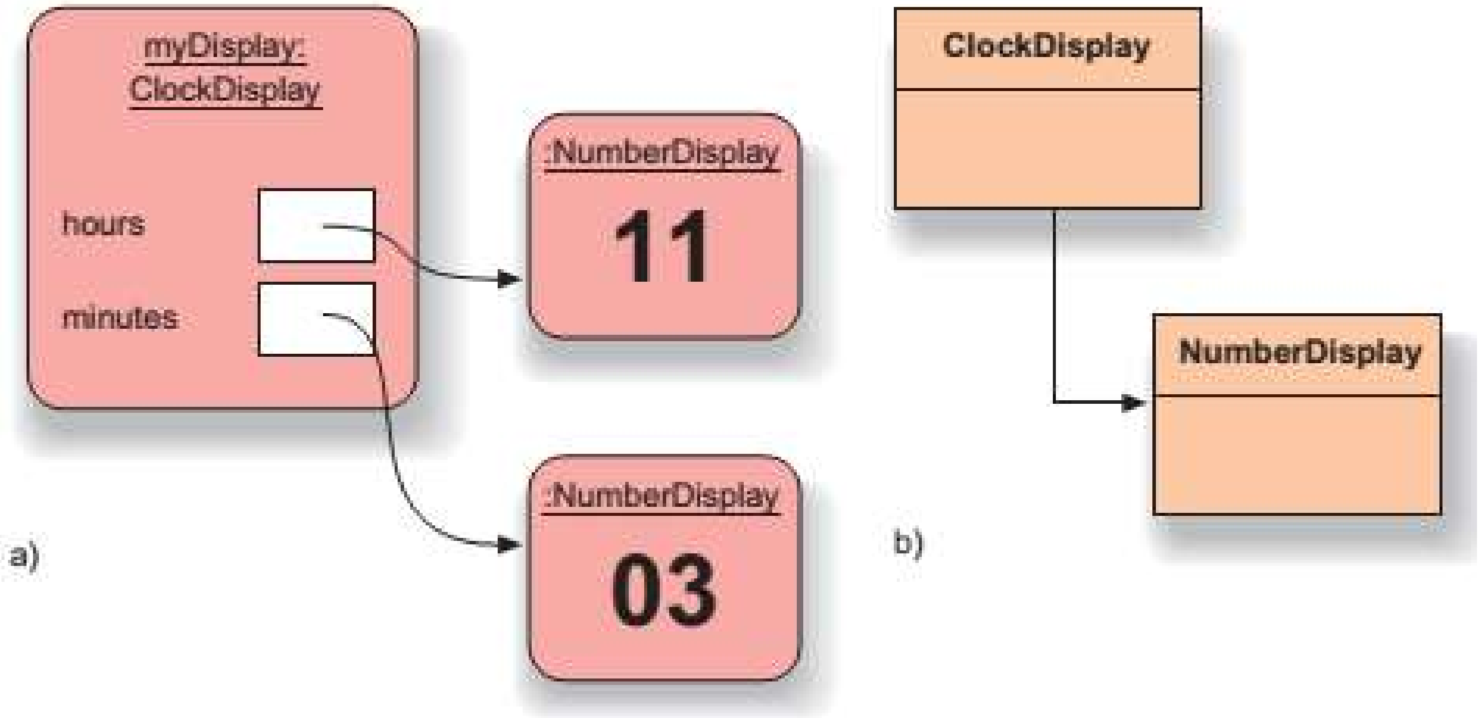
Interaksi Object

- Manipulasi attribute dan method melalui class lain
- Hak akses
- Static
- Overloading

Class ClockDisplay

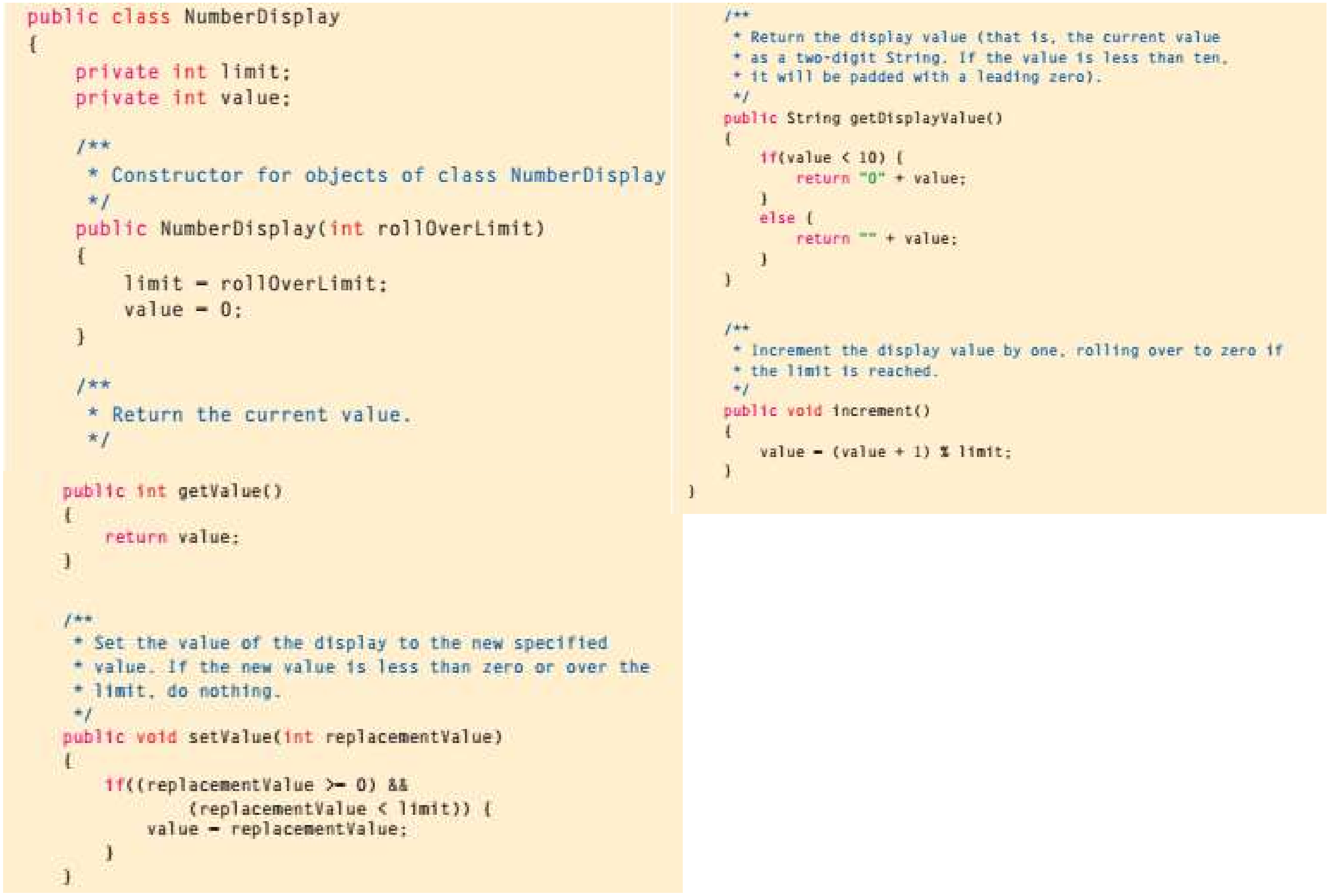


Implementasi Class

```
public class NumberDisplay
{
    private int limit;
    private int value;

    Constructor and methods omitted.
}
```

```
public class ClockDisplay
{
    private NumberDisplay hours;
    private NumberDisplay minutes;

    Constructor and methods omitted.
}
```

