Dalam kasus seperti itu, kita mendapatkan pesan kesalahan yang dihasilkan sistem. Hal yang baik tentang *exception* adalah bahwa dapat ditangani (*handling*). Dengan *Exception Handling*, kita dapat memberikan pesan yang bermakna kepada pengguna tentang masalah tersebut daripada pesan yang dihasilkan sistem, yang mungkin tidak dapat dimengerti oleh pengguna.

Mengapa terjadi exception? ada beberapa alasan yang dapat menyebabkan program mengeluarkan exception. Misalnya: Membuka file yang tidak ada di program kita, kesalahan input yang diberikan oleh pengguna, mengakses array yang melebihi batas indeks dll.

1. Kelebihan menggunakan Exception handling

Jika terjadi *Exception*, yang belum ditangani oleh programmer maka eksekusi program akan dihentikan dan pesan kesalahan yang dihasilkan sistem akan ditampilkan kepada *user*. Contoh *exception* yang dihasilkan sistem di bawah ini:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero at ExceptionDemo.ma
ExceptionDemo : The class name
main : The method name
ExceptionDemo.java : The filename
java:5 : Line number
```

Gambar 17. 5 Contoh pesan *Excetion*

Pesan ini tidak di mengerti oleh user sehingga users tidak akan dapat memahami apa yang salah. Untuk memberi tahu kesalahan pada contoh program sederhana ini, kita dapat melakukan *exception handling*. kita menangani kondisi seperti itu dan kemudian menampilkan pesan peringatan kepada pengguna, yang memungkinkan mereka memperbaiki kesalahan karena sebagian besar waktu exception terjadi karena data yang salah yang diberikan oleh pengguna.

Exception handling memastikan bahwa aliran program tidak terputus saat ekspresi terjadi. Misalnya, jika sebuah program memiliki banyak pernyataan dan eksepsi terjadi di tengah jalan setelah menjalankan pernyataan tertentu, maka pernyataan setelah eksepsi tidak akan dijalankan dan program akan berhenti secara tiba-tiba.

Dengan exception kita pastikan bahwa semua pernyataan dijalankan dan aliran program tidak terputus.

2. Perbedaan error dan exception

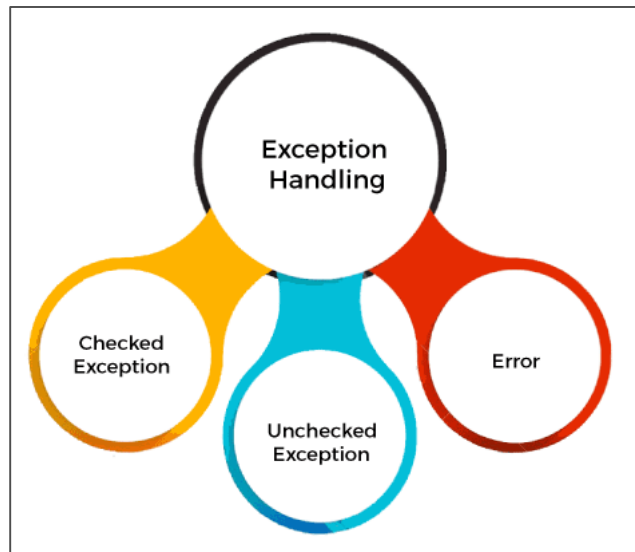
Error menunjukkan bahwa suatu kesalahan berupa operasi yang *illegal*. Kesalahan penulisan *keyword* perintah, kesalahan logical kode program Kesalahan operasi ini menyebabkan masalah pada kode program, Selama menulis kode program, kemunculan *error* sangat sulit dikenali.

Exception adalah peristiwa yang terjadi dalam kode. Seorang programmer dapat menangani kondisi seperti itu dan mengambil tindakan korektif yang diperlukan. Exception dapat ditangani oleh blok try-catch sehingga dapat membuat program tetap berjalan normal jika exception ini muncul. Sementara memulihkan *Error* adalah hal yang sangat tidak mungkin. Satu-satunya cara yang dilakukan adalah menghentikan program

