Pengenalan pemrograman berorientasi obyek

Oleh:

Ali Ridho Barakbah

Keuntungan OOP

- Reusabilitas
- Pembangunan program lebih cepat
- Fleksibilitas lebih tinggi
- Ekstensibilitas
- Less maintenance

Kata kunci OOP

- **Objek** → dapat berupa Class atau Instances. Harus berasal dari entitas atau konsep dunia nyata.
- **Atribut** → identitas unik dari obyek
- **Metode** → fungsi untuk pengaksesan atribut atau tugas tertentu
- **Enkapsulasi** → tersembunyi struktur data dan implementasi dari obyek lain.
- **Inheritansi** → merepresentasikan keterhubungan struktural antar obyek
- **Polymorphism** → kemampuan untuk merepresentasikan 2 bentuk yang berbeda

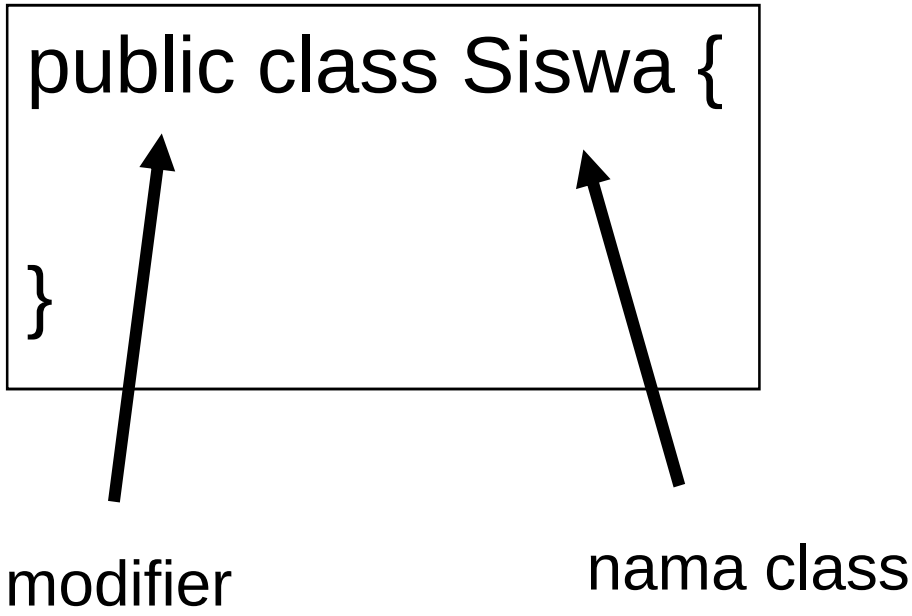
Fitur OOP

- Encapsulation
- Inheritance
- Polymorphism

Deklarasi class

```
<modifier> class <classname> {
        [deklarasi_atribut]
        [deklarasi_konstruktor]
        [deklarasi_metode]
}
```

Contoh



Deklarasi Atribut

<modifier> <tipe> <nama_atribut>;

Contoh

```
public class Siswa {
        public int nrp;
        public String nama;
}
```

atribut

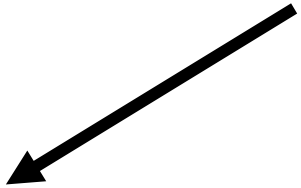
Deklarasi metode

```
<modifier> <return_type> <nama_metode>
                ([daftar_argumen]) {
                        [<statement>]
                }
```

Contoh

```
public class Siswa {
        public int nrp;
        public String nama;
        public void info() {
                System.out.println(nrp + " " + nama + " "
                        + "adalah siswa PENS");
        }
}
```

metode



Pengaksesan anggota obyek

```
public class IsiData {
        public static void main(String args[ ]) {
                Siswa IT2=new Siswa();
                IT2.nrp=5;
                IT2.nama=”Andi”;
                IT2.info();
        }
}
```

Reference type

- Tipe selain tipe primitif dinamakan reference type
- Reference type adalah tipe berbentuk suatu class
- Pembuatan suatu reference type untuk mengalokasikan memori dilakukan dengan menggunakan kata kunci **new** XXX() dimana XXX adalah konstruktor dari reference type

