

Bab 0

Dasar-Dasar Pemrograman Terstruktur

Sebuah analogi yang cukup baik menyamakan proses memrogram mirip seperti apa yang dikerjakan oleh Harry Potter dan teman-temannya. Mereka mampu menggerakkan atau membuat efek pada sesuatu hanya dengan menggunakan mantra (spell). Seorang programmer juga menggerakkan dan memberi efek pada sesuatu (komputer) dengan menggunakan bahasa (pemrograman). Ketepatan pengucapan mantra, membuat efek yang diinginkan tepat berlaku seperti yang dikehendaki. Mirip dengan itu, ketepatan penggunaan bahasa pemrograman oleh seorang programmer, mempengaruhi ketepatan proses dan efek yang berjalan pada sebuah komputer. Salah matra akan berakibat fatal, demikian pula salah program (karena salah menerapkan kaidah bahasa, baik salah sintaks, salah logika maupun konteks penerapan) dapat berakibat fatal pada proses yang berjalan pada komputer.

Tugas programmer adalah melakukan kontrol terhadap proses yang berjalan dalam komputer melalui bahasa pemrograman. Sehingga untuk memrogram kita perlu memahami aspek bahasanya dan aspek proses yang terjadi akibat eksekusi bahasa itu dalam komputer.

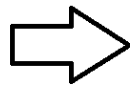
Bab ini akan melakukan review terhadap dasar-dasar pemrograman yang perlu diingat dan dikuasai sebelum kita beranjak lebih lanjut dalam membahas konsep-konsep berorientasi objek. Pembahasan pada bab ini akan mulai dari review terkait jenis bahasa pemrograman, kemudian pembahasan mengenai tipe data dasar, struktur kontrol dalam memprogram, serta pembahasan mengenai struktur data dan function (method). Karena bersifat ulasan ringkas, pembaca yang ingin lebih detail dapat mempelajari pada berbagai referensi atau tutorial yang banyak tersedia.

Bahasa Pemrograman

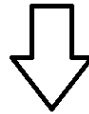
Bahasa pemrograman yang digunakan dapat kita kategorikan menjadi dua, bahasa yang di-compile dan bahasa yang di-interpretasi. Sebagai contoh bahasa yang di-compile adalah bahasa C. Source-code yang dibuat dalam bahasa C (pada file dengan ekstensi .c) perlu di-compile terlebih dahulu menjadi file lain, file dengan ekstensi .exe. File executable inilah yang kemudian dapat dijalankan oleh sistem operasi. Program yang digunakan untuk melakukan proses kompilasi ini disebut sebagai compiler.

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(int argc, char *argv[]){
    printf("Hello World \n");
    return 0;
}
```

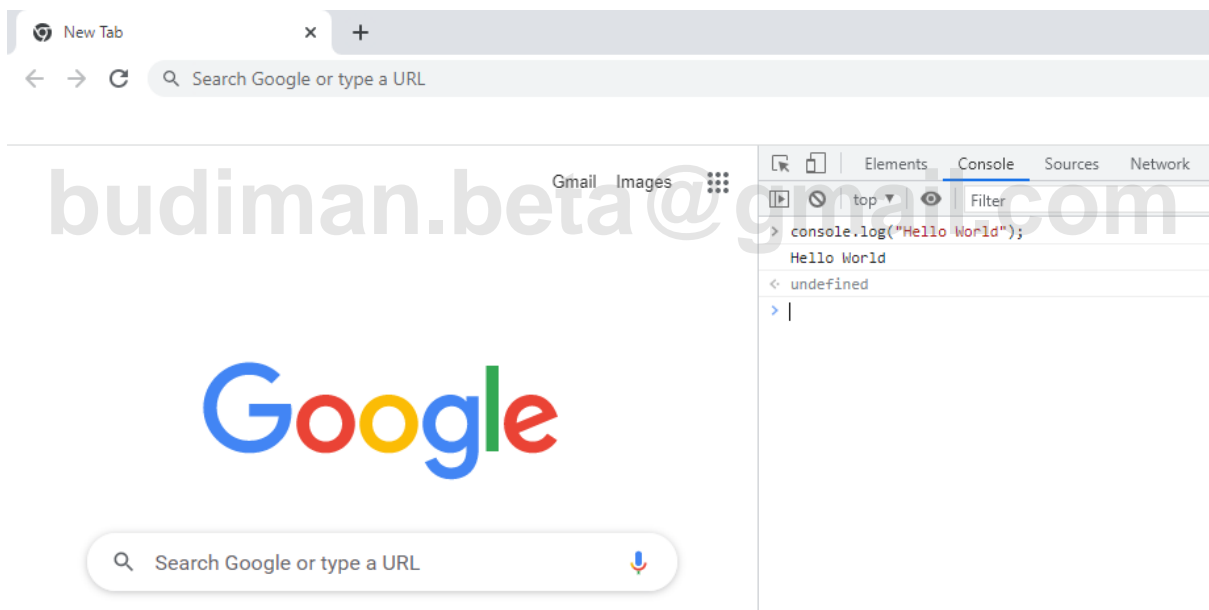


main.c	12/07/2022 10:50	C File
main	12/07/2022 10:50	Application