3. Jenis jenis dari exception

Pada umumnya terdapat dua jenis *exception* pada bahasa Java namun menurut oracle terdapat 3 jenis exception yaitu :

- a. *checked exception*
- b. *unchecked exception*
- c. *error*



Gambar 18. 1 Jenis Jenis *Exception*

a. *Checked exception*

Merupakan eksepsi yang dikenal dan diketahui pada saat program di *compile class* yang secara langsung mewarisi adalah class *Throwable* misalnya, *IOException*, *SQLException*, dll.

b. *Unchecked Exception*

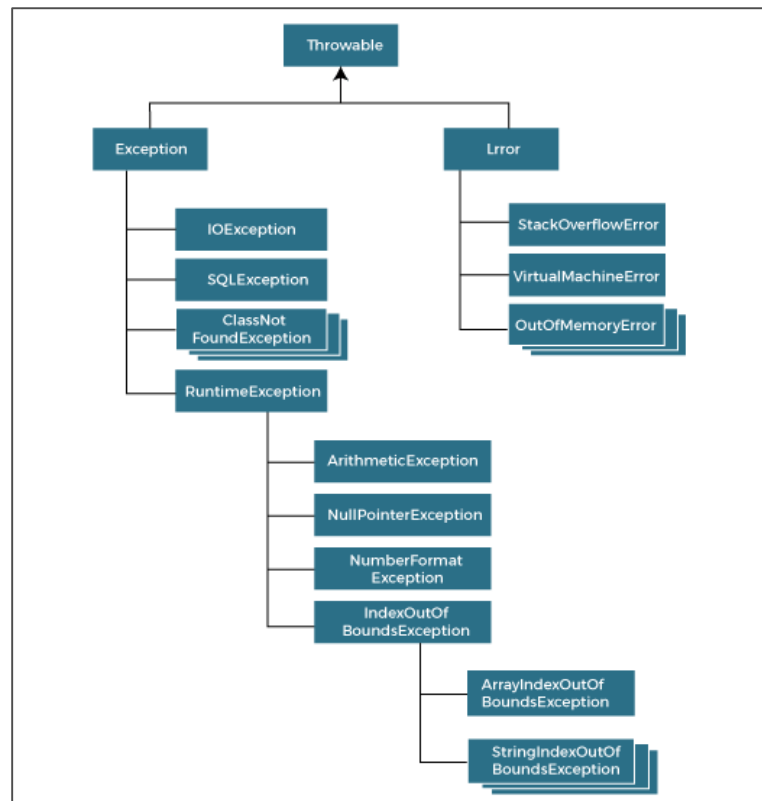
Merupakan class yang mewarisi *RuntimeException* dikenal Misalnya, *ArithmeticException*, *NullPointerException*, *ArrayIndexOutOfBoundsException*, dll. *Unchecked Exception* tidak diperiksa pada waktu kompilasi, tetapi diperiksa saat runtime.

c. *Error*

Kesalahan yang tidak dapat dipulihkan. Beberapa contoh kesalahan adalah *OutOfMemoryError*, *VirtualMachineError*, *AssertionError* dll.

4. Hirarki dari class exception

Class `java.lang.Throwable` adalah kelas akar dari hierarki exception Java yang diwarisi oleh dua subkelas: `checked` dan `error`. Hirarki class exception Java diberikan di bawah ini:



Gambar 18. 2 Hirarki *Exception*

5. Keyword Java exception

Java menyediakan lima kata kunci yang digunakan untuk menangani *exception*. Tabel berikut menjelaskan masing-masing.

Tabel 18. 1 Keyword pada exception

Keyword	Deskripsi
try	Kata kunci " <i>Try</i> " digunakan untuk menentukan blok di mana kita harus menempatkan kode exception . Artinya kita tidak bisa menggunakan blok <i>try</i> saja. Blok <i>try</i> harus diikuti oleh <i>catch</i> atau <i>last</i> .
catch	Blok " <i>catch</i> " digunakan untuk menangani <i>exception</i> harus didahului dengan blok <i>try</i> yang artinya kita tidak bisa menggunakan blok <i>catch</i> saja. bisa diikuti dengan keyword <i>finally</i> setelahnya
finally	Blok " <i>finally</i> " digunakan untuk mengeksekusi kode program yang diperlukan. dieksekusi apakah exception ditangani atau tidak.
throw	Kata kunci " <i>throw</i> " digunakan untuk melempar eksepsi .
throws	Kata kunci " <i>throws</i> " digunakan untuk menyatakan ekspresi atau mendeklarasikan exception untuk menentukan bahwa mungkin ada exception dalam metode.

6. Penggunaan Blok try catch

Blok *try* digunakan untuk menyertakan kode yang mungkin menimbulkan *exception*. Maka *try* harus digunakan dalam metode. Jika exception terjadi pada pernyataan tertentu di blok *try*, sisa kode dari blok tidak akan dieksekusi. Jadi, disarankan untuk tidak menyimpan kode di blok *try* yang tidak akan mengeluarkan exception.

Blok Java *try* harus diikuti oleh blok *catch* atau *finally*.

Bentuk Penulisan *Blok try catch*