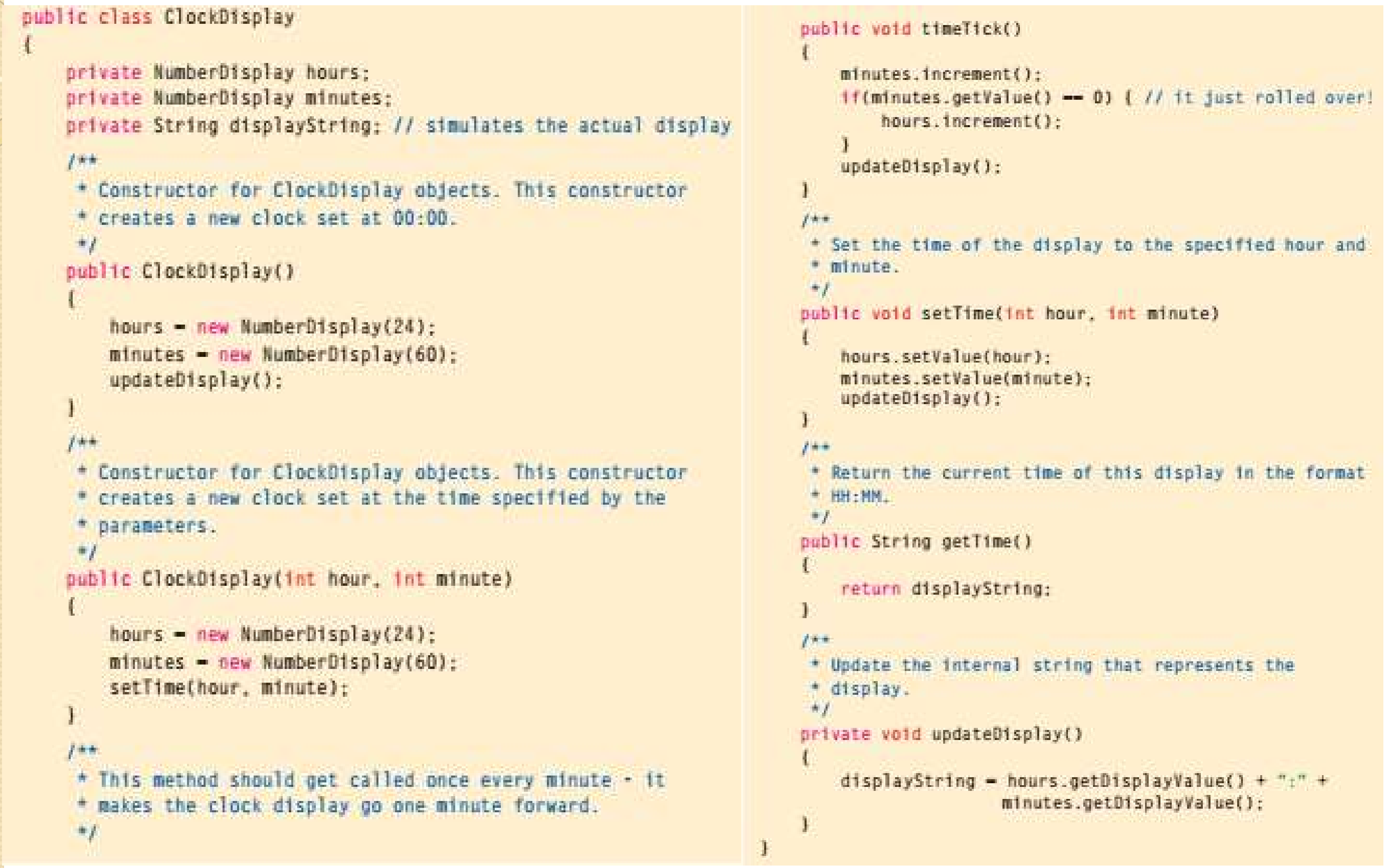
Implementasi NumberDisplay



```
public class NumberDisplay
{
    private int limit;
    private int value;

    /**
     * Constructor for objects of class NumberDisplay
     */
    public NumberDisplay(int rollOverLimit)
    {
        limit = rollOverLimit;
        value = 0;
    }

    /**
     * Return the current value.
     */
    public int getValue()
    {
        return value;
    }

    /**
     * Set the value of the display to the new specified
     * value. If the new value is less than zero or over the
     * limit, do nothing.
     */
    public void setValue(int replacementValue)
    {
        if((replacementValue >= 0) &&
                (replacementValue < limit)) {
            value = replacementValue;
        }
    }
```