```
C:\PRIVATE>main
Hello World
```

Sedangkan contoh bahasa yang di-interpretasi adalah JavaScript, seperti yang dapat anda lihat pada beragam browser yang ada. Pada proses interpretasi, code tidak diubah menjadi jenis lain (binary executable) tetapi langsung dieksekusi oleh interpreter bahasa itu.



Bahasa Pemrograman Java

Java agak berbeda, karena source-code Java (file dengan ekstensi .java) perlu di-compile menjadi jenis lain (disebut sebagai Byte-Code, file dengan ekstensi .class), setelah itu byte-code ini dieksekusi oleh JVM (Java Virtual Machine atau disebut juga Java Runtime Environment [JRE]) dengan cara di-interpretasi. Byte-Code tidak dieksekusi langsung oleh sistem operasi, tetapi oleh virtual machine (JVM) di atas sistem operasi itu.

Untuk membuat dan menjalankan aplikasi Java kita memerlukan compiler dan interpreter Java. Kita dapat menggunakan JDK dari Oracle maupun OpenJDK untuk mendapatkan compiler dan interpreter Java. Langkah-langkah konfigurasi JDK dapat dibaca pada

Appendix A (yang dapat di-download dari link google drive berikut ini, <https://bit.ly/3aZj4Ni>).

Hello World

Mengikuti tradisi dalam belajar pemrograman (tradisi yang dimulai oleh C), kita akan membuat Hello World program untuk melakukan test-drive terhadap compiler dan interpreter yang sudah kita install dalam komputer.

Buatlah sebuah file dalam sebuah folder, simpan dengan nama HelloWorld.java. Ketikkan kode di bawah ke dalam file itu.

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


Java cukup ketat dalam sintaksis, huruf kecil dan huruf besar adalah berbeda. Nama file harus sama dengan nama class yang dibuat.

Compiler

Untuk mengkompilasi file di atas, gunakan command prompt pada folder dimana file tersimpan dan ketikkan perintah berikut :

javac HelloWorld.java

Perintah **javac** akan menjalankan proses kompilasi terhadap file HelloWorld.java, yang akan menghasilkan file baru dengan ekstensi .class.

 HelloWorld.class

Compiler akan menampilkan pesan error jika ada sintaksis yang tidak tepat.

Interpreter

File HelloWorld.class selanjutnya dapat kita eksekusi melalui perintah berikut :

java HelloWorld

Perintah **java** di atas meminta JVM untuk mengeksekusi kode pada HelloWorld.class. JVM akan memulai eksekusi dari sebuah class yang memiliki method **main**. Class HelloWorld di atas sudah memiliki method main, sehingga dapat dieksekusi langsung oleh JVM (JRE).

