PEMOGRAMAN JAVA

Yoannita, S.Kom

- **Class(ii) [Constructor]**
  [keyword this]
- **Modifier**
- **Passing parameter**
  [by value]
  [by references]

Class

Dalam pendefinisian *class*, dituliskan :

```
<modifier> class <name> {
    <attributeDeclaration>*
    <constructorDeclaration>*
    <methodDeclaration>*
}
```

dimana :

<modifier> adalah sebuah *access modifier*, yang dapat dikombinasikan dengan tipe *modifier* lain.

Constructor

- ***Constructor*** merupakan method khusus yang berfungsi untuk inisialisi atau menciptakan suatu object dari class

  ➔Constructor tidak mengembalikan nilai

  ➔Constructor mempunyai nama sama dengan nama class.

- Jika constructor tidak didefinisikan, Java memberikan constructor dengan nama constructor_default.
- Constructor default tidak melakukan apa-apa, namun semua variabel yang tidak diinisialisasi dianggap sebagai berikut :
  - ✓Variabel numerik diset ke 0
  - ✓String diset ke null
  - ✓Variabel bolean diset ke false

  ➔Constructor tidak memiliki tipe hasil, walaupun constructor bisa public, private atau protected. Sebagian constructor bersifat public

Keyword this

- Keyword this digunakan untuk menyatakan bahwa yang sedang diakses adalah variabel instans dari class dan bukan parameter suatu method.

```
class ordinat {
   int x;
   int y;
   void inisial(int x, int y){
              this.x = x;
              this.y = y;
    }
}

class dmThis{
 public static void main(String arg[]){
    ordinat t1;
    ordinat t2;
    t1 = new ordinat();
    t2 = new ordinat();
    t1.inisial(12,5);
    t2.inisial(20,15);
    System.out.println("nilai x t1 = " + t1.x);
    System.out.println("nilai x t2 = " + t2.x);
   }
}
```

Keyword this

```
class Ordinat {
        int x;
        int y;

        void Inisial(int x, int y){

            this.x = x;
            this.y = y;

         }
}
```

Keyword this

Kata kunci *this* dapat digunakan untuk beberapa alasan berikut:

- 1. Adanya ambigu pada atribut lokal dari variabel lokal
- 2. Menunjuk pada object yang meminta method non-static
- 3. Menunjuk pada constructor lain.

Keyword this (i)

- Sebagai contoh pada maksud pertama, perhatikan kode berikut dimana variabel *data* disediakan sebagai sebuah atribut dan parameter lokal pada saat yang sama.

```
class ThisDemo1 {
    int data;
    void method(int data) {
            this.data = data;
            /* this.data menunjuk ke atribut
            sementara data menunjuk ke variabel lokal */
    }
}
```

Keyword this (ii)

- Contoh berikut menunjukkan bagaimana object *this* secara mutlak menunjuk ketika anggota non static dipanggil.

```
class ThisDemo2 {
    int data;
    void method() {
        System.out.println(data);    //this.data
    }
    void method2() {
        method();                    //this.method();
}
```

Keyword this (iii)

- Contoh selanjutnya memiliki constructor overloaded dan referensi *this* yang dapat digunakan untuk menunjuk versi lain dari constructor.

```
class ThisDemo3 {
    int data;

    ThisDemo3() {
      this(100);
     }

    ThisDemo3(int data) {
      this.data = data;
     }
}
```

Modifier

- Access Modifier
  - default
  - public
  - private
  - protected
- Static Modifier
- Final Modifier
- Abstract Modifier
- Synchronized Modifier
- Native Modifier
- Default

