2016

Modul Praktikum Implementasi Algoritma

LABORATORIUM PRIDE

KELOMPOK KEAHLIAN PROGRAMMING

FAKULTAS ILMU TERAPAN

UNIVERSITAS TELKOM

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR PROGRAM

No table of figures entries found.

DAFTAR TABEL

Modul 0 : Running Modul

0.1 Tujuan

Setelah mengikuti Running Modul mahasiswa diharapkan dapat:

1. Memahami peraturan kegiatan praktikum.
2. Memahami Hak dan Kewajiban praktikan dalam kegiatan praktikum.
3. Memhami komponen penilaian kegiatan praktikum.

0.2 Peraturan Praktikum

1. Praktikum diampu oleh **Dosen Kelas** dan dibantu oleh **Asisten Laboratorium** dan **Asisten Praktikum**.
2. Praktikum dilaksanakan di Gedung FIT lantai 2 (**PRIDE LAB**) sesuai jadwal yang ditentukan.
3. Praktikan wajib membawa **modul praktikum, kartu praktikum, dan alat tulis**.
4. Praktikan wajib mengisi **daftar hadir** dan **BAP praktikum** dengan bolpoin **bertinta hitam**.
5. Durasi kegiatan praktikum **D3 = 4 jam (200 menit)**.
    a. 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
    b. 60 menit untuk penyampaian materi
    c. 125 menit untuk pengerjaan jurnal dan tes akhir
6. Jumlah **pertemuan praktikum**:
    - 10 kali di lab (praktikum rutin)
    - 3 kali di luar lab (terkait Tugas Besar dan/atau UAS)
    - 1 kali berupa presentasi Tugas Besar dan/atau pelaksanaan UAS
7. Praktikan **wajib hadir minimal 75%** dari seluruh pertemuan praktikum di lab.Jika total kehadiran kurang dari 75% maka nilai UAS/ Tugas Besar = 0.
8. Praktikan yang datang terlambat :
    - <= 30 menit : diperbolehkan mengikuti praktikum tanpa tambahan waktu Tes Awal
    - > 30 menit : tidak diperbolehkan mengikuti praktikum
9. Saat praktikum berlangsung, asisten praktikum dan praktikan:
    - Wajib menggunakan **seragam** sesuai aturan Institusi.
    - Wajib mematikan/ men-silent semua **alat komunikasi**(smartphone, tab, iPad, dsb).
    - Dilarang membuka **aplikasi yang tidak berhubungan** dengan praktikum yang berlangsung.
    - Dilarang mengubah **setting software maupun hardware** komputer tanpa ijin.
    - Dilarang **membawa makanan maupun minuman** di ruang praktikum.
    - Dilarang **memberikan jawaban ke praktikan lain** (pre-test, TP, jurnal, dan post-test).
    - Dilarang **menyebarkan soal pre-test, jurnal, dan post-test**.
    - Dilarang**membuang sampah/sesuatu apapun** di ruangan praktikum.
10. Setiap praktikan dapat mengikuti praktikum susulan maksimal 2 modul untuk satu praktikum.
    - Praktikan yang dapat mengikuti praktikum susulan hanyalah praktikan yang memenuhi syarat sesuai ketentuan Institusi, yaitu rawat inap di Rumah Sakit

(menunjukkan bukti rawat inap dan resep obat dari RS), tugas dari Institusi (menunjukkan surat dinas dari Institusi), atau mendapat musibah (menunjukkan surat keterangan dari orangtua/ wali mahasiswa).

- Persyaratan untuk praktikum susulan diserahkan sesegera mungkin ke Asisten Praktikum untuk keperluan administrasi.

11. Pelanggaran terhadap peraturan praktikum ini akan ditindak secara tegas secara berjenjang di lingkup Kelas, Laboratorium, Program Studi, Fakultas, hingga Institusi.

0.3 Penilaian Praktikum

1. Komponen penilaian praktikum:
   60% nilai permodul dan **40%** nilai Tugas Besar (atau UAS praktek)
2. Seluruh komponen penilaian beserta pembobotannya ditentukan oleh dosen **PJMP**
3. Penilaian permodul dilakukan oleh **asisten praktikum**, sedangkan nilai Tugas Besar/ UAS diserahkan kepada **dosen kelas,** dilaporkan ke **PJMP.**
4. Baik praktikan maupun asisten tidak diperkenankan meminta atau memberikan **tugas tambahan** untuk perbaikan nilai.
5. Standar **indeks dan range nilai** ditentukan oleh dosen PJMP atas sepengetahuan Ketua Kelompok Keahlian

Modul 1 : Pengantar

1.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengenal struktur dasar bahasa pemrograman
2. Mengetahui sejarah singkat bahasa Java
3. Menginstal IDE IntelliJ IDEA
4. Membuat program Hello World.

1.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software IntelliJ IDEA yang telah terinstall pada masing-masing PC

1.3 Dasar Teori

1.3.1 Algoritma

1.3.1.1 Definisi Algoritma

Dalam menghadapi suatu masalah, setiap orang pasti mempunyai cara-cara tersendiri untuk menyelesaikannya. Di dunia pemrograman, istilah algoritma bisa dianalogikan sebagai hal tersebut. Algoritma merupakan langkah-langkah yang perlu dilakukan agar dapat menyelesaikan suatu masalah. Masalah disini maksudnya adalah kasus yang harus dicari jalan keluarnya.

1.3.1.2 Struktur Dasar Algoritma

Sebuah algoritma dapat disusun dari 3 dasar algoritma yaitu sequence, selection, dan repetition.

1.3.1.3 Sequence

Sequence jika diartikan adalah berurutan. Maksudnya setiap baris instruksi harus dieksekusi secara berurutan. Tidak boleh instruksi pertama langsung mengeksekusi instruksi terakhir.

bilangan1 = 5 bilangan2 = 3 bilangan1 = bilangan2 + 2 bilangan1 = 4	Apakah hasil akhirnya? bilangan1 = …….. bilangan2 = ……..

Isilah kotak di sebelah kanan dengan nilai terakhir bilangan1 dan bilangan2 jika instruksi yang ada di kotak sebelah kiri dijalankan secara berurutan.

1.3.1.4 Selection

Adakalanya sebuah instruksi dikerjakan menurut kondisi tertentu. Tiap instruksi akan diseleksi terlebih dahulu untuk dieksekusi. Jika sesuai dengan kondisi yang diberikan maka instruksi tersebut akan dieksekusi.

A = 5
B = 4
If A > B then
  Tampilkan di layar “A > B”
Else
  Tampilkan di layar “B > A”
	Apakah hasil akhirnya?
	
	……..

Isilah kotak di sebelah kanan dengan hasil akhirnya jika instruksi yang ada di kotak sebelah kiri dijalankan secara berurutan.

1.3.1.5 Operator Aritmatik

Aritmatik adalah bentuk operasi angka (perkalian (*), pembagian (/), penjumlahan (+), pengurangan (-)). Berikut adalah contoh penggunaan aritmatika dalam algoritma.

A = 5 B = 4 Tampilkan di layar “A-B” Tampilkan di layar “A+B” Tampilkan di layar “A*B” Tampilkan di layar “A/B”	Apakah hasil akhirnya? ……..

1.3.1.6 Operator Logika

Operator untuk penghubung logika yaitu !, &&, ||. Operator tersebut digunakan ketika kita ingin menambahkan beberapa syarat di pengkondisian.

Tabel 1-1 Operator Logika

Operator	Keterangan
!	Negasi (NOT)
&&	Dan (AND)
||	Atau (OR)

1.3.1.7 Repetition

Repetition maksudnya kondisi yang dilaksanakan secara berulang ulang. Contoh di bawah adalah algoritma untuk menuliskan “Aku cinta Informatika” sebanyak 100 kali.

For 1 to 100 do Tampilkan di layar “Aku cinta informatika”	Apakah hasil akhirnya? ……..

Isilah kotak di sebelah kanan dengan hasil akhirnya jika instruksi yang ada di kotak sebelah kiri dijalankan secara berurutan.

1.3.2 Mengenal JAVA

Java adalah bahasa tingkat-tinggi yang hanya memiliki platform software. Dua komponen utama dari platform Java adalah Java Application Programming Interface (API) yang merupakan library dari Java dan Java Virtual Machine (JVM), interpreter yang mengubah source code Java menjadi bahasa mesin.

Menggunakan Java, suatu aplikasi dapat dijalankan pada platform yang berbeda. Java merupakan bahasa pemrograman berorientasi object (object-oriented programming), banyak fitur object-oriented Java mendapatkan pengaruh dari bahasa C++.

1.3.2.1 Sejarah Bahasa Java

Pada awal diciptakan komputer hanya sebagai alat bantu perhitungan, dan bahasa yang waktu itu digunakan masih sangat primitif karena hanya mengenal angka biner 1 dan 0. Beberapa waktu kemudian mulai diperkenalkan bahasa mesin yang bisa sedikit dipahami oleh manusia, yaitu bahasa Assembly yang termasuk bahasa tingkat menengah.

Tahun 1969, Laboratorium Bell AT&T di New Jersey menggunakan Assembly untuk mulai mengembangkan sistem operasi UNIX. Setelah UNIX berkembang, Ken Thompson seorang developer

di laboratorium tersebut mengembangkan compiler baru dengan bahasa B. Bahasa B ini masih bersifat intpreter dan terbilang lambat. Sehingga pada tahun 1971, UNIX kembali dibuat dengan menggunakan bahasa C, yang dikembangkan oleh Dennis Ritchi (developer dari lab yang sama). Tetapi bahasa C masih bersifat prosedural murni sehingga masih sulit dipelajari. Pada tahun 1983 Bjarne Stroustrup yang juga berasal dari lab Bell AT&T memperkenalkan bahasa C++ yang merupakan hybrid dari bahasa C.

Pada tahun 1991, ilmuwan dari Sun Microsystem yang dipimpin oleh James Gosling dan Patrick Naughton membuat bahasa pemrograman baru yang diberi nama “Green”. Pada awalnya, Green diperuntukkan bagi perangkat elektronik konsumer, bahasa ini didesain sederhana dan tidak tergantung pada arsitektur tertentu. Saat web menjadi populer pada tahun 1993, Sun melihat kesempatan untuk menggunakan Java untuk menambah konten dinamis pada halaman web, seperti interaktifitas dan animasi. Java diluncurkan tahun pada pameran Sun World di tahun 1995.

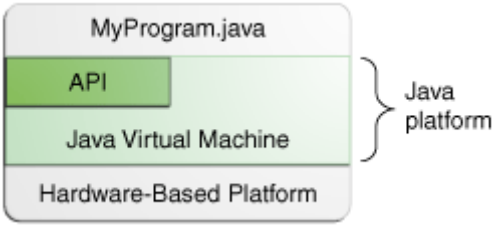
Tahun 2009, Sun Microsystem diakuisisi oleh Oracle. Pada konferensi JavaOne di tahun 2010, Oracle mengumumkan bahwa 97% komputer desktop perusahaan, tiga milyar handset dan 80 juta perangkat TV menggunakan bahasa Java. Terdapat lebih dari 9 juta Java developer, naik dua kali lipat dari tahun 2005. Saat ini Java merupakan bahasa pemrograman yang paling banyak digunakan di dunia.

1.3.2.2 Java Platform

Platform adalah lingkungan perangkat keras dan perangkat lunak tempat suatu program berjalan. Beberapa platform yang terkenal diantaranya adalah Microsoft Windows, Linux, dan Mac OS. Platform dapat dideskripsikan sebagai kombinasi antara OS dan perangkat keras dari perangkat. Namun, platform Java berbeda karena pada platform pada Java hanya berupa perangkat lunak saja (software-only platform) yang dapat dijalankan pada platform perangkat keras lainnya.

Platform Java terdiri atas dua komponen:

- Java Virtual Machine (JVM), merupakan dasar dari platform Java, interpreter yang mengubah source code menjadi… <<cari>>
- Java Application Programming Interface (API): kumpulan library Java, komponen perangkat lunak yang dapat langsung digunakan.



Gambar 1-1 Java Platform (API dan JVM)

Saat ini, terdapat empat jenis platform Java, yaitu:

a. Java Standard Edition (Java SE): Merupakan bahasa standar Java dengan core library. Dapat digunakan untuk membangun aplikasi Java berbasis desktop, server, maupun tertanam. Komponen yang terdapat pada Java SE adalah Java Development Kit (JDK), Java Runtime Environment (JRE) dan Java SE API. Di dalam JDK terdapat JRE, compiler dan debugger, sementara di dalam JRE terdapat library Java dan JVM.
b. Java FX, Java User Interface Platform: Merupakan user interface Java tingkat lanjut bagi perusahaan. Dapat digunakan untuk membangun aplikasi client-server yang lebih robust dan reliable.

c. Java Platform, Enterprise Edition (Java EE): Merupakan versiJava bagi industry untuk membangun web dan aplikasi perusahaan.
d. Java Embedded dan Java ME : Java Embedded digunakan untuk sistem tertanam pada perangkat elektronik pengguna, sementara Java ME merupakan lingkungan bagi aplikasi yang berjalan pada sistem tertanam dan bergerak.

Pada praktikum Implementasi Algoritma digunakan Java Standar Edition 8.