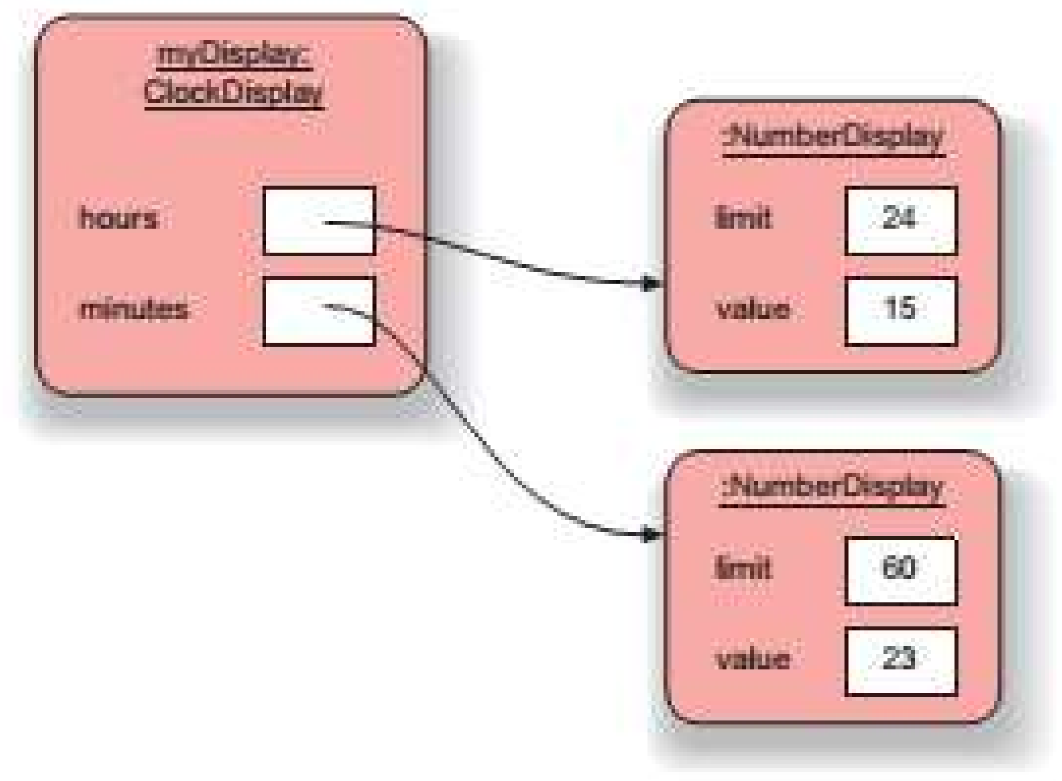
```
    /**
     * Return the display value (that is, the current value
     * as a two-digit String. If the value is less than ten,
     * it will be padded with a leading zero).
     */
    public String getDisplayValue()
    {
        if(value < 10) {
            return "0" + value;
        }
        else {
            return "" + value;
        }
    }

    /**
     * Increment the display value by one, rolling over to zero if
     * the limit is reached.
     */
    public void increment()
    {
        value = (value + 1) % limit;
    }
}
```

Implementasi ClockDisplay

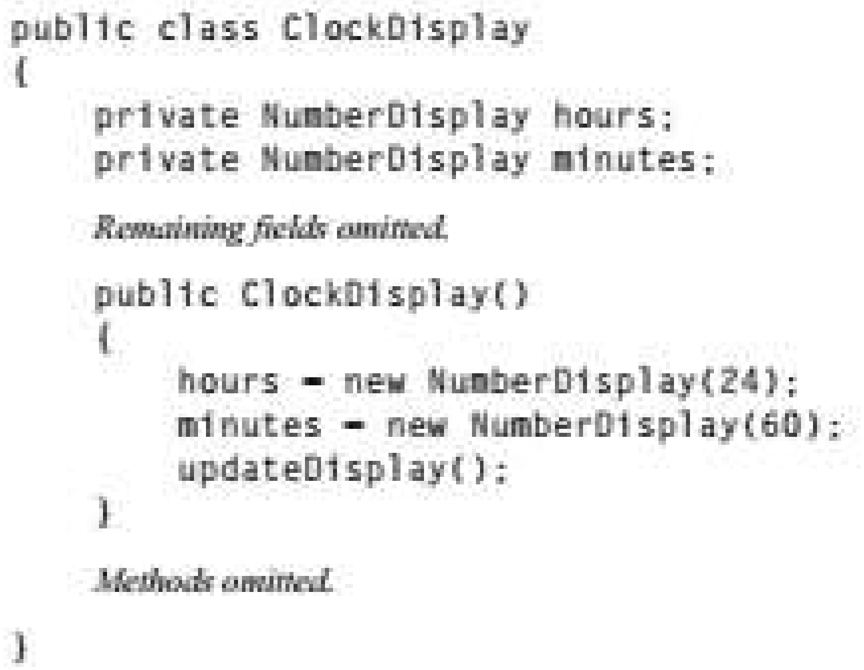


```java
public class ClockDisplay
{
    private NumberDisplay hours;
    private NumberDisplay minutes;
    private String displayString; // simulates the actual display

    /**
     * Constructor for ClockDisplay objects. This constructor
     * creates a new clock set at 00:00.
     */
    public ClockDisplay()
    {
        hours = new NumberDisplay(24);
        minutes = new NumberDisplay(60);
        updateDisplay();
    }

    /**
     * Constructor for ClockDisplay objects. This constructor
     * creates a new clock set at the time specified by the
     * parameters.
     */
    public ClockDisplay(int hour, int minute)
    {
        hours = new NumberDisplay(24);
        minutes = new NumberDisplay(60);
        setTime(hour, minute);
    }

    /**
     * This method should get called once every minute - it
     * makes the clock display go one minute forward.
     */
    public void timeTick()
    {
        minutes.increment();
        if(minutes.getValue() == 0) { // it just rolled over!
            hours.increment();
        }
        updateDisplay();
    }

    /**
     * Set the time of the display to the specified hour and
     * minute.
     */
    public void setTime(int hour, int minute)
    {
        hours.setValue(hour);
        minutes.setValue(minute);
        updateDisplay();
    }

    /**
     * Return the current time of this display in the format
     * HH:MM.
     */
    public String getTime()
    {
        return displayString;
    }

    /**
     * Update the internal string that represents the
     * display.
     */
    private void updateDisplay()
    {
        displayString = hours.getDisplayValue() + ":" +
                        minutes.getDisplayValue();
    }
}
```

Object Diagram ClockDisplay



Objects creating Object



```
public class ClockDisplay
{
    private NumberDisplay hours;
    private NumberDisplay minutes;

    Remaining fields omitted.

    public ClockDisplay()
    {
        hours = new NumberDisplay(24);
        minutes = new NumberDisplay(60);
        updateDisplay();
    }

    Methods omitted.

}
```

- Syntax membuat object

  new ClassName (parameter-list)

Object creating object

- Ada 2 operasi :
  - Membuat obyek dari nama kelas (*NumberDisplay*)
  - Eksekusi konstruktor dari class

Ex :

```
public NumberDisplay (int  ollOverLimit)
new NumberDisplay (24);
```

Class Access Level

Specifier	Class	Package	SubClass	World
private	✓			
no specifier	✓	✓		
protected	✓	✓	✓	
public	✓	✓	✓	✓