```
try{
    //code that may throw an exception
} catch(Exception_class_Name ref){}
```

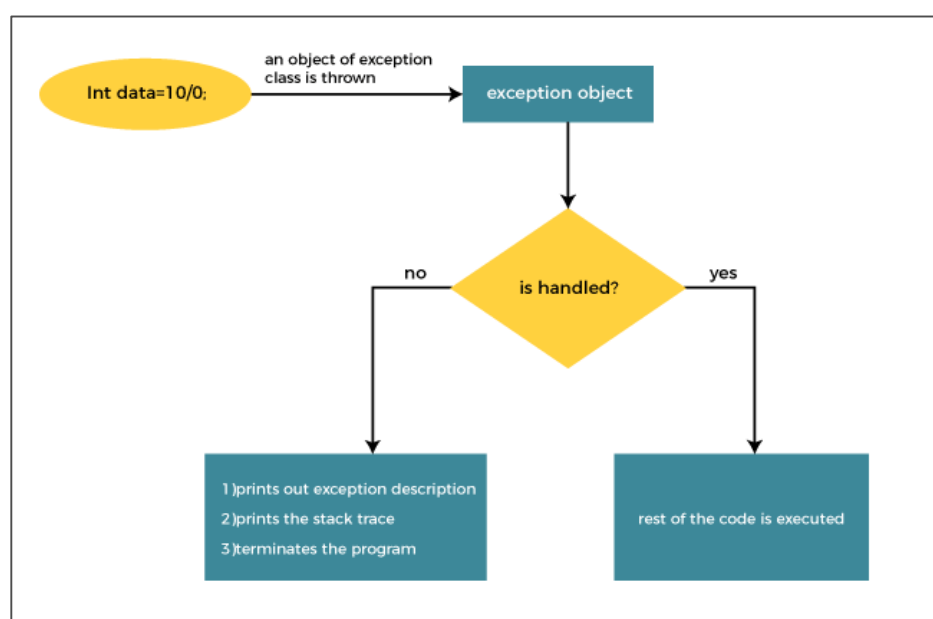
Bentuk Penulisan Blok *try finally*

```
try{
    //code that may throw an exception
}
finally{}
```

Blok `catch` Java digunakan untuk menangani *Exception* dengan mendeklarasikan tipe *exception* di dalam parameter. *Exception* yang dideklarasikan harus merupakan *Exception* kelas induk atau jenis *Exception* yang dihasilkan.

7. Cara kerja blok `try – catch`

JVM pertama-tama memeriksa apakah *exception* ditangani atau tidak. Jika *exception* tidak ditangani, JVM menyediakan penanganan pengecualian default yang melakukan tugas-tugas berikut:



Gambar 18. 3 cara kerja blok `try catch`

- Mencetak deskripsi exception.
- Mencetak jejak tumpukan (Hirarki metode tempat exception terjadi).
- Menyebabkan program berhenti.

Contoh program tanpa *exception handling*

TryCatchcontoh1.java

```
public class TryCatchcontoh1 {  
  
    public static void main(String[] args) {  
  
        int data=50/0; //may throw exception  
  
        System.out.println("rest of the code");  
  
    }  
  
}
```

Output :

```
Exception in thread "main" java.lang.ArithmeticException:  
    / by zero
```

Contoh program menggunakan *exception handling* (blok *try catch*)

TryCatchcontoh2.java

```
public class TryCatchcontoh2 {  
  
    public static void main(String[] args) {  
  
        try  
        {  
  
            int data=50/0; //may throw exception  
  
        }  
  
        //handling the exception  
  
        catch(ArithmeticException e)  
        {  
  
        }  
  
    }  
  
}
```

```
        System.out.println(e);  
    }  
  
    System.out.println("rest of the code");  
}  
}
```

Output :

```
java.lang.ArithmeticException: / by zero  
  
rest of the code
```

Seperti yang ditampilkan dalam contoh di atas, *rest of code* dieksekusi.

Contoh lain program blok *try catch* :

trycatchContoh3.java