1.3.2.3 Instalasi Java untuk Windows

File instalasi Java SE 8 untuk Windows bisa diunduh dari tautan berikut :
http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html#javasejdk
Atau:
https://java.com/en/download/manual.jsp
Pilih untuk JDK. Setelah itu, lakukan instalasi dengan langkah berikut:

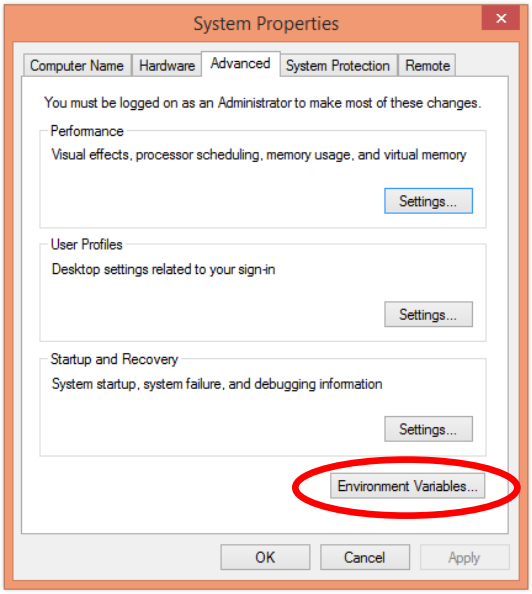
1. Klik installer. File jdk-8version-windows-i586-i.exe merupakan installer untuk Windows 32-bit, sementara file jdk-8version-windows-x64.exe merupakan installer untuk Windows 64-bit.
2. Klik Next, lalu Continue
3. Ikuti langkah instalasi sampai selesai.

Aplikasi berbasis Java dijalankan pada Command prompt DOS. Path dari file executable Java harus dituliskan setiap kali aplikasi dijalankan, misalnya:

```
C:\> "C:\Program Files\Java\jdk1.8.0\bin\javac" MyClass.java
```

Untuk mencegah hal tersebut, lakukan setting variable PATH dengan cara berikut:

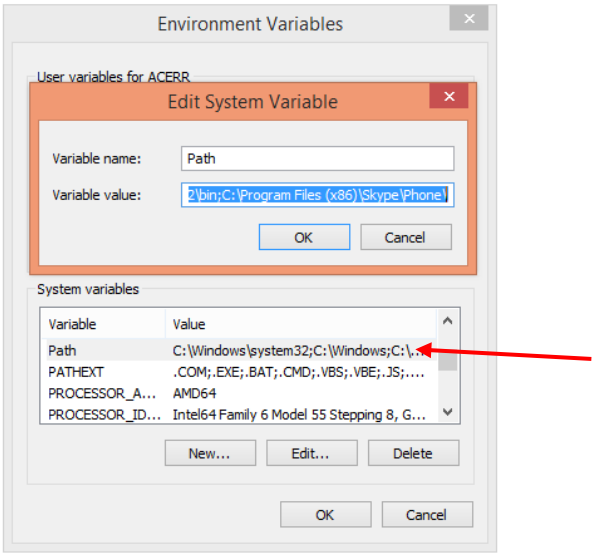
1. Klik **Start** >> **Control Panel** >> **System**.
2. Klik **Advanced** >> **Environment Variables**.



Gambar 1-2 Setting Environment Variables

3. Tambahkan lokasi folder bin JDK pada variabel PATH dalam System Variables:
   a. Pilih **Path** >> **Edit**

b. Tambahkan path JDK pada variable value. Path dari folder bin biasanya adalah sebagai berikut:
C:\WINDOWS\system32;C:\WINDOWS;C:\Program Files\Java\jdk1.8.0\bin



Gambar 1-3 Setting PATH

Membuat source code Java pada suatu editor dan menjalankannya lewat DOS terkadang tidak praktis dan menyulitkan bagi programmer baru. Akan jauh lebih mudah jika membuat program tersebut pada suatu Integrated Development Environment (IDE). Sepanjang praktikum ini, aplikasi akan dibuat pada IDE IntelliJ IDEA.

1.3.3 Mengenal IDE IntelliJ IDEA

Intellij IDEA (IntelliJ) merupakan IDE (*Integrated Development Environment*), yaitu alat pengembangan terpadu dari JetBrains. Di dalamnya terdapat fasilitas untuk programmer mengembangkan program / aplikasi.

IntelliJ merupakan IDE yang dapat dijalankan pada berbagai platform, seperti Windows, OS X dan Linux. IntelliJ dipilih karena keseuaiannya dengan pengembangan berbasis mobile dikemudian hari. Bundel IntelliJ telah mencakup JRE, jadi tidak perlu instalasi Java untuk menjalankan aplikasi yang dibuat pada IntelliJ, namun bundel ini tidak termasuk JDK.

1.3.3.1 Cara menginstall IntelliJ IDEA

File instalasi IntelliJ IDEA bisa diunduh dari tautan berikut :

https://www.jetbrains.com/idea/download/index.html#section=windows

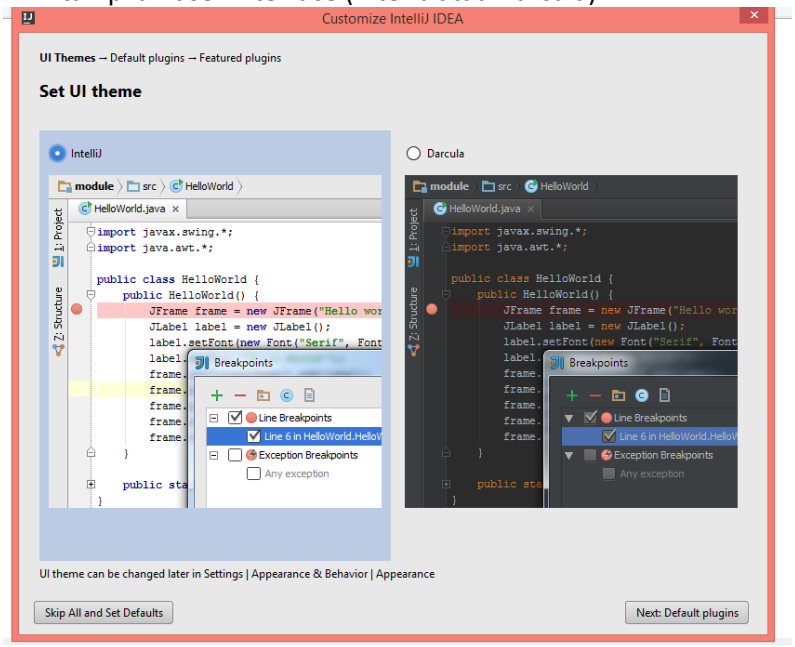
Minimum hardware requirement bagi instalasi IntelliJ adalah:

- RAM 1 GB, direkomendasikan 2 GB
- Hard disk space 300 MB + setidaknya 1 GB untuk cache
- Screen resolution: 1024x768

Langkah instalasi IntelliJ adalah sebagai berikut:

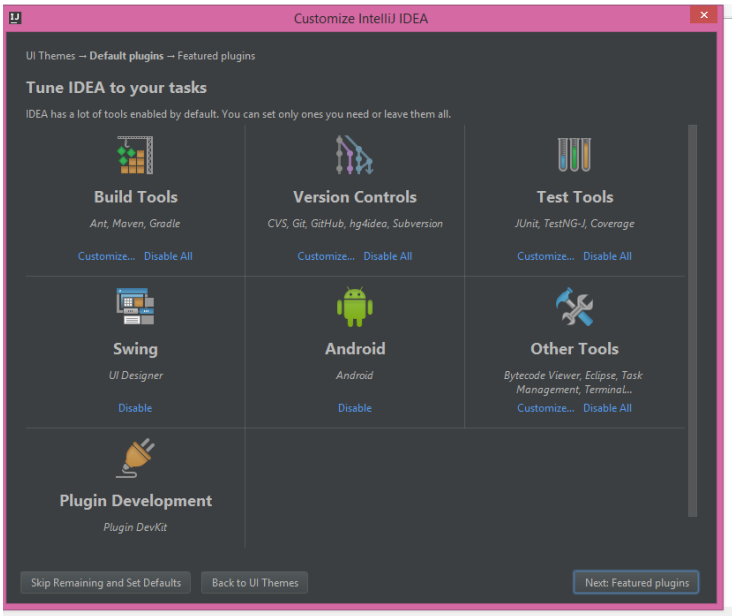
1. Klik installer. File idealC atau idealU-*.exe
2. Klik Next, lalu Continue

3. Ikuti langkah instalasi sampai selesai.
   a. Pilih untuk membuat icon IntelliJ pada desktop
   b. Pilih tampilan user interface (IntelliJ atau Darcula)



Gambar 1-4 Pilihan User Interface IntelliJ

   c. Pilih PlugIn (enable all)



Gambar 1-5 Pilihan Plug-in IntelliJ

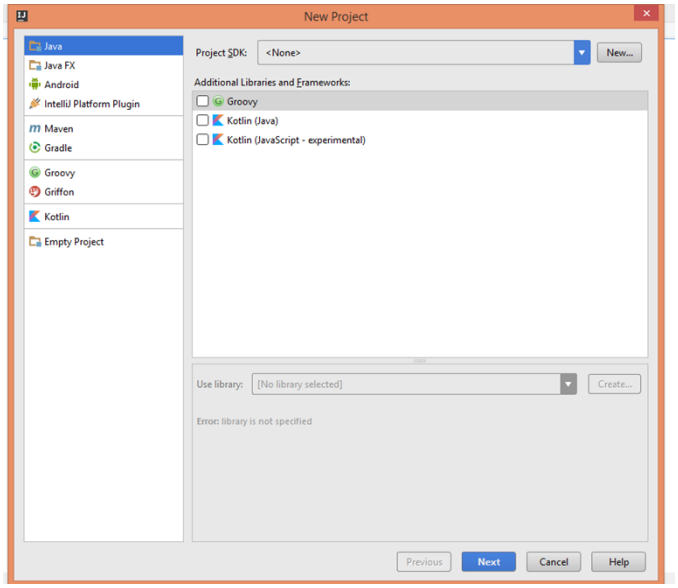
4. Untuk menjalankan IntelliJ, klik icon IntelliJ pada desktop, atau bisa dicari pada kotak search Windows.