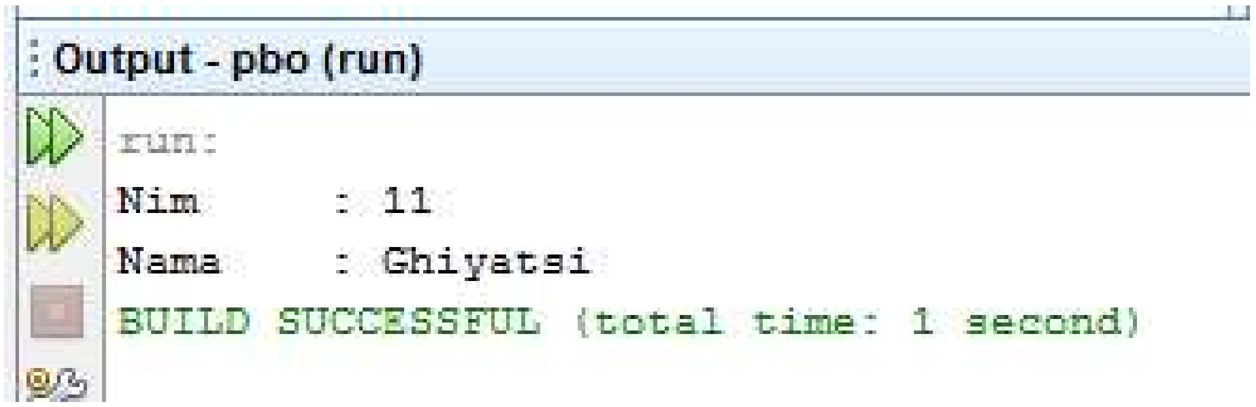
Encaptulation

- Information hiding
- Interface to access data (cara untuk mengubah nilai pada suatu variabel yang telah lakukan *information hiding*)

```
public class Mahasiswa{
  private int nim;
  private String nama;
  public Mahasiswa (int nim, String nama){
  this.nim = nim;
  this.nama = nama;
  }
  public int getNim(){
      return nim;
  }
  public String getNama(){
      return nama;
  }}
```

MahasiswaDemo

```
Public class MahasiswaDemo{
Public static void main(String[] args){
    Mahasiswa mhs1=new Mahasiswa(11,"ghiyatsi");
    System.out.println("Nim     : "+mhs1.getNim());
     System.out.println("Nama :  "+mhs1.getNama());
}
```

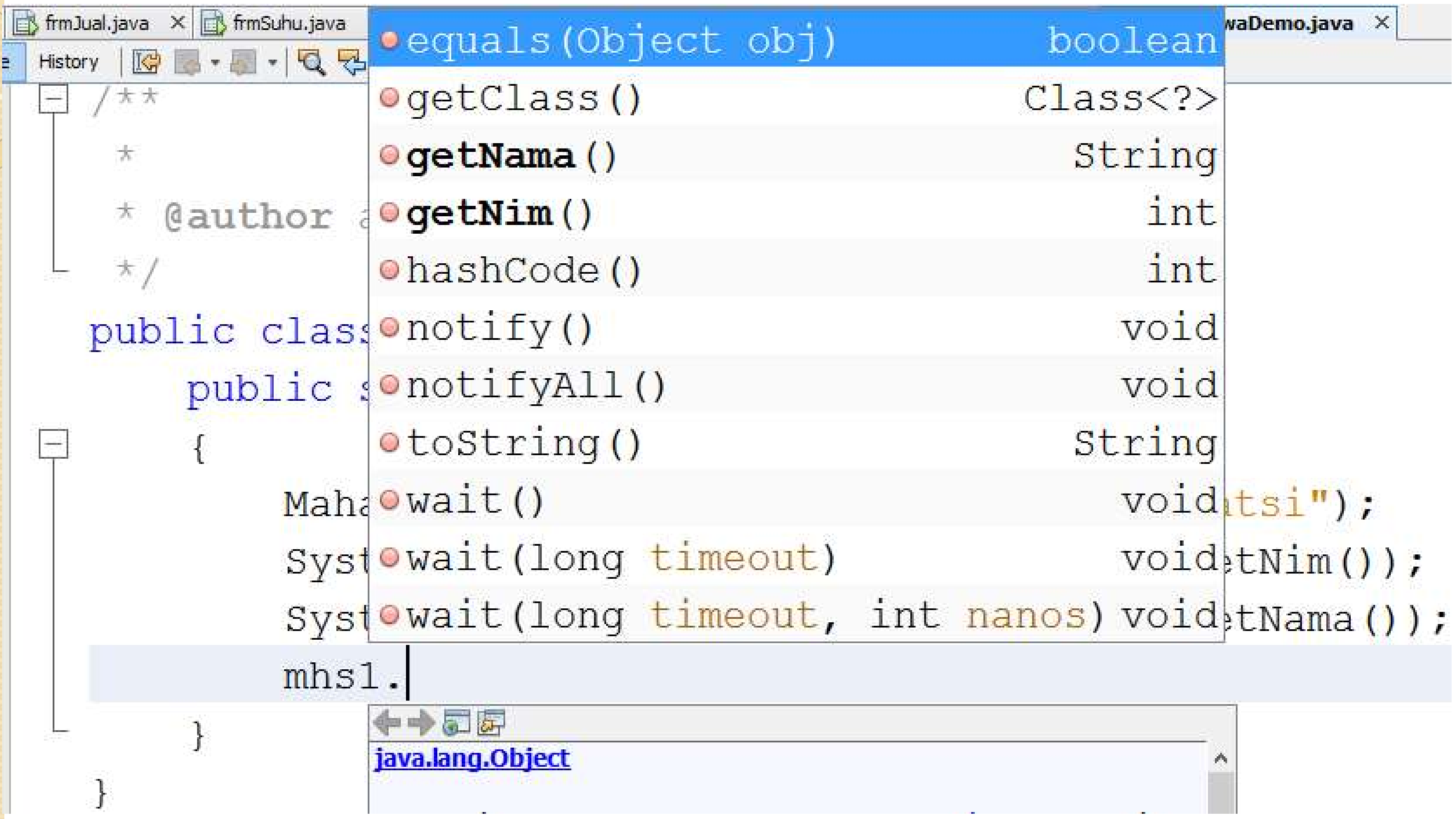

Output - pbo (run)

```
run:
Nim        : 11
Nama       : Ghiyatsi
BUILD SUCCESSFUL (total time: 1 second)
```

Add Atribute IPK

```
public class Mahasiswa{
  private int nim;
  private String nama;
 private float IPK;
  public Mahasiswa (int nim, String
  nama){
  this.nim = nim;
  this.nama = nama;
  }
  public int getNim(){
      return nim;
  }
  public String getNama(){
      return nama;
  }}
```

