BAB I
DASAR PEMROGRAMAN JAVA

A. PENGENALAN JAVA

Java diciptakan oleh James Gosling dan rekan-rekannya di Sun Microsystems pada tahun 1995. Java menggunakan sintaks yang mirip dengan bahasa C dan C++, tetapi dengan fitur-fitur tambahan untuk memudahkan pengembangan aplikasi, seperti sistem pengelolaan memori otomatis, manajemen kesalahan, dan dukungan untuk pemrograman berorientasi objek.Java dirancang untuk tujuan umum (general-purpose) dan sepenuhnya menganut paradigma OOP (Object Oriented Programming).

OOP adalah paradigma pemrograman yang berbasis pada objek. Setiap kita membuat program, maka wajib hukumnya membuat objek terlebih dahulu.Berbeda dengan python yang merupakan bahasa Interpreter (membaca kode sumber baris perbaris dan dapat dijalankan langsung oleh komputer), Java merupakan bahasa compiler yang mana akan membaca seluruh kode sumber program dan menerjemahkannya ke dalam bentuk kode mesin yang dapat dijalankan secara langsung oleh komputer. Hasil dari proses ini adalah file yang disebut executable yang dapat dijalankan tanpa memerlukan program compiler.

Proses yang akan kita lakukan saat coding dengan java adalah:

1. Menulis kode program Java, yang akan menghasilkan file dengan extensi .java
2. Melakukan Compile dengan compiler (javac) dan akan menghasilkan file .class
3. File .class berisi *bytecode* yang mana merupakan kode yang dipahami JVM. kemudian JVM akan mengeksekusi kode tersebut dan program pun berjalan.

B. PERSIAPAN MEMBANGUN DAN MENJALANKAN JAVA

Beberapa tools yang diperlukan untuk membangun program dengan Java ialah SDK, IDE dan *build tools.*

1. **Software Development Kit (SDK**)

   SDK merupakan seperangkat alat pengembangan perangkat lunak yang digunakan untuk mempermudah pengembangan aplikasi dalam platform tertentu. Setiap platform biasanya menyediakan SDK khusus, misalnya Android dengan Android SDK-nya, iOS dengan iOS SDK-nya, ataupun Java dengan JDK-nya

JDK, JRE dan JVM sendiri terlihat sama karena ketiganya merupakan inti dari bahasa pemrograman Java. Meski terlihat sama, masing-masing dari ketiganya, punya peran sendiri-sendiri. JDK (**Java Development Kit**) adalah sebuah perangkat lunak yang menyediakan beberapa tools untuk pengembangan dan berkas binary yang diperlukan untuk proses kompilasi dari kode Java ke bytecode.

Selanjutnya, JVM atau **Java Virtual Machine** bertanggung jawab untuk melakukan konversi bytecode kedalam bahasa mesin. JVM juga menyediakan fungsi inti dari Java seperti memory management, garbage collection, security dan sebagainya. JVM disebut virtual karena memiliki antarmuka yang tidak bergantung pada sistem operasi dan perangkat keras yang menjadi dasar dari JVM itu sendiri.

Terakhir, JRE atau **Java Runtime Environment**, merupakan implementasi dari JVM yang menyediakan sebuah platform untuk menjalankan program. Berbeda dengan JDK dan JVM, JRE tidak menyediakan tools untuk pengembangan seperti compiler, debugger dan sebagainya. Tetapi JRE diperlukan untuk menjalankan program.

Pada dasarnya setiap JDK yang bisa kita gunakan, punya basis OpenJDK dan bersifat open-source. Yang membedakannya adalah bagaimana JDK tersebut didistribusikan. Contoh distribusinya bisa dari Oracle (Oracle JDK), OpenJDK Distribution atau Azul Zulu JDK.

2. Build Tools

Build tools adalah program atau utilitas yang digunakan oleh pengembang perangkat lunak untuk mengotomatiskan dan menyederhanakan proses pembangunan (build) perangkat lunak. Build tools membantu pengembang untuk mengelola dependensi, membangun (build), mengemas (package), dan mempublikasikan kode sumber program. Build tools juga dapat membantu dalam proses pengujian, integrasi, dan distribusi perangkat lunak. Beberapa contoh build tools yang populer di kalangan pengembang perangkat lunak meliputi Apache Maven, Gradle dan Ant. dan dari ketiganya, Gradle lah yang paling sering digunakan. Karna cukup fleksibel dalam membantu proses kompilasi.