1.3.3.2 Membuat Project baru di Intellij IDEA

Tampilan pada saat kali pertama membuka IntelliJ

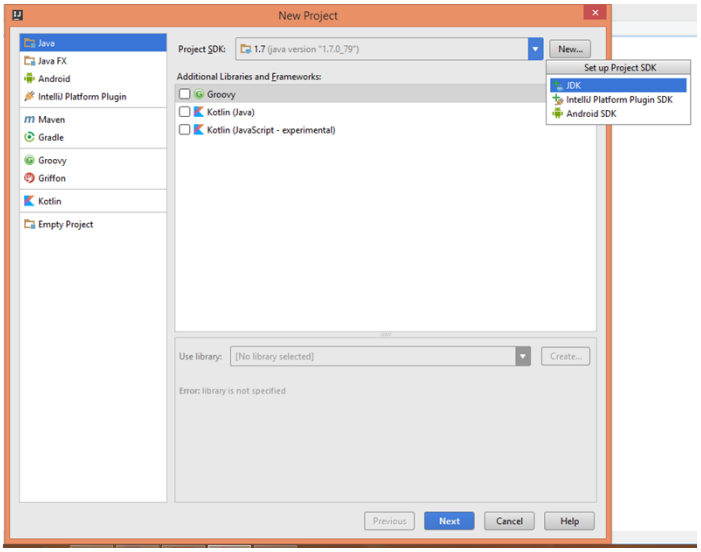
Gambar 1-6 Tampilan awal IntelliJ

1. Pilih Create New Project
2. Tampilan pada saat memulai project baru >> pilih Java



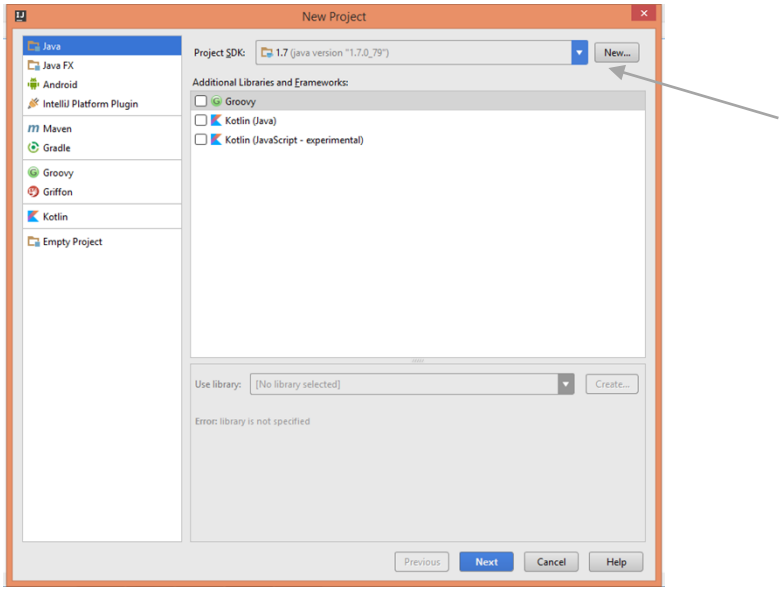
Gambar 1-7 Tampilan projek baru

3. Pilih New >> Set up Project SDK >> JDK



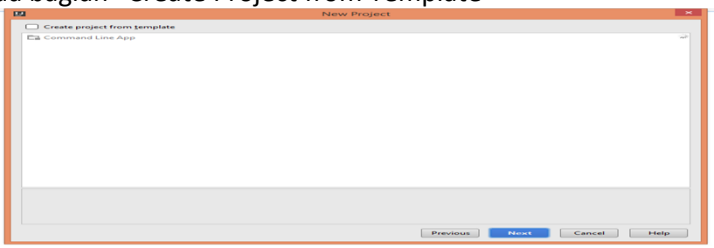
Gambar 1-8 Setting SDK

4. Masukkan folder bin dari Java JDK yang sudah diinstal sebelum instalasi IntelliJ (IntelliJ tidak men-support JDK).



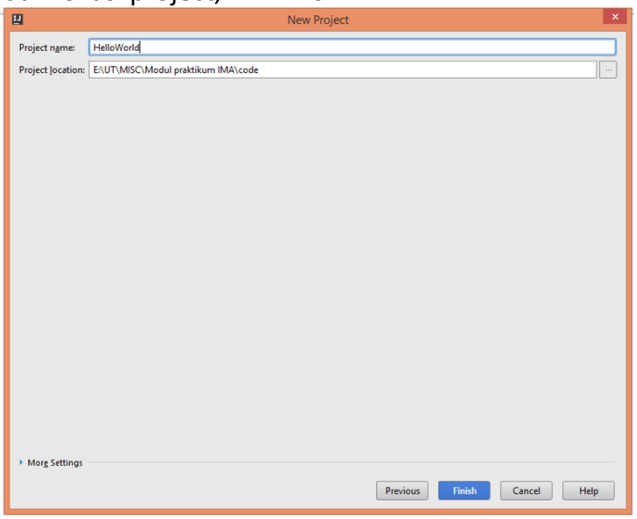
Gambar 1-9 Setting folder JDK

5. Klik Next (karena yang akan dibuat hanya project Java sederhana)
6. Klik Next pada bagian “Create Project from Template”



Gambar 1-10 Create poject from template

7. Tentukan nama dan lokasi project, klik finish.

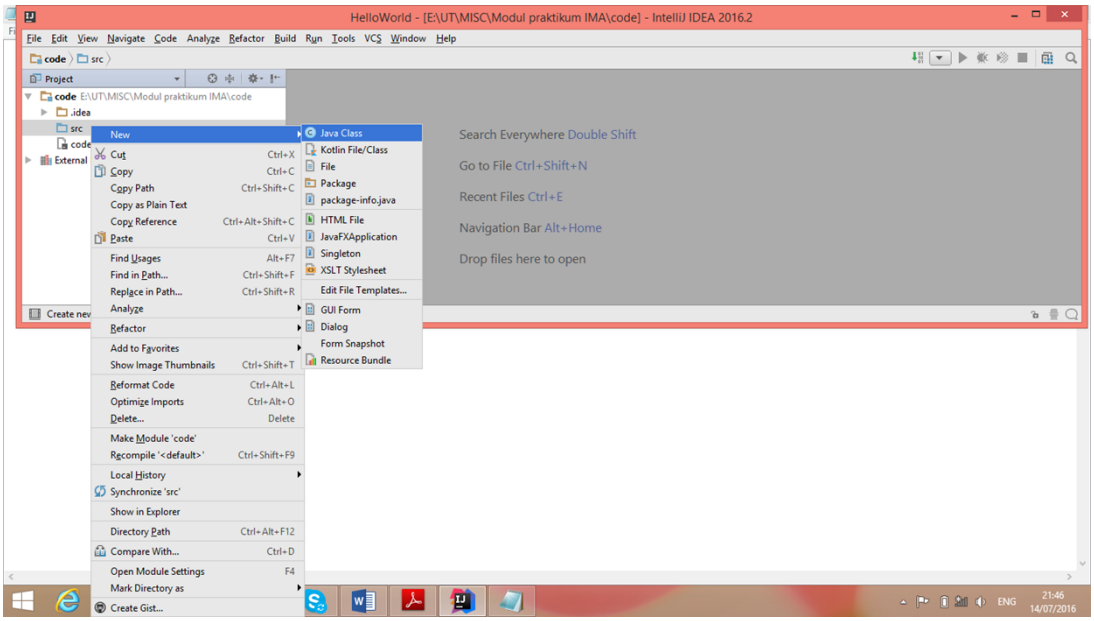


Gambar 1-11 Setting nama dan lokasi penyimpanan project

1.3.4 Hello World!

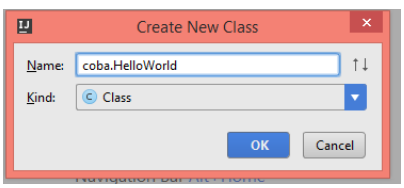
Hello World merupakan program sederhana pertama yang akan Anda buat. Pada Java, program dibuat di dalam Class yang terdapat pada package. Untuk membuat program pertama Anda, lakukan langkah berikut:

1. Pada project yang dibuat, terdapat folder src. Klik kanan folder ini >> **New** >> **Java Class**



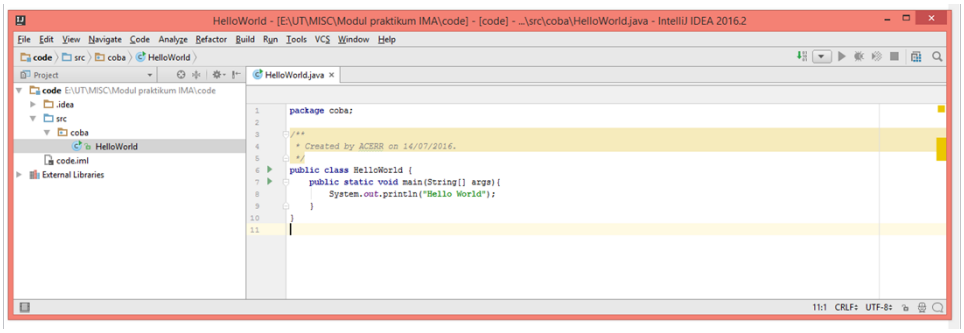
Gambar 1-12 Pembuatan class baru

2. Beri nama Class ini, misal coba.HelloWorld. Coba merupakan nama package tempat Class HelloWorld ditempatkan.



Gambar 1-13 Pemberian nama class

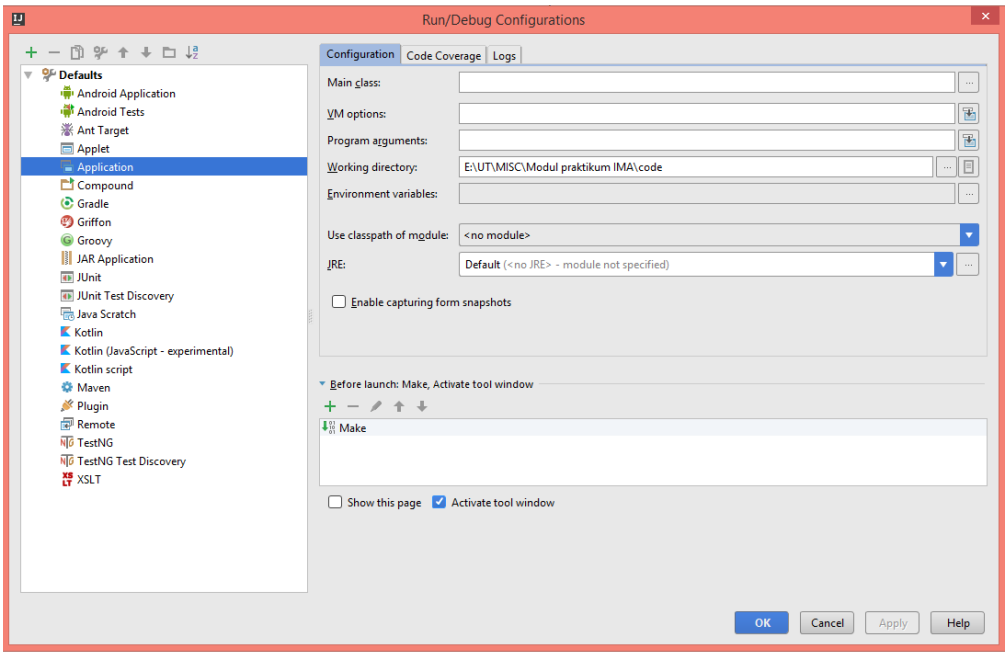
3. Ketikkan *code* berikut:



Gambar 1-14 Penulisan kode pada jendela kerja

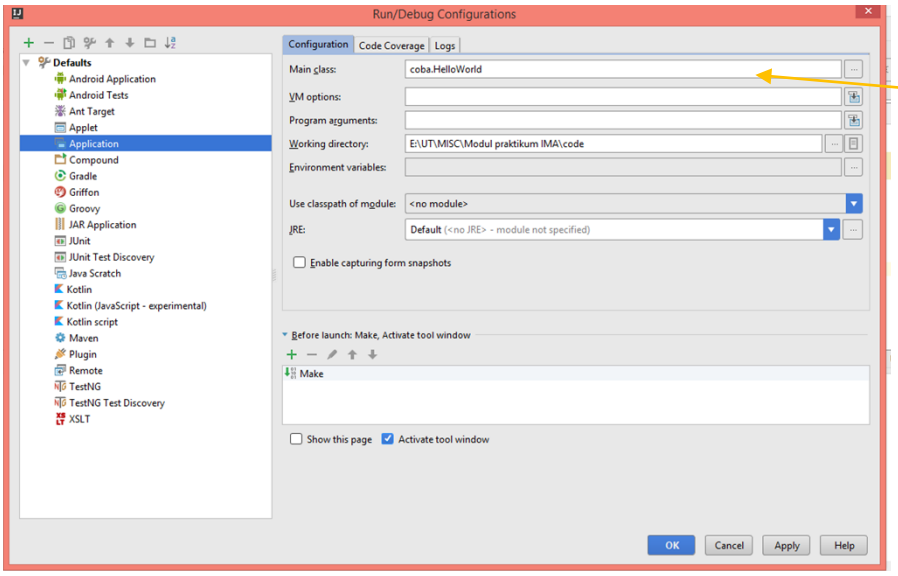
```
package coba;

public class HelloWorld {

    public static void main(String args[]){
        System.out.println(“Hello World”);
    }
}
```

4. Sebelum menjalankan program, terlebih dahulu harus dilakukan konfigurasi pada project yang dibuat. Pilih **Configuration** >> **Application**



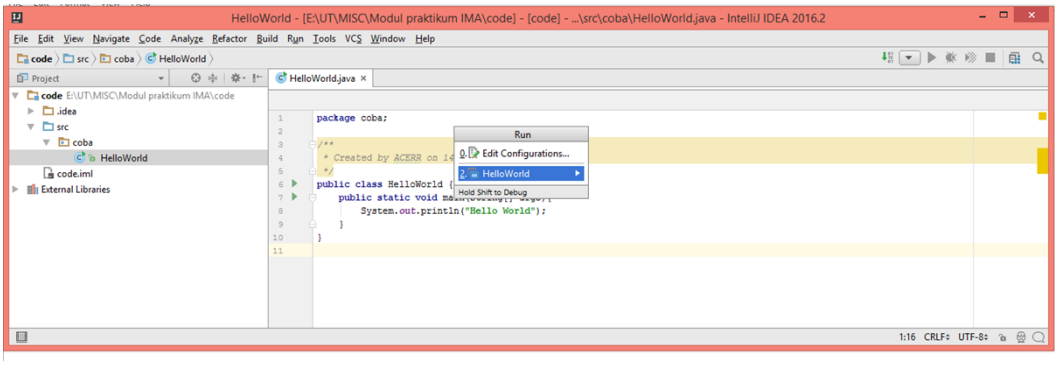
Gambar 1-15 Konfigurasi projek

5. Pilih main class dari project yang dibuat (klik tombol di sebelah kanan kotak Main class).



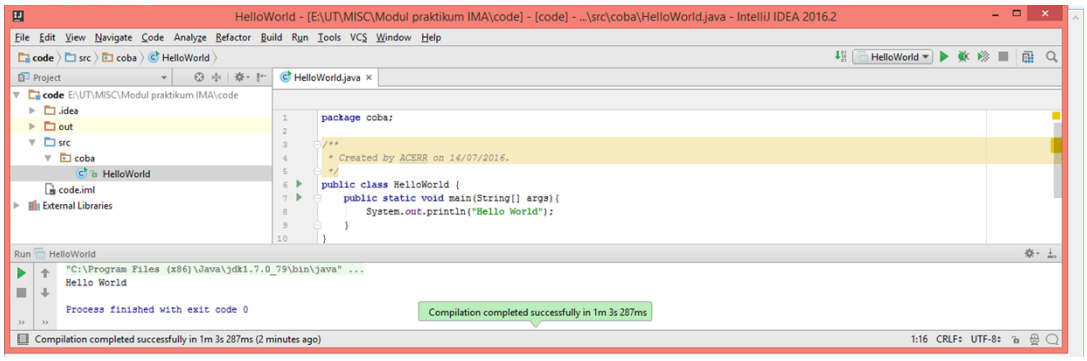
Gambar 1-16 Pemilihan main class

6. Jalankan program dengan meng-klik kanan kelas yang berisi main method.



Gambar 1-17 Running program

7. Hasil akhir akan terlihat pada console di bagian bawah jendela IntelliJ.



Gambar 1-18 Hasil program

1.3.5 Struktur Program pada Java

Perhatikan potongan program Hello World berikut.