Access Modifier

- **default**
  - Hanya kelas dalam paket dapat mengakses variabel dan metode dari kelas.
  - Dapat diakses oleh subclass asalkan berada dalam package yang sama
- **public**
  - Variabel dan metode dapat diakses dari dalam maupun luar kelas
- **private**
  - Atribut kelas hanya dapat diakses oleh metode dalam kelas dimana didefinisikan
  - tidak ada kode satupun dari luar class tersebut yang diizinkan mengakses / mengubah nilai dari member tersebut
- **protected**
  - Atribut kelas hanya dapat diakses oleh kelas dan subkelas dari kelas tersebut

Access Modifier

	private	default/package	***protected***	***public***
class yang sama	Yes	Yes	Yes	Yes
package yang sama		Yes	Yes	Yes
package yang berbeda (subclass)			Yes	Yes
package yang berbeda (non-subclass)				Yes

Static Modifier

- Memberikan spesifikasi bahwa atribut atau metode sama untuk semua objek dari kelas tertentu.
- Keuntungan dari modifier static adalah dapat diakses tanpa membuat objek dalam class.

Final Modifier

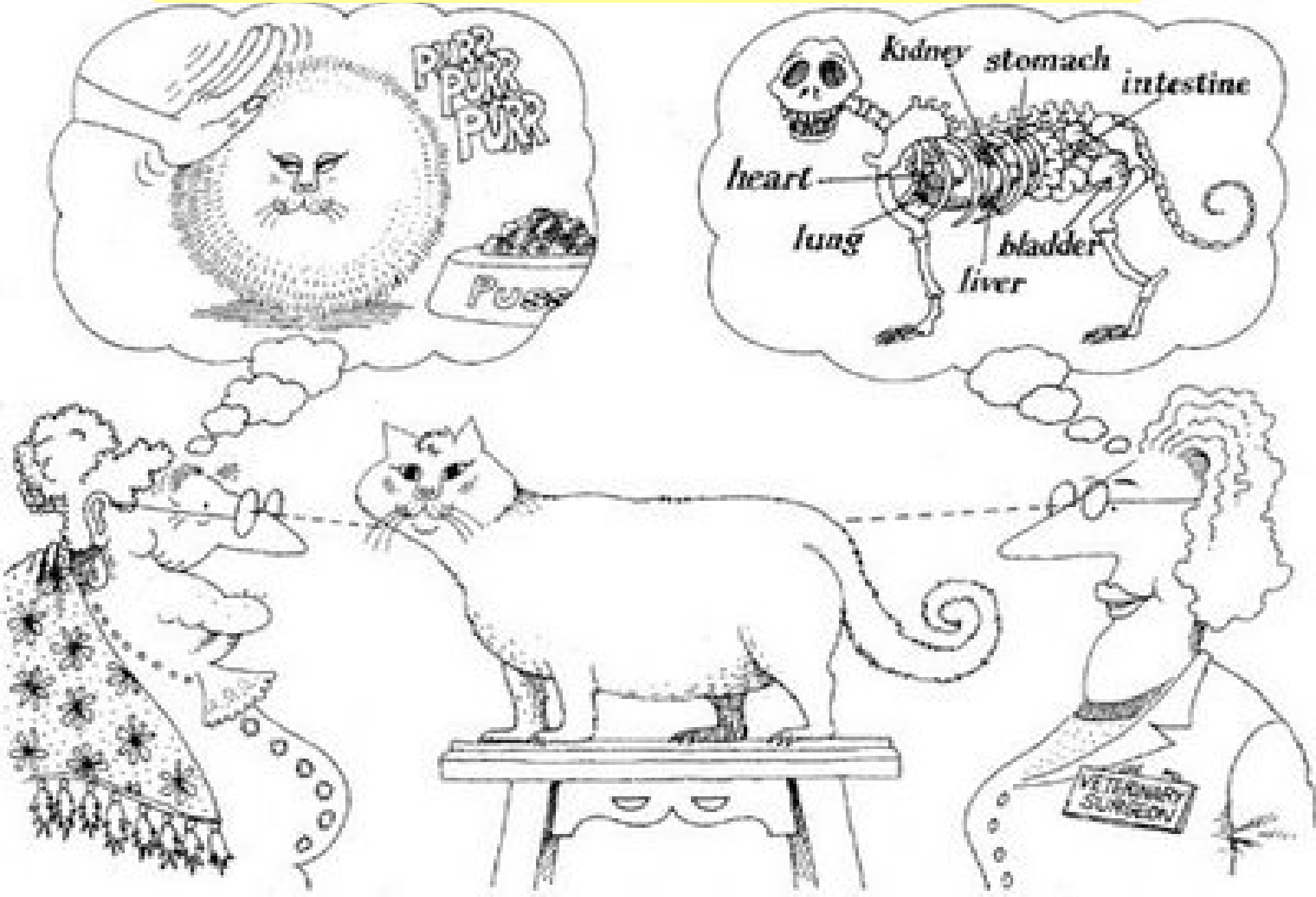
- Memberikan spesifikasi bahwa variabel mempunyai nilai konstan sama atau metode yang tidak dapat di–*overriden* dalam subkelas.

