

Pengenalan pemrograman berorientasi obyek

Topik

- MENGENAL OBJEK & CLASS
- Fitur OOP
- Deklarasi class
- Deklarasi Atribut
- Deklarasi metode
- Pengaksesan anggota obyek
- Life Cycle dari Objek
- Tipe Reference
- Pass by value



MENGENAL OBJEK & CLASS

- Paradigma Objek
 - Paradigma adalah suatu cara pandang atau cara berpikir
 - Paradigma objek adalah cara pandang yang memandang SEGALA SESUATU sebagai OBJEK
 - Semua aspek dalam Java programming dapat dianggap sebagai objek, -kecuali TIPE DATA PRIMITIF-, karena semua library dan objek dalam Java memiliki akar awal class `java.lang.Object`
 - Berbagai benda di sekitar kita adalah objek nyata yang dapat dilihat, seperti : kucing, meja, rumah, orang , dll



MENGENAL OBJEK & CLASS

- Persoalannya, bagaimana memindahkan pemikiran objek di dunia nyata menjadi objek di dunia software atau pemrograman, khususnya Java
- Ambil contoh objek nyata yang akan dipindahkan adalah objek orang



MENGENAL OBJEK & CLASS

- Data Member
 - Setiap objek yang dinamakan ‘orang’ pasti memiliki : nama, tinggi badan, berat badan, warna rambut, warna kulit, jenis kelamin, menggunakan kaca mata, dll
 - Ciri-ciri tersebut dapat dipindahkan menjadi variabel-variabel dari class yang sering disebut sebagai : data member

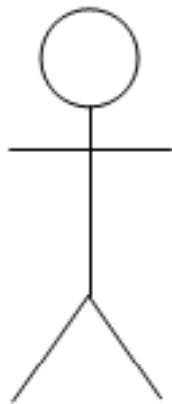


MENGENAL OBJEK & CLASS

- Contoh pemisalan objek orang nyata menjadi kode program dalam class Orang :

```
class Orang {  
    String nama;                //nama orang  
    int tinggiBadan;           //dalam cm  
    int beratBadan;            //dmlm kg  
    String warnaRambut;        //hitam, pirang, coklat  
    String warnaKulit; //sawoMatang, hitam, putih  
    String jenisKelamin;       //pria atau wanita  
    boolean berkacamata;       //bila berkacamata berarti true  
}
```

MENGENAL OBJEK & CLASS



```
class Orang  
  
nama;  
tinggiBadan;  
beratBadan;  
warnaRambut;  
warnaKulit;  
jenisKelamin;  
berkacamata;  
  
menangis()  
tertawa()
```

Memindahkan orang dari dunia nyata menjadi class Orang



MENGENAL OBJEK & CLASS

- Class dapat diumpamakan seperti spesifikasi atau blueprint. Dalam hal ini, Tuhan menciptakan manusia dengan spesifikasi tertentu.
- Jadi dapat diumpamakan bahwa Tuhan memiliki class Orang yang kemudian membuat banyak objek dari class Orang tsb, dan contoh objek tersebut adalah Anda sendiri.
- Objek dalam pemrograman adalah objek yang dibuat dari class tertentu.



MENGENAL OBJEK & CLASS

- Dari definisi class Orang di atas, kita bisa membuat objek-objek berdasar class tersebut.
- Objek-objek yang dibuat perlu disimpan dalam variabel yang akan menyimpan referensi/address dari objek yang dibuat.
- Proses pembuatan objek sering disebut sebagai **instansiasi class**, sedangkan objeknya disebut **sebagai instance dari class**