```
C:\PRIVATE>java HelloWorld  
Hello World
```

Struktur Dasar Program Java

```
public class HelloWorld{ class
    public static void main (String[] args){ method
        System.out.println("Hello World");
    }
}
```

Struktur dasar program Java terdiri atas class dan method (function pada class itu). Konsep class akan dipelajari secara detail pada bab ke-2. Pada bab ini cukup kita pahami bahwa batu bata dasar pembangun program Java adalah sebuah class, seperti contoh di atas. Sebuah class dapat memiliki beragam function di dalam-nya. Function dalam class ini disebut sebagai method. Tidak ada function yang berdiri sendiri di luar class di Java.

Method main pada class di atas adalah method khusus, JVM (JRE) akan memulai eksekusi proses dari method main ini. Method khusus ini juga memiliki signature (tanda) yang baku seperti di bawah ini.

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

Method dengan nama sama namun tidak ada kata kunci **static**, atau memiliki return value (integer misalnya) bukan **void**, atau bersifat **private** bukan **public** atau menerima argumen lain selain **array String**, maka tidak akan dieksekusi oleh JVM.

Variabel dan Operator

Proses komputasi dalam komputer banyak ragamnya, perintah di bawah ini merupakan proses sederhana menampilkan teks Hello World ke console komputer. (Semua perintah atau statement di Java selalu diakhiri oleh **titik koma ;**).

```
| System.out.println("Hello World");
```

Proses komputasi umumnya terkait dengan manipulasi informasi. Informasi atau data dalam komputer di simpan dalam sebuah lokasi memori tertentu. Dalam bahasa pemrograman, abstraksi terhadap lokasi memori yang menyimpan data itu dikenal sebagai variabel. Jadi variabel merupakan abstraksi memori yang menyimpan data tertentu. Sebuah variabel memiliki tipe data (data type) tertentu. Tipe data dasar (primitive) yang dapat digunakan

misalnya int, double, boolean, char. Berikut ini cara kita mendeklarasikan penggunaan sebuah variabel.

```
| int score ;
```

Pada contoh di atas, score adalah nama variabel, int merupakan tipe datanya. Jadi score dapat menampung data berupa integer atau bilangan bulat. Secara internal mesin Java akan mengalokasikan memori tertentu untuk menampung data berupa bilangan bulat. Seberapa besar ukuran memori yang diperlukan tergantung dari tipe datanya.

Kita dapat melakukan operasi terhadap sebuah variabel. Salah satu operasi dasar pada variabel adalah perintah **assignment**.

```
| score = 90;
```

Assignment adalah proses memasukkan nilai ke dalam variabel. Tanda = di sini adalah operator assignment.

Beberapa contoh tipe data dasar (**primitive**) dapat dilihat pada tabel di bawah ini.

Tipe Data	Contoh
int	int i = 1; //bilangan bulat
double	double d = 4.5; //bilangan pecahan
boolean	boolean b = true; //nilai logika
char	char c = 'c' ; // nilai karakter

Selain primitive, ada juga yang dikenal sebagai variabel reference (secara detail akan dibahas pada bab-bab selanjutnya). Salah satu variabel reference yang sering digunakan adalah String untuk menampung teks.

```
| String str = "Hello World";  
| System.out.println(str);
```

Perbedaan antara primitive dan reference adalah, pada variabel primitive nilai disimpan pada lokasi memori yang dialokasikan untuk variabel itu, sedangkan pada variabel reference, nilai dari variabel di simpan di lokasi lain, lokasi memori variabel hanya menyimpan semacam alamat ke lokasi lain itu.

Sesuai dengan tipe data masing-masing, kita dapat menerapkan operasi dengan menggunakan operator yang cocok dengan tipe datanya.

```
| int a = 2;  
| int b = 3;
```

```

a = a + b;
a = a * b;

System.out.println("Nilai a sekarang adalah " + a);

System.out.println(true || false);
System.out.println(true && true);

boolean isTrainee = false;
System.out.println(!isTrainee);
System.out.println(a == 15);

