BUILT-IN CLASS

DEFINISI

Built in class adalah class-class yang secara langsung disediakan oleh Java user hanya perlu memakain tanpa perlu mendefinisikan sendiri

KEUNTUNGAN DAN KEKURANGAN

Keuntungan

- User tidak perlu repot-repot membuat dari awal class yang umumdigunakan..
- Bersifat Reusable.
- Mengefisiensikan waktu dalam pembuatan program.

Kekurangan

- Cangkupannya yang terbatas karena hanya menyediakan proses yang sering digunakan

Mengakses Built-in class

Ada beberapa langkah yang harus dilakukan untuk mengakses suatu *built-in class*.

1. Modul/Kamus yang menyediakan *class* tersebut harus di-**import** terlebih
2. Buat *instance* dari *class* tersebut
3. Tanda titik digunakan oleh Java sebagai pemisah antara *instance* dengan *method/atribut*

Contoh

```
import java.util.Scanner;

public class Main {
int masukan;

    public static void main(String[] args) {
        Main m = new Main();
        Scanner s =new Scanner(System.in);
        m.masukan = s.nextInt();
    }
```

Kamus diimportkan

Dibuat intance class

Pemisah memanfaatkan titik

```
s.nextInt();
```

nextInt()

next()

nextFloat()

Remote Scanner

s disini dimaslkan sebagai remote sedang tombol-tombol pada remote digunakan untuk mengakses (meninvoke) method dan atribut yang dimiliki sang remote Dalam hal ini object Scanner

Built in class tipe primitive (Wrapper)

1. NUMBERIC VALUE

Primitive	Java.lang class	Conversion to the Primitive Type	Returns
byte	Byte	byteValue()	byte
short	Short	shortValue()	short
int	Int	intValue()	int
long	Long	longValue()	long
float	Float	floatValue()	float
double	Double	doubleValue()	double

Built in class tipe primitive (Wrapper)

2. Character Values

   Untuk Kelas Karakter didefinisikan dalam package **java.lang**

   beberapa diantanya :

   – Class String
   – Class Character

Class String

Karena dalam java tidak disediakan variable tipe primitive yang langsung mengakses kata maka disediakanlah *class* String Pada java string direpresentasikan dengan dua kelas

Pseucode :

String <nama_variabel> = "deklarasi";

String merupakan Class

Namun demikian khusus untuk String,

instansiasi dapat dilakukan tanpa menyertakