Modul 2:
Class, Objek, data, variabel dan operator

Setelah mengikuti mata kuliah ini mahasiswa dapat mendefinisikan class java, menggunakan modifier, dan membuat program yang menggunakan data dan variable

Pengantar :

Dalam modul ini akan diuraikan beberapa topik bahasan yaitu :

1. Pengertian class dan objek :
2. Modifier : public, static
3. Data primitiveJAVA, operator dan Konsep encaptulation
4. String sebagai sebuah class

Contoh definisi class dan objek kasus : class Dog

- **Class dan Objek merupakan paradigma dasar dari OOP (Object Oriented Programming)**
- **Class** : suatu frame yang merupakan definisi yang memuat data dan metod pengolah data
- **Objek** : Bentuk konkrit dari class yang dengannya data dan metod dapat direalisasikan
- Contoh class : **Dog**
- Contoh data : warna, umur , harga

  definisi data menggunakan sintak : tipe_data nama_var;

  atau : tipe_data nama_var=nilai_data;
- Contoh metod : bark(), sleep()

  definisi metod :

  [public/static] return_value nama_metod(parameter){ … }

  { isi metod}

Definisi class Dog

```
class Dog
{private String nama=“Noname”;
 private int umur =0;
 public Dog(String nm, int um){nama=nm; umur=um;}
 public void bark()
    {System.out.println(“Gug..gug..!”);}
 public void sayHello()
    {System.out.println(“Hello my name is “+nama+”
  I’m a “+umur+” years old”);
      }
}
```

Membuat instant (objek)

Membuat objek pada dasarnya sama dengan mendefinisikan data dan mengisi nilai datanya.

Contoh :

Mendifinisikan data :

```
String nama;
```

Mendefinisikan data sekaligus mengisi data :

```
String nama="Heli";
```

Mendefinisikan objek : (nama objek adalah myDog, merupakan objek dari class Dog)

```
Dog myDog;
```

Mendefinisikan objek sekalligus menginisialisasi nilai objek.

```
Dog myDog = new Dog("Heli", 3);
```

Parameter nama

Parameter umur

Memanggil metod dari objek

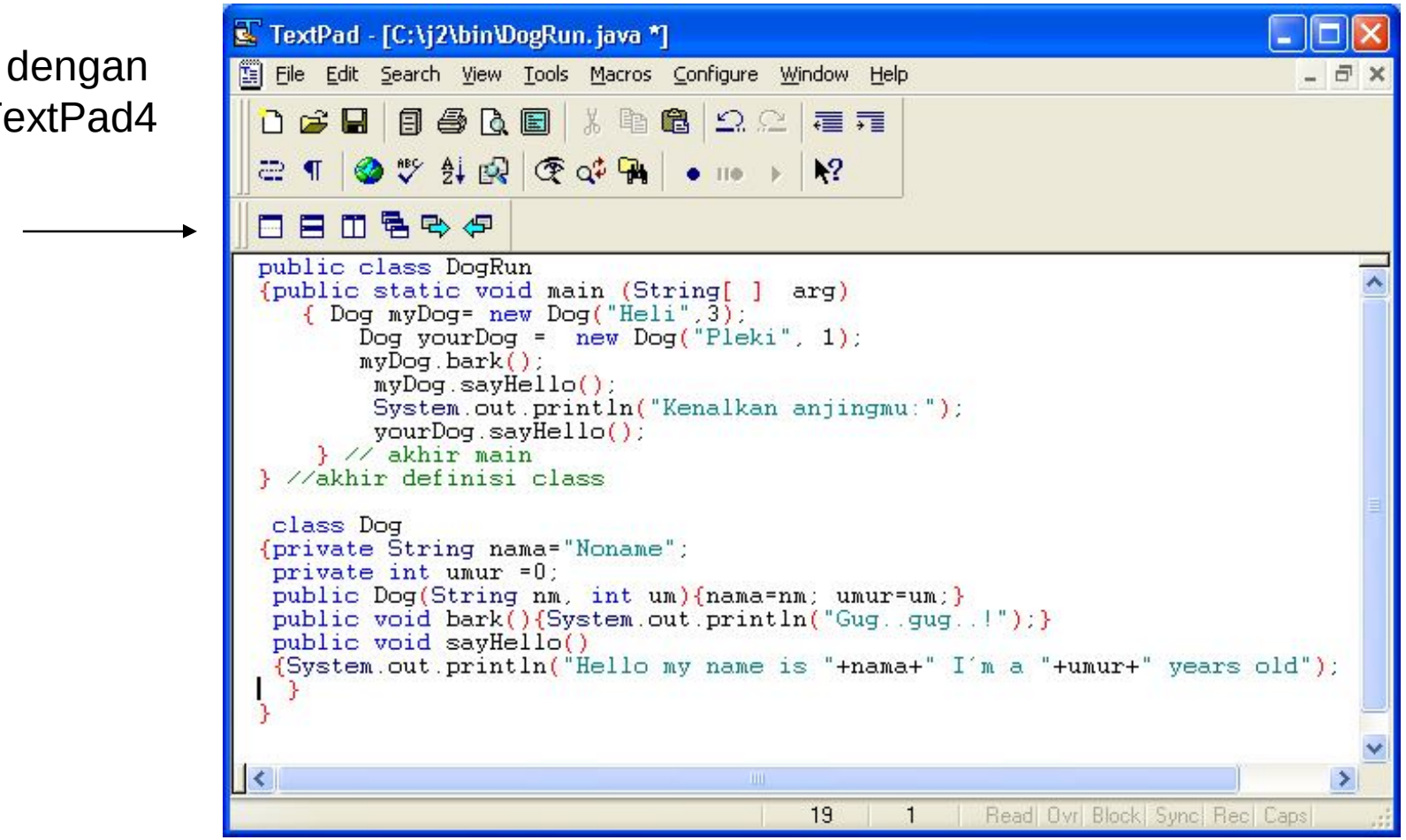
- Berikut ini class yang memuat program utama dan memanggil metod bark() dan seyHello().
- Rumus memanggil metod adalah :

```
nama_class.nama_metod()
```

```
public class DogRun
{public static void main (String[ ]  arg)
   { Dog myDog= new Dog(“Heli”,3);
      Dog yourDog =  new Dog(“Pleki”, 1);
       myDog.bark();
      myDog.sayHello();
      System.out.println(“Kenalkan anjingmu:”);
      yourDog.sayHello();
    } // akhir main
} //akhir definisi class
```

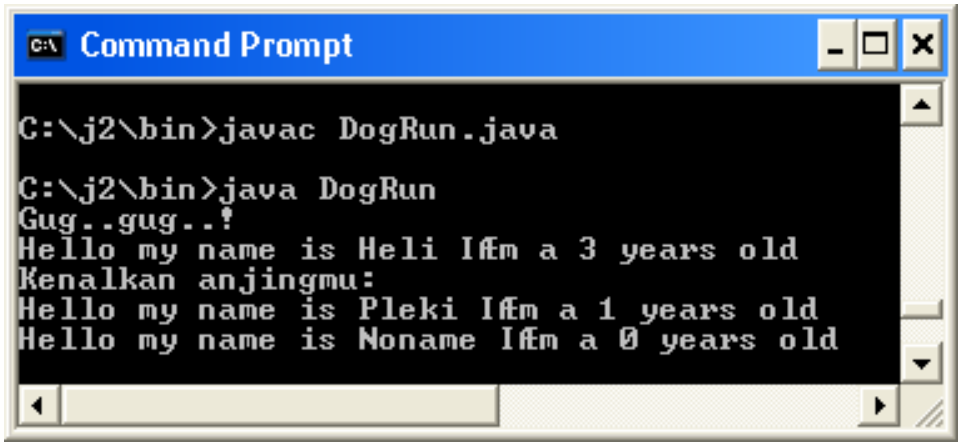
Edit program dengan TextPad dan kompilasi dengan javac.exe, eksekusi dengan java.exe

Edit dengan TextPad4



Selanjutnya

- Setelah selesai edit disimpan dalam file :DogRUn.java
- Saat dikompilasi dengan :
- >javac DogRun.java
- Akan terbentuk dua file class, yiatu :DogRun.class dan Dog.class
- Saat dieksekusi dengan :
- >java DogRun, hasilnya sbb:



2. Modifier public , void, static, private

- Modifier **public** :
- Pada definisi **class**, merupakan class yang memuat metod main(), dan nama file disimpan dengan nama class public
- Pada definisi **metod**, bermakna bahwa metod tersebut dapat diases dari luar class yang bersangkutan
- Pada daefinisi **data**, bermakna data tersebut dapat diases dari luar class
- Modifier **void** pada metod: berarti Metod tersebut tidak ada return valuenya
- Modifier **static** pada metod : berarti metod tersebut dapat dipanggil hanya dari dalam class yang sama
- Modifier **private** pada data : berarti data hanya dapat diases dari class yang sama
- Modifier **protected** paad metod : dapat diases dari class turunannya

Construktur

- Construktor adalah jenis metod yang akan otomatis dipanggil jika suatu objek dibuat
- Construktor didefinisikan secara public dan namanya sama dengan nama classnya
- Construktor dapat didefinisikan lebih dari satu definisi
- Contoh : untuk Dog ditambah definisi construktor tanpa parameter, menjadi :

```
public Dog( ) { }
public Dog(String nm, int um){nama=nm; umur=um;}
```

Misalkan dalam class DogRun didefinisikan objek yourDog

```
Dog yourDog=new Dog();
```

Maka jika dalam class DogRun dipanggil :

```
yourDog.seyHello();
```

Outputnya : (nilai nama dan umur digunakan nilai default)

```
Hello my name is Noname I'am a 0 years old
```

3. Data primitive dan operator

Data primitive : Data-data dasar java, yang hampir sama dengan C++

Data Integer :

- **int** 4 byte kisaran nilai :-2.147.486.648 s/d 2.147.486.647
- **short** 2 byte kisaran nilai :-32.768 s/d 32.767
- **long** 8 byte kisaran nilai :-9.223.372.036.854.775.808L s/d 9.223.372.036.854.775.807L
- **byte** 1 byte kisaran nilai :-128 s/d 127

Data floating point :

- **float** 4 byte kisaran nilai :±3.40282347E+38F (7 digit ignifikan)
- **double** 8 byte kisaran nilai :±1.79769313486231570E+308 (15 digit signifikan)

karakter dan boolean

- **char** 1 byte 1 karakter
- **boolean** bernilai true, false

Operator

Operator aritmatik :

- +, - , * dan / untuk : penambahan, pengurangan, perkalian dan pembagian
- += , yakni : x+=4 ekivalen dengan operasi : x=x+4
- -= ; *= dan /= memiliki ekivalensi yang sama dengan -=
- ++ yaitu : n++ ekivalen dengan n=n+1 (*increment by one*)
- -- yaitu : n-- ekivalen dengan n=n-1 (*decrement by one*)
- % untuk MODULO, yaitu : 5 % 3 sama dengan 2

Operator relational dan logika

- < , > , <= dan >= adalah : lebih kecil, lebih besar, lebih kecil atau sama dan lebih besa atau sama
- == untuk SAMA DENGAN
- != untuk TIDAK SAMA DENGAN
- && untuk operator AND
- || untuk operator OR

Contoh program dengan data dan operator :

```
class OperasiDATA
{ public static void main (String args[])
  {int a;    float  b;  double c;   String d;
  a=(int)1/3;b=(float)1/3 ; c=(double)1/3 ;
  d="JON SHOLEH";
  System.out.println("a = " +a);
  System.out.println("b = " +b);
  System.out.println("c = " +c);
  System.out.println("d = " +d);
  System.out.println("d.length() = " +d.length());
  System.out.println("d.substring(0,3) = "
   +d.substring(0,3));
  }
}
```

Output program :

a = 0

b = 0.333333

c = 0.333333

d = JON SHOLEH

d.length() = 10

d.substring(0,3) =JON

Encaptulation

- **Encaptulation**: konsep dalam OOP yang bertujuan membungkus (mang-kapsul) data dan metode dalam suatu class sehingga data dan metode yang tersembunyi dalam class tidak dapat diases dari luar class.
- Encaptulation direalisir dengan modifier : **private**
- Data yang dapat diases dari luar diberi modifier dengan **public**

Contoh :

```
class Dataku
 { public int N_umum=0;
   private int N_khusus=0;
  public void Cetak1(){System.out.println("N_umum="+N_umum);}
   public void setN_Khusus(int n){N_khusus=n;}
   public void Cetak2()
  {System.out.println("N_khusus="+N_khusus);}
 }
```

Ases data public

Jika dimiliki program utama :

```
public class DatakuT
{public static void main(String[] arg)
 {Dataku Dt=new Dataku();
  Dt.N_umum=15;
  Dt.Cetak1();
}
}
```

maka hasilnya :

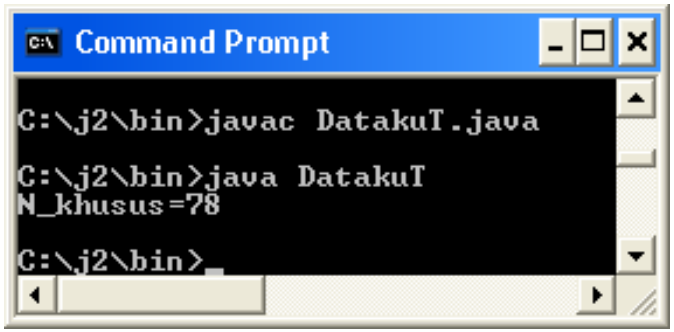
Jika di program utama : dituliskan Dt.N_khusus = 78; maka hasilnya :

Mengases data **private** : Terjadi error saat kompilasi

Mengases data private

- Jika suatu data dideklarasikan secara private biasanya disediakan mekanisme metode public untuk mengasesnya
- Metode ases public itu umumnya didefinisikan dengan **`set().`** Dalam contoh diatas jika nilai N_khusus akan diubah digunakan **`setN_khusus(n)`**
- **Contoh** :
- Program diubah menjadi : hasil program :

```
public class DatakuT
{public static void main(String[] arg)
 {Dataku Dt=new Dataku();
  Dt.setN_khusus(78);
  Dt.Cetak2();
}
}
```



4. String sebagai class

- String merupakan nama khusus class java yang masih mempertahankan kompatibilitas dengan tipe data primitif yang lama sebagaimana int, long, float dan lain-lain
- Definisi string dapat digunakan cara luwes, seperti deklarasi data primitif atau seperti deklarasi poembuatan objekl string.