System.out.println(isTrainee == true);

```

Simbol `=` di atas adalah operator assignment. Operator `+` dan `*` adalah operator aritmatika, yang berlaku pada tipe data bilangan. Tetapi `+` juga merupakan operator untuk menggabungkan dua nilai, seperti pada contoh `System.out.println("Nilai a sekarang adalah " + a);` Operator yang memiliki interpretasi yang berbeda jika diterapkan pada tipe data tertentu dikenal sebagai operator overloading.

Operator `||` dan `&&` adalah operator logika. Simbol `||` merupakan operator OR, yang akan menghasilkan nilai true jika salah satu bagian bernilai true. Simbol `&&` adalah operator AND, yang akan bernilai true jika semua bagian bernilai true. Untuk menegaskan nilai boolean tertentu digunakan operator `!`, pada contoh di atas `!isTrainee` akan bernilai true.

Tanda `==` adalah operator perbandingan, digunakan untuk membanding nilai (variabel primitif). Pada contoh di atas `a == 15` akan bernilai true, sedangkan `isTrainee == true` akan bernilai false.

Terkait tipe data String, operasi terhadap sebuah String dapat dilakukan dengan menggunakan method yang disediakan oleh objek String itu.

```

String str = " Hello World ";
String str2 = str.toLowerCase();
String str3 = str.substring(4);
String str4 = str.trim();

System.out.println(str);
System.out.println(str2);
System.out.println(str3);
System.out.println(str4);

```

Proses Sekuensial

Proses komputasi secara default berjalan secara sekuensial, maksudnya perintah akan dieksekusi satu persatu secara berurutan.

```

String str = "Hello World";
System.out.println(str);

```

Pada contoh di atas, setelah proses assignment dijalankan, proses menampilkan isi variabel ke console komputer di jalankan.

Struktur Kontrol (Control Structure)

Karena untuk melakukan pemecahan masalah dengan menggunakan komputer kita perlu menerapkan algoritma yang sesuai, dan keperluan untuk menjalankan sebuah algoritma pemecahan masalah tidak cukup dengan eksekusi secara sekuensial saja, kita memerlukan struktur untuk melakukan kontrol atas proses itu. Ada dua jenis struktur kontrol utama (1) Kontrol untuk seleksi (**selection control**) , dan (2) Kontrol untuk repetisi atau perulangan (**repetition control**).

(1) Selection Control

Kontrol seleksi membantu kita untuk melakukan percabangan proses berdasarkan kriteria tertentu. Untuk membantu seleksi proses kita dapat menggunakan if else.

```
int score ;
Scanner keyboard= new Scanner(System.in);
System.out.println("Nilai ujianmu berapa ? ");

score = keyboard.nextInt();
if(score > 75) {
    System.out.println("Lulus");
}else {
    System.out.println("Gagal");
}
```

Object scanner digunakan untuk mendapatkan response dari System.in, umumnya adalah dari keyboard. Nilai yang diberikan oleh user melalui keyboard akan disimpan dalam variabel score. Selanjutnya statemen if akan melakukan pengecekan apakah nilai score lebih besar dair 75, jika evaluasi ini bernilai true maka akan ditampilkan pesan “Lulus” ke console komputer. Jika evaluasi bernilai false, maka perintah pada blok else akan dieksekusi (menampilkan pesan “Gagal”).

Kita dapat mengkombinasikan penggunaan if else dalam statement if else yang lain, atau dikenal sebagai nested if.

```
if(score > 75) {
    if(score > 90) {
        System.out.println("Grade A");
    }else {
        System.out.println("Grade B");
    }
}else if(score > 60 && score <= 75){
    System.out.println("Grade C");
}else {
    System.out.println("Grade D");
}
```

Untuk melakukan seleksi berdasarkan kriteria nilai variabel tertentu (int, boolean, double, float, char, dan String) kita dapat juga menggunakan switch.

```
int carType ;
Scanner keyboard = new Scanner(System.in);
System.out.println("Masukkan kode jenis angkutan umum ");

carType = keyboard.nextInt();

switch (carType) {
    case 1:
        System.out.println("Transjakarta");
        break;
    case 2:
        System.out.println("MRT");
        break;
    case 3:
        System.out.println("Taksi");
        break;
    case 4:
        System.out.println("Ojek");
        break;
    default:
        System.out.println("Bajaj");
        break;
}
```

Statemen switch di atas akan mengevaluasi nilai dari variabel carType, jika bernilai 1 maka akan dieksekusi perintah pada blok **case 1**. Jika tidak ada satupun nilai yang cocok, maka akan dieksekusi perintah pada blok **default**. Pada setiap blok case ditambahkan perintah **break**, agar tidak mengeksekusi case selanjutnya.

(2) Repetition Control

Untuk melakukan perulangan kita dapat menggunakan beberapa control di Java, dengan menggunakan while, for maupun do while.

while

