PERTEMUAN 11
KONSEP PEMROGRAMAN BERORIENTASI OBJEK *CONSTRUCTOR*

A. TUJUAN PEMBELAJARAN

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

B. URAIAN MATERI

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

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

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

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

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

1. Bagaimana constructor bekerja

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

Selanjutnya buat objek MyClass seperti contoh dibawah ini :

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

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

a. Buat class Myclass ketikan kode dibawah ini :

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

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

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

    }

}
```

Output :

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

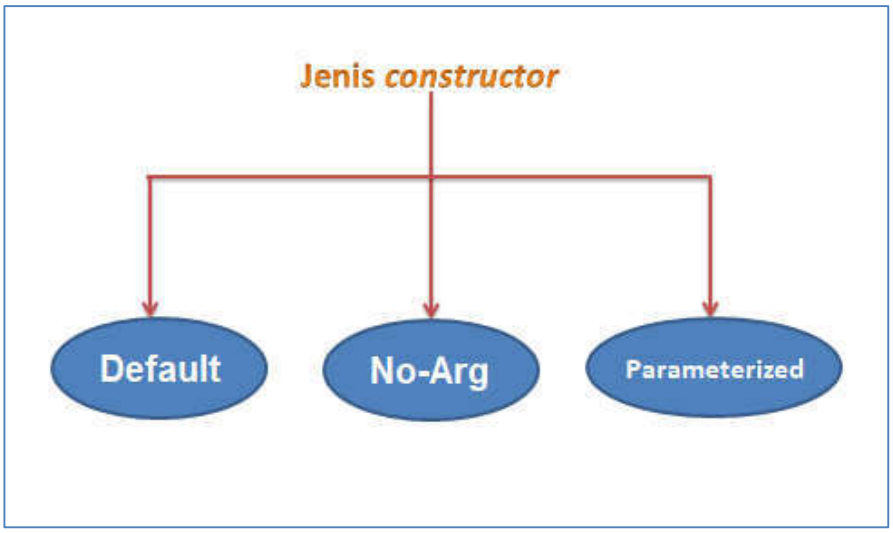
```
BUILD SUCCESSFUL (total time: 1 second)
```

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

2. Jenis jenis constructor

Terdapat tiga jenis *constructor* yaitu :

a. Default
b. No-arg
c. Parameterized



Gambar 11. 1 Jenis Constructor

a. *Default Constructor*

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

Bentuk umum penulisan **Default *Constructor***

```
<class_name>(){

}
```

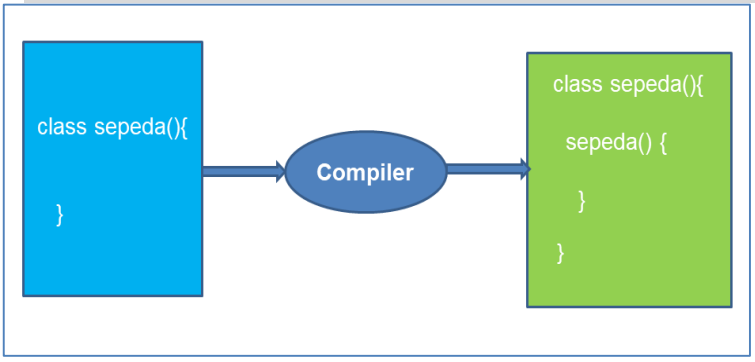
Contoh program sederhana penulisan *default constructor* :

```
// Program untuk membuat dan memanggil sebuah default constructor
class sepeda1{
//membuat  default constructor
sepeda1()
{System.out.println("sepeda sedang dibuat");
}
```

```
  //main method
  public static void main(String args[]){
  //calling a default constructor
  Sepeda1 b=new sepeda1();
  }
  }
```

Output :

```
  sepeda sedang dibuat
```



Gambar 11. 2 Default Constructor

b. *No-Arg Constructor*

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

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

Contoh : constructor *class demoNoArg*

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

```
public class NoArgMain {

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

Output :

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

c. *Paramterized Constructor* (Konstruktor ber parameter)

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

Contoh program konstruktor berparameter

```
public class Karyawan {


    int IdKaryawan;
   String NamaKaryawan;


   // constructor dengan 2 parameter
   Karyawan (int id, String nama){
       this.IdKaryawan = id;
       this.NamaKaryawan = nama;
   }
   void info(){
        System.out.println("Id Karyawan : "+IdKaryawan+"
Nama: "+NamaKaryawan);
   }
}
```

```
public class KaryawanMain {


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

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



    }


}
```

Output :

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

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

Contoh 2 program konstruktor berparameter

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

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

```
public class COntoh2Main {

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

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

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


    }


}
```

Output :

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

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

d. *Keyword Super*()

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

Contoh program keyword Super ()

```
public class IndukClass {

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

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

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

Output :

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

C. LATIHAN/TUGAS

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

Tabel 11. 1 Tabel Golongan

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

Contoh Output program :

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

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

D. REFERENSI

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

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

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

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

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

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