```
public class TryCatchContoh3{  
  
    public static void main(String[] args) {  
        int i=50;  
        int j=0;  
        int data;  
  
        try  
        {  
            data=i/j; //  
        }  
  
        // Penanganan dengan exception  
        catch(Exception e)  
        {  
            // solusi dengan blok try catch exception  
            System.out.println(i/(j+2));  
        }  
    }  
}
```

Output :

25

8. Penggunaan Keyword Throw

Keyword Java **throw** digunakan untuk melempar exception secara eksplisit. Kita menentukan objek exception yang akan dilempar. Exception ini memiliki beberapa pesan yang memberikan deskripsi kesalahan. Exception ini mungkin terkait dengan kesalahan input pengguna, server, dll.

Kita dapat membuang exception yang *unchecked* atau *checked* di Java dengan kata kunci *throw*.

Bentuk umum penulisan keyword *throw*

```
throw new exception_class("error message");
```

contoh penulisan *throw* IOException

```
throw new IOException("sorry device error");
```

Contoh program *throw* unchecked exception

Dalam contoh ini, kita telah membuat metode validasi yang mengambil nilai integer sebagai parameter. Jika usia kurang dari 17, kita melempar *ArithmeticException* jika tidak, cetak pesan selamat datang untuk memilih.

ThrowContoh1.java

```
public class ThrowContoh1{  
    //function mencheck jika seseorang eligible untuk  
    vote atau tidak  
    public static void validate(int age) {  
        if(age<17) {  
            //throw Arithmetic exception jika tidak eligible  
            throw new ArithmeticException("Orang ini  
            Belum eligible untuk vote");  
        }  
        else {
```

```
        System.out.println("  Orang  ini  sudah  eligible
untuk  vote!!");

    }

}

//main method

public static void main(String args[]){

    //calling the function

    validate(13);

    System.out.println("rest of the code...");

}

}
```

Output

```
Exception in thread "main" java.lang.ArithmeticException:
Person is not eligible to vote

    at
pkg01sifk001.ThrowContoh1.validate(ThrowContoh1.java:16)

    at
pkg01sifk001.ThrowContoh1.main(ThrowContoh1.java:27)
```

Contoh program *throw User defined exception*

UserDefinedException.java

```
public class UserDefinedException extends Exception
{

    public UserDefinedException(String str)

    {

        // Calling constructor of parent Exception

        super(str);

    }

}
```

```
}
```

throwContoh2.java

```
public class throwContoh2 {  
    public static void main(String[] args) {  
        // TODO code application logic here  
        try  
        {  
            // throw an object of user defined  
exception  
            throw new UserDefinedException("Ini Adalah  
user-defined exception");  
        }  
        catch (UserDefinedException ude)  
        {  
            System.out.println("Menangkap  
exception");  
            // Print the message from MyException  
object  
            System.out.println(ude.getMessage());  
        }  
    }  
}
```

Output

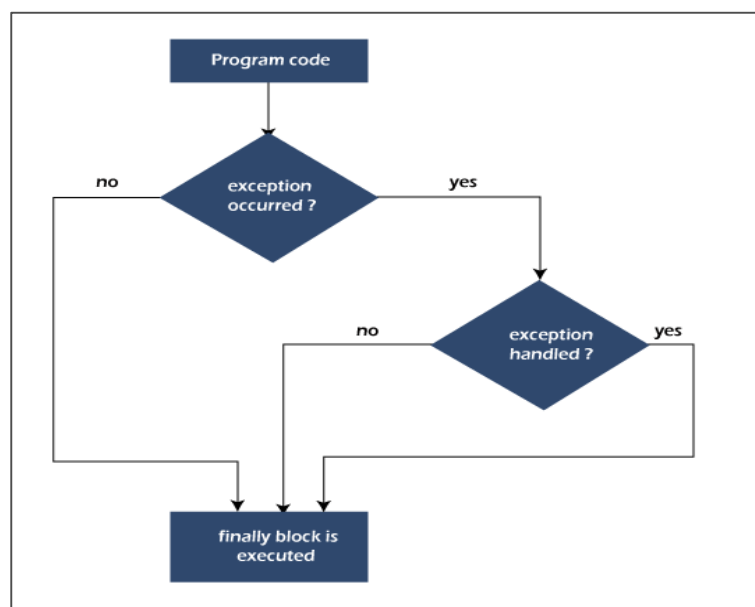
```
Menangkap exception  
Ini Adalah user-defined exception
```

9. Penggunaan Blok Finally

Blok finally pada Java adalah blok yang digunakan untuk mengeksekusi kode penting seperti menutup koneksi, dll. Blok ini selalu dieksekusi apakah *exception* ditangani atau tidak. Oleh karena itu, berisi semua pernyataan yang diperlukan yang perlu dicetak terlepas dari exception terjadi atau tidak.

Mengapa menggunakan blok finally ?

- Blok Finally di Java dapat digunakan untuk meletakkan kode "pembersihan" seperti menutup file, menutup koneksi, dll.
- Pernyataan-pernyataan penting yang akan dicetak dapat ditempatkan di blok terakhir.



Gambar 18. 4 Blok Finally

Contoh penggunaan blok finally

FinallyBlokContoh.java