```
int i = 0;
while(i < 10) {
    System.out.println("Hello >> "+ i);
    i = i + 1;
}
```

Eksekusi perintah di atas akan menampilkan hasil berikut,


```
Hello >> 0
Hello >> 1
Hello >> 2
Hello >> 3
Hello >> 4
Hello >> 5
Hello >> 6
Hello >> 7
Hello >> 8
Hello >> 9
```

Variabel `i` pada program di atas berfungsi sebagai counter. Perintah `while` akan mengevaluasi kondisi `i`, apakah `i` bernilai lebih kecil dari 10. Jika `true` maka perintah-perintah pada blok `while` akan dijalankan. Jika evaluasi `false`, perulangan akan berhenti.

Nilai variabel counter `i` dinaikkan setiap kali proses perulangan (loop) pada blok setelah `while` dijalankan (jika evaluasi kondisi bernilai `true`). Penambahan nilai counter ini dilakukan oleh perintah `i = i + 1`. Perintah menaikkan nilai sebuah variabel seperti perintah `i = i + 1` dikenal sebagai proses increment (sebaliknya `i = i - 1` adalah proses decrement). Kita dapat menyingkat proses increment itu dengan menggunakan operator increment berikut, `i++` atau `++i`. Program di atas dapat dituliskan juga seperti di bawah ini.

```
int i = 0;
while(i < 10) {
    System.out.println("Hello >> " + i);
    i++; //atau ++i;
}
```

do while

Perintah `do while` mirip dengan `while`, hanya saja menjamin bahwa perintah-perintah di dalam blok `do-while` akan dieksekusi minimal sekali.

```
int k = 0;
do {
    System.out.println("do ...while ..>>" + k);
    --k;
}while(k > 0);
```

Program di atas kan menghasilkan output berikut,

```
do ...while ..>> 2
do ...while ..>> 1
```

for

Perintah `for` juga mirip dengan `while`, hanya saja deklarasi counter, evaluasi nilai dan increment hanya dilakukan pada satu baris saja.

```
for(int j=0; j < 10; ++j) {
    System.out.println("Loop with for >> " + j);
}
```

Perintah for di atas akan menghasilkan output berikut,

```
Loop with for >> 0
Loop with for >> 1
Loop with for >> 2
Loop with for >> 3
Loop with for >> 4
Loop with for >> 5
Loop with for >> 6
Loop with for >> 7
Loop with for >> 8
Loop with for >> 9
```

Beda i++ dan ++i

Pada contoh-contoh program di atas, kita menemukan statement berikut i++ maupun ++i.

```
int i = 5;
i++;
System.out.println(i);
```

Nilai i yang ditampilkan program di atas adalah 6.

```
int i = 5;
++i;
System.out.println(i);
```

Demikian juga nilai i yang akan ditampilkan oleh program di atas juga 6.

Operator ini akan memberikan hasil yang berbeda jika digabungkan dalam sebuah ekspresi seperti contoh berikut,

```
int i = 5;
int b;
b = i++;
System.out.println(b);
```

Pada akhir eksekusi program di atas, variabel b akan bernilai 5. Mengapa ? Ekspresi `b = i++` akan dievaluasi dengan memasukkan nilai i saat itu (5) sebelum dilakukan proses increment. Jadi variabel b akan bernilai 5, tetapi variabel i berniali 6.