MENGENAL OBJEK & CLASS

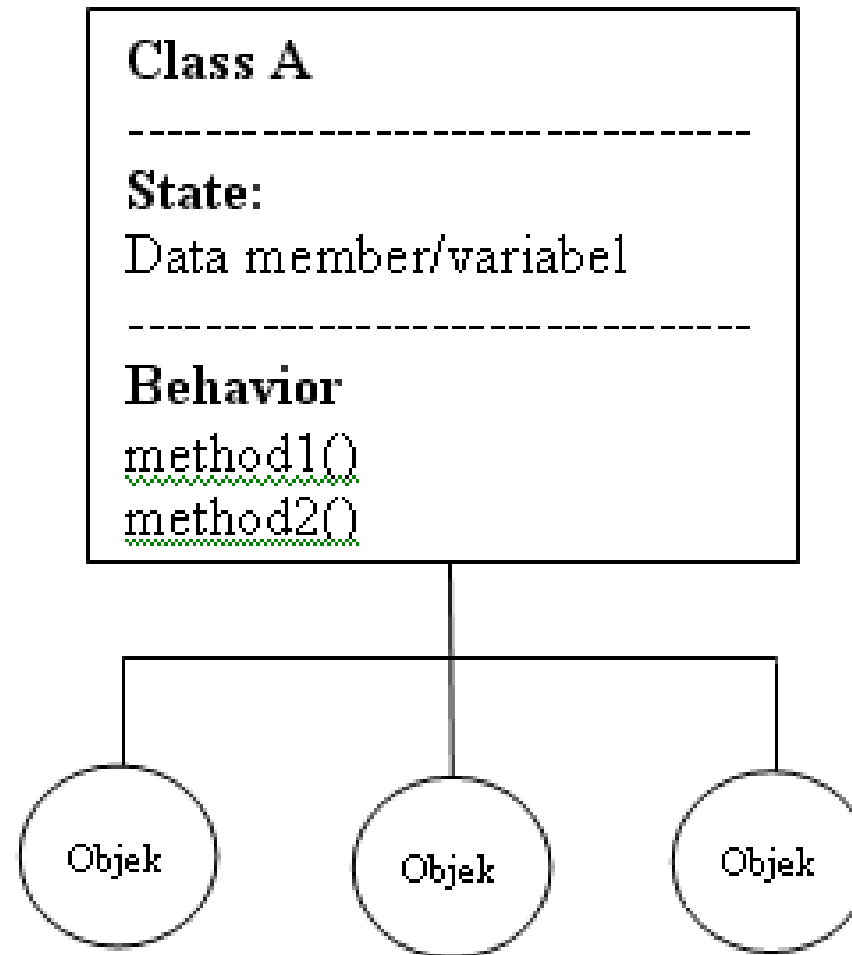
- Method
 - Selain memiliki atribut(STATE) yang diimplementasikan sebagai data member di atas, manusia juga dapat melakukan suatu aksi atau pekerjaan tertentu (BEHAVIOR)
 - Contoh aksi/behavior yang umum adalah menangis dan tertawa
 - Kedua behavior tsb bisa dipindahkan ke dalam bahasa pemrograman menjadi method sbb :

```
void menangis() {  
    System.out.println("hik..hikk..hik...");  
}  
  
void tertawa() {  
    System.out.println("ha..ha..ha..ha..");  
}
```



MENGENAL OBJEK & CLASS

- Method merupakan perwujudan aksi atau tindakan dari dunia nyata di dalam pemrograman komputer.
- Method dalam dunia pemrograman juga “pasti melakukan sesuatu aksi”, misalnya menampilkan String di konsol