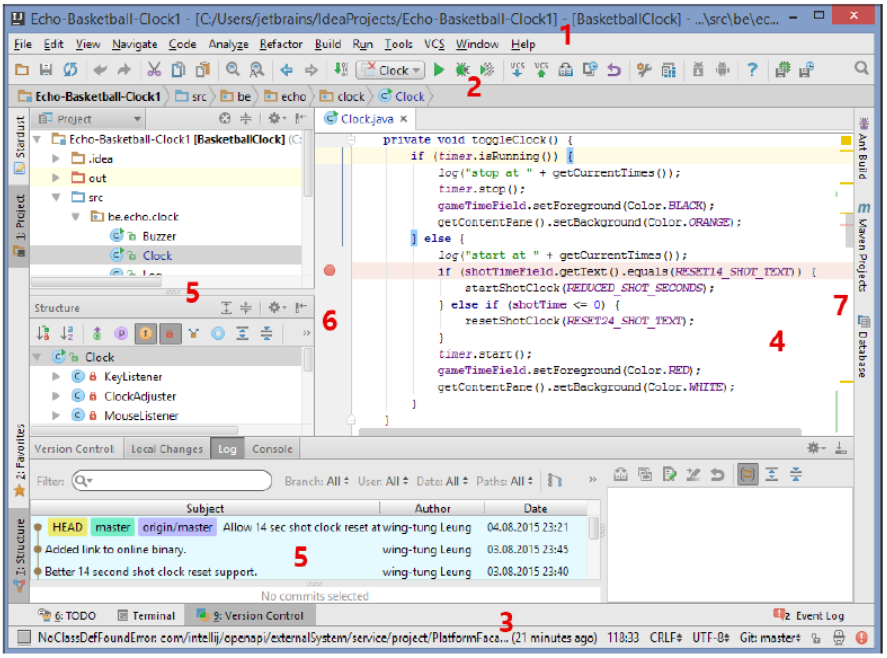
package coba;	(1)
class HelloWorld {	(2)
public static void main(String args[]){	(3)
System.out.println(“Hello World”);	(4)
}	(5)
}	

Program Hello World tersebut terdiri atas struktur berikut:

1. Package: Setiap kelas yang dibuat sebaiknya dimasukkan ke dalam satu package tertentu.Package ini dapat diumpamakan sebagai kontainer tempat menyimpan semua kelas yang dibuat.
2. Kelas: Suatu program Java dapat terdiri atas beberapa kelas, deklarasi kelas ditandai dengan keyword class. Nama kelas dituliskan sesudah keyword class, diawali dengan huruf besar. Apabila nama kelas terdiri dari beberapa kata, kata berikutnya ditulis bersambung dan awal kata diawali dengan huruf besar. Pada praktikum Implementasi Algoritma, kelas yang dibuat hanya satu kelas untuk sebagian besar praktikum, atau paling banyak dua kelas.
3. Main method: main merupakan method utama, walaupun suatu kelas atau aplikasi Java dapat terdiri atas beberapa method, salah satu method tersebut harus meupakan method main, dengan bentuk seperti yang terlihat pada contoh program Hello World.
4. Keluaran: System.out merupakan objek keluaran standar pada Java. Hasil dari objek ini akan ditampilkan pada command line. Method System.out.println akan menampilkan (mencetak) teks yang berada di antara tanda kutip (tapi tidak mencetak tanda kutip tersebut). Teks tersebut merupakan argumen bagi method ini. Setelah mencetak teks, kursor akan diposisikan pada baris baru dari command line.
5. Blok program: Suatu blok program pada Java akan ditandai dengan kurung kurawal buka ({) dan diakhiri dengan kurung kurawal tutup (}). Blok program dapat berupa bagian dari class, method maupun percabangan dan perulangan.

1.3.6 Lingkungan program IntelliJ

Kode yang dibuat dalam bahasa Java bersifat *case sensitive* yang berarti huruf besar dan huruf kecil berpengaruh. Berikut ruang lingkup dari jendela IntelliJ.



Gambar 1-19 Jendela IntelliJ

Perintah yang sering digunakan pada IntelliJ diletakkan di menu dan toolbars. Selain itu, perintah juga ditampilkan pada menu pop-up. Area pada jendela IntelliJ adalah:

1. Navigation bar
2. Tab dari file-file yang terbuka
3. Status bar: Mengindikasikan status dari project, IDE dan berbagai pesan informasi dan peringatan lainnya.
4. Editor tempat source code diketikkan.
5. Tool window: Memungkinan kita meng-eksplorasi, juga navigasi kode dan file, juga tempat untuk memperlihatkan hasil pencarian dan debug, serta lainnya.
6. Left gutter – merupakan garis vertikal yang menunjukkan break point, dan navigasi pada hirarki code seperti mencari definisi/ deklarasi perintah.
7. Right gutter: Menunjukkan error, warning dan status lainnya. Kotak di pojok kanan atas menunjukkan status keseluruhan dari suatu file.

1.3.7 Comment

Baris komentar tidak akan dieksekusi oleh komputer. Terdapat tiga jenis komentar pada Java; single-line, multi line dan documentation:

1. Single-line

   Bentuk komentar: //

   Merupakan end-of-line comment. Komentar akan berakhir pada ujung baris tempat komentar berada (baris berikutnya tidak termasuk komentar). Komentar jenis ini tidak harus berada pada awal baris, bisa juga berada di tengah baris.
2. Multiline

   Bentuk komentar: /*

   */

Merupakan komentar tradisional dari bahasa C/ C++. Semua baris perintah yang berada di antara pembatas komentar ini tidak akan dieksekusi.

3. *Documentation* (Javadoc comment)
   Bentuk perintah: /**
   */
   Merupakan komentar yang berasal dari Java. Seperti komentar multiline, semua baris perintah diantara tanda komentar tidak akan dieksekusi.

1.3.8 Escape Character

Di dalam Java, terdapat beberapa khusus yang penulisannya singkat tapi sangat berguna dalam program. Karakter tersebut disebut escape character.

Tabel 1-2 Escape sequence pada Java

Escape Sequence		Keterangan
\n	Newline	Kursor berada pada awal baris berikutnya
\t	Tab horisontal	Memindahkan kursor pada tab berikutnya
\r	Carriage retrun	Meletakkan kursor di awal baris asal, tidak pindah ke baris berikutnya. Karakter yang dikeluarkan sesudah perintah ini akan menimpa karakter yang sudah ada sebelumnya
\\	Backslash	Mencetak backslash
\”	Double quote	Mencetak kutip dua

Modul 2 : Tipe Data, Variabel dan Operator pada Java

2.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengenal konsep tipe data primitif.
2. Mengenal operator pada Java.
3. Dapat melakukan type casting.
4. Mengenal konsep variabel dan konstanta.

2.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software IntelliJ IDEA yang telah terinstall pada masing-masing PC

2.3 Dasar Teori

2.3.1 Tipe Data

Java merupakan bahasa yang sangat ketat mengatur tipe data. Dalam Java, setiap variabel dan ekspresi harus memiliki tipe data, dan tipe ini didefinisikan dalam aturan yang ketat. Selanjutnya, setiap *assignment*, baik yang berupa *assignment* langsung maupun yang berasal dari melewatkan parameter dalam method, akan di-cek kompatibilitas tipenya.

Dalam Java, variabel dapat memiliki tipe primitif maupun *reference*. Variabel bertipe primitif hanya dapat menyimpan satu nilai sesuai dengan tipe yang dideklarasikan dalam satu waktu. Variabel dengan tipe *reference* menyimpan lokasi objek pada memori, dengan demikian variabel tersebut merujuk pada suatu objek dalam program yang dijalankan. Variabel tipe reference akan dibahas pada pertemuan mengenai tipe data objek.

2.3.1.1 Tipe Data Primitif

Terdapat delapan tipe data primitif dalam Java, yaitu byte, short, int, long, char, float, double dan boolean. Kedelapan tipe ini dapat dikelompokkan dalam empat grup:

- Integer: Meliputi byte, short, int dan long. Merupakan angka bertanda (plus atau minus).
- Floating-point: Meliputi float dan double, menyatakan angka dalam bentuk pecahan.
- Character: Hanya terdiri atas char, merepresentasikan simbol dalam set karakter (huruf dan angka). Java menggunakan metode unicode untuk merepresentasikan tipe data ini.
- Boolean: Hanya terdiri atas boolean, merepresentasikan nilai true atau false.

Untuk lebih jelasnya, perhatikan tabel tipe data dasar berikut.

Tabel 2-1 Tipe data primitif pada Java

Tipe	Panjang	Jangkauan
long	64	-9.223.372.036.854.775.808 sampai 9.223.372.036.854.775.807
int	32	-2.147.483.648 sampai 2.147.483.647
short	16	-32.768 sampai 32.767
byte	8	-128 sampai 127
Double	64	4.9e-324 sampai 1.8e+308
float	32	1.4e-045 sampai 3.4e+038

2.3.1.2 Literal

Nilai konstan pada Java disebut dengan literal. Terdapat literal integer, floating-point, boolean, character, dan string.

a. Literal integer

Semua nilai angka merupakan literal integer dalam Java. Selain mengenal angka desimal, Java juga menyediakan angka oktal dan heksadesimal. Setiap literal integer memiliki tipe integral (int), jadi, pada dasarnya literal ini harus di-assign pada variable bertipe sama (integer). Jika tipe variabel bukan integer, literal tetap dapat di-assign selama masih dalam range tipe variabel tersebut.

Tabel berikut menunjukkan jenis literal integer dan *assignment*-nya.

Tabel 2-2 Jenis literal integer

Jenis	Keterangan
Desimal	Menyatakan bilangan basis sepuluh (1 – 10) Contoh: 1, 390, 42
Oktal	Menyatakan bilangan basis delapan, pada Java ditandai dengan angka 0 di depan bilangan Contoh: 01, 03, 06
Heksadesimal	Menyatakan bilangan basis enam belas, pada Java ditandai dengan angka 0-kali (0x). Contoh: 0x4, 0xA, 0x12
Assignment	int: int data = 10 short/ byte: short data2 = 10 (masih tidak masalah karena dalam range short) long: long data3 = 10L (menandakan tipe long) char: char data4 = ‘10’ (masih dalam range char)

Sumber: Schildt 2007

b. Literal floating-point

Angka floating-point menyatakan nilai decimal dengan bagian pecahannya. Nilai ini dapat dinyatakan baik dalam bentuk standar maupun notasi ilmiah. Tipe data default bagi floating-point dalam Java adalah double, jika ingin menyatakan dalam float harus ditambahkan F/f di belakang bilangan.

Contoh:

- Notasi standar: 2.0; 5.8; 125.258
- Notasi ilmiah: 2.523651E12, 4.56323E-5
- Assignment float: float bil = 2.34F

c. Literal boolean

Hanya memiliki nilai true atau false, tidak memiliki padanan 0 atau 1. Digunakan pada variabel boolean, atau sebagai ekspresi pada operator boolean.

d. Literal character

Merupakan set karakter Unicode. Dapat dikonversikan ke dalam bentuk integer dan dimanipulasi oleh operator integer (seperti ditambah atau dikurang). Literal ini ditulis diantara kutip tunggal. Semua karakter ASCII dapat langsung ditulis sebagai literal ini, untuk yang tidak memungkinkan, digunakan *escape sequence*.

128 karakter pertama dari Unicode dan ASCII adalah sama.

Contoh:

- Notasi literal: ‘a’, ‘12’, ‘@’
- Notasi dengan escape sequence: ‘\’’, ‘\n’
- Notasi dengan octal atau heksadesimal: ‘\141’ (‘a’), ‘\u0061’ (\u menandakan heksadesimal, ‘\u0061’ adalah notasi ISO-Latin-1 untuk ‘a’)

e. Literal string
Literal string ditandai dengan tanda kutip dua (“)
Contoh:
“Hello World”, “two\nlines”, “\’Dalam tanda kutip\’”

2.3.1.3 Assignment

Assignment adalah perintah untuk memasukan sebuah nilai ke dalam variabel. Operator *assignment* adalah tanda “sama dengan (=)”. Bentuk umum dari operator adalah:

var = expression

Tipe data dari *var* harus kompatibel dengan tipe *expression*.

```
package praktikum2;

class Assigment {
   public static void main (String args[]){
        int A = 5; // assignment
        System.out.println(A);
   }
}
```

Apakah Outputnya?

……

Kode baris di atas akan melakukan pemberian nilai (*assignment*) pada variabel A dengan 5, dan menampilkan kembali isi dari variabel A.

2.3.1.4 Operator

Operator dalam bahasa Java pada dasarnya dapat dibagi ke dalam empat bagian: aritmatika, *bitwise*, relasional dan logika.