- *final int persenDiskon = 0.1;*

Abstract Modifier

- Memberikan spesifikasi bahwa metode tidak dapat dijalankan, dan harus digunakan subkelas yang tidak abstrak

Abstraction focuses on the essential characteristics of some object, relative to the perspective



Synchronized Modifier

- Memberikan spesifikasi bahwa metode thread safe, yang berarti eksekusi diperbolehkan pada synchronized method

Native Modifier

- Mengidentifikasi metode yang mempunyai implementasi dasar
- Modifier native memberikan informasi java kompiler bahwa implementasi metode berada di luar
- Contoh : native int nilaiTotal();

Passing Parameter

- Passing parameter by Value

  ➔ pengiriman / passing parameter by value **merupakan pengiriman searah**.

  ➔ Karena data yang diproses dalam method **tidak akan dikirim kembali** ke pemanggil method tersebut.

- Passing Parameter By Reference

  ➔ artinya perubahan nilai dalam method yang dipanggil akan mempengaruhi nilai parameter pemanggil

Passing parameter by Value

```
class pass1 {
  void hitung(int p){
      p = p * 2;
  }
}


class byValue{
  public static void main(String arg[]){
     pass1 x1;
     int r = 5;

     x1 = new pass1();
       System.out.println("sebelum pemanggilan hitung r = " + r);

     x1.hitung(r);
       System.out.println("setelah pemanggilan hitung r = " + r);
  }
}
```

Hasil:
sebelum pemanggilan hitung r = 5
setelah pemanggilan hitung r = 5

Nilai r tidak berubah.
Pada saat method hitung dipanggil *(x1.hitung(r);)* dengan **membawa nilai r**, dan didalam method hitung nilai r diterima oleh parameter p. dan **p di proses** sehingga p bernilai 10.
Namun nilai p ini **tidak dikembalikan** ke parameter pemaggil (r), sehingga nilai r tetap.

Passing Parameter By Reference

```
class pass2 {
   int angka;
   void setNilai(int param1) {
      angka = param1;
    }
    void hitung(pass2 p) {
      p.angka = p.angka * 2;
    }
}

class byRef{
   public static void main(String arg[]) {
      pass2 y;
      y = new pass2();

      y.setNilai(8);
      System.out.println("sebelum pemanggilan hitung = " +  y.angka);

      y.hitung(y);
      System.out.println("setelah pemanggilan hitung = " +  y.angka);
   }
}
```

Hasil :
sebelum pemanggilan hitung = 8
setelah pemanggilan hitung = 16

Method ini memiliki parameter p bertipe pass2.
Pada saat memanggil method ini yang dikirim adalah objek y, sehingga *objek y dan objek p merujuk pada satu lokasi yang sama* dan apa pun yang terjadi pada lokasi tersebut akan mempengaruhi semua objek yang merujuk ke lokasi tersebut.