3. Integrated Development Environment (IDE)

Proses pengembangan aplikasi tak lepas dari bantuan IDE. Memang tanpa IDE kita tetap bisa membuat program dengan text editor. Namun, fitur-fitur yang DItawarkan akan membuat proses pengembangan menjadi jauh lebih mudah dan efisien.

Pada umumnya IDE menyediakan beberapa fitur seperti text editor yang akan kita gunakan untuk menulis kode, tools untuk mengotomatisasi proses build dari program yang kita buat dan sebuah

debugger yang akan membantu kita mendeteksi dan memperbaiki kesalahan yang terdapat pada program.

Meskipun terdapat berbagai macam IDE yang mendukung java seperti IntelliJ IDEA, Android Studio, dan Eclipse namun pada moduli ini kita akan menggunakan Visual Studio Sode.

4. Instalasi JDK

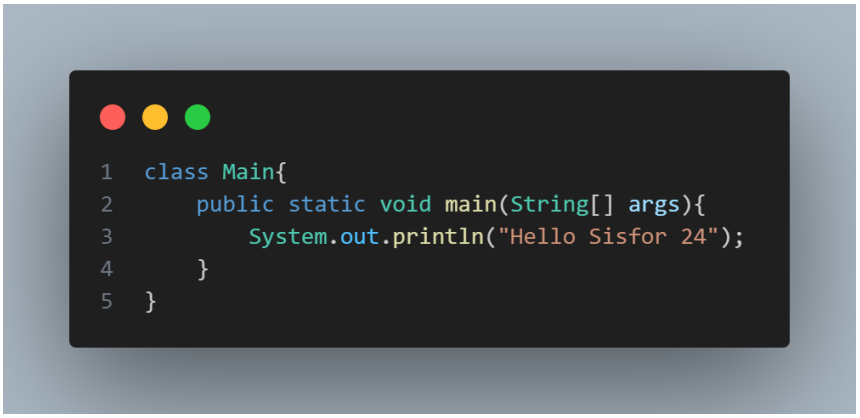
a. Silahkan download Java SDK (JDK) dari link berikut:
   a. **Windows**
      https://download.oracle.com/java/17/archive/jdk-17.0.12_windows-x64_bin.exe
   b. **macOs**
      https://download.oracle.com/java/17/archive/jdk-17.0.12_macos-aarch64_bin.dmg
b. Silahkan jalankan dan lakukan instalasi JDK sampai selesai.
c. Pergi ke lokasi folder instalasi JDK yang umumnya ada di Program File komputer dan buka folder tersebut.
d. Buka folder bin. Salin lokasi folder yang terletak di bagian atas folder. Umumnya C:\Program Files\Java\jdk-17\bin.
e. Tambahkan path tersebut ke Environment Variable.
f. Jalankan java --version untuk memastikan JDK telah terinstall

```
C:\Users\ADVAN>java -version
java version "17.0.12" 2024-07-16 LTS
Java(TM) SE Runtime Environment (build 17.0.12+8-LTS-286)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.12+8-LTS-286, mixed mode, sharing)
```

Jika hasilnya sama seperti diatas, maka JDK telah berhasil diinstal.

C. STRUKTUR DAN SYNTAKS JAVA

Struktur program Java secara umum dibagi menjadi 3 bagian:



1. **Import Library**
   Pada bagian ini kita melakukan import library yang dibutuhkan program.

2. **Bagian Class**
   Karena java merupakan bahasa dengan pure paradigma OOP, maka setiap program harus dibungkus di dalam class agar nanti bisa dibuat menjadi objek. Blok class dibuka dan ditutup dengan tanda {}.

3. **Method Main**
   Method main() atau fungsi main() merupakan blok program yang akan dieksekusi pertama kali. Method main() wajib kita buat agar program bisa dieksekusi.
   Lalu didalam method main(), terdapat statement atau expresi.

"**Statement dan ekspresi di Java harus diakhiri dengan titik koma(;)**"

Blok program merupakan kumpulan dari statement yang dibungkus menjadi satu. Blok program selalu dibuka dan ditutup dengan tanda kurung kurawal {}.