1. Operator Aritmatika

   Penggunaan operator aritmatika pada bahasa Java sama seperti dalam aljabar. Operan dari operator ini harus dalam tipe numerik, tidak bisa dalam boolean. Namun, tipe char dapat digunakan sebagai operan pada operator ini karena pada Java, char merupakan subset dari int. Tabel berikut menampilkan daftar operator aritmatika pada Java.

Tabel 2-3 Operator aritmatika

Jenis	Keterangan
+	Penjumlahan
-	Pengurangan (juga untuk menyatakan minus, suatu operator uner)
*	Perkalian
/	Pembagian
%	Modulus
++	Increment
+=	Assignment penjumlahan
-=	Assignment pengurangan
*=	Assignment perkalian
/=	Assignment pembagian
%=	Assignment modulus
--	Decrement

Sumber: Schildt 2007

2. Operator *Bitwise*

Operator *bitwise* melakukan operasi pada bilangan integer (int, long, byte, short dan char) per bit. Jadi, operasi akan dilakukan pada bit-bit yang menyusun suatu bilangan, misal nilai 10 akan direpresentasikan oleh bit 1010. Yang perlu diingat, bilangan integer pada Java merupakan bilangan bertanda (*signed*), jadi bilangan negatif dan positif harus dibedakan. Untuk menyatakan bilangan negatif pada Java dilakukan operasi komplemen 2 dengan cara menginversikan bit suatu bilangan dan menambah satu setelahnya.

Tabel 2-4 Operator bitwise

Jenis	Keterangan
~	Bitwise unary NOT
&	Bitwise AND
|	Bitwise OR
^	Bitwise Exclusive OR
>>	Shift right
>>>	Shift right zero fill
<<	Shift left
&=	Bitwise AND assignment
|=	Bitwise OR assignment
^=	Bitwise Exclusive OR assignment
>>=	Shift right assignment
>>>=	Shift right zero fill assignment
<<=	Shift left assignment

Sumber: Schildt 2007

3. Operator Relasional

Operator relasional menyatakan hubungan antara satu operan dengan operan lainnya. Lebih jelasnya, operator ini menyatakan kesetaraan dan urutan (lebih besar, lebih kecil). Keluaran dari operasi ini adalah nilai boolean.

Tabel 2-5 Operator relasional

Jenis	Keterangan
==	Sebanding dengan (sama dengan)
!=	Tidak sama dengan
>	Lebih besar dari
<	Lebih kecil dari
>=	Lebih besar atau sama dengan
<=	Lebih kecil atau sama dengan

Sumber: Schildt 2007

4. Operator Logika Boolean

Hanya bekerja dengan nilai boolean, dan menghasilkan nilai boolean juga.

Tabel 2-6 Operator logika boolean

Jenis	Keterangan
&	Logika AND
|	Logika OR

^	Logika XOR
||	Short-circuit OR
&&	Short-circuit AND
!	Logika NOT
&=	Assignment AND
|=	Assignment OR
^=	Assignment XOR
==	Sama dengan
!=	Tidak sama dengan
?:	Operator ternary if-then-else

Sumber: Schildt 2007

5. Urutan operator

Tabel berikut menyatakan urutan operator, dari yang paling tinggi (jadi harus dilaksakan terlebih dahulu) sampai yang paling rendah. Jika operator dengan tingkatan yang sama ada dalam satu baris, maka urutan pengerjaan adalah dari kiri ke kanan.

Tabel 2-7 Urutan operator

Paling tinggi			
( )	[ ]	.	
++	--	~	!
*	/	%	
+	-		
>>	>>>	<<	
>	>=	<	<=
==	!=		
&			
^			
|			
&&			
||			
?:			
=	op=		
Paling rendah			

Sumber: Schildt 2007

Berikut contoh kode untuk beberapa operator.

```
package praktikum 3;

class DemonstrateOp {
  public static void main(String args[]){
    //Contoh operator aritmatika
    System.out.println("Operator aritmatika pada
integer");
    int a = 1 + 1, b = a * 3, c = b / 4;
    System.out.println("a = " + a);
    System.out.println("b = " + b);
    System.out.println("c = " + c);
    System.out.println("Operator modulus");
    int x = 8;
    System.out.println("x = " + x);
    System.out.println("x mod 10 = " + x%10);
```

Apakah Outputnya?

……

……

……

……

……

```
   //Contoh increment / decrement
   System.out.println("Operator modulus");
   int e = 3, f = ++e, g = f--;
   System.out.println("e = " + e);
   System.out.println("f = " + f);
   System.out.println("g = " + g);
   //Shift right
   int i, num = 0xFFFFFFE;
   for(i=0; i<4; i++) {
       num = num << 1;
       System.out.println(num);
   }
   //Operator logika Boolean
   boolean bool1 = true, bool2 = false;
   boolean bool3 = bool1 | bool2;
   boolean bool4 = bool1 ^ bool2;
   boolean bool5 = (!bool1 & bool2) | (bool1 &
!bool2);
   System.out.println("  bool1 = " + bool1);
   System.out.println("  bool2 = " + bool2);
   System.out.println("  bool1|bool2 = " + bool3);
   System.out.println("  bool1^bool2 = " + bool4);
   System.out.println("!bool1 & bool2| bool1 &
!bool2 = " + bool6);
   }
}
```

2.3.1.5 Compound Assignment

Compound assignment merupakan kombinasi dari operator yang bertujuan untuk merperpendek suatu baris perintah.

Simbol	Sama Dengan
A += 5	A = A +5
A -= 5	A = A – 5
A /= 5	A = A / 5
A *= 5	A = A * 5
A %= 5	A = A % 5

Simbol	Sama Dengan
A++	A = A +1
++A	A = A + 1
A--	A = A - 1
--A	A = A - 1

Perhatikan A++ dan ++A. Memang fungsinya sama, yaitu menambahkan satu ke variable A. Tetapi ada perbedaannya pada saat pemakaian, yaitu urutan eksekusinya dalam suatu perintah. Begitu juga untuk A—dan –A. Untuk lebih jelasnya perhatikan contoh kode berikut.

```
package praktikum 4;

Class DemoIncr {
   public static void main (String args[]){
      int A = 5;
      int B = A++;
      System.out.println("Nilai A " + A);
      System.out.println("Nilai B " + B);
      A = 6;
      B = ++A;
      System.out.println("Nilai A " + A);
     System.out.println("Nilai B " + B);
  }
}
```

Apakah Outputnya?

……

……

……

2.3.1.6 Inisialisasi Variabel

Sebuah variabel bisa diartikan sebagai sebuah penampung. Kita bisa menempatkan nilai apa saja ke variabel tersebut. Proses penempatan tersebut dinamakan assignment, sementara penempatan nilai pertama pada variabel dinamakan inisialisasi. Namun, sebelumnya variabel harus dideklarasikan terlebih dahulu. Bentuk dasar deklarasi variabel pada Java adalah sebagai berikut:

type identifier [=value][, identifier[=value] …;

Type dapat berupa tipe data primitif atau nama kelas atau interface. *Identifier* adalah nama variabel, juga digunakan pada method dan kelas. Aturan pemberian nama *identifier* pada Java adalah sebagai berikut:

1. *Identifier* dapat terdiri atas huruf, angka, garis bawah (_) dan tanda dolar ($), namun tidak boleh diawali dengan angka.
2. Tidak boleh terpisah atau mengandung simbol (seperti ? atau %)
3. Tidak boleh merupakan reserved word

Sementara konvensi penulisan identifier dalam Java adalah sebagai berikut:

1. Variabel dan method diawali dengan huruf kecil. Jika identifier terdiri atas lebih dari satu suku kata, gunakan aturan “camel case” dimana huruf awal kata berikutnya dijadikan huruf besar.
2. Nama kelas, sebaliknya, diawali dengan huruf besar.
3. Sebaiknya tidak menggunakan tanda dolar karena biasanya merupakan nama yang dibuat otomatis oleh suatu tools (tapi tidak berarti dilarang)

Inisialisasi variabel dapat dengan cara memasukkan nilai literal langsung ke dalam variabel, dapat juga berupa nilai dari variabel lainnya, atau nilai kembalian dari suatu method (diterangkan pada bab lain). Untuk lebih jelasnya, perhatikan contoh berikut.

```
package praktikum 5;

class DemoVar {
  int i = 10;
  int x = i;
  public static void main (String args[]){
    System.out.println(“Isi dari variabel i = ” + i + “
dan isi dari variabel x = ” + x);
    x = 15;
    System.out.println(“Isi dari variabel i = ” + i + “
dan isi dari variabel x = ” + x);
   }
}
```

Apakah Outputnya?

……

……

Pada program diatas, variabel i diisi dengan nilai literal 10, sedangkan variabel x diisi dengan nilai dari variabel i. Walaupun awalnya kedua variabel memiliki nilai yang sama, namun perubahan nilai pada salah satu variabel tidak otomatis akan mempengaruhi variabel lainnya.

2.3.1.7 const (konstanta)

Selain variabel, dapat digunakan konstanta, yaitu nilai yang tidak akan berubah. Penulisan konstanta dalam bahasa Java adalah dengan menambahkan keyword final. Melakukan perubahan pada suatu konstanta akan memunculkan error. Untuk lebih jelasnya, perhatikan contoh berikut.

```
package praktikum 6;

class DemoConst {
  final int LUCKY_NUMBER = 7;
  final float PI = 3.14159;
```

Apakah Outputnya?

……

final char NEW_LINE = '\n';
 	
public static void main (String args[]){
System.out.println(LUCKY_NUMBER + "" + NEW_LINE);
System.out.println(PI + "" + NEW_LINE);
PI = 5;
}
}
	……

Konstanta ditulis dengan huruf kapital, jika terdapat lebih dari satu suku kata, maka dipisahkan dengan tanda garis bawah. Baris terakhir program akan menyebabkan program tidak bisa di-*compile*.

2.3.1.8 Type conversion and Casting

Seperti yang telah disebutkan pada sub-bab literal, kita dapat menempatkan nilai dengan tipe data yang berbeda dari variabel, selama nilai tersebut masih dalam range tipe data variabel (kompatibel). Pengubahan (konversi) tipe data yang masih berada dalam range yang kompatibel disebut konversi otomatis (*automatic conversion*). Selain itu, harus dilakukan casting terhadap data.

1. *Automatic Conversion*
   Automatic conversion dari literal dengan tipe data yang berbeda dari variabel dapat dilakukan jika:
   - Kedua tipe (bilangan dan variabel) kompatibel
   - Tipe data tujuan (variabel) lebih besar dari tipe data asal (literal)
2. *Casting*
   Casting dilakukan pada *assignment* dari dua tipe data yang tidak kompatibel, biasanya karena tipe data tujuan lebih kecil daripada tipe data asal (misal, memasukkan nilai int ke dalam variabel bertipe byte). Cara melakukan casting adalah dengan menuliskan tipe data yang di-cast di depan nilai:

   (target-type) value

   Untuk lebih jelasnya perhatikan contoh kode berikut.

```
package praktikum6;

class Conversion{
   public static void main (String args[]){
     byte b; int i = 153; double d = 112.45;
     System.out.println("Konversi int menjadi
byte");
     b = (byte) i;
     System.out.println("i dan b " + i + " " +
b);
     System.out.println("Konversi double
menjadi int.");
     i = (int) d;
     System.out.println("i dan d "+i+" " + d);
     System.out.println("Konversi double
menjadi byte");
     b = (byte) d;
     System.out.println("b dan d "+b+" "+ d);
   }
}
```

Apakah Outputnya?

……

……

……

2.3.2 Tips menulis Program

Berikut beberapa tips dalam menulis program:

1. Comment
   Comment diperlukan dalam sebuah program untuk memudahkan si *Programmer* itu sendiri. Setidaknya juga masih berguna untuk *programmer* lain jika ingin melanjutkan programnya.

2. Penjelasan singkat (about)
   Setiap program sebaiknya dibuat sebuah penjelasan singkat tentang program tersebut, misalnya nama program, nama pembuat, fungsi program, dan lain-lain.

3. Penulisan class
   Nama class merupakan nama dari file Java yang dibuat. Jadi, untuk class HelloWorld misalnya, maka nama filenya pun harus HelloWorld.java

4. Indentansi
   Dalam penulisan kode program ada baiknya jika memperhatikan pengindentasian/penjorokan ke dalam. Indentasi sangat berguna untuk membuat program yang mudah dibaca. Indentasi tidak akan mempengaruhi kerja compiler (dengan ataupun tanpa indentasi akan sama saja), namun program yang ditulis dengan indentasi yang baik akan lebih mudah ditelusuri.

5. Penulisan main
   Seperti telah disebutkan pada Bab 1, main merupakan bagian yang akan dieksekusi kali pertama oleh Java compiler. Penulisan main harus mengikuti urutan sebagai berikut:

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

   Urutan penulisan tidak boleh diubah, juga isi dari argument main (yang berada dalam kurung sesudah main) Namun, nama variabel args boleh saja diganti dengan nama lain, selama tidak merupakan keyword Java.

Modul 3 : PERCABANGAN

3.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami esensi percabangan.
2. Menyelesaikan kasus percabangan.
3. Mengetahui dan mengaplikasikan jenis-jenis percabangan.