Contoh :

Deklarasi seperti data primitive

```
int n=10;  String s="JOKO";
```

Deklarasi seperti membuat objek :

```
String s= new String("JOKO");
```

Hasil nya sama

Contructor dan Beberapa fungsi String

- String memiliki beberapa konstruktor :

  String(String s) : argumen string s

  String(byte [ ] b) : argumen berupa array of byte b

  String(char [ ] c) : argumen berupa array of char

- length() : mengetahui panjang string
- substring(n,m) : mengambil substring dari string sebanyak m mulai dari posisi n
- equals(s) : untuk mengetahui apakah suatu string s nilainya sama dengan string tersebut
- charAt(n) : mengambil satu data char dari string pada posisi n

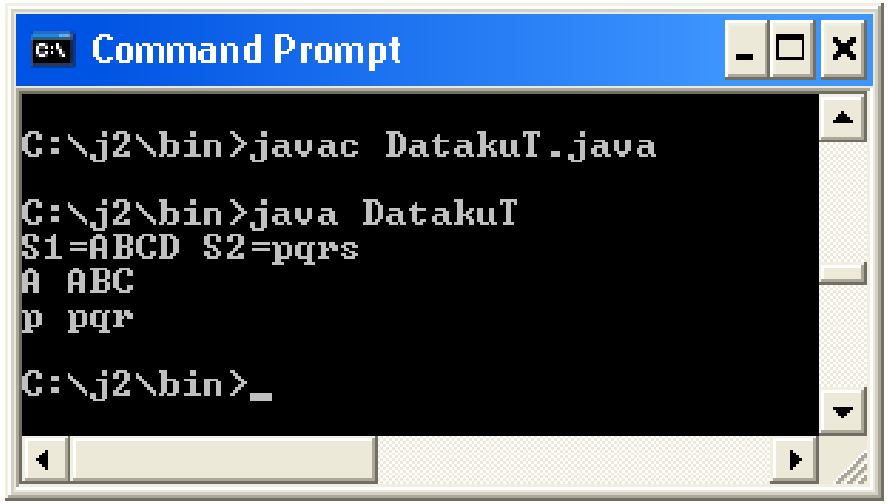
Contoh constructor dan pemanggilan metod String

```
public class ContohString
{public static void main(String[] arg)
 { byte [] b={65,66,67,68};
   char [] c={'p', 'q','r','s'};
   String S1=new String(b) ;
   String S2=new String(c) ;
   System.out.println("S1="+S1+" S2="+S2);
   System.out.println(S1.charAt(0)+" "+S1.substring(0,3));
   System.out.println(S2.charAt(0)+" "+S2.substring(0,3));
}
}
```

Contoh program:

b = array of byte

c = array of char



Hasil program

Rangkuman

1. Class adalah frame dari suatu data yang berisi definisi data, dan metod pengolah data
2. Objek adalah bentuk konkrit dari data dimana ia didefinisikan untuk mengisi data, mengaktifkan metod dan operasi data sebanarnya
3. Operasi data java hampir semua sama dengan operasi data dalam C++
4. Dalam definisi metod ada metod khusus yang otomatis dipanggil yaitu construktor
5. Konsep OOP yang disebut encaptulation memungkinkan programmer menyembunyikan data yang tidak diperlukan dengan mendefinisikan secara private
6. Data dan metod yang berkomunikasi keluar didefinisikan secara publik

Latihan

1. Buatlah definisi class Motor, yang memiliki data : nama (String) , status (boolean)

   memiliki metod : tampilkan() : mencetak keterangan nama keadaan motor (status mati /hidup)

   memiliki metod : nyalakan() :berfungsi men-set status dari mati (false) menjadi hidup (true). Jika status sudah hidup dan dipanggil metod nyalakan(), maka akan ada komentar “Mesin sudah hidup”, jika mesin belum hidup dan meetod nyalakan() dipanggil maka status diubah menjadi true.

2. Buatlah definisi class TesMotor , untuk membuat objek motor :

   motorku , nama motor HONDA kondisi mati, panggil metod tampilkan()

   motormu, nama motor YAMAHA , panggil metod nyalakan() dan tampilkan()