MODUL IV PBO I
Constructor, Overload, Static

Informatika

Universitas Sanata Dharma

Kompetensi

Setelah membaca modul ini mahasiswa dapat:

- Memahami Constructor
- Memahami Overload
- Mendefinisikan static methods dan static atribut

CONSTRUCTOR

Constructor

- Ketika obyek dibuat dengan kata kunci new, maka secara otomatis metode constructor akan dieksekusi.

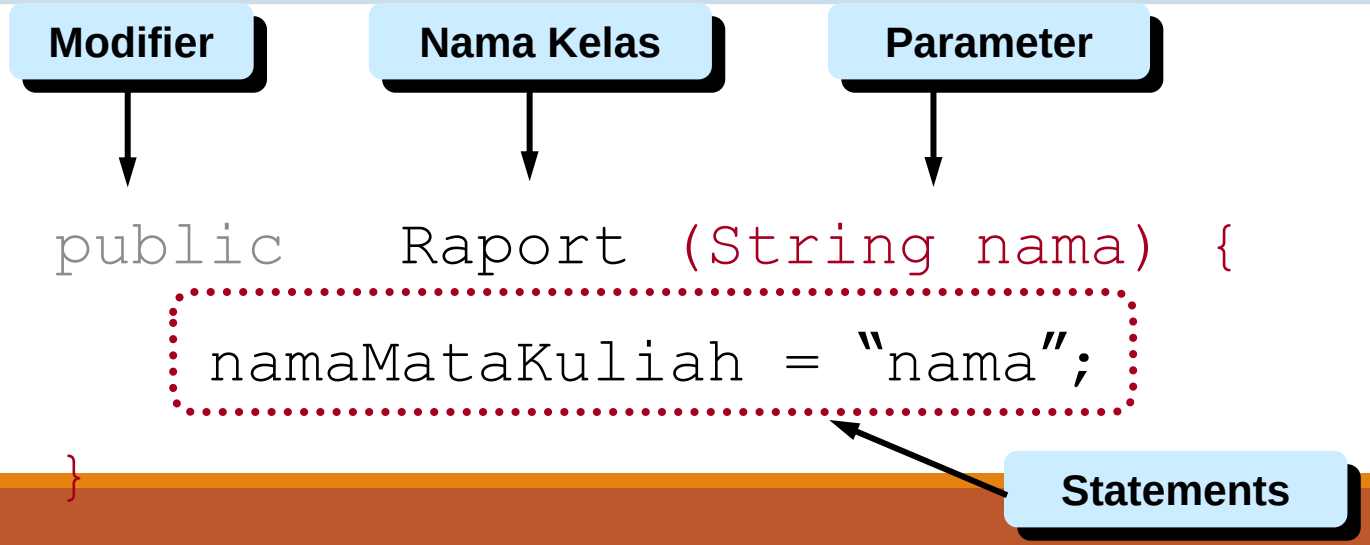
```
Raport myRaport = new Raport();
```

- Ketika constructor dijalankan, maka :
  - Blok memori akan dialokasikan untuk kelas
  - Obyek diinisialisasi
- Contsructor memainkan peran untuk menginisialisasi obyek

Constructor

- Nama metode *constructor* sama dengan nama kelas.
- Constructor tidak memiliki tipe *return*
- Deklarasi constructor sebagai berikut :

```
public <nama kelas> ( <parameters> ){
    <statement>
}
```



Default Constructor

- Setiap kelas yang tidak mendeklarasikan constructor secara eksplisit maka secara otomatis akan memiliki constructor default.
- Tetapi jika constructor dideklarasikan, maka constructor default tidak terpakai
- Constructor ini tidak memiliki argumen
- Constructor ini akan mengalokasikan obyek dan menginisialisasi variabel instance dalam deklarasi

```java
// Raport.java
// Kelas Raport yang memuat  variabel instance namaMatakuliah instance
// dan metode set dan get nya

public class Raport
{
   private String namaMatakuliah; // nama matakuliah didefinisikan scr privat
   // constructor initializes courseName with String supplied as argument
   public Raport( String nama )
   {
      namaMatakuliah = nama; // initializes courseName
   } // end constructor

   // method set untuk nama matakuliah
   public void setNamaMatakuliah( String nama )
   {
      namaMatakuliah = nama; // simpan ke namaMatakuliah
   } // akhir dari method set

   // method get namaMatakuliah
   public String getNamaMatakuliah()
   {
      return namaMatakuliah;
   } // akhir dari method get

   // Menampilkan pesan kepada user
   public void tampilPesan()
   {
      // perintah berikut memanggil getNamaMatakuliah untuk mendapatkan
      // nama matakuliah di buku raport
      System.out.printf( "Selamat datang di buku raport\n%s!\n",
         getNamaMatakuliah() );
   } // akhir dari method tampil pesan

} // akhir dari kelas raport
```