MahasiswaDemo



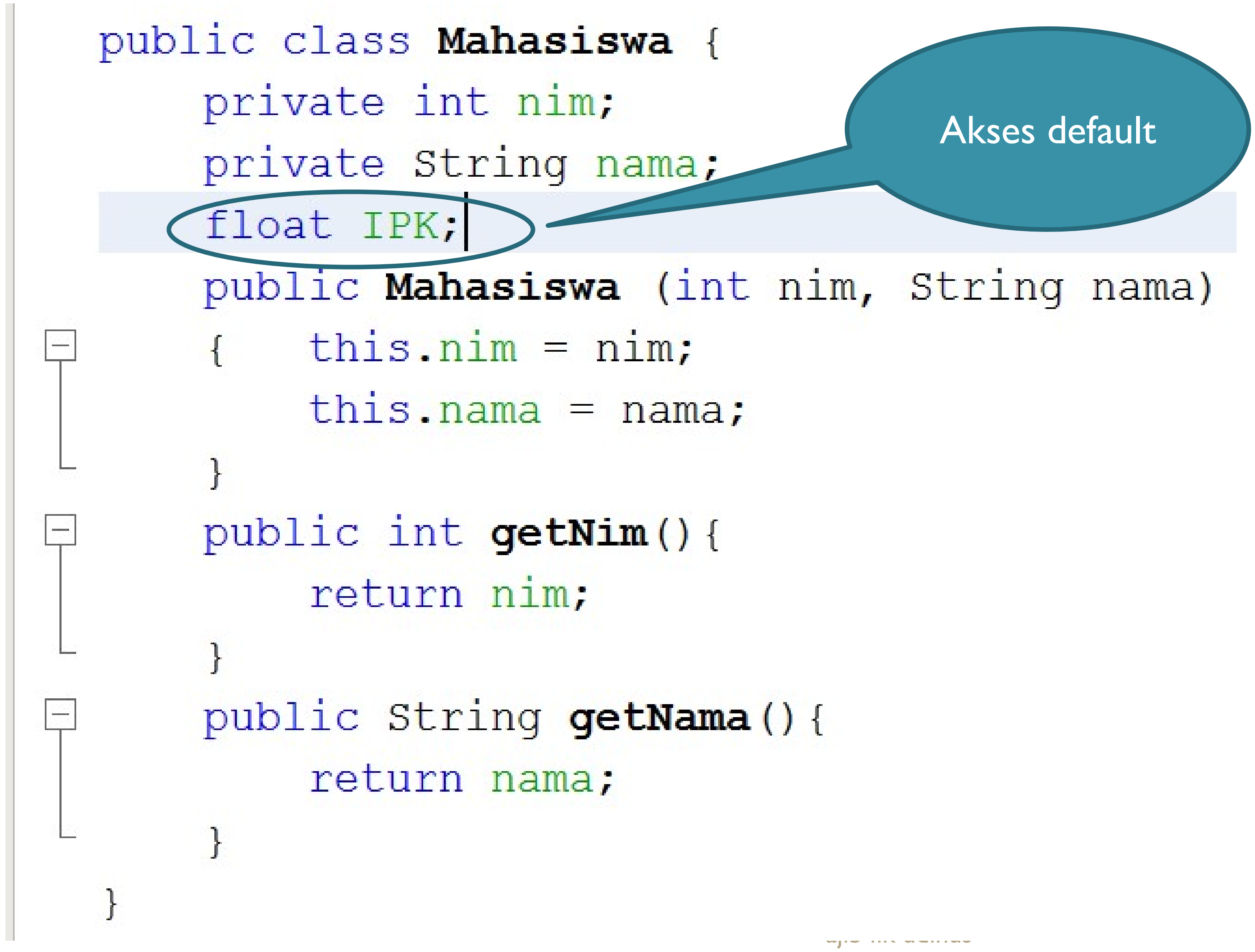
```
/**
 *
 * @author a
 */
public class
    public s
    {
        Maha                                          tsi");
        Syst                                          tNim());
        Syst                                          tNama());
        mhs1.
    }
}
```

```
equals(Object obj)                    boolean
getClass()                            Class<?>
getNama()                             String
getNim()                              int
hashCode()                            int
notify()                              void
notifyAll()                           void
toString()                            String
wait()                                void
wait(long timeout)                    void
wait(long timeout, int nanos)         void
```

java.lang.Object

Karena IPK access private, tdk dpt diakses di class lain

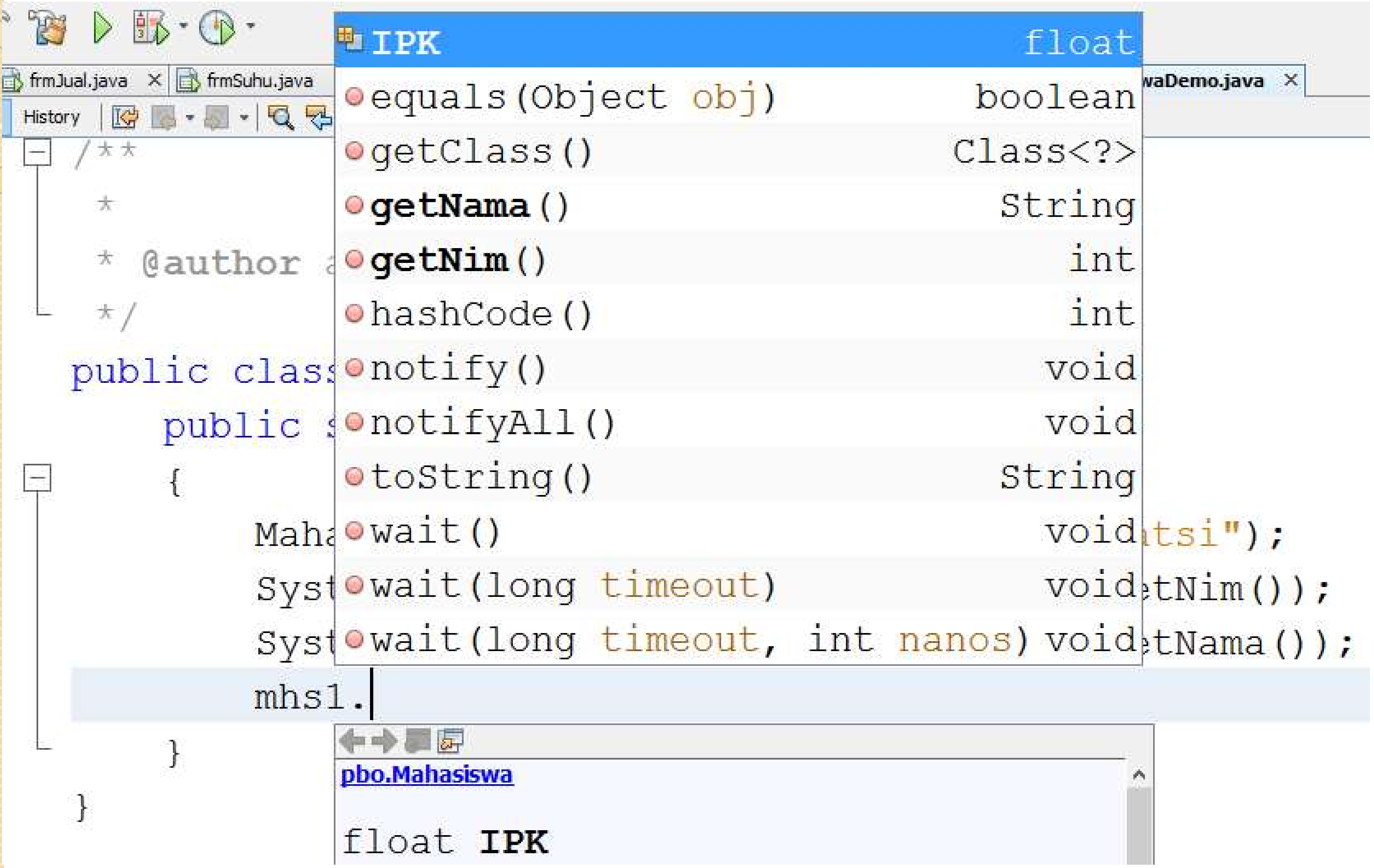
Class Mahasiswa Update



```
public class Mahasiswa {
    private int nim;
    private String nama;
    float IPK;
    public Mahasiswa (int nim, String nama)
    {   this.nim = nim;
        this.nama = nama;
    }
    public int getNim(){
        return nim;
    }
    public String getNama(){
        return nama;
    }
}
```