```
int i = 5;
int b;
b = ++i;
System.out.println(b);
```

Sedangkan jika ekspresinya adalah `b = ++i`, seperti pada contoh ke-2 di atas, maka ekspresi itu akan dievaluasi dengan melakukan increment terlebih dahulu (sekarang i bernilai 6) baru kemudian dilakukan assignment ke variabel b (sekarang b berniali 6). Variabel b dan i pada akhir eksekusi program bernilai 6.

break dan continue

```
for(int j=0; j < 10; ++j) {  
    if( j == 5) {  
        break;  
    }  
    System.out.println("Loop with for >> " + j);  
}
```

```
Loop with for >> 0  
Loop with for >> 1  
Loop with for >> 2  
Loop with for >> 3  
Loop with for >> 4
```

Perintah break akan menghentikan proses perulangan. Sedangkan perintah continue akan melakukan skip (penghentian) proses jika kondisi sebelum perintah continue terpenuhi saja, selanjutnya proses perulangan tetap akan dilanjutkan.

```
for(int j=0; j < 10; ++j) {  
    if( j == 5) {  
        continue;  
    }  
    System.out.println("Loop with for >> " + j);  
}
```

```
Loop with for >> 0  
Loop with for >> 1  
Loop with for >> 2  
Loop with for >> 3  
Loop with for >> 4  
Loop with for >> 6  
Loop with for >> 7  
Loop with for >> 8  
Loop with for >> 9
```

Pemrograman Terstruktur (Structured Programming)

Penelitian ilmuwan ilmu komputer sejak akhir tahun 60-an abad lalu menyatakan bahwa untuk menyelesaikan (semua) masalah komputasi (yang dapat dikerjakan oleh komputer) kita hanya memerlukan tiga jenis kontrol proses yang sudah kita pelajari sebelumnya.

1. Sekuensial (sequential control)
2. Seleksi (selection control)
3. Repetisi (Repetition control)

Pengetahuan kita (sebagai manusia) banyak yang tersimpan sebagai deklarative knowledge. Misalnya kita tahu akar 4 adalah 2. Bagi komputer, untuk menyelesaikan sebuah masalah, komputer perlu tahu langkah-langkah yang harus dia lakukan untuk menyelesaikan masalah itu (imperative knowledge). Langkah-langkah menyelesaikan masalah itu perlu didefinisikan dengan menggunakan tiga control proses yang dipahami oleh komputer di atas.

Program = Struktur Data + Algoritma

Untuk menyelesaikan sebuah masalah (komputasional) kadangkala kita perlu merepresentasikan masalah itu dalam sebuah struktur yang dapat membantu algoritma bekerja menyelesaikan masalah kita. Misal kita memiliki variabel-variabel berikut,

```
int a = 10;  
int b = 8;  
int c = 7;  
int d = 22;  
int e = 15;
```

bagaimana kita mengurutkan data di atas dari nilai yang paling kecil hingga paling besar ? Pikiran kita akan dengan mudah mengurutkannya. Tetapi bagi komputer tidak mudah. Dia tidak tahu cara mengurutkan secara otomatis. Kita bisa membayangkan cara mengurutkan data di atas, tetapi untuk eksekusinya akan sulit dengan variabel nilai yang terpisah-pisah seperti itu. Di sini kita memerlukan sebuah struktur untuk membantu kita merepresentasikan masalah kita (misal pengurutan data) dan membantu algoritma kita untuk mudah diimplementasikan. Struktur itu kita kenal sebagai struktur data (**data structure**). Persamaan, **program = struktur data + algoritma**, dipopulerkan oleh pembuat bahasa pemrograman Pascal pada tahun 70-an abad lalu.

Array

Salah satu struktur data dasar yang umumnya diberikan oleh bahasa pemrograman adalah array. Sebuah array adalah sebuah struktur berbaris, berbasis indeks, dengan panjang terbatas, untuk menampung data dengan tipe tertentu. Untuk mendeklarasikan sebuah array yang dapat menampung beberapa bilangan bulat (misal untuk merepresentasikan umur seseorang) dapat dilakukan dengan perintah berikut.