Komentar pada Java menggunakan :

1. Garis miring ganda **(//)** untuk komentar satu baris;
2. Garis miring bintang **(/*...*/)** untuk komentar yang lebih dari satu baris.

Penulisan **String** pada Java, harus diapit dengan tanda petik ganda seperti **"Hello World".** Apabila diapit dengan tanda petik tunggal, maka akan menjadi karakter (**char**).

Java bersifat ***case sensitive***, jadi nama tidak sama dengan Nama.
untuk gaya penulisan sendiri java menggunakan:

1. **camelCase** pada nama variabel, nama objek dan nama method
2. **PascalCase**, pada nama class
3. **ALL_UPPER**, pada nama konstanta.

D. TIPE DATA, VARIABEL DAN OPERATOR

1. TIPE DATA

Berikut macam-macam tipe data pada Java :

Tipe Data	Penjelasan	Contoh
char	karakter	'Z'
float	bilangan desimal	3.2
double	bilangan desimal dengan kapasitas lebih besar	13.12
String	kumpulan dari karakter yang membentuk teks	"Hello World"
boolean	tipe data yang bernilai true atau false	true

2. VARIABEL

Variabel merupakan representasi dari alamat memori yang digunakan untuk menyimpan nilai dari data. Java adalah bahasa pemrograman *Strongly Typed* dan *Statically Typed*, artinya untuk menggunakan variabel, haruslah didahului deklarasi tipe data dan nama variabelnya.

Syntax pembuatan variabel di Java

<tipe data> namaVariabel;

atau

<tipe data> namaVariabel = <nilai variabel>;

3. OPERATOR

a. Operator Assignment (Penugasan)

Assignment operator adalah operator yang digunakan untuk mengatur alamat dan nilai dari variabel, operator ini ditandai dengan simbol sama dengan **"="**.

b. Operator Aritmatika

Nama	Simbol
Penjumlahan	+
Pengurangan	-
Perkalian	*
Pembagian	/
Sisa Bagi(Modulo)	%

c. Operator Unary

Nama	Simbol
Increment, peningkatan nilai sebesar 1 poin	++
Decrement, pengurangan nilai sebesar 1 poin	--
Pengisian dan Penambahan	+= atau =+
Pengisian dan Pengurangan	*= atau =*
Pengisian dan Pembagian	/= atau =/
Pengisian dan Sisa Bagi	%= atau =%

d. Operator Perbandingan

Nama	Simbol
Lebih Besar	>
Lebih Kecil	<
Sama Dengan	==
Tidak Sama Dengan	!=
Lebih Besar Sama Dengan	>=
Lebih Kecil Sama Dengan	<=

e. Operator Logika

Nama	Simbol
Logika AND	&&
Logika OR	||
Negasi/Kebalikan	!

f. Operator Bitwise

Nama	Simbol
AND	&

OR	|
XOR	^
Nagasi	~
Left Shift	<<
Right Shift	>>
Left Shift (Unsigned)	<<<
Right Shift (Unsigned)	>>>

g. Operator Ternary

Simbol ternary operator menggunakan tanda tanya **(?)** dan titik-dua **(:)** untuk memisah jawabannya.

Contoh :

String status = nilai>90 ? “Lulus” : “Tidak Lulus”;

E. CONTROL FLOW

I. if else

Logika dasar *If Else* adalah jika kondisi pada if terpenuhi, maka perintah di dalam if akan dijalankan, jika *if* tersebut memiliki else maka perintah *else* akan dijalankan ketika kondisi *if* tidak terpenuhi.

Contoh If-else:

```
public static void main(String[] args){
 int score = 80;
 char grade;
 if (score >= 80) {
   grade = 'A';
 } else if (score >= 75){
   grade = 'B';
 } else if (score >= 65){
   grade = 'C';
 } else {
   grade = 'E';
 }
 System.out.println("Your grade is " + grade);
}
```

2. **SWITCH CASE**

Switch Case Statement adalah bentuk penyeleksian kondisi dengan satu variabel uji. Switch berfungsi menyatakan variabel yang akan diuji, kemudian case yang tersedia akan dibandingkan dengan variabel yang ada di switch, jika salah satu case terpenuhi maka statement yang didalamnya akan dijalankan hingga terdapat break atau return statement.