Contoh

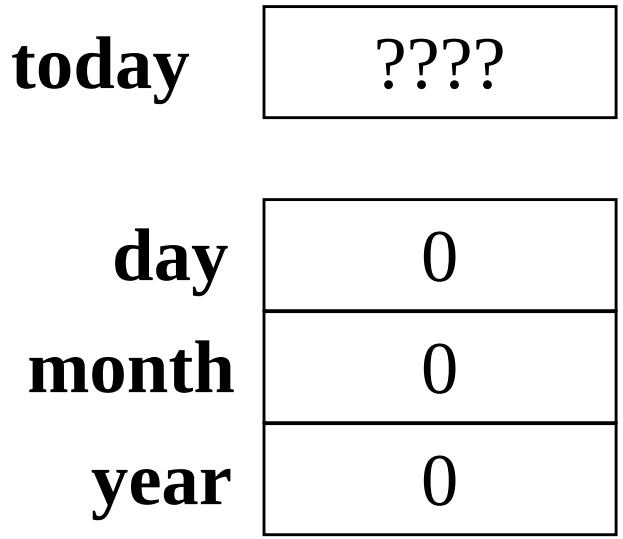
```
public class MyDate {
   private int day=1;
   private int month=1;
   private int year=2000;

   public MyDate(int day, int month, int year)
   {…}
}
```

```
public class TestMyDate {
     public static void main(String args[]) {
       MyDate today = new MyDate(10,11,2006);
     }
}
```

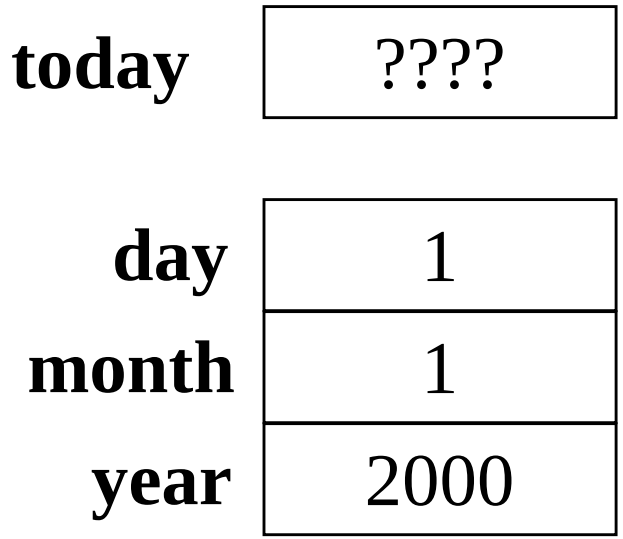
Alokasi memori

```
MyDate today = new MyDate(10, 11, 2006);
```

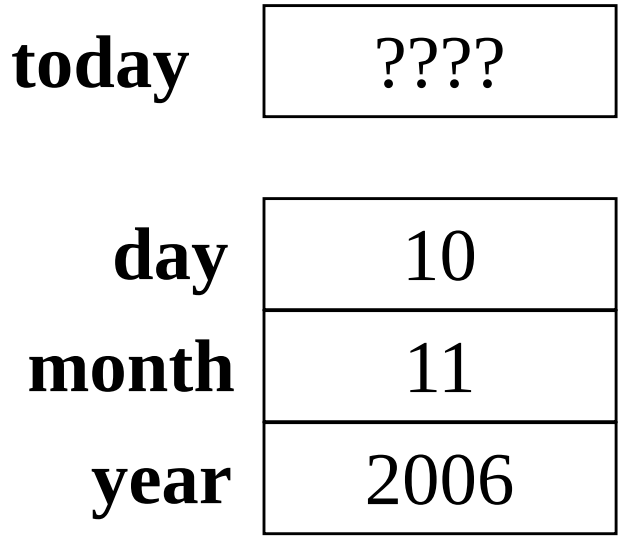
```
MyDate today = new MyDate(10, 11, 2006);
```