```
int[] ageArray = new int[4];  
ageArray[0] = 17;  
ageArray[1] = 10;  
ageArray[2] = 20;  
ageArray[3] = 5;
```

Variabel **ageArray** bertipe array of integer, `int[]` adalah tipe array yang dapat menampung nilai integer (bilangan bulat). Perintah `new int[4]` membuat objek array dengan panjang 4. Jadi `ageArray` dapat menampung 4 nilai integer. Di Java, array adalah objek sehingga untuk membuatnya kita perlu menggunakan operator **new** (lebih jauh mengenai objek dan operator `new` akan dibahas pada bab-bab selanjutnya).

Untuk menempatkan sebuah nilai ke dalam sebuah array kita menggunakan indeks dari array itu. Indeks array di Java dimulai dari indeks 0. `ageArray[0] = 17`; mengisi array pada indeks 0 dengan nilai 17.

	17	10	20	5
indeks	0	1	2	3

Untuk membaca data pada array, kita juga menggunakan indeks-nya.

```
System.out.println(ageArray[0]);  
System.out.println(ageArray[1]);
```

Perintah `for` dapat digunakan untuk membaca atau menelusuri data pada sebuah array. Karena array adalah objek, objek array menyediakan variabel **length** untuk mendapatkan ukuran dari array itu.

```
for(int i=0; i < ageArray.length;i++) {  
    System.out.println(ageArray[i]);  
}
```

Java juga menyediakan struktur control yang spesifik untuk penelusuran sebuah struktur data seperti array dan struktur data lain yang dikenal sebagai collection (seperti List, Set dsb).

```
for(int age : ageArray) {  
    System.out.println(age);  
}
```

Perintah di atas dapat kita terjemahkan sebagai berikut, **untuk setiap (for each)** data yang ada pada **ageArray** ambil datanya kemudian simpan dalam variabel **age**, kemudian tampilkan **isi/nilai** age ke console komputer.

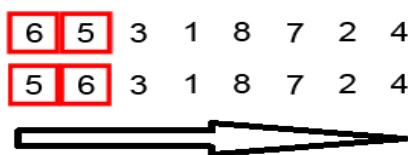
Berikut ini adalah contoh lain dari sebuah array yang menyimpan data String.

```
String[] strArray = new String[4];  
strArray[0] = "Budiman";  
strArray[1] = "Topati";  
strArray[2] = "Hendra";  
strArray[3] = "Teja";  
  
for(int i=0; i < strArray.length;i++) {  
    System.out.println(strArray[i]);  
}  
  
for(int str : strArray) {  
    System.out.println(str);  
}
```

Sorting (Pengurutan Data)

Untuk menyelesaikan problem pengurutan data, seperti diungkapkan di atas, kita perlu merepresentasikan data kita dalam struktur data tertentu (dalam konteks ini array). Kemudian kita dapat menerapkan algoritma pengurutan (sorting) atas data dalam array itu. Algoritma untuk sorting cukup banyak. Pada bagian ini kita hanya menerapkan algoritma sederhana **bubble sort** untuk mengurutkan data. Seperti namanya ide dari algoritma ini adalah bagian yang ringan akan ke atas (atau yang berat ke bawah). Jika dibayangkan secara horizontal bagian yang ringan (lebih kecil) akan pindah ke kiri.

Persoalannya adalah bagaimana kita mengetahui yang lebih ringan, tentu saja dengan melakukan perbandingan data. Selanjutnya, bagaimana memindahkan data yang lebih ringan ke samping kiri, ini dapat dilakukan dengan penukaran data. Ilustrasi berikut dapat membantu menjelaskan algoritma bubble sort.



Untuk setiap data pada array, lakukan perbandingan antara data itu dengan data selanjutnya. Jika data selanjutnya lebih kecil tukar posisi data itu dengan data sebelumnya. Ulangi proses itu kembali sesuai jumlah array yang ada. Dengan menggunakan pseudocode, algoritma itu dapat kita tuliskan sebagai berikut :

```
for i=0 to array.length
    for j = i + 1 to array.length
        if array[j] < array[i]
            swap(i,j)
```

Untuk melakukan penukaran data pada sebuah array, kita dapat memanfaatkan sebuah variabel untuk menyimpan secara sementara variabel yang akan kita tukar.