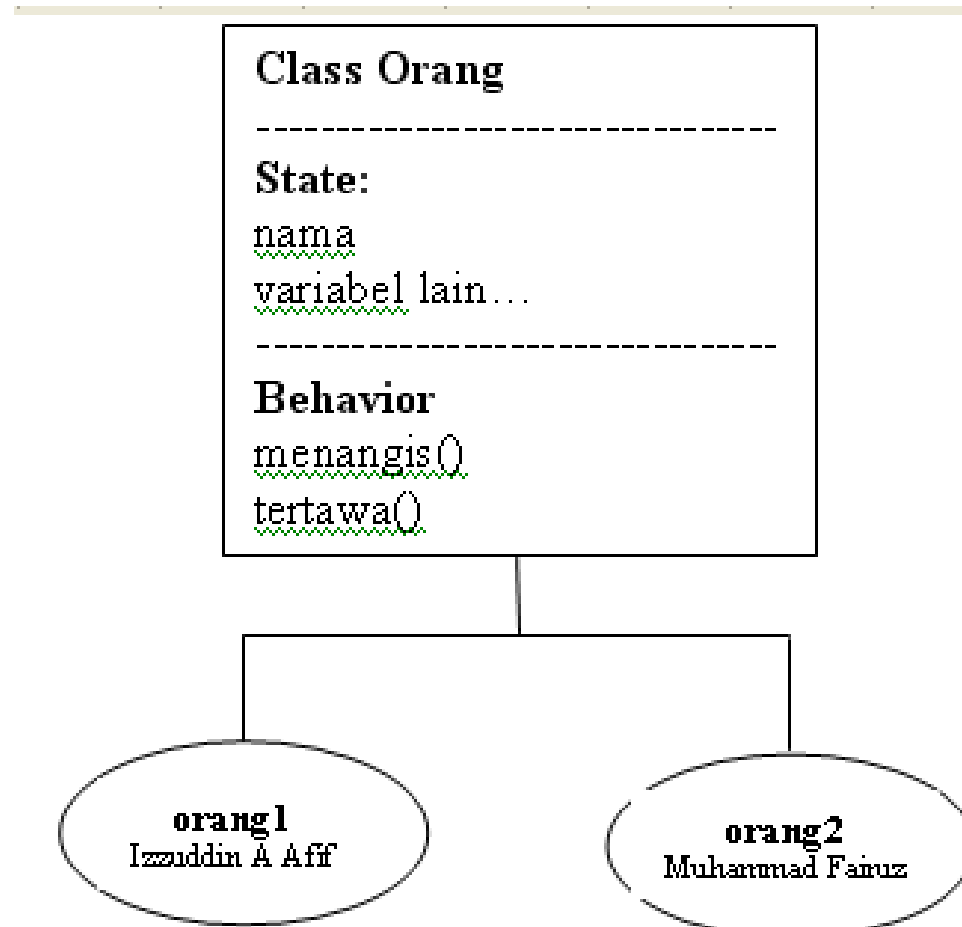


Ilustrasi perbedaan antara class dan objek



MENGENAL OBJEK & CLASS

- Dari gambar di atas dapat dipahami bahwa suatu class dapat memiliki banyak objek, dan setiap objek akan mewarisi data member dan method yang sama dari class
- Untuk membuat objek Orang dari class Orang, gunakan keyword **new** sbb :
Orang orang1 = new Orang("Izzuddin A Afif");
Orang orang2 = new Orang("Muhammad Fairuz");
- setiap objek dapat memiliki state atau nilai data member yang berbeda (hanya nama dan tipe variabel yang sama)



Ilustrasi pembuatan objek dari class

Fitur OOP

- Encapsulation
- Inheritance
- Polymorphism

Fitur OOP

- **Enkapsulasi** → suatu cara untuk menyembunyikan implementasi detail dari suatu class dalam rangka menghindari akses yang ilegal
- **Inheritansi** → dimana suatu entitas/obyek dapat mempunyai entitas/obyek turunan. Dengan konsep inheritance, sebuah class dapat mempunyai class turunan
- **Polymorphism** → kemampuan untuk merepresentasikan 2 bentuk yang berbeda

Deklarasi class

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

Contoh

```
public class Siswa {
```

```
}
```

modifier


nama class

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>]  
}
```

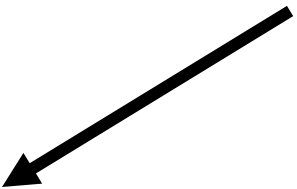
Type Reference

- Tipe selain tipe primitif dinamakan tipe reference
- Tipe reference adalah tipe berbentuk suatu class

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

- Struktur untuk mengakses anggota obyek.