Constructor untuk mengawali variabel `Nama matakuliah`

```
1  // Fig. 3.11: GradeBookTest.java
2  // GradeBook constructor used to specify the course name at the
3  // time each GradeBook object is created.
4
5  public class NilaiRaport
6  {
7     // main mulai dari sini
8     public static void main( String args[] )
9     {
10       // bentuk obyek myRaport1 dan myRaport2
11       Raport myRaport1 = new Raport(
12          "CS101 Introduction to Java Programming" );
13       Raport myRaport2 = new Raport(
14          "CS102 Data Structures in Java" );
15
16       // tampilkan nilai awal matakuliah
17       System.out.printf( "Nama Matakuliah pertama: %s\n",
18          myRaport1.getNamaMatakuliah();
19       System.out.printf( "NamaMatakuliah kedua: %s\n",
20          myRaport2.getNamaMatakuliah();
21    } // akhir main
22
23 } // akhir kelas raport
```

Panggil constructor untuk membentuk obyek pertama

Bentuk obyek kedua

```
gradeBook1 course name is: CS101 Introduction to Java Programming
gradeBook2 course name is: CS102 Data Structures in Java
```

Diagram Kelas UML

- Constructor diletakkan pada bagian metode
- Tuliskan “<<constructor>>” sebelum nama constructor
- Letakkan constructor pada baris pertama dari bagian metode

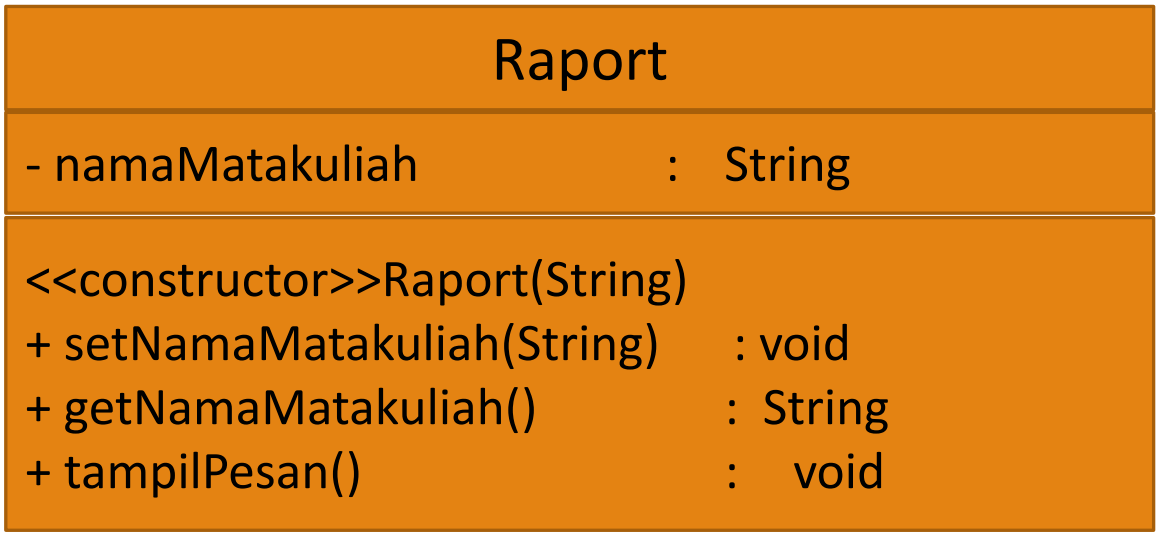


Fig. 3.12 | UML class diagram indicating that class **`Raport`** has a constructor that has a **name** parameter of UML type **`String`**.

Contoh constructor

– Perhatikan constructor kelas RekeningBank yang akan memberi nilai saldo awal=0.