3.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software IntelliJ IDEA yang telah terinstall pada masing-masing PC

3.3 Dasar Teori

3.3.1 Mengapa harus ada Percabangan?

Misalkan anda dihadapkan pada suatu persoalan seperti ini. Anda diminta membuat sebuah program sederhana dengan Java untuk bisa menghitung jumlah yang harus dibayar dari suatu transaksi dan jika jumlah pembayaran sudah mencapai nilai tertentu, maka pembeli akan mendapatkan bonus barang. Bagaimana kita membuat logika dari penyelesaian kasus tersebut? Oleh karena itu, dibutuhkan percabangan dengan kata jika (IF).

3.3.2 Percabangan

3.3.2.1 if

if digunakan jika terdapat hanya satu pengkodisian. Baris perintah sesudah if akan dilaksanakan jika kondisi pada if terpenuhi. Perhatikan kode berikut.

```
package praktikum 3_1

class DemoIf {
   public static main void (String args[]) {
      int A = 10;
      int B = 50;
         if (A > B)
            System.out.println("A > B");
   }
}
```

Apakah Outputnya?

……

Perhatikan, penulisan kode program sesudah kondisi pada if tidak menggunakan tanda titik koma (;) Jika baris perintah yang harus dilaksanakan jika kondisi terpenuhi lebih dari satu baris, maka gunakan tanda kurung kurawal ({ … }).

3.3.2.2 if - else

if…else merupakan percabangan jika terdapat 2 kondisi. Jika tidak memenuhi kondisi pertama maka akan masuk kondisi kedua. Perhatikan kode berikut.

```
package praktikum 3_2

class DemoIf_Else{
   public static void main (String args[]) {
      int A = 10;
      int B = 50;
      if (A > B)
```

Apakah Outputnya?

……

```
            System.out.println("A > B");
        else
            System.out.println("B > A");
    }
}
```

Pada percabangan tersebut, saat baris perintah if, dilakukan pengecekan, apakah kondisi dari if terpenuhi. Jika terpenuhi, maka blok perintah sesudah if akan dieksekusi, jika tidak, maka blok perintah else yang akan dieksekusi.

3.3.2.3 Nested if

Nested if adalah If yang bersarang, yaitu di dalam if ada if lagi. Perhatikan kode berikut.

```
package paktikum3_3

class NestedIF {
   public static void main (String args[]){
      int Basics = 10;
      int Phys = 90;
      int Bio = 90;

      if (Basics >= 70) {
        System.out.println("Basics passed");
        if (Phys >= 70) {
            System.out.println("Physics passed");
            if (Phys >= 70) {
                System.out.println("Bio passed");
            }
        }
      }
      else {
         System.out.println("You failed in
Basics");
      }
   }
}
```

Apakah Outputnya?

……

……

……

3.3.3 if dengan banyak kondisi

Digunakan jika kita mempunyai banyak syarat untuk satu kondisi. Perhatikan kode berikut.

```
package praktikum3_4

class MultiConditionIF {
   public static void main (String args[]){
      int Basics = 90;
      int Phys = 90;
      int Bio = 90;
      if (Basics > 70 && Phys > 70 && Bio > 70)
         System.out.println("You passed it all");
   }
}
```

Apakah Outputnya?

……

3.3.4 switch

Digunakan jika kita mempunya banyak syarat pada satu variabel. Perhatikan kode berikut.

```
package praktikum3_5

class DemoSwitch {
   int Bulan = 7;
   public static void main (String args[]){
     switch (Bulan) {
        case 1:
             System.out.println("Januari");
             break;
        case 2:
             System.out.println("Februari");
             break;
        case 3:
             System.out.println("Maret");
             break;
        case 4:
             System.out.println("April");
             break;
        case 5:
             System.out.println("Mei");
             break;
        case 6:
             System.out.println("Juni");
             break;
        case 7:
             System.out.println("Juli");
             break;
        case 8:
             System.out.println("Agustus");
             break;
        case 9:
             System.out.println("September");
             break;
        case 10:
             System.out.println("Oktober");
             break;
         case 11:
             System.out.println("November");
             break;
         case 12:
             System.out.println("Desember");
             break;
         default:
             System.out.println("Angka  yang  Anda
masukkan salah");
         }
   }
}
```

Apakah Outputnya?

……

Ekspresi yang dicek sesudah perintah switch (dalam program diatas adalah Bulan) harus bertipe byte, short, int atau char, dan nilai pada masing-masing case harus kompatibel dengan ekspresi tersebut. Setiap case juga harus berupa literal (tidak boleh variabel) dan nilainya harus berbeda satu dengan yang lain. Perintah default bersifat opsional, jadi bisa saja suatu switch tidak memiliki nilai default. Perintah break berfungsi untuk keluar dari blok switch.

3.4 Ternary If

Suatu perintah if-else dapat disingkat penulisannya. Perhatikan kode program di bawah berikut.

```
package praktikum3_6

class DemoIFAwal {
   public static voin main (String args[]){
      int Math = 10;
      int Final = 0;
```

Apakah Outputnya?

……

```
      if (Math >= 10)
         Final = 1;
      else
         Final = 0;
      System.out.println(Final);
   }
}
```

Dapat disingkat menjadi:

```
package praktikum3_7

class DemoTernaryIF {
   public static void main(String args[]){
      int Math = 9;
      int Final = 0;
      Final = Math >= 10 ? 1 : 0;

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

Apakah Outputnya?

……

Penyingkatan if menggunakan operator <kondisi> ? <nilai1> : <nilai2> disebut juga ternary operator. Operator ternary akan mengevaluasi kondisi sebelum tanda tanya, jika kondisi terpenuhi, maka nilai1 akan dikerjakan, jika tidak maka nilai2 yang akan dikerjakan.

3.5 Latihan

1. Buatlah program untuk menghitung total harga barang yang mengikuti aturan berikut:
   - Jika jumlah barang yang dibeli < 100 buah, maka harga per barang adalah Rp.10.000,00.
   - Jika jumlah barang yang dibeli ≥ 100 buah dan < 150, maka harga per barang adalah Rp.9.500,00.
   - Jika jumlah barang yang dibeli ≥ 150, maka hargar per barang adalah Rp.9.000,00.
2. Tuliskan program yang mengeluarkan nama bulan tertentu berdasarkan angka inputan user.

   Format tampilan:

   Masukkan angka: <mengisi angka>
   < selang 2 baris >
   Bulan ke <angka> adalah bulan <nama bulan ke-angka>

   Contoh:

   Masukkan angka: 4
   < selang 2 baris >
   Bulan ke 4 adalah bulan April
3. Buatlah sebuah program untuk menghitung nilai akhir dari suatu mata kuliah seorang mahasiswa. Terdapat daftar nilai UTS, UAS dan Tubes. Jika Rata-rata nilai semua berjumlah 75, maka mahasiswa tersebut lulus. Akan tetapi jika dengan rata-rata tersebut mahasiswa tersebut mendapatkan nilai dibawah 60 untuk Tubes, maka mahasiswa tersebut dinyatakan tidak lulus.

Modul 4 : PERULANGAN

4.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengerti esensi penggunaan perulangan.
2. Memilih bentuk perulangan yang benar dan tepat untuk kasus-kasus tertentu.
3. Mengaplikasikan bentuk-bentuk perulanganan tersebut ke dalam bahasa Java.

4.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software IntelliJ IDEA yang telah terinstall pada masing-masing PC

4.3 Dasar Teori

4.3.1 Mengapa harus ada Perulangan?

Apabila kita ingin menuliskan angka 1 sampai dengan 5 secara berurutan maka kita bisa saja menuliskan semua angka tersebut secara manual karena range yang ditulis masih kecil. Lain halnya apabila angka yang ingin kita tulis rangenya dari 1 s.d. 10000 apakah kita akan menulisnya secara manual 1, 2, 3, 4, …, 10000? Tentu tidak. Diperlukan suatu cara yaitu perulangan (looping). Perulangan digunakan untuk mengefisienkan waktu dan meringkas kode program dalam pengeksekusian sub-program yang sama. Hal yang terpenting dalam perulangan adalah harus ada kondisi berhenti.

4.3.2 Perulangan

Bentuk perulangan dalam Java ada tiga yaitu while, do while, dan for. Statement tersebut membentuk apa yang biasanya disebut sebagai kalang (*loop*). Kalang akan mengeksekusi secara berulang satu set instruksi yang sama sampai kondisi berhentinya terpenuhi.

4.3.2.1 while

While merupakan bentuk paling dasar dari statement loop Java. Perintah atau blok perintah dalam while akan diulang sampai ekspresi kontrolnya terpenuhi. Bentuk umum kode program dari while adalah sebagai berikut.

```
while (kondisi) {
  <perintah1>;
  <perintah2>;
  ...
  <perintahN>;
}
```

Kondisi merupakan suatu ekspresi boolean. Badan/ isi dari kalang akan dieksekusi selama kondisi tersebut bernilai benar. Ketika kondisi menjadi salah, program akan menjalankan baris kode yang berada sesudah kalang tersebut. Tanda kurung kurawal tidak diperlukan jika hanya terdapat satu perintah pada kalang. Perhatikan contoh kode berikut.

```
package praktikum4_1

class While {
   public static void main(String args[]) {
     int n = 10;
     while(n > 0) {
       System.out.println("angka ke- " + n);
```

Apakah Outputnya?

……

……

……

```
            n--;
            }
          }
}
```

Pada program diatas, kondisi yang harus terpenuhi adalah n > 0. Variabel n harus dideklarasikan dan diinisialisasi sebelum kondisi while di-cek. Terkadang, jika kondisi awal dari while tidak terpenuhi, maka isi dari while tidak akan dieksekusi sama sekali.

Kemudian, harus diperhatikan agar tidak terjadi perulangan tak terhingga (*infinite loop*) pada kalang. Harus dipastikan ada perintah yang menyebabkan kondisi tidak terpenuhi, sehingga dapat keluar dari kalang. Pada program di atas, perintah tersebut dipenuhi oleh decrement n-- yang akan mengurangi nilai n sehingga dapat memenuhi kondisi awal.

4.3.2.2 do-while

Ketika kondisi perulangan pada while tidak terpenuhi sejak awal, maka isi dari perulangan tidak akan dieksekusi sama sekali. Terkadang isi dari perulangan tersebut perlu untuk dieksekusi minimal sekali, walaupun kondisi awal untuk perulangan adalah salah. Bentuk perulangan tersebut adalah do-while. Bentuk umum kode program dari do-while adalah sebagai berikut.

```
do {
  <perintah1>;
  <perintah2>;
  ...
  <perintahN>;
} while (<kondisi>);
```

Bentuk umum tersebut dapat diartikan seperti berikut. Selama kondisi terpenuhi (bernilai true), maka lakukan perintah 1 sampai dengan N. Dengan kata lain, iterasi dari do-while akan dieksekusi terlebih dahulu sebelum kondisi dicek. Kondisi ini harus merupakan ekspresi Boolean. Perhatikan contoh kode berikut.

```
package praktikum4_2

class DoWhile {
   public static void main(String args[]) {
     int n = 10;
     do {
       System.out.println("Angka ke- " + n);
       n--;
     }while(n > 0);
   }
 }
```

Apakah Outputnya?

……

……

……

Program di atas akan menampilkan hasil yang sama persis dengan perulangan while sebelumnya. Namun, jika n diberi nilai awal 1, maka perulangan akan tetap dijalankan dahulu sekali (dan menghasilkan tampilan “Angka ke- 1”. Perintah untuk mengurangi *counter* (n) juga dapat diletakkan pada bagian while, seperti terlihat pada contoh kode berikut.

```
do {
      System.out.println("klik " + n);
      n--;
    } while(n > 0);
```

Perulangan do-while akan sangat berguna ketika menampilkan menu, karena biasanya setidaknya menu akan tampil setidaknya sekali.

4.3.2.3 for

Seperti pada bahasa pemrograman lain, Java memiliki bentuk perulangan for. Bentuk umum kode program dari for adalah sebagai berikut.

```
for (inisialisasi; kondisi; banyaknya perulangan) {
  <perintah1>;
  <perintah2>;
  …
  <perintahN>;
}
```

Pada perulangan for, inisialisasi akan menentukan nilai awal dari variabel kontrol pada perulangan. Kondisi berupa ekspresi Boolean yang menguji variabel pengontrol perulangan. Selama hasil tes masih bernilai benar, kalang loop masih terus berulang. Ekspresi iterasi menentukan bagaimana variabel pengontrol perulangan berubah pada setiap iterasi. Untuk lebih jelasnya, perhatikan contoh kode berikut.

```
package praktikum4_4;

class For {
    public static void main (String args[]) {
      int n;
      for (n=10; n>0; n--)
        System.out.println("Angka ke- " + n);
    }
}
```

Apakah Outputnya?

……

……

……

Biasanya, counter pada perulangan for (dalam contoh diatas, variabel n) dideklarasikan dan diinisialisasi di dalam perintah for, seperti terlihat pada contoh kode berikut.