Akses default

Class MahasiswaDemo



```
/**
 *
 * @author a
 */
public class
    public s
    {
        Maha                                      tsi");
        Syst                                      tNim());
        Syst                                      tNama());
        mhs1.
    }
}
```

```
IPK                                          float
equals(Object obj)                           boolean
getClass()                                   Class<?>
getNama()                                    String
getNim()                                     int
hashCode()                                   int
notify()                                     void
notifyAll()                                  void
toString()                                   String
wait()                                       void
wait(long timeout)                           void
wait(long timeout, int nanos)                void
```

pbo.Mahasiswa

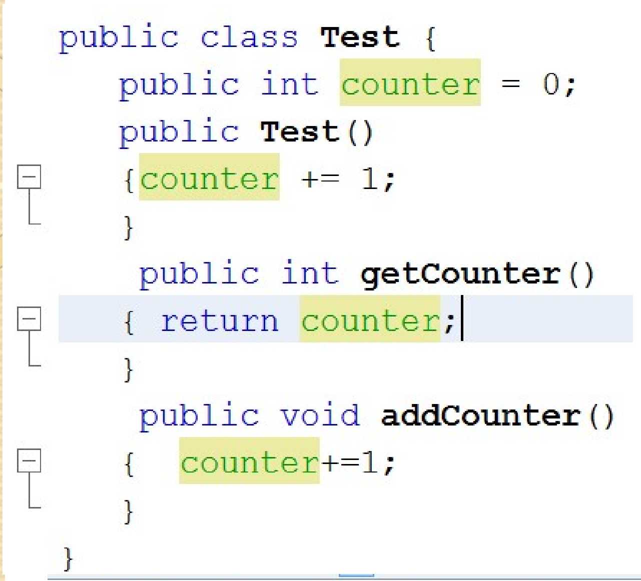
```
float IPK
```

Karena IPK access default, dpt diakses di class lain

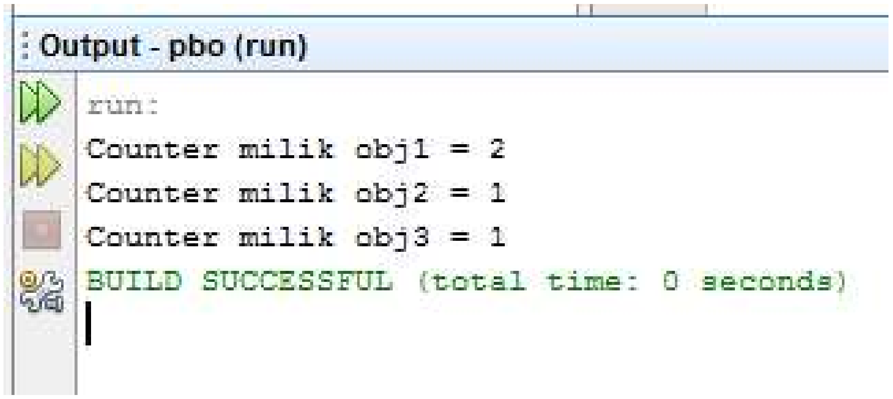
Keyword Static

- Dengan menggunakan static maka method dan variable akan menjadi milik class, bukan menjadi milik suatu instance.
- Bila di suatu class terdapat static variable dan static method, maka apabila ada class lain yang ingin menggunakannya dapat langsung memanggil variable atau method tersebut dengan format: NamaClass.namaStaticVariableAtauMethod. Kita tidak perlu lagi membuat suatu objek dari class tersebut.