Contoh switch-case

```
public static void main(String args[]){
    int month = 9;
    switch (month){
      case 12:
        System.out.println("December");
        break;
      case 9:
        System.out.println("September");
        break;
      default:
        break;
    }
  }
```

3. **ternary operator**

Untuk mempersingkat penulisan If Else Statement, Java menyediakan *Ternary Operator (? :)*.

```
public static void main(String args[]){
    int score = 75;
    char grade = score >= 80 ? 'A' : 'B';
    System.out.println("Your Grade is " + grade);
  }
```

4. **try catch**

Dalam alur sebuah program, terdapat sebuah kejadian yang disebut *Exception. Try Catch Statement* berfungsi khusus untuk mengatasi *exception,* Try Catch Statement terdiri dari blok *try, catch,* dan *finally*.

Berikut contoh cara menangani exception dengan try catch statement

```
public static void main(String args[]){
    try{
      int result = 10/0;
    }catch(Exception e){
```

```
      System.out.println(e.toString());
    } finally {
      System.out.println("end");
    }
  }
```

F.LOOPING

1. for loop

For loop adalah perulangan yang jumlah perulangannya bisa dihitung, perulangan ini terdiri dari inisialisasi, kondisi, dan iterasi.

```
class Main{
  public static void main(String args[]){
    for(int i=0;i<10;i++){
      System.out.println("Iterasi Ke-" + i);
    }
  }
}
```

2. while loop

While loop adalah perulangan yang perulangannya akan terus dilakukan sampai kondisi yang dicek sudah tidak terpenuhi lagi.

```
public static void main(String args[]){
    Scanner in = new Scanner(System.in);
    int answer = 9;
    boolean guess = false;
    while(!guess){
      int tryAnswer = in.nextInt();
      guess = tryAnswer == answer;
    }
    System.out.println("end");
  }
```

3. do while loop

Do While melakukan pengecekan kondisi setelah perintah pada bagian *body* dijalankan, sehingga sekalipun kondisi tidak terpenuhi, minimal program akan berjalan sekali sebelum akhirnya dihentikan.

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

```
    int i = 10;
    do {
      System.out.print(i);
      i++;
    } while(i<10);
    System.out.println("end");
  }
```

4. **jump statement**
   a. **Break**
      Break berfungsi untuk menghentikan alur sebuah perulangan, namun yang dihentikan hanya satu blok yang membungkus break tersebut.
   b. **Continue**
      Umumnya continue digunakan di perulangan, yang mana ketiak terdapat continue program akan kembali melakukan iterasi selanjutnya tanpa menjalankan perintah setelah continue.
   c. **Return**
      Return merupakan perintah untuk mengembalikan nilai dan keluar dari suatu method.

G. ARRAY

Array digunakan untuk menyimpan beberapa nilai dalam satu variabel, alih-alih mendeklarasikan variabel terpisah untuk setiap nilai. Array mempunyai ukuran yang *fix* sehingga tidak dapat diubah setelah deklarasi. Indeks array dimulai dari 0, indeks digunakan untuk mengakses elemen array, jika terdapat array sepanjang *n,* maka indeksterakhirarraytersebutn-1.

a. **Deklarasi array**

Deklarasi array ditandai dengan kurung siku

```
public static void main(String args[]){
    //1. tipeData[] namaVariabel;
    int[] arrayOfInt;

    //2. tipeData[] namaVariabel;
    char arrayOfChar[];

    //3. tipeData[] namaVariabel = new tipeData[n]
    double[] arrayOfDouble = new double[10];
```

b. Inisialisasi Array

Inisialisasi array dapat ditandai dengan {} (kurung kurawal) atau dengan mengakses indeks array.

```
public static void main(String args[]){
   //Cara Pertama
   int[] arrayOfInt = new int[]{1, 2, 3};

   //Cara Kedua
   char[] arrayOfChar = new char[4];
   arrayOfChar[0] = 'h';
   arrayOfChar[1] = 'a';
   arrayOfChar[2] = 'l';
   arrayOfChar[3] = 'o';

   //Cara Ketiga
   float[] arrayOfFloat = {1f, 2f, 3f};
 }
```