```
class FinallyBlokContoh {  
  
    public static void main(String args[]){  
  
        try{  
  
            //below code do not throw any exception
```

```
int data=50/5;

System.out.println(data);

}

//catch won't be executed

catch (NullPointerException e){
System.out.println(e);
}

//executed regardless of exception occurred or not

finally {
System.out.println("blok finally akan selalu di eksekusi
");
}

System.out.println("rest of the code...");

}

}
```

Output

```
10

blok finally akan selalu di eksekusi

rest of the code
```

C. LATIHAN/TUGAS

1. Terdapat 2 cara untuk *exception* yaitu :
 - a. Blok *try catch* (Menangkap Exception)
 - b. *throw* (Melempar exception)

Apa perbedaan dari kedua exception tersebut, buat contoh programnya untuk masing exception.

2. Buat program blok finally dibawah ini lalu apa hasil dari outputnya :

```
class FinallyBlokContoh {  
  
    public static void main(String args[]){  
  
        try{  
  
            //below code do not throw any exception  
  
            Float data=500/15;  
  
            System.out.println(data);  
  
        }  
  
        catch (NullPointerException e){  
  
            System.out.println(e);  
  
        }  
  
        finally {  
  
            System.out.println("blok finally akan selalu di eksekusi  
");  
  
        }  
    }  
}
```

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

Wahono Satrio Romi, Java Fundamental, <https://romisatriawahono.net/java/>, diakses pada tanggal 5 Desember 2022

DAFTAR PUSTAKA

- Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.
- Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, *publishing as prentice hall*.
- Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.
- Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press
- Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.
- Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021
- Wahono Satrio Romi, Java Fundamental, <https://romisatriawahono.net/java/>, diakses pada tanggal 5 Desember 2022

RENCANA PEMBELAJARAN SEMESTER (RPS)

Program Studi	: Sistem Informasi S-1	Mata Kuliah/Kode	: Pemrograman Berorientasi Objek (Java-1)/ KK1108
Prasyarat	: -	SKS	: 3 Sks
Semester	: 2	Kurikulum	: KKNi
Deskripsi Mata Kuliah	: Mata kuliah ini membahas mengenai Pemrograman Java dengan konsep <i>Object Oriented programming (OOP)</i> mata kuliah ini merupakan mata kuliah wajib Program Studi S-1 sistem Informasi yang membahas tentang Pengantar Java, variable, tipe data, operator, Control Flow If, Control Flow If-else, Switch case, loop for, loop while, loop do while, Oop Class, Object, Constructor, Inheritance, polymorphism, abstraction, Encapsulation, Access modifier, array, dan exception handling.		Capaian Pembelajaran : Setelah pembelajaran diharap mahasiswa mampu membuat aplikasi sederhana dengan bahasa java dengan benar berdasarkan konsep OOP.
Penyusun	Ir. Agus Suharto, M.Kom		

PERTEMUAN KE-	KEMAMPUAN AKHIR YANG DIHARAPKAN	BAHAN KAJIAN (MATERI AJAR)	METODE PEMBELAJARAN	PENGALAMAN BELAJAR MAHASISWA	KRITERIA PENILAIAN	BOBOT NILAI
---------------	---------------------------------	----------------------------	---------------------	------------------------------	--------------------	-------------

(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	Mahasiswa mampu memahami sejarah bahasa java , memahami Terminologi bahasa java , mengerti JDK , JVM, JRE , dapat membuat program java pertama mengimplementasi kedalam bahasa pemrograman	Pengantar Pemrograman Java	Ceramah, diskusi	Mengungkapkan pendapat dan mengajukan Pertanyaan tentang Sejarah Bahasa java dan konsep Pemrograman Berorientasi Objek	Mahasiswa aktif bertanya materi yang diberikan dan mengungkapkan pendapat mengenai Bahasa java JDK, JVM, ,JRE , dan program pertama java	5%