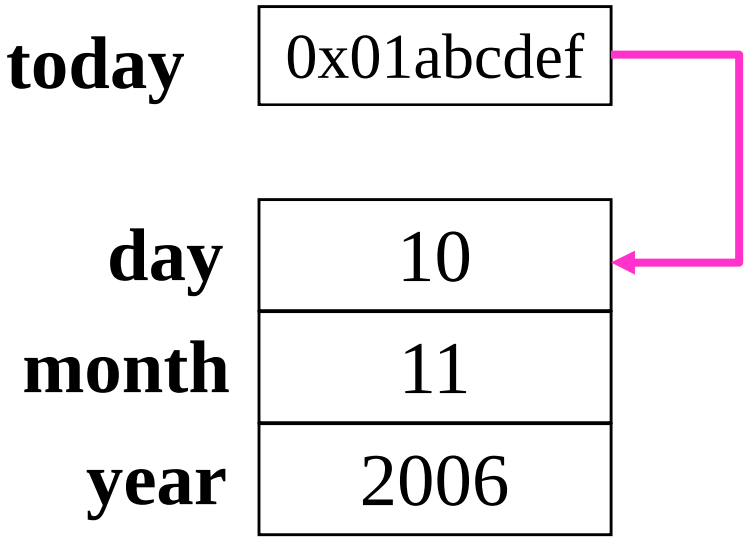
```
MyDate today = new MyDate(10, 11, 2006);
```



```
MyDate today = new MyDate(10, 11, 2006);
```

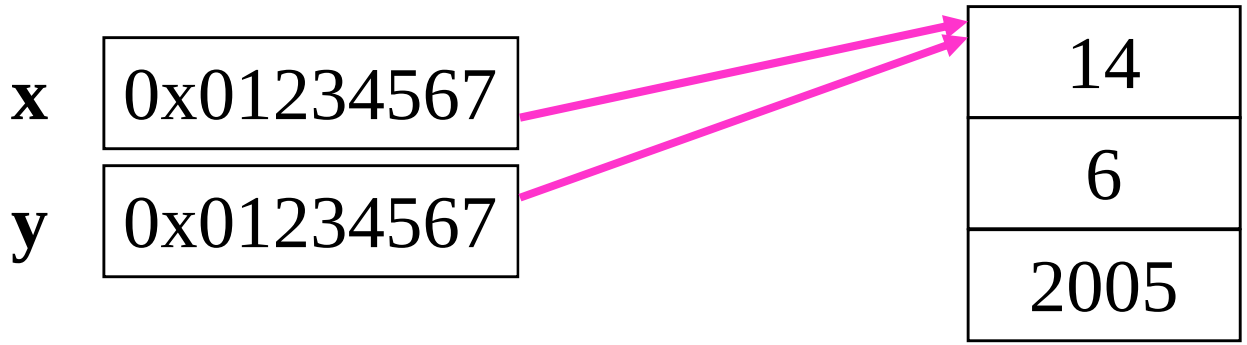


```
MyDate today = new MyDate(10, 11, 2006);
```



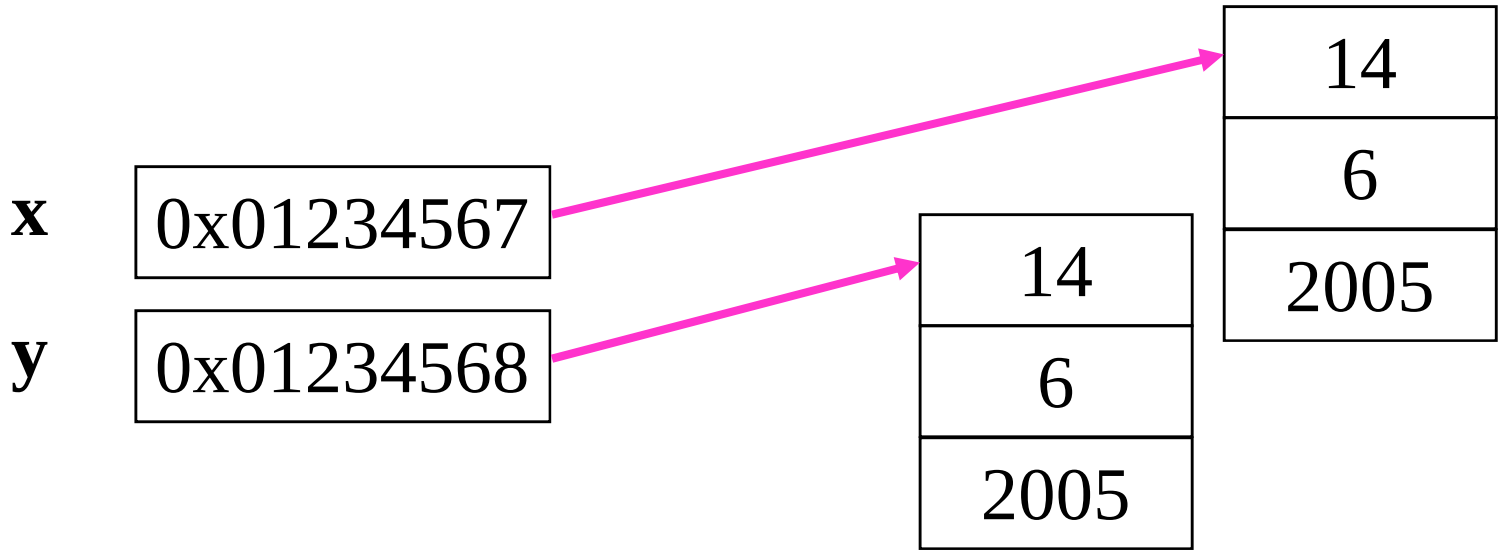
Men-assign reference variable

```
MyDate x = new MyDate(14, 6, 2005);
MyDate y = x;
```