```
public static void main (String args[ ]) {
       //int n tidak dideklarasikan pada baris ini
       for (int n=10; n>0; n--)
           System.out.println("Angka ke- " + n);
    }
```

Perlu diperhatikan, variabel yang dideklarasikan dalam suatu perulangan (for,while, do-while) hanya akan dikenali oleh perulangan tersebut, tidak akan bisa digunakan diluar perulangan. Kemudian, kontrol perulangan for dapat lebih dari satu variabel. Untuk lebih jelasnya, perhatikan contoh kode berikut.

```
package praktikum4_5;

class ForDuaVariabel {
   public static void main(String args[]) {
      int a, b;
      for(a=4, b=1; b<a; a--, b++) {
        System.out.println("a = " + a);
        System.out.println("b = " + b);
      }
   }
}
```

Apakah Outputnya?

……

……

……

4.3.2.4 for-each (*enhanced* for)

Selain bentuk for konvensional, Java menyediakan bentuk “for-each”, yang bisa digunakan pada JDK 5 keatas. Perulangan for-each akan melakukan iterasi pada kumpulan objek, seperti array, secara sekuensial dari awal sampai akhir (tidak boleh ada yang terlewat). Pada Java, bentuk for ini disebut juga dengan *enhanced* for. Bentuk umum kode program dari for adalah sebagai berikut.

```
for(type itr-var : collection) statement-block
```

Type menyatakan tipe data dan itr-var menyatakan nama variabel yang akan menerima elemen dari kumpulan data, satu-per satu, dari awal hingga akhir data. Pada setiapiterasi, elemen dari kumpulan data akan diambil dan disimpan pada itr-var. Kalang akan terus diulang sampai semua elemen pada kumpulan data diperoleh. Karena itr-var menerima nilai dari kumpulan data, type harus memiliki tipe data yang sama (atau kompatibel) dengan tipe data elemen yang disimpan pada kumpulan data. Untuk lebih jelasnya, perhatikan contoh kode berikut.

```
package praktikum4_6;

class ForEach {
  public static void main(String args[]) {
    int nums[] = {10,20,30,40,50,60,70,80,90,100};
    int sum = 0;
    for (int x : nums) {
      System.out.println ("Nilai x adalah: "
                          + x);
      sum += x;
    }
    System.out.println("Jumlahnya: " + sum);
    }
 }
```

Apakah Outputnya?

……

……

……

Pada program diatas, variabel x akan mengambil isi dari array nums, dari awal sampai akhir, secara berurutan. Nilai tersebut kemudian dijumlahkan dan hasilnya ditampilkan sesudah perulangan.

4.4 Latihan

1. Buatlah program login yang akan mengeluarkan pesan "Selamat datang!" jika memasukkan username="iflab" dan password="balfi". Kesempatan untuk memasukkan username dan password hanya 3 kali. Jika sudah lebih dari 3 kali maka program akan langsung berhenti dan mengeluarkan pesan "Anda penyusup!"

2. Tuliskan program untuk membuat sebuah pola angka. Inputan dari program adalah sebuah angka N. Jika N diberi nilai 5, maka output-nya adalah :

   1 2 3 4 5
   2 3 4 5 1
   3 4 5 1 2
   4 5 1 2 3
   5 1 2 3 4

3. Buatlah program untuk mencari nilai rata-rata, nilai tertinggi dan nilai terendah dari inputan yang dinputkan oleh user.

   Contoh:

   banyak data : 5

data ke-1 : 8
data ke-2 : 10
data ke-3 : 7
data ke-4 : 5
data ke-5 : 9

Nilai Rata-Rata = 7.8
Nilai Tertinggi = 10

Nilai Terendah = 5

Modul 5 : PERULANGAN LANJUTAN

5.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami konsep break dan continue.
2. Mengaplikasikan break dan continue dengan tepat.
3. Memahami dan mengaplikasikan perulangan bersarang.

5.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software IntelliJ IDEA yang telah terinstall pada masing-masing PC

5.3 Dasar Teori

5.3.1 Keluar dari perulangan

Tidak selamanya perulangan harus dimulai dari awal sampai kondisi perulangan berakhir. Terkadang, terdapat suatu kondisi dimana program harus keluar dari perulangan. Perintah yang dapat mengakomodir kemungkinan tersebut antara lain adalah break dan continue.

5.3.2 break

Sebelumnya pasti sudah mencoba break, yaitu pada percabangan dengan menggunakan switch. Perintah break juga dapat digunakan untuk keluar dari perulangan tanpa melihat kondisi perulangan tersebut. Ketika terdapat break di dalam suatu perulangan, perulangan tersebut akan dihentikan dan eksekusi program akan dilanjutkan pada perintah berikutnya yang berada sesudah perulangan. Perhatikan contoh kode berikut.

```
package praktikum5_1;

class BreakLoop {
   public static void main (String args[]) {
     for (int i=0; i <= 50; i++) {
        if (i == 10) break;
           System.out.println("i: " + i);
     }
     System.out.println("Perulangan selesai");
     }
}
```

Apakah Outputnya?

……

……

……

……

Dapat dilihat pada contoh kode di atas, walaupun seharusnya perulangan for dilakukan sampai 50, perintah break menyebabkan perulangan terhenti pada angka 10. Perintah break dapat digunakan pada jenis perulangan apapun pada Java.

5.3.3 break sebagai pengganti goto

Walaupun Java tidak mengenal dan menggunakan goto, break dapat digunakan untuk menggantikan fungsi goto. Bentuk umum kode program dari penggunaan break tersebut adalah sebagai berikut.

```
break label
```

Biasanya, label merupakan blok kode yang akan dituju oleh perintah break. Blok ini dapat berupa blok yang berdiri sendiri maupun blok yang merupakan target dari perintah lain. Ketika perintah break

dieksekusi, program akan keluar dari blok yang dimaksud. Blok ini harus mencakup perintah break, tapi tidak harus langsung mengakhirinya.

Untuk memberi nama blok, letakkan label diawal blok tersebut, diikuti dengan tanda titik dua (:). Label harus mengikuti aturan penulisan Java. Ketika perintah break mengacu pada suatu label, maka program yang akan dieksekusi merupakan baris perintah sesudah blok tersebut. Perhatikan contoh kode berikut.

```
package praktikum5_2;

class Break {
   public static void main (String args[ ]) {
      boolean t = true;
      pertama: {
        kedua: {
          ketiga: {
            System.out.println("Sebelum break");
            if (t) break second; // break keluar dari
blok kedua
            System.out.println("Perintah  ini  tidak
dijalankan");
            }
          System.out.println("Perintah   ini   tidak
dijalankan ");
        }
      System.out.println("Perintah   sesudah   blok
kedua.");
      }
   }
}
```

Apakah Outputnya?

……

……

……

……

Sumber:

Pada program diatas, masing-masing blok perintah (dibatasi dengan kurung kurawal) diberi label pertama, kedua dan ketiga. Perintah break menyebabkan program “meloncat” keluar, sampai akhir dari blok kedua, melewati dua perintah println().

5.3.4 continue

Perintah continue dapat digunakan untuk melewati perulangan yang digunakan. Perintah yang mengikuti continue akan di-bypass, tidak dikerjakan. Iterasi akan melakukan perulangan selanjutnya Perhatikan contoh kode berikut.

```
}
package praktikum5_3;

class DemoContinue {
   public static void main(String args[]) {
      for (int i = 0; i < 15; i++) {
         System.out.print (i + " ");
         if (i % 2 == 0) continue;
         System.out.println("");
         }
    }
```

Apakah Outputnya?

……

……

……

……

Dapat dilihat pada contoh kode di atas, dalam perulangan 0 sampai dengan 14, ketika bertemu dengan angka genap, maka perintah sesudah continue (mencetak baris baru) tidak akan dijalankan, program langsung melanjutkan pada perulangan selanjutnya.

5.3.5 Nested loop

Seperti pada bahasa pemrograman lainnya, terdapat perulangan bersarang pada Java. Jadi, suatu perulangan dapat berada dalam perulangan lainnya. Perhatikan contoh kode berikut.

```
package praktikum5_4;

class NestedLoop {
   public static void main(String args[]) {
      int i, j;
      for(i = 0; i < 10; i++) {
        for(j = 0; j < i; j++)
           System.out.print("+");
        System.out.println();
        }
   }
}
```

Apakah Outputnya?

……

……

……

……

Modul 6 : ARRAY SATU DIMENSI

6.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami bentuk array dalam algoritma dan penerapannya dalam Java.
2. Memahami penggunaan dan pengimplementasian array satu dimensi.

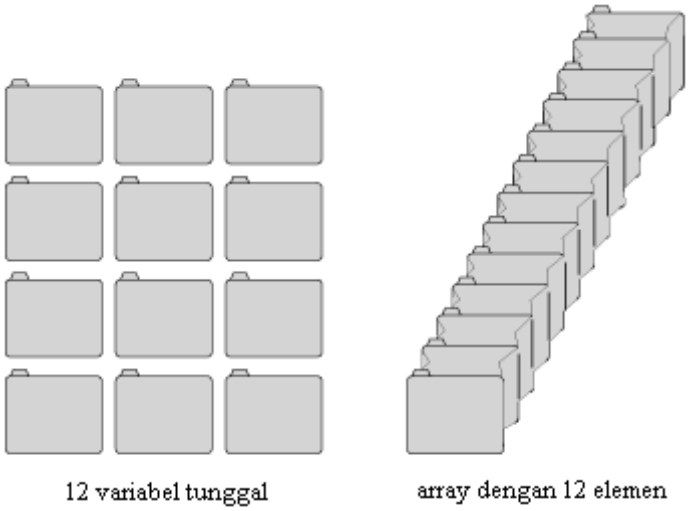
6.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software IntelliJ IDEA yang telah terinstall pada masing-masing PC

6.3 Dasar Teori

6.3.1 Apa itu Array?

Array merupakan sekelompok data sejenis yang disimpan ke dalam suatu variabel yang mana variabel tersebut memiliki indeks untuk pengaksesan datanya. Berikut contoh ilustrasi antara variabel dengan array.



Gambar 6-1 Ilustrasi Array

Jika kita lihat pada gambar yang pertama, kita mendeklarasikan 12 buah variabel dengan nama yang berbeda tentunya untuk tujuan yang sama, misalnya untuk menyimpan total pengeluaran setiap bulan atau menyimpan identitas dari mahasiswa. Sekarang kita lihat gambar yang kedua, kita mendeklarasikan sebuah array dengan 12 element dan fungsinya sama untuk menyimpan total pengeluaran setiap bulan. Jika kita bandingkan tentunya akan lebih mudah jika kita mempergunakan array karena kita hanya mempergunakan satu buah variabel dalam prosesnya nanti, dan juga dalam implementasi ke dalam program kita tidak mungkin akan mendeklarasikan variabel sebanyak seribu buah untuk tujuan yang sama namun dengan menggunakan array untuk mendeklarasikan variabel sebanyak itu akan dapat dilakukan dengan mudah.

6.3.2 Deklarasi

Secara umum bentuk pendeklarasian tipe data array pada bahasa Java dapat dilakukan dengan format sebagai berikut.

Tipe_Data_Array Nama_Array [ukuran]

Layaknya variable, sebuah array juga memiliki tipe data dan perlu diperhatikan suatu array hanya bisa memiliki data yang bertipe sama saja. Tipe data dari suatu array bisa berupa integer, float, char, bahkan tipe objek. Tanda kurung [] pada pendeklarasian array di atas digunakan untuk menunjukkan jumlah elemen larik. Jika dideklarasikan berupa int x[10], maka x merupakan array yang berisi 10 elemen dengan tipe data integer. Selain itu, yang perlu diperhatikan lagi bahwa penghitungan elemen dari suatu array dimulai dari 0, bukan 1.
Namun, perlu diperhatikan bahwa pada Java, deklarasi array tidak otomatis menjadikan variabel array tersebut dapat digunakan, karena pada saat deklarasi hanya nama array saja yang terbentuk, sementara objek array belum terbentuk. Saat deklarasi, nilai array masih null, yang artinya tidak ada objek array tersebut. Agar terbentuk suatu objek array, perlu mengalokasikan array tersebut dengan keyword new. Perhatikan kode berikut.

Tipe_Data_Array Nama_Array [ukuran]; *array-var = new type[size]*

Pada kode program diatas, setelah melakukan deklarasi array, dilakukan alokasi pada variabel array tersebut dengan menggunakan new. Type pada bagian new harus memiliki tipe data yang sama dengan tipe data pada saat deklarasi array. Jika tidak dilakukan inisialiasi, otomatis akan diisi dengan nilai default masing-masing tipe data (misal, untuk tipe integer akan diisi nol).
Jadi, pembuatan array pada Java membutuhkan dua langkah; pertama deklarasi variabel array, kemudian alokasikan memori untuk variabel tersebut dengan menggunakan keyword new. Karena itu, proses alokasi ini bisa saja dilakukan pada baris program yang terpisah jauh dengan baris program deklarasi array.

6.3.3 Inisialisasi Array

Inisialisasi array dapat dilakukan langsung saat deklarasi array, dengan memasukkan serangkaian nilai yang dipisahkan oleh koma pada dalam kurung kurawal. Koma berfungsi memisahkan nilai antar elemen array. Jika melakukan insialisasi pada saat deklarasi, maka keyword new tidak diperlukan. Perhatikan contoh berikut.