2	Mahasiswa dapat memahami apa itu variabel,Mahasiswa dapat membedakan jenis jenis variable ,Mahasiswa dapat aturan penulisan variable , serta mempraktekan penggunaan variabel pada bahasa program java	Variabel pada bahasa java	Ceramah, diskusi	Mendeklarasikan variabel , Aturan Penamaan variabel, Jenis jenis variabel, latihan	Mahasiswa mampu merespon, bertanya dan berdiskusi aktif	5%

3	Mahasiswa dapat memahami Tipe tipe Data pada Bahasa java , Mahasiswa dapat membedakan Tipe Data Primitf dan Non Primitif , Mahasiswa dapat mempraktekan penggunaan tipe data bahasa java	Tipe data pada java	Ceramah, diskusi dan penyelesaian soal	Latihan 3	Mahasiswa mampu merespon, bertanya dan berdiskusi aktif	5%
4	Mahasiswa dapat memahami jenis jenis operator pada java, membedakan Jenis operator pada bahasa java dapat mempraktekan penggunaan Jenis operator pada bahasa program java	Operator Pada java	Ceramah, Tanya jawab dan penyelesaian soal	Latihan 4	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	5%

5	Mahasiswa dapat memahami penggunaan input output pada java ,dapat mempraktekan menggunakan input Scanner , dapat mempraktekan menggunakan input ,BufferReader dapat memahami Keyword pada java	Input Output Stream dan Keyword ada java	Ceramah, Tanya jawab dan penyelesaian soal, praktek	Latihan 5	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	6%
6	Mahasiswa dapat memahami penggunaan Kondisi aliran program IF ,dapat memahami penggunaan Kondisi aliran program IF ..Else ,dapat mempraktekan menggunakan If -Else	Mengontrol Aliran Program (IF-IF..ELSE)	Ceramah, diskusi, praktek	Mengungkapkan pendapat dan mengajukan pertanyaan mengenai fungsi If, IF..Else, Latihan 6	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	6%

7	<p>Mahasiswa dapat memahami penggunaan Kondisi aliran SWITCH..CASE</p> <p>,dapat mempraktekan penggunaan SWITCH..CASE,</p> <p>Mahasiswa dapat memilih penggunaan opsi struktur kendali /control flow</p>	Mengontrol Aliran Program Switch –Case	Ceramah, diskusi, praktek	Mengungkapkan pendapat dan mengajukan pertanyaan tentang teknik search	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	6%
8	<p>Mahasiswa dapat memahami Konsep Kontrol pengulangan / Looping</p> <p>,dapat mengerti aturan kode Kontrol pengulangan For,dapat mempraktekan penggunaan Kontrol pengulangan For , nested for</p>	Mengontrol Aliran Program Perulangan For	Ceramah, Tanya jawab dan penyelesaian soal, praktek	Latihan 8	Mahasiswa mampu merespon, bertanya dan membuat program Perulangan For	6%

9	Mahasiswa dapat memahami Konsep Kontrol pengulangan / Looping While Do While, dapat mengerti aturan penulisan Kontrol pengulangan While Do While, dapat membedakan dan mempraktekan penggunaan Kontrol pengulangan While dan Do While	Mengontrol Aliran Program Perulangan While , Do While	Ceramah, Tanya jawab dan penyelesaian soal, Praktek	Latihan 9	Mahasiswa mampu merespon, bertanya dan membuat program loop While dan do While	6%
UTS						
10	Mahasiswa memahami Konsep Pemrograman berorientasi Objek ,memahami apa itu object , <i>class</i> pada Pemrograman berorientasi Objek, memahami apa itu <i>attribute</i> ,	Konsep PBO <i>Class</i> , <i>Object</i> , <i>Attribute</i> , <i>Method</i>	Ceramah, Tanya jawab dan penyelesaian soal. Praktek	Latihan 10	Mahasiswa mampu merespon, bertanya dan membuat program konsep OOP(Object Oriented Programming	5%

	<i>method</i> pada Pemrograman berorientasi Objek dapat mempraktekan pembuatan class,object , attribute dan method pada bahasa java					