Men-assign reference variable

```
MyDate x = new MyDate(14, 6, 2005);
MyDate y = x;
y = new MyDate(14, 6, 2005);
```



Pass by value

- Java tidak membolehkan adanya pass by reference, jadi hanya mengijinkan pass by value.
- Ketika argumen yang di-passing adalah bertipe reference type, maka anggota-anggota (data member) dari argumen tersebut diperlakukan sebagai pass by reference, sedangkan argumennya tetap (dianggap) sebagai pass by value

Contoh

```
public class MyDate {
        private int day=1;
        private int month=1;
        private int year=2000;
        public MyDate(int day, int month, int year) {
                …
        }
        public void setDay(int day) {
                // change the day
        }
        public void print() {
                // print the day, month and year
        }
}
```

```
public class TestMyDate {
        public static void changeInt(int value) {
                value = 10;
        }
        public static void changeObjectRef(MyDate ref) {
                ref = new myDate(3, 5, 2003);
        }
        public static void changeObjectAttr(Mydate ref) {
                ref.setDay(5);
        }
        public static void main(String args[]) {
                int x=5;
                changeInt(x);
                System.out.println(x);
                MyDate today=new MyDate(10,10,2005);
                changeObjectRef(today);
                today.print();
                changeObjectAttr(today);
                today.print();
        }
}
```

Hasil eksekusi

> java TestMyDate

5

10-10-2005

5-10-2005

Class Fundamentals: main method

- The *main()* Method

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

  - public : method main() dapat diakses oleh apa saja, termasuk java technology interpreter.
  - static : keyword ini berfungsi untuk memberi tahu kompiler bahwa method main bisa langsung digunakan dalam context class yang bersangkutan. Untuk mengeksekusi/menjalankan method yang bertipe static, tidak diperlukan instance nya.
  - void : menunjukkan bahwa method main() tidak mengembalikan nilai
  - main : merupakan nama method utama dari program java
  - String [] args : Menyatakan bahwa method main() menerima single parameter yaitu args yang bertipe array. Digunakan pada saat memasukkan parameter pada saat menjalankan program.

Contoh: `java TestGreeting args[0] args[1] …`

Declaring Constructors

- Basic syntax of a constructor:

```
<modifier> <class_name> ([<argument_list>]) {
      [<statements>]
   }
```

- Example:

```
1 public class Dog {
2   private int weight;
3
4   public Dog() {
5      weight = 42;
6   }
7
8   public int getWeight() {
9     return weight;
10  }
11  public void setWeight(int newWeight) {
12    weight = newWeight;
13  }
14 }
```

- **Modifier : public, private, protected, dan default**
- **Constructor adalah bukan method, sehingga tidak punya return values dan tidak diturunkan/diwariskan**

Access Control

Modifier	Same Class	Same Package	Subclass	Universe
`private`	Yes			
default	Yes	Yes		
`protected`	Yes	Yes	Yes	
`public`	Yes	Yes	Yes	Yes

- **Modifier : public, private, protected, dan default**

The Default Constructor

- There is always at least one constructor in every class.
- If the writer does not supply any constructors, the default constructor is present automatically:
  - The default constructor takes no arguments
  - The default constructor has no body
- Enables you to create object instances with `new Xxx()` without having to write a constructor.