```
NamaObject>NamaVariabel  
NamaObject>NamaMethod(parameter-list)
```

```
Siswa siswa = new Siswa();  
siswa.nrp=10;  
Siswa.info();
```

Cara mengakses method
object

Cara mengakses
variabel object

Pengaksesan anggota obyek

```
1 public class TestSiswa{  
2     public static void main(String args[]){  
3         Siswa anak = new Siswa();  
4         anak.nama = "Andika" ;  
5         anak.nrp = 1 ;  
6         anak.info();  
7     }  
8 }
```

Cara mengakses
variabel object

Cara mengakses method
object

Output

1 Andika adalah siswa PENS

Contoh Class

```
public class Coin {
    public final int HEADS = 0;
    public final int TAILS = 1;
    private int face;
    public Coin () {
        flip();
    }
    public void flip (){
        face = (int) (Math.random() * 2);
    }
    public int getFace (){
        return face;
    }
    public String toString(){
        String faceName;
        if (face == HEADS)
            faceName = "Heads";
        else
            faceName = "Tails";
        return faceName;
    }
}
```

Contoh Class

```
public class Circle {  
  
    public double x, y; // centre of the circle  
    public double r; // radius of circle  
  
    //Methods to return circumference and area  
    public double circumference() {  
        return 2*3.14*r;  
    }  
    public double area() {  
        return 3.14 * r * r;  
    }  
}
```

Using Circle Class

```
// Circle.java: Contains both Circle class and its user class
//Add Circle class code here
class MyMain
{
    public static void main(String args[])
    {
        Circle aCircle; // creating reference
        aCircle = new Circle(); // creating object
        aCircle.x = 10; // assigning value to data field
        aCircle.y = 20;
        aCircle.r = 5;
        double area = aCircle.area(); // invoking method
        double circumf = aCircle.circumference();
        System.out.println("Radius="+aCircle.r+" Area="+area);
        System.out.println("Radius="+aCircle.r+" Circumference =" +circumf);
    }
}
```

```
[raj@mundroo]?: java MyMain
Radius=5.0 Area=78.5
Radius=5.0 Circumference =31.400000000000002
```

Executing Methods in Object/Circle

- Using Object Methods:

sent 'message' to aCircle

```
Circle aCircle = new Circle();  
  
double area;  
aCircle.r = 1.0;  
area = aCircle.area();
```

Life Cycle dari Objek

- Creation (Membuat objek)
- Use (Menggunakan objek)
- Destruction (Menghapus objek)

Contoh

```
public class MyDate {  
    private int day=1;  
    private int month=1;  
    private int year=2000;  
  