```
int temp = ageArray[2];
ageArray[2]=ageArray[3];
ageArray[3] = temp;
```

Jika kita terjemahkan algoritma di atas ke dalam kode Java kurang lebih dapat kita buat sebagai berikut,

```
public static void main(String[] args) {
    int[] ageArray = {1,2,100,4,5,6,7,8,9,10,11,12,13,14,15};

    System.out.println("Data sebelum algoritma pengurutan");
    for(int i=0; i < ageArray.length;i++) {
        System.out.print(ageArray[i] +" ");
    }
}
```

```

    }

    for(int i=0; i < ageArray.length; i++) {
        for(int j = i + 1; j < ageArray.length; j++) {
            if(ageArray[j] < ageArray[i]) {
                int temp = ageArray[i];
                ageArray[i] = ageArray[j];
                ageArray[j] = temp;
            }
        }
    }

    System.out.println();
    System.out.println("Data setelah algoritma pengurutan");
    for(int i=0; i < ageArray.length;i++) {
        System.out.print(ageArray[i] +" ");
    }
}

```

Data sebelum algoritma pengurutan

1 2 100 4 5 6 7 8 9 10 11 12 13 14 15

Data setelah algoritma pengurutan

1 2 4 5 6 7 8 9 10 11 12 13 14 15 100

Fungsi (Function)

Ide tentang fungsi (function) atau prosedur berhubungan dengan ide mengenai **reuse** atau guna ulang kode yang sudah dibuat. Pada proses sorting di bagian sebelumnya, jika kita merubah nama variabel datanya, maka kita perlu merubah implementasi algoritma kita. Demikian pula kita perlu mengimplementasikan algoritma sorting itu kembali di bagian lain dari program kita yang memerlukan pengurutan data. Di sinilah letak pentingnya pemrograman secara modular, yang dapat dipakai ulang pada bagian lain aplikasi kita. Konsep modul yang utama adalah fungsi.

Karena fungsi di Java tidak bisa berdiri sendiri maka fungsi harus ada dalam sebuah class. Fungsi dalam sebuah class kita sebut sebagai method. Konsep fungsi dapat kita dekati dengan menambahkan kata kunci static pada method. (Pembahasan mengenai static method akan dipelajari pada bab-bab selanjutnya). Sebagai contoh, kita dapat membuat fungsi untuk mencari nilai mutlak dari sebuah bilangan bulat.

```

public static int absValueInt(int x) {
    if(x >= 0) {
        return x;
    } else {
        return -x;
    }
}

```

Sebuah fungsi terdiri dari dua bagian, (1) signature-nya, (2) implementasinya. Signature dari fungsi di atas adalah public static int absValueInt(int x). Kata kunci public static sementara

kita asumsikan dulu harus ada. `int` merupakan return value dari function. Fungsi yang tidak mengembalikan nilai (tidak mereturn value) harus menambahkan `void` pada bagian ini. `absValueInt` adalah nama fungsi. `(int x)` adalah parameter dari fungsi.

Setelah tanda kurung kurawal pada signature merupakan implementasi fungsi `absValueInt`. Perintah `return`, digunakan untuk mengembalikan nilai kepada pemanggil fungsi.

```
int x = -100;
System.out.println();
int z = absValueInt(x);
System.out.println(z);
System.out.println(x);
```

Perhatikan contoh di atas untuk memanggil fungsi yang sudah dibuat dari bagian lain program.

Untuk kasus pengurutan data menggunakan bubble sort di atas kita dapat menulis ulang algoritma sorting-nya dalam sebuah fungsi (`sortArray`), seperti tampak pada class berikut. (Perhatikan cara mendefinisikan dan memanggil fungsi-nya).

```
public class ArrayIntSortTest {

    public static void printArrayInt(int[] arr) {
        for(int i=0; i < arr.length;i++) {
            System.out.print(arr[i] + " ");
        }
    }

    public static void main(String[] args) {
        int[] ageArray = {1,2,100,4,5,6,7,8,9,10,11,12,13,14,15};

        printArrayInt(ageArray);

        System.out.println("\n== sorting =====");
        int[] sortedArray = sortArrayInt(ageArray);

        printArrayInt(sortedArray);

    }

    public static int[] sortArrayInt(int[] arr) {
        for(int i=0; i < arr.length; i++) {
            for(int j = i + 1; j < arr.length; j++) {
                if(arr[j] < arr[i]) {
                    int temp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
            }
        }

        return arr;
    }
}
```