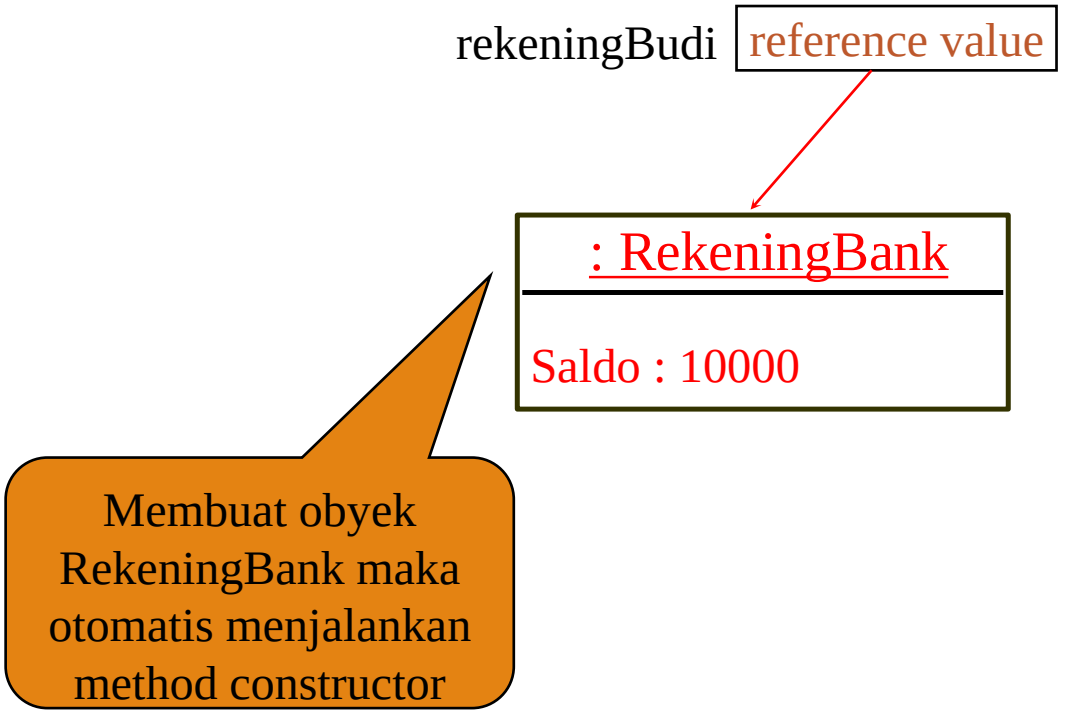
```
public class RekeningBank
{
  private int saldo;

  RekeningBank()
  {
   saldo=10000;
  }
  public int saldo()
  {
    return saldo;
  }

  public void simpanan( int jumlah )
  {
    saldo = saldo + amount;
  }
}
```

Contoh Constructor

```
RekeningBank rekeningBudi = new RekeningBank();
```



OVERLOAD

Overloaded Methods

2 atau lebih metode dapat memiliki nama yang sama asalkan

- Mereka memiliki jumlah parameter yang berbeda (Aturan 1) atau
- Parameternya memiliki tipe yang berbeda jika jumlahnya sama (Aturan 2)
- Urutan tipe data parameter berbeda (Aturan 3)

```
public void myMethod(int x, int y) { ... }
public void myMethod(int x) { ... }
```

✓ Aturan 1

```
public void myMethod(double x) { ... }
public void myMethod(int x) { ... }
```

✓ Aturan 2

```
public void myMethod(double x, int y) { ... }
public void myMethod(int x, double y) { ... }
```

✓ Aturan 3

Overload : Jumlah Parameter Beda

```
class DisplayOverloading
{
    public void disp(char c)
    {
        System.out.println(c);
    }
    public void disp(char c, int num)
    {
        System.out.println(c + " "+num);
    }
}

class Sample {
    public static void main(String args[])
    {
        DisplayOverloading obj = new DisplayOverloading();
        obj.disp('a’);
        obj.disp('a',10);
    }
}
```

Overload : Tipe Parameter Beda

```
class DisplayOverloading2
{
    public void disp(char c)
    {
        System.out.println(c);
    }
    public void disp(int c)
    {
        System.out.println(c);
    }
}

class Sample2 {
    public static void main(String args[])
    {
        DisplayOverloading2 obj = new DisplayOverloading2();
        obj.disp('a');
        obj.disp(5);
    }
}
```

Overload : Urutan Tipe Parameter Beda

```
class DisplayOverloading3
{
    public void disp(char c, int num)
    {
        System.out.println("first");
    }
    public void disp(int num, char c)
    {
        System.out.println("second");
    }
}

class Sample3 {
    public static void main(String args[])
    {
        DisplayOverloading3 obj = new DisplayOverloading3();
        obj.disp('a',50);
        obj.disp(5,'y');
    }
}
```

Overloaded Constructor

Aturan yang sama berlaku juga untuk metode konstruktor

- Oleh karena itu kita dapat mendefinisikan lebih dari 1 konstruktor dalam satu kelas

```
public Person( ) { ... }
public Person(int age) { ... }
```

✓ Aturan 1

```
public Pet(int age) { ... }
public Pet(String name) { ... }
```

✓ Aturan 2

```
public Person(double x, int y) { ... }
public Person(int x, double y) { ... }
```