    //konstruktor  
    public MyDate(int day, int month, int year)  
    {...}  
}
```

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

Declaring Objek (Membuat Objek)

- **MyDate today** = new MyDate(10, 11, 2006);
- Pernyataan diatas terdiri dari 3 langkah :
 - Deklarasi objek → MyDate today ;
 - Alokasi Memori → menggunakan kata kunci **new** MyDate(10, 11, 2006);
 - Inisialisasi Objek → tergantung dari konstruktornya

Membuat Objek

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

today

????

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

today

????

day

0

month

0

year

0

**Alokasi Memori
objek today dengan
tipe class MyDate**

**Mengisi atribut
dengan default value**

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

today

????

day

1

month

1

year

2000

Mengisi atribut
dengan nilai
inisialisasi eksplisit

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

today

????

**Menjalankan
konstruktor**

day

10

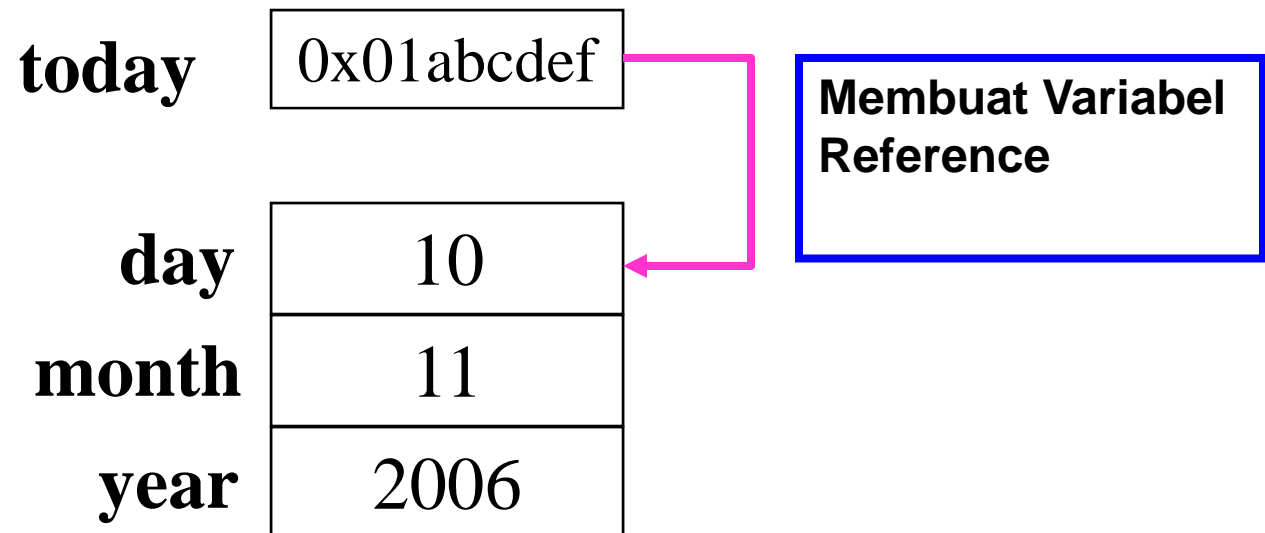
month

11

year

2006

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

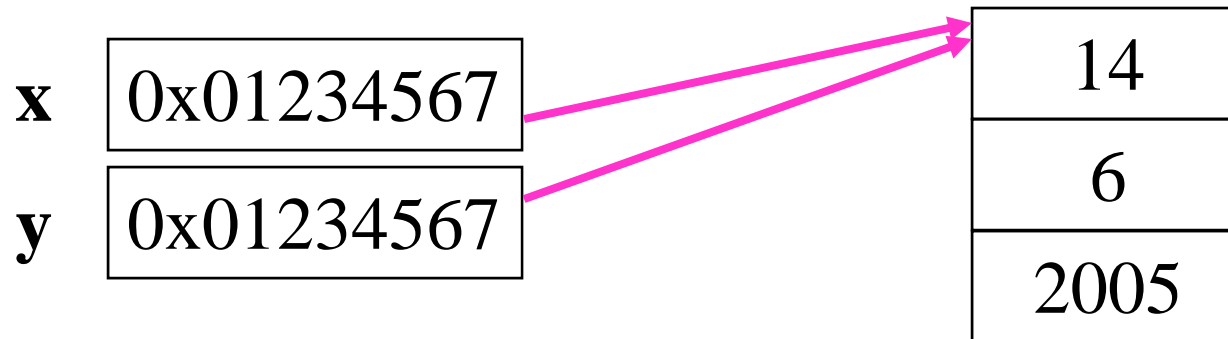




Men-assign reference variable

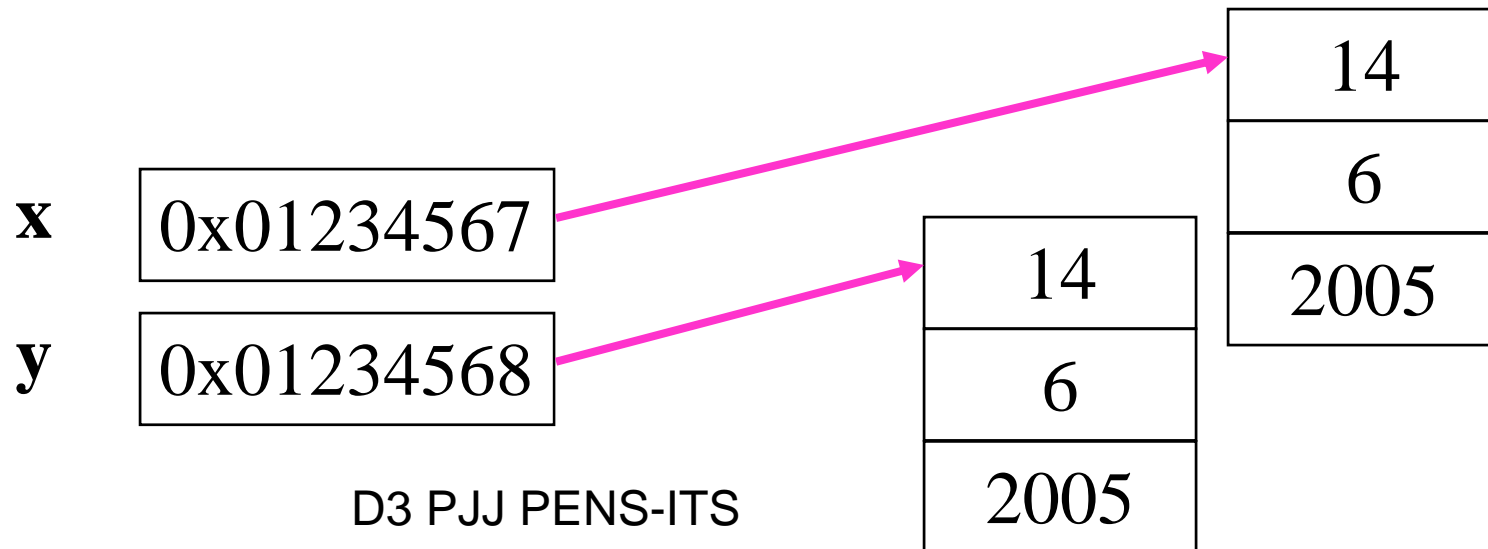
```
MyDate x = new MyDate(14, 6, 2005);
MyDate y = x;
```

Variabel Reference x dan y



Men-assign reference variable

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



Menggunakan Objek

- Ada 2 cara :
 - Memanipulasi variabelnya
 - Menggunakan metode dari objek tersebut

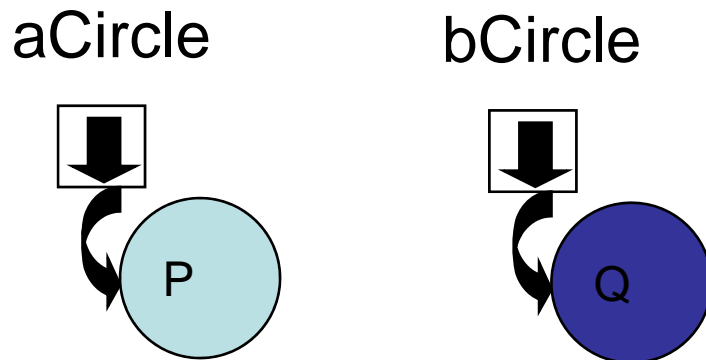
Membuat object dari sebuah Class