c. Mengakses Elemen array

Array memiliki indeks, indeks dapat digunakan untuk mengakses elemen array dengan cara memanggil nama array diikuti kurung siku dan indeks yang ingin diakses.

```
public static void main(String args[]){
  int[] arr = {0, 4, 9};
  int n = arr[2];
  System.out.println(n);
 }
```

d . Mengakses array dengan *for each*

Jika ingin mengakses beberapa elemen dalam array akan lebih efisien menggunakan perulangan, selain perulangan biasa terdapat juga perulangan khusus array yaitu *for each loop.*

```
public static void main(String args[]){
  int[] numbers = {1, 5, 2};
  // dengan perulangan biasa
  for (int i=0; numbers.lengthl; i++){
```

```
    System.out.prinln(unhas[i]);
   }

   // menggunakan for each
   for (int number : numbers){
    System.out.println(e);
   }
  }
```

Fungsi *.length* digunakan untuk mendapatkan panjang array, karena jika mengakses array diluar batas indeksnya maka akan terjadi *ArrayIndexOutOfBoundsException.* for each dapat dikatakan lebih aman terhadap index out of bounds, karena secara otomatis, banyaknya perulangan akan menyesuaikan dengan panjang array.

e. Array Multidimensi

Multidimensi array dapat diilustrasikan sebagai array dalam array, jadi setiap satu elemen array mengandung array, dimensi dalam array ditandai dengan banyaknya [] (kurung siku)

```
public static void main(String args[]){
   int[][] matrix = {{ 1,0,0 },{0,1,0},{0,0,1}};
   for(int i = 0; i < matrix.length; i++){
    for(int j=0; j<matrix[i].length; j++){
      System.out.print(matrix[i][j] + " ");
    }
    System.out.println();
   }
  }
```

H. METHOD

Procedure, Function, atau Method merujuk pada hal yang sama, yaitu Kumpulan *statement* yang disatukan menjadi subprogram.

a. Mendeklarasikan dan Memanggil Method

```
public static void main(String args[]){
    int[] ages = {19, 20, 25};
    String name = "john";
    int age = 25;
    // memanggil method
    printHello(name, age);
    System.out.println(isAgeExist(ages, age));
```

```
}

// deklarasi method void
public static void printHello(String name, int age){
  System.out.println("Hello I'm " + name);
  System.out.println("I'm " + age + "years old.");
}

// deklarasi method dengan return value
public static boolean isAgeExist(int[] arr, int find){
  for(int element : arr){
    if(element==find){
      return true;
    }
  }
  return false;
}
```

b. Method Overloading

Method overloading merupakan method dengan nama yang sama tetapi dengan jumlah, tipe parameter atau return type yang berbeda.

```
public static void main(String args[]){
  System.out.println(sum(3, 9));
  System.out.println(sum(3, 9, 10, 5));
  System.out.println(sum(3.1, 9.2, 7.1, 3.5));
}

public static int sum (int a, int b){
  return a + b;
}
public static int sum (int ... nums){
  int result = 0;
  for (int n : nums){
    result += n;
  }
  return result;
}
```

```
public static double sum (double ... nums){
  double result = 0;
  for (double n : nums){
    result += n;
  }
  return result;
}
```

c. Method Rekursif

Method rekursif adalah method yang bodynya memanggil dirinya sendiri, method rekursif bisanya digunakan untuk mempersingkat kode dan membuatnya elegan, akan tetapi rekursif membutuhkan banyak memori dan lebih lambat.

```
public static void main(String args[]){
    System.out.println(factorial(8));
  }

  //method rekursif
  public static int factorial(int n){
    return n == 0 ? 1 : n * factorial(n-1);
  }
```

I. STRING MANIPULATION

1. Deklarasi String

Deklarasi string dapat dilakukan dengan string literal dengan cara membuat layaknya tipe data primitif dan nilainya ditandai dengan “” (kutip dua). atau dengan instansiasi objek.

```
public static void main(String args[]){
    //String Literal
    String a = "sisfo";
    //String Objek
    String b = new String("sisfo");
  }
```

2. String dan Array Of Char

Terdapat method toCharArray() pada string yang akan membuat array bertipe data char, sebaliknya char array dapat dijadikan objek

```
public static void main(String args[]){
    String prodi = "Sistem Informasi";
    char[] prodiChar = prodi.toCharArray();
    String prodiString = new String(prodiChar);
  }
```

Selain diubah ke array of char, string juga dapat dimanipulasi layaknya array menggunakan method charAt(i), dimana i adalah indeks yang dimulai dari 0.