11	1. Mahasiswa memahami Konsep OOP constructor, memahami cara kerja dan jenis jenis <i>constructor</i> ,dapat mempraktekan penggunaan constructor dalam bahasa java	Konsep PBO <i>Constructor</i>	Ceramah, diskusi,Praktek	Latihan 11	Mahasiswa mampu merespon, bertanya dan membuat program konsep OOP <i>Constructor</i>	5%
12	Mahasiswa memahami Konsep OOP <i>Inheritance</i> /Pewarisan memahami cara kerja dan jenis jenis <i>Inheritance</i> ,	Konsep PBO <i>Inheritance</i> ,/ Pewarisan	Ceramah, diskusi,praktek	Mengungkapkan pendapatdan mengajukan pertanyaan tentang inheritance,latihan 12	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	5%

	dapat mempraktekan penggunaan Inheritance dalam bahasa java, dapat mempraktekan dan membedakan jenis jenis Inheritance dalam bahasa java					
13	<p>Mahasiswa memahami Konsep OOP Polymorphism ,membedakan jenis jenis Polymorphism.</p> <p>Dapat memahami method overloading dan method override</p> <p>dapat mempraktekan penggunaan Polymorphism dan jenis jenisnya</p>	<p>Konsep Polymorphism</p> <p>PBO</p>	<p>Ceramah, Tanya jawab dan penyelesaian soal, praktek</p>	<p>Mengungkapkan pendapat dan mengajukan pertanyaan tentang polymorphism, latihan 13</p>	<p>Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskus dan mempraktekan</p>	5%

14	Mahasiswa memahami apa itu abstract class pada bahasa java oop, memahami apa itu interface Oop, ,dapat mengetahui aturan yang harus digunakan abstract class dan Interface ,dapat mempraktekan kapan penggunaan abstract class dan Interface	<i>Abstract,class, Interface</i>	Ceramah, Tanya jawab dan penyelesaian soal, praktek	Latihan 14	Mahasiswa mampu merespon, bertanya dan membuat program konsep <i>abstract, class, interface</i>	6%
15	Mahasiswa dapat memahami apa itu encapsulation, package, memahami kelebihan dari encapsulation , dapat mengetahui cara membuat package , dapat mempraktekan	<i>Encapsulation , package</i>	Ceramah, diskusi	Mengungkapkan pendapat dan mengajukan pertanyaan, latihan 15	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi dan mempraktekan nya	6%

	encapsulation dan package					
16	Mahasiswa dapat memahami apa itu access modifier, dapat membedakan jenis jenis access modifier, dapat mengetahui aturan membuat access modifier, dapat mempraktekan penggunaan access modifier	<i>Access modifier</i>	Ceramah, Tanya jawab dan penyelesaian soal,	Mengungkapkan pendapat dan mengajukan pertanyaan mengenai access modifier, latihan 16	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi.	6%
17	Mahasiswa memahami konsep dari Array, dapat membedakan jenis array satu dimensi dan dua dimensi ,dapat mengerti fitur pada array dan mendeklarasikan array , dapat mempraktekan penggunaan array satu dimensi dan multi dimensi	Array / Larik	Ceramah, Tanya jawab dan penyelesaian soal, praktek	Mengungkapkan pendapat dan mengajukan pertanyaan mengenai array / larik, latihan 17	Mahasiswa mampu merespon secara aktif materi yang diberikan dengan cara bertanya dan berdiskusi	6%

	konsep OOP					
18	Mahasiswa mampu membuat aplikasi sederhana dengan konsep dari <i>Exception Handling</i> .	<i>Exception Handling</i>	Ceramah, Tanya jawab dan penyelesaian soal , latihan	Latihan 18	Mahasiswa mampu merespon, bertanya dan membuat program dengan penanganan eksepsi	6%
UAS						

REFERENSI

Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

Wahono Satrio Romi, Java Fundamental, <https://romisatriawahono.net/java/>, diakses pada tanggal 5 Desember 2022

Ketua Program Studi
S1 Sistem Informasi

Dede Supriyadi, S.Kom, . M.Kom
NIDN. 0403078402

Tangerang Selatan, 10 Desember 2021
Ketua Tim Penyusun

Ir. Agus Suharto, M.Kom
NIDN. 0329056504