```
aCircle = new Circle();  
bCircle = new Circle();
```

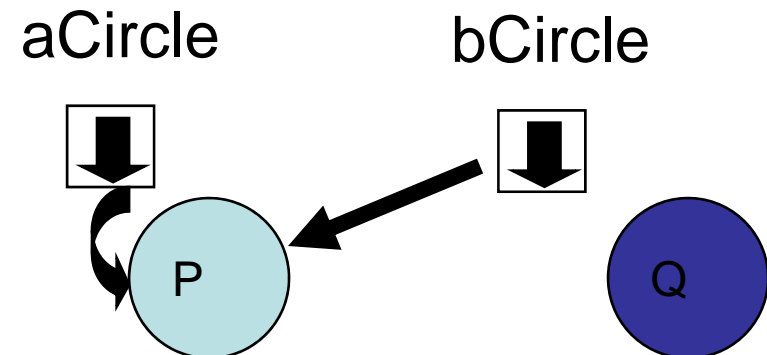
Buatlah object dari
Class Circle

```
bCircle = aCircle;
```

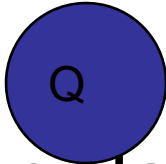
Sebelum Assignment



Setelah Assignment



Automatic garbage collection

- Object  sudah tidak mempunyai reference dan tidak bisa digunakan lagi.
- Maka object tersebut menjadi kandidat dari automatic **garbage collection**.
- Java secara otomatis mengumpulkan garbage secara periodik dan membersihkan memori yang sudah dipakai, supaya bisa digunakan lagi untuk selanjutnya

Pass by value

- Java tidak membolehkan adanya pass by reference, jadi hanya mengizinkan 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

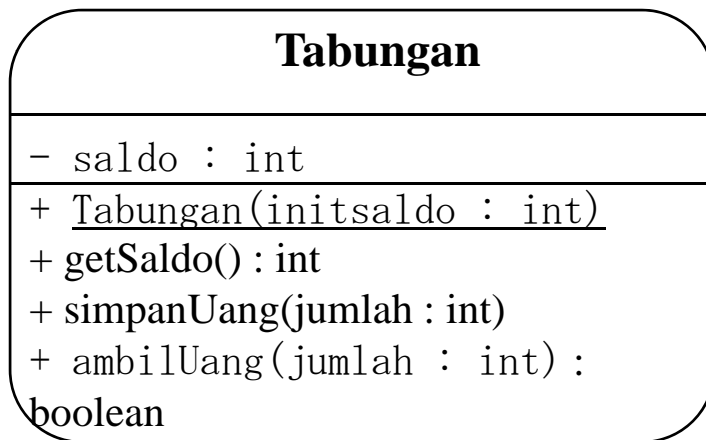
```
public static void main(String[] args)
```

- **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] ...`

Contoh Program

- Implementasikan UML class diagram dalam program untuk class Tabungan



Output

Jumlah uang yang disimpan : 8000
Jumlah uang yang diambil : 6000 true
Jumlah uang yang disimpan : 5500
Jumlah uang yang diambil : 4000 true
Jumlah uang yang diambil : 1600 false
Jumlah uang yang disimpan : 3500
Saldo : 3500


```
1 public class Tabungan{
2     private int saldo ;
3     public Tabungan(int initsaldo){
4         saldo = initsaldo;
5     }
6     public void simpanUang(int jumlah){
7         saldo = saldo + jumlah ;
8     }
9     public boolean ambilUang(int jumlah){
10        if (jumlah > saldo)
11            return false ;
12        else
13            saldo = saldo - jumlah ;
14        return true;
15    }
16    public int getSaldo(){
17        return saldo;
18    }
19 }
```

```
1 public class TestTabungan{
2     public static void main(String args[]){
3         Tabungan t1 = new Tabungan(5000);
4         t1.simpanUang(3000);
5         System.out.println("Jumlah uang yang disimpan : " + t1.getSaldo());
6         System.out.println("Jumlah uang yang diambil : 6000 " + t1.ambilUang(6000));
7         t1.simpanUang(3500);
8         System.out.println("Jumlah uang yang disimpan : " + t1.getSaldo());
9         System.out.println("Jumlah uang yang diambil : 4000 " + t1.ambilUang(4000));
10        System.out.println("Jumlah uang yang diambil : 1600 " + t1.ambilUang(1600));
11
12        t1.simpanUang(2000);
13        System.out.println("Jumlah uang yang disimpan : " + t1.getSaldo());
14        System.out.println("Saldo : " + t1.getSaldo());
15    }
16 }
```

Class Customer

```
1 public class Customer {
2     private String firstName;
3     private String lastName;
4     private int age;
5
6     public String getFirstName() {
7         return firstName;
8     }
9     public void setFirstName(String firstName) {
10        this.firstName = firstName;
11    }
12    public String getLastName() {
13        return lastName;
14    }
15    public void setLastName(String lastName) {
16        this.lastName = lastName;
17    }
18    public int getAge() {
19        return age;
20    }
21    public void setAge(int age) {
22        this.age = age;
23    }
24 }
```

```
1 public class TestCustomer {
2     public static void main(String args[]){
3         Customer[] customers = new Customer[4];
4         customers[0] = new Customer();
5         customers[0].setFirstName("Yuliana");
6         customers[0].setLastName("Setiowati");
7         customers[0].setAge(29);
8
9         customers[1] = new Customer();
10        customers[1].setFirstName("Stanley");
11        customers[1].setLastName("Clark");
12        customers[1].setAge(8);
13
14        customers[2] = new Customer();
15        customers[2].setFirstName("Jane");
16        customers[2].setLastName("Graff");
17        customers[2].setAge(16);
18    }
```