3. String Format

Dengan menggunakan method format() atau printf(), string dapatdimanipulasi formatnya sesuai kebutuhan seperti mengatur banyaknya angka di belakang koma, spasi dll.

```
public static void main(String args[]){
    //String.format()
    String result = String.format("%.2f", 3.12412);
    System.out.println(result);
    //printf()
    System.out.printf("%.1f", 3.123123);
  }
```

tiap tipe data memiliki format specifier yang berbeda berikut beberapa contohnya

Format Specifier	Tipe Data	Fungsi
%d	Integer	Menampilkan angka
%s	String	Menampilkan String
%n		Menampilkan New Line
%f	Floating Point	Menampilkan Pecahan
%c	Char	Menampilkan Unicode Character

4. Reference Equality dan Value Equality pada String

Untuk tipe data primitif, perbandingan nilai dapat diketahui menggunakan operator “==”, sedangkan pada string terdapat sedikit perbedaan, jika String yang dibandingkan menggunakan string literal maka dapat menggunakan operator “==”, karena operator ini hanya

membandingkan alamat memori dari objek, sedangkan jika string dibuat dengan String Object, maka dibutuhkan method equals() untuk membandingkannya. karena method ini membandingkan konten dari string.

```
public static void main(String args[]){
    String a = "sisfo";
    String b = new String("sisfo");
    String c = "sisfo";
    System.out.println(a.equals(b));
    System.out.println(a == b);
    System.out.println(a == c);
  }
```

5. Method pada Class String

Untuk melihat list method pada class string silahkan cek di halaman berikut :
https://docs.oracle.com/en/java/javase/19/docs/api/java.base/java/lang/String.html

I COLLECTION

Collection merupakan salah satu framework dalam Java yang berisi kumpulan interface yang dapat digunakan untuk menampung beberapa data dalam sebuah variabel.

I. List

List merupaka collection yang memiliki struktur mirip dengan array dimana data disimpan secara teratur dan memiliki indeks berupa integer dari nol. Salah satu implementasi dari interface list adalah ArrayList.

```
public static void main(String args[]){
   // Membuat array list
   ArrayList<String> names = new ArrayList<>();
   // Menambah data kedalam array list
   names.add("Rojali");
   names.add("Eko");
   // Mengakses element arraylist
   System.out.println(names.get(0));
   // Mendapatkan panjang dari arrayList
   System.out.println(names.size());
   // Menghapus elemen arraylist indeks ke 0
   names.remove(0);
   // Menghapus semua element di array list
   names.clear();
```

```
  }
```

Masih terdapat beberapa built in method yang bisa digunakan di arraylist, untuk lebih jelasnya silahkan buka dokumentasi berikut: https://docs.oracle.com/en/java/javase/19/docs/api/java.base/java/util/ArrayList.html

2. **Set**

Set mirip List hanya saja elemen dalam set tidak bisa duplikat. salah satu implementasi dari interface set adalah HashSet.

```
import java.util.HashSet;
class Main{
  public static void main(String args[]){
    // Membuat hashset
    HashSet<String> names = new HashSet<>();
   // menambah data
   names.add("Devon");
   names.add("Devon");
   System.out.println(names);
  }
}
```

Method-method pada arraylist seperti add(e) remove(e) juga dapat digunakan di set.

3. **Map**

Map merupakan struktur data yang berbentuk Key-Value Pair, Map dapat dikatakan array namun bisa menggunakan indeks yang bukan integer, Salah satu implementasi dari Map adalah HashMap.

```
import java.util.HashMap;

class Main{
  public static void main(String args[]){
    // Membuat HashMap <key, value>
    HashMap<String, Integer> scores = new HashMap<>();
   // menambah data
   scores.put("Abdul", 87);
   scores.put("Yoyo", 89);
   // Mengakses elemen
   System.out.println(scores.get("Abdul"));
   // Menghapus
```

```
    scores.remove("Yoyo");
    // Merubah nilai
    scores.replace("Abdul",  89);
  }
}
```

Berikut link dokumentasi method HashMap

https://docs.oracle.com/en/java/javase/19/docs/api/java.base/java/util/HashMap.html