Contoh

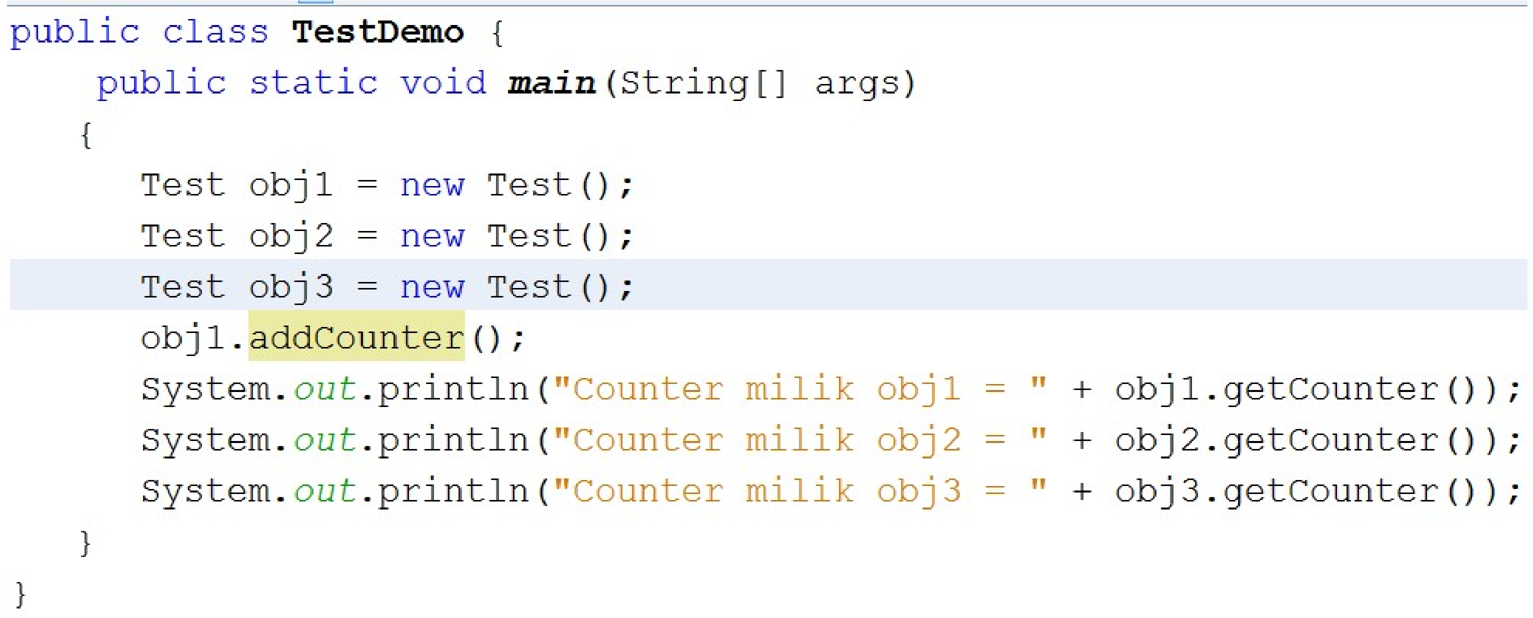


```
public class Test {
   public int counter = 0;
   public Test()
   {counter += 1;
   }
    public int getCounter()
   { return counter;
   }
    public void addCounter()
   {   counter+=1;
   }
}
```



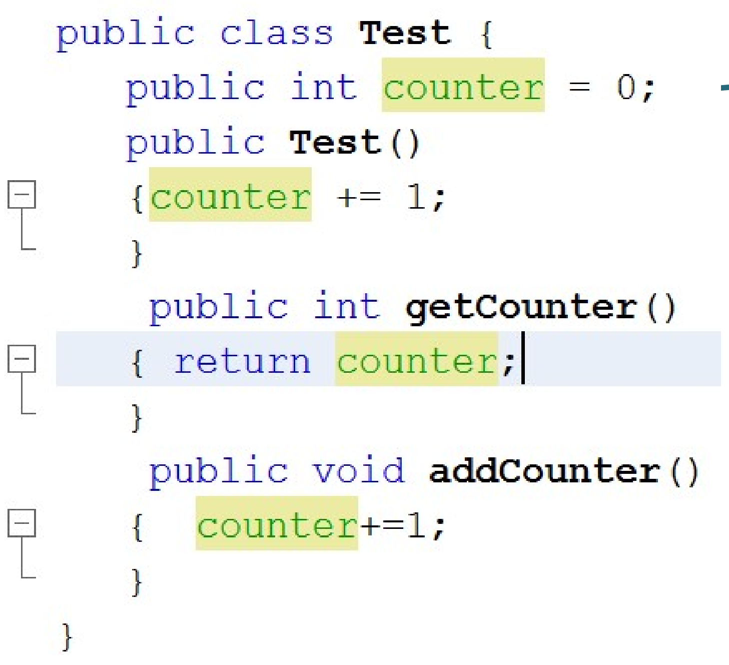
Output - pbo (run)

```
run:
Counter milik obj1 = 2
Counter milik obj2 = 1
Counter milik obj3 = 1
BUILD SUCCESSFUL (total time: 0 seconds)
```



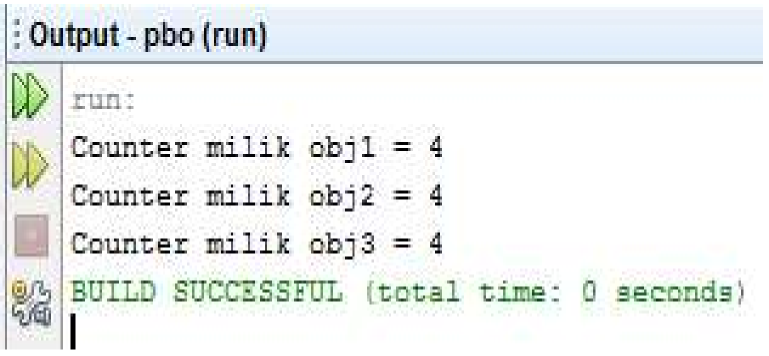
```
public class TestDemo {
    public static void main(String[] args)
   {
      Test obj1 = new Test();
      Test obj2 = new Test();
      Test obj3 = new Test();
      obj1.addCounter();
      System.out.println("Counter milik obj1 = " + obj1.getCounter());
      System.out.println("Counter milik obj2 = " + obj2.getCounter());
      System.out.println("Counter milik obj3 = " + obj3.getCounter());
   }
}
```

Static variable



```
public class Test {
    public int counter = 0;
    public Test()
    {counter += 1;
    }
     public int getCounter()
    { return counter;
    }
     public void addCounter()
    {  counter+=1;
    }
}
```

public static int counter = 0;