✓ Aturan 3

Overload Constructor

```
class StudentData {
    private int stuID;
    private String stuName;
    private int stuAge;
    StudentData()
    { //Default constructor
        stuID = 100;
        stuName = "New Student";
        stuAge = 18;
    }
    StudentData(int num1, String str, int num2)
    { //Parameterized constructor
        stuID = num1;
        stuName = str;
        stuAge = num2;
    }
    StudentData(String str, int num2)
    { //Parameterized constructor
        stuID = 100;
        stuName = str;
        stuAge = num2;
    }
    public String toString()
    {
        return String.format("%3d %12s %3d",stuID,stuName,stuAge);
    }}
```

Overload Constructor

```
class StudentMain {
    public static void main(String[] args)
    {
        StudentData sd1 = new StudentData();
        System.out.println(sd1.toString());
        StudentData sd2 = new StudentData("Nani", 16);
        System.out.println(sd2.toString());
        StudentData sd3 = new StudentData(300, "Budi", 17);
        System.out.println(sd3.toString());

}
```

Konstruktor dan this

Untuk memanggil suatu konstruktor dari konstruktor lain dalam kelas yang sama, digunakan kata kunci this().

```
class StudentData {
   private int stuID;
   private String stuName;
   private int stuAge;
   StudentData()
   { //Default constructor
      this(100, "New Student", 18);
   }
   StudentData(int num1, String str, int num2)
   { //Parameterized constructor
      stuID = num1;
      stuName = str;
      stuAge = num2;
   }
   StudentData(String str, int num2)
   { //Parameterized constructor
      this(100, str, num2);
   }
```

STATIC

Metode / variable static

Kata kunci static digunakan untuk mendefinisikan metode milik suatu kelas

```
public static int gcd(int m, int n) {

   //the code implementing the Euclidean algorithm
}


public static Fraction min(Fraction f1, Fraction f2)
{

   //convert to decimals and then compare

}
```

Metode / variable static

Dengan kata kunci static, maka jika kita memanggil method tidak perlu membuat instance dari kelas.

Maka method main adalah static

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

Static Fº to Cº Convert Example

```
public class FtoC
{
 public static double convert( double degreesF )
        { return 5.0 / 9.0 * (degreesF - 32 ); }
}
```

```
public class F2CDemo
{
 public static void main( String[ ] args )
 {
        double degreesF = 100;

        // Since convert is static, no object is needed
        // The class name is used when convert is called

        double degreesC = FtoC.convert( degreesF );
        System.out.println( degreesC );
 }
}
```

Metode / variable static

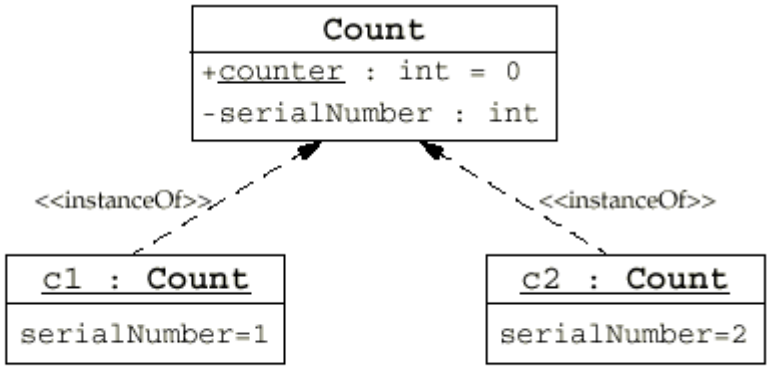
Anggota class yang bersifat static ini sering disebut dengan “class members” (class variable dan class methods).

Class variable bersifat milik bersama dalam arti semua instance/obyek dari class yang sama akan mempunyai class variable milik bersama.

Class variable mirip dengan global variable.

```java
public class Count {
    private int serialNumber;
    public static int counter=0;
    public Count(){
        counter++;
        serialNumber = counter ;
    }

    public int getSerialNumber() {
        return serialNumber;
    }
}
```




```java
class TestCount{
    public static void main(String args[]){
        Count c1 = new Count();
        System.out.println("Objek c1");
        System.out.println("serial number : " + c1.getSerialNumber());
        System.out.println("counter : " + c1.counter) ;

        Count c2 = new Count();
        System.out.println();
        System.out.println("Objek c1 setelah create object c2");
        System.out.println("serial number : " + c1.getSerialNumber());
        System.out.println("counter : " + c1.counter) ;
        System.out.println();
        System.out.println("Objek c2");
        System.out.println("serial number : " + c2.getSerialNumber());
        System.out.println("counter : " + c2.counter) ;
    }
}
```

Output:

Objek c1

serial number : 1

counter : 1

Objek c1 setelah create object c2

serial number : 1

counter : 2

Objek c2

serial number : 2

counter : 2

The best tip to be a good programmer:

practice, practice and practice