```
19 customers[3] = new Customer();
20 customers[3].setFirstName("Nancy");
21 customers[3].setLastName("Goodyear");
22 customers[3].setAge(69);
23
24 for (int i=0; i<4; i++) {
25     String lastName = customers[i].getLastName();
26     String firstName = customers[i].getFirstName();
27     int age = customers[i].getAge();
28     System.out.println(firstName + ", " + lastName + "    Age:" + age);
29 }
30 }
31 }
```

Output

```
Yuliana, Setiowati Age:29
Stanley, Clark Age:8
Jane, Graff Age:16
Nancy, Goodyear Age:69
```

Class Customers

```
1 public class Customer2 {
2     private Tabungan tabungan ;
3     private String firstName;
4     private String lastName;
5     private int age;
6
7     public Customer2(String f, String l, int a) {
8         firstName = f;
9         lastName = l;
10        age = a ;
11    }
12    public String getFirstName() {
13        return firstName;
14    }
15    public String getLastName() {
16        return lastName;
17    }
18    public int getAge() {
19        return age;
20    }
21    public Tabungan getTabungan() {
22        return tabungan;
23    }
24    public void setTabungan(Tabungan t) {
25        tabungan = t ;
26    }
27 }
```

Output

Yuliana Setiowati Age:29
Stanley Clark Age:8
Jane Graff Age:16
Nancy Goodyear Age:69
12000

Class TestCustomer2

```
1 public class TestCustomer2{
2     public static void main(String args[]){
3         Customer2 customers[] = new Customer2[20];
4         customers[0] = new Customer2("Yuliana","Setiowati",29);
5         customers[1] = new Customer2("Stanley","Clark",8);
6         customers[2] = new Customer2("Jane","Graff", 16);
7         customers[3] = new Customer2("Nancy","Goodyear",69);
8
9         for (int i=0; i<4; i++) {
10            String lastName = customers[i].getLastName();
11            String firstName = customers[i].getFirstName();
12            int age = customers[i].getAge();
13            System.out.println(firstName + " " + lastName + "   Age:" + age);
14        }
15        //customers[0]
16        customers[0].setTabungan(new Tabungan(5000));
17        customers[0].getTabungan().ambilUang(3000);
18        customers[0].getTabungan().simpanUang(10000);
19        System.out.println(customers[0].getTabungan().getSaldo());
20    }
21 }
```

Data Member

- Disebut juga variabel atau atribut
- Variabel dibagi menjadi dua :
 - Variabel instance : variabel yang dimiliki oleh setiap objek. Masing-masing objek mempunyai nilai variabel instance yang berbeda
 - Variabel class : variabel yang dimiliki oleh class. Semua objek dari class tersebut akan mempunyai nilai yang sama. Ciri dari variabel class dengan menambahkan kata kunci static contoh `private static double bunga`

Contoh Program

- Class `Tabungan2` terdiri dari dua variabel:
 - Variabel instance : `saldo`
 - Variabel class : `bunga` (kata kunci `static`)
- Method `public String toString()` →
mengubah objek menjadi `String`



```

public class Tabungan2 {
    private int saldo ;
    private static double bunga ;

    public Tabungan2(int saldo, double bunga){
        this.saldo = saldo;
        this.bunga = bunga ;
    }
    public void simpanUang(int jumlah){
        saldo = saldo + jumlah ;
    }
    public boolean ambilUang(int jumlah){
        if (jumlah > saldo)
            return false ;
        else
            saldo = saldo - jumlah ;
        return true;
    }
    public int getSaldo(){
        return saldo;
    }

    public String toString(){
        return saldo+" " + bunga ;
    }
}

```



```

public class Test {
    public static void main(String args[]) {
        Tabungan2 t1 = new Tabungan2(100000,0.01) ;
        System.out.println("T1");
        System.out.println(t1.toString());

        Tabungan2 t2 = new Tabungan2(200000,0.02) ;

        System.out.println("T1");
        System.out.println(t1.toString());

        System.out.println("T2");
        System.out.println(t2.toString());

    }
}

```