```
package praktikum6_1;


class InitArray {
   public static void main(String args[]) {
     int jum_hari[] =
                   {31,28,31,30,31,30,31,31,30,31,30,31};
     System.out.println("Februari memiliki " + jum_hari[1] +
" hari.");

     //deklarasi array terpisah
     String nama_hari;
     nama_hari = new String[7];
     nama_hari[0] = "Senin";
     nama_hari[1] = "Selasa";
     }
 }
```

Apakah Outputnya?

……

……

……

Pada program diatas, terdapat dua cara deklarasi dan inisialisasi array. Yang pertama adalah dengan mendeklarasikan dan sekaligus melakukan inisialisasi array (pada array jum_hari), sementara pada

array nama_hari, deklarasi dilakukan terpisah dengan inisialisasi. Inisialisasi pada array nama_hari hanya dilakukan pada elemen pertama dan kedua aarray. Selain itu, inisialisasi array juga dapat dilakukan dengan menggunakan perulangan. Perhatikan contoh berikut.

```
package praktikum6_2;


import.java.util.Scanner;


class DemoArray {
   public static void main(String args[]){
      int[] a = new int[100];
      Scanner in = new Scanner (System.in);

      System.out.println("Masukkan banyaknya nilai : ");
      int x = in.nextInt(); //menerima input dari console

      for (int i = 0; i < x; i++) {
         System.out.println("Input angka ke - "+(i+1)+ " :" );
         a[i] = in.nextInt();
       }

      System.out.println("Angka yang di-input-kan: " );

      for (i = 0; i < x; i++) {
         System.out.println(a[i]);
      }
   }
}
```

Apakah Outputnya?

……

……

……

Perhatikan bahwa deklarasi array dapat dilakukan dengan meletakkan kurung siku pada tipe data (bukan hanya pada nama variabel). Untuk menerima input dari console, digunakan Scanner. Karena Scanner merupakan objek dari kelas util dari java (bukan pada kelas yang sedang dijalankan) maka Scanner tersebut harus di-import dengan perintah import setelah nama package. Berikut contoh lain program yang menggunakan array.

```
package praktikum6_3;

import.java.util.Scanner;

class AnotherArray{
   publlic static void main (String args[]) {
      int a[] = new int[100];
      int x, sum, ganjil, genap;
      Scanner in = new Scanner (System.in);
      System.out.println("Masukkan banyaknya nilai: ");
      x = in.nextInt();
      sum = ganjil = genap = 0;

      for (int i = 0; i < x; i++) {
        System.out.println("Input angka ke- "+(i+1)+":");
          a[i] = in.nextInt();
          sum += a[i];
          if (a[i] % 2 == 0)
            genap += 1;
          else
```

Apakah Outputnya?

……

……

……

……

……

……

```
            ganjil+=1;
      }
      System.out.println("Angka yang di-input-kan: " );
      for (i = 0; i < x; i++) {
        System.out.println(a[i]);
      }

      System.out.println("Total bilangan : " + sum );
      System.out.println("Jumlah bilangan ganjil : " + ganjil );
      System.out.println("Jumlah bilangan genap : "+ genap);

   }
}
```

6.4 Latihan

1. Buatlah sebuah program yang dapat menentukan bahwa bilangan-bilangan di dalam suatu array tersebut bilangan prima atau tidak.

   Contoh :

   Banyaknya bilangan : 3
   Masukkan bilangan ke-1 : 9
   Masukkan bilangan ke-2 : 11
   Masukkan bilangan ke-3 : 51
   Bilangan Prima : 11

2. Buatlah program yang dapat menentukan rata-rata, median, dan modus dari beberapa bilangan (Gunakan array)!

   Contoh:

   Banyaknya bilangan : 5
   Masukkan bilangan ke-1 : 20
   Masukkan bilangan ke-2 : 12
   Masukkan bilangan ke-3 : 3
   Masukkan bilangan ke-4 : 15
   Masukkan bilangan ke-5 : 20
   Rata–Rata : 14
   Median : 15
   Modus : 20

Modul 7 : ARRAY LANJUTAN

7.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami konsep array lebih dalam lagi.
2. Memahami dan mengimplementasikan array n dimensi.
3. Memahami dan mengimplementasikan ArrayList.

7.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software IntelliJ IDEA yang telah terinstall pada masing-masing PC

7.3 Dasar Teori

7.3.1 Array n Dimensi

Pada modul sebelumnya telah dibahas bagaimana menggunakan array, namun array tidak hanya bisa menampung nilai 1 dimensi saja. Array nyatanya juga bisa dibuat seperti matriks atau yang biasa disebut dengan array 2 dimensi. Array 2 dimensi ini merupakan bagian dari array n dimensi (multi dimensi). Lalu bagaimana mendeklarasikannya? Deklarasi dari array 1 dimensi dengan 2 dimensi tidak jauh berbeda, adapun cara deklarasinya sebagai berikut.

```
Tipe_Data_Array Nama_Array[ukuranbaris][ukurankolom]
```

Dalam Java, array multi dimensi (termasuk didalamnya array 2 dimensi) adalah array di dalam array. Hal ini sangat berpengaruh pada saat dilakukan alokasi memori untuk array multi dimensi. Alokasi memori yang diperlukan hanya pada dimensi pertama (dimensi paling kiri) dari array. Sisa dimensi dapat dialokasikan secara terpisah. Dengan demikian, alokasi memori bagi tiap dimensi dapat berbeda, tidak perlu sama. Untuk lebih jelasnya, lihat contoh kode berikut ini.

```
package praktikum7_1;

class InitArrayDuaD {
   public static void main(String args[]){
      int duaD[][] = new int[3][];
      duaD [0] = new int[2]; duaD [1] = new int[3];
      duaD [2] = new int[4];
      int i, j, k = 0;
      for(i=0; i<3; i++) {
         for(j=0; j<i+1; j++) {
             twoD[i][j] = k;
             k++;}
      /*Tampilkan isi array*/
      for(i=0; i<3; i++){
         for(j=0; j<i+1; j++)
             System.out.print(duaD[i][j] + " ");
         System.out.println();
         }
      }
}
```

Apakah Outputnya?

……

……

……

Untuk memperjelas penggunaannya, perhatikan contoh berikut.

```
package praktikum7_2;

import.java.util.Scanner;

class DemoArray2D() {
   public static void main(){
       int [][] matriks = new int [10][10]; // deklarasi maksimum array 2 dimensi
       int baris, kolom, i, j;
       Scanner in = new Scanner (System.in);

       System.out.println("Masukkan jumlah baris: ");
       baris = in.nextInt();

       System.out.println("Masukkan jumlah kolom: ");
       kolom = in.nextInt();

       for (i = 0; i < baris; i++) { //Pengisian array
           for (j = 0; j < kolom; j++) {
               System.out.println("Isi [" + (i+1) + "," + (j+1) + "]");
               matriks [i][j] = in.nextInt();
           }
       }

       System.out.println("Hasil transposnya: ");

      for (i = 0; i < kolom; i++) {
         for (j = 0; j < baris; j++) {
             System.out.println(matriks[i][j] + " ");
             }
           System.out.println();
         }
      }
}
```

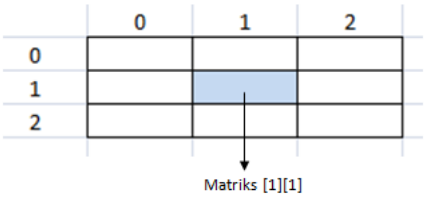
Apakah Outputnya?

……

……

……

Bila diilustrasikan, bentuk dari suatu array 2 dimensi adalah sebagai berikut.



7.3.2 Inisialisasi Array Multidimensi

Pada dasarnya, inisialisasi pada array multidimensi sama dengan inisialisasi pada array satu dimensi. Inisialisasi pada array multi dimensi dapat dilakukan pada masing-masing dimensi dengan menempatkan nilai awal dalam kurung kurawal untuk setiap dimensi. Untuk lebih jelasnya, lihat contoh kode berikut ini.

```
package praktikum7_3;


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

Apakah Outputnya?

……

……

……

7.3.3 ArrayList

ArrayList merupakan bagian dari data collection yang disediakan oleh Java API untuk menyimpan grup dari objek yang saling berhubungan. Pengaturan data di dalam ArrayList sama seperti pada array, dimana objek dengan tipe data yang sama disimpan secara berurutan. Namun, kita tidak perlu membuat kode program untuk pengaturan data tersebut, karena sudah disediakan oleh Java. Selain itu, dengan menggunakan ArrayList, alokasi data dapat dilakukan secara dinamis (dapat ditambah dan dikurangi, jika menggunakan array tidak bisa). Adapun cara deklarasi ArrayList adalah sebagai berikut.

```
ArrayList<T> nama_variabel = new ArrayList<T>
```

Perintah <T> merupakan tipe data yang digunakan untuk ArrayList. Misal, jika ingin membuat suatu ArrayList dari String, dituliskan ArrayList<String>. Seperti pada array, keyword new digunakan untuk mengalokasikan ArrayList pada memori. Method-method yang akan digunakan dalam ArrayList dapat dilihat pada Tabel 7-1.

Table 7-1 Method pada ArrayList

Method	Keterangan
add	Menambahkan elemen di ujung AraryList
get	Mengambil elemen pada indeks tertentu dalam ArrayList
remove	Menghapus elemen yang pertama kali ditemukan pada ArayList
size	Mengembalikan besar ArrayList

Method yang dipelajari pada praktikum ini terbatas untuk add, get dan remove. Penjelasan lebih mendalam mengenai ArrayList akan dipelajari pada mata kuliah Implementasi Struktur Data. Karena ArrayList berada pada kelas lain, maka harus di-import terlebih dahulu. Untuk lebih jelasnya, lihat contoh kode berikut ini.

```
Package praktikum7_4;

import java.util.ArrayList; // import ArrayList

class DemoArrayList {
   public static void main( String[] args ) {
       ArrayList<String> baju = new ArrayList<String>();
       baju.add( seragam");
       baju.add("gaun");
       baju.add(0,”kaos kaki");

       /*tampilkan isi array*/
       for ( int i=0; i < items.size(); i++ )
          System.out.print (baju.get( i ) + “ “ );

       baju.add(“gaun”); //menambahkan gaun diujung array
       baju.remove( "gaun" ); // menghapus elemen gaun
pertama yang ditemui
       /*tampilkan isi array*/
       for ( int i=0; i < items.size(); i++ )
          System.out.print (baju.get( i ) + “ “ );
       }
}
```

Apakah Outputnya?

……

……

……

Perintah add akan menambahkan elemen pada ujung array. Namun, jika ditentukan indeks spesifik, seperti pada perintah add (0, “kaos kaki”) maka elemen akan diletakkan pada tempat yang ditunjukkan oleh indeks (dalam hal ini, indeks pertama array). Perintah remove hanya menghapus elemen “gaun” yang ditemukan pertama kali.

7.4 Latihan

1. Buatlah program untuk menghitung total belanja pada suatu minimarket. Program akan menanyakan apakah masih ada lagi data yang akan ditambahkan, dan berhenti ketika kasir menyatakan tidak ada data lagi yang akan dimasukkan. Jika ternyata total belanja lebih dari Rp 150.000, maka akan diberikan diskon 5% (gunakan ArrayList untuk menyimpan data)

   Contoh keluaran:

   Masukkan jumlah belanja:
   55000
   Masih ada pembelanjaan (Y/T)
   Y
   Masukkan jumlah belanja:
   100000
   Masih ada pembelanjaan (Y/T)
   T
   Total belanja: 147250

2. Diberikan sebuah matriks (array 2 dimensi) integer yang berukuran m x n. Tuliskan algoritma untuk menentukan apakah didalam sebuah matriks tersebut ada baris yang semua elemennya 0.

3. Buatlah sebuah program untuk mencari hasil perkalian matriks. Inputan matriks (kolom, baris, dan isi matriks) terserah kepada user tapi dengan syarat kolom matriks pertama harus sama dengan matrik kedua.

Contoh:

```
Baris matrik 1: 3
Kolom matrik 1: 2
Baris matrik 2: 2
Kolom matrik 2: 1

Isi matrik 1
matrik1 [1,1] = 1
matrik1 [1,2] = 2
matrik1 [2,1] = 3
matrik1 [2,2] = 4
matrik1 [3,1] = 5
matrik1 [3,2] = 6

Isi matrik 2
matrik2 [1,1] = 1
matrik2 [2,1] = 2

Hasil perkalian matriks
Hasil [1,1] = 5
Hasil [2,1] = 11
        Hasil [3,1] = 17
```

Program 7- 10 mahasiswa.h

DAFTAR PUSTAKA

- Anonim. 2014. Modul Implementasi Algoritma. Fakultas Ilmu Terapan-Universitas Telkom, Bandung.
- Deitel, Paul J dan Deitel, Harvey M. 2012. Java How to Program 9th Ed. Prentice Hall,
- Hortsmann, Cay. 2010. Big Java: Compatible with Java 5,6 and 7, 4th edition. John Wiley&Sons,
- Schildt, Herbert. 2007. Java: The Complete Reference, 7th Ex. Mc Graw Hill,
- http://www.oracle.com/technetwork/java/index.html diakses tanggal 10 Juli 2016
- Nugroho Adi. 2011. Perancangan dan Implementasi Sistem Basis Data. Penerbit Andi. Yogyakarta