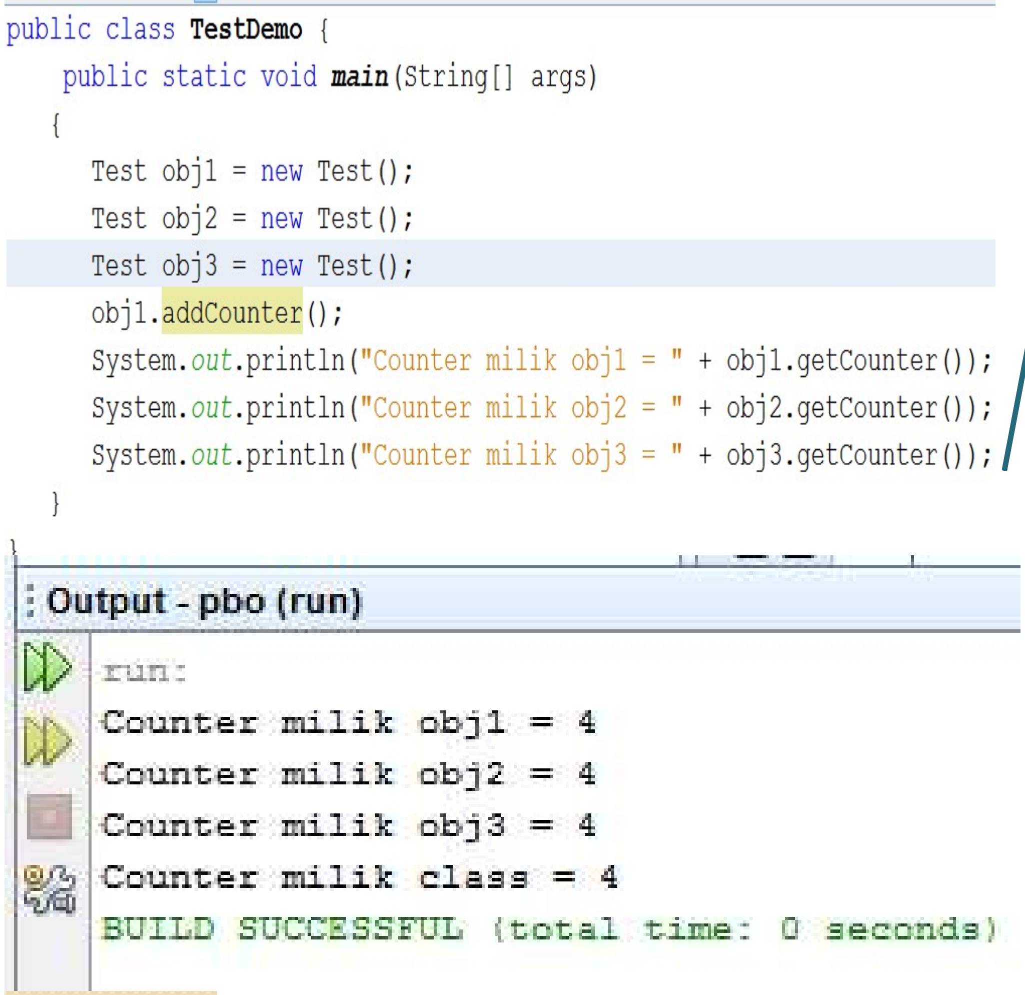


Output - pbo (run)

```
run:
Counter milik obj1 = 4
Counter milik obj2 = 4
Counter milik obj3 = 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

Bila suatu class memiliki static variable, maka variable tersebut akan dipakai bersama² oleh object² dari class tersebut. Setiap objek dari class tersebut akan mengakses variable yang sama. Sehingga obj1, obj2, obj3 menggunakan variable yang sama (shared variable).

Static variable



```
public class TestDemo {
    public static void main(String[] args)
    {
        Test obj1 = new Test();
        Test obj2 = new Test();
        Test obj3 = new Test();
        obj1.addCounter();
        System.out.println("Counter milik obj1 = " + obj1.getCounter());
        System.out.println("Counter milik obj2 = " + obj2.getCounter());
        System.out.println("Counter milik obj3 = " + obj3.getCounter());
    }
}
```

Output - pbo (run)

```
run:
Counter milik obj1 = 4
Counter milik obj2 = 4
Counter milik obj3 = 4
Counter milik class = 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

System.out.println("Counter milik class = " + Test.counter);

Dengan menggunakan static variable maka kita bisa langsung mengakses suatu **state** tanpa harus membuat suatu object terlebih dahulu. Dan juga perlu diingat bahwa untuk instance variable adalah 1 per instance dan untuk static variable adalah 1 per class

Static Method

```
public static void fungsiStatic()
{
    System.out.println("ini fungsi static");
}

public void fungsiBiasa()
{
    System.out.println("ini fungsi biasa");
}
```

Tambahkan method di class Test

```
public class TestDemo {
    public static void main(String[] args)
   {
      Test obj1 = new Test();
      Test obj2 = new Test();
      Test obj3 = new Test();
      obj1.addCounter();
      System.out.println("Counter milik obj1 = " + obj1.getCounter());
      System.out.println("Counter milik obj2 = " + obj2.getCounter());
      System.out.println("Counter milik obj3 = " + obj3.getCounter());
      System.out.println("Counter milik class = " + Test.counter);
      Test.fungsiStatic();
      Test obj = new Test();
      obj.fungsiBiasa();
   }
}
```

Akses langsung dr class

OverLoading

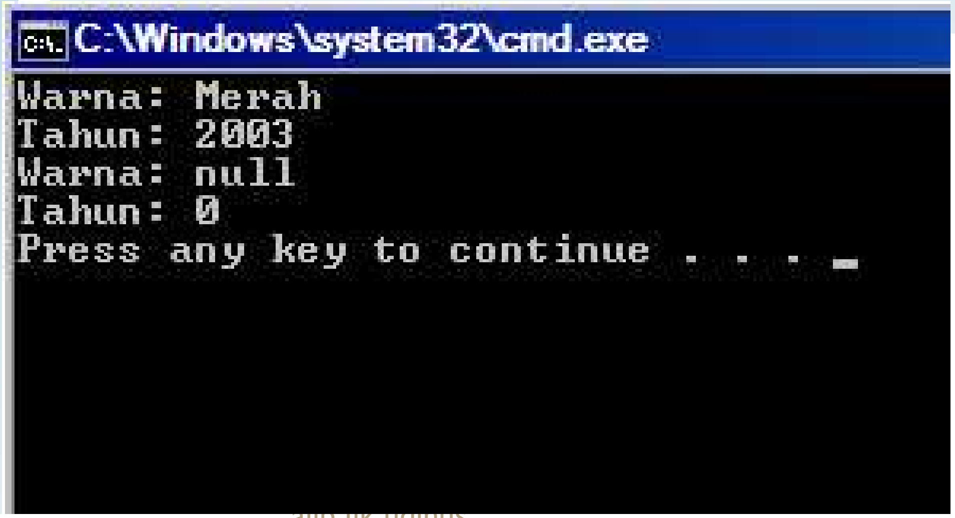
- Penggunaan satu nama untuk beberapa method yang berbeda (beda parameter)
- *One name different parameter*
- Contoh :
- Class A{

```
void info(String title){
…
}
Void info(String title, int x){
…}
```

OverLoading

```
class Mobil {
    private String warna;
    private int tahunProduksi;

    public Mobil(String warna, int
    tahunProduksi){
        this.warna = warna;
        this.tahunProduksi = tahunProduksi;
    }

    public Mobil(){
    }

    public void info(){
        System.out.println("Warna: " +
    this.warna);
        System.out.println("Tahun: " +
    this.tahunProduksi);
    }
}
```

```
public class Konstruktor{
  public static void main(String[] args){
    Mobil mobilku = new Mobil("Merah",
    2003);
    mobilku.info();

    Mobil mobilmu = new Mobil();
    mobilmu.info();
    }
}
```


C:\Windows\system32\cmd.exe

```
Warna: Merah
Tahun: 2003
Warna: null
Tahun: 0
Press any key to continue . . . _
```

Ada pertanyaan

?

Rehat Sejenak

- Mc Donald Arab
- Cepat Langsing