Note:

- Jika kita mendeklarasikan constructor pada suatu class yang sebelumnya tidak mempunyai constructor, maka default constructor class tersebut akan hilang.
- Sehingga bila constructor yang kita buat tadi mempunyai argumen, kemudian kita buat obyek dengan cara new Xxx(), proses kompilasi akan menghasilkan error.

Constructing and Initializing Objects

- Calling `new Xxx()` to allocate space for the new object results in:
  - Memory allocation: Space for the new object is allocated and instance variables are initialized to their default values (for example, `0`, `false`, `null`, and so on)
  - Explicit attribute initialization is performed
  - A constructor is executed
  - Variable assignment is made to reference the object
- Example:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```

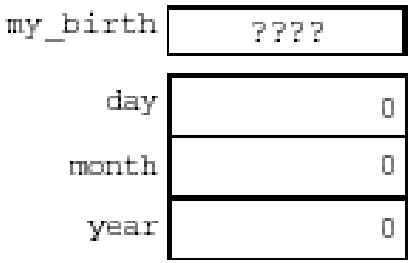
Memory Allocation and Layout

- A declaration allocates storage only for a reference:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```

- Use the `new` operator to allocate space for `MyDate`:

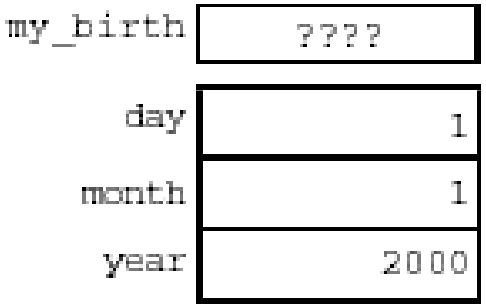
```
MyDate my_birth = new MyDate(22, 7, 1964);
```



Explicit Attribute Initialization

- Initialize the attributes:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```

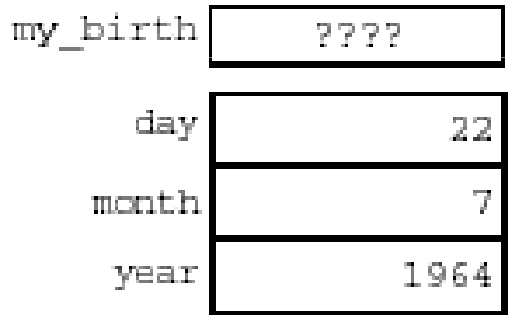


- The default values are taken from the attribute declaration in the class.

Executing the Constructor

- Execute the matching constructor:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```

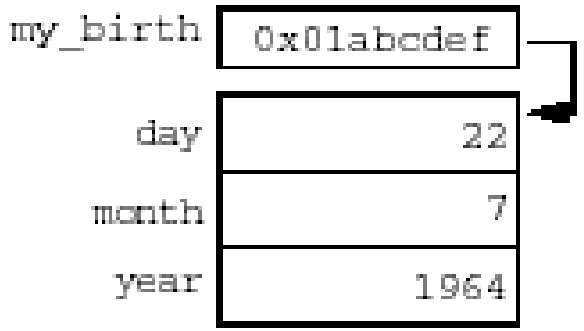


- In the case of an overloaded constructor, the first constructor may call another.

Assigning a Variable

- Assign the newly created object to the reference variable:

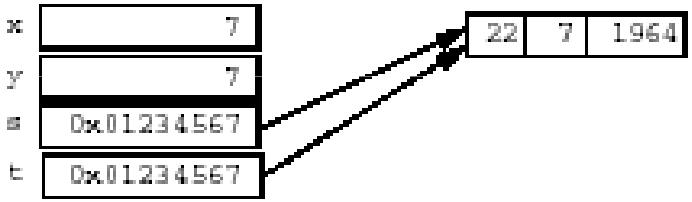
```
MyDate my_birth = new MyDate(22, 7, 1964);
```



Assigning Reference Variables

```
int x = 7;
int y = x;
MyDate s = new MyDate(22, 7, 1964);
MyDate t = s;
```

- Two variables refer to a single object:




```
t = new MyDate(22, 12, 1964);
```

- Reassignment makes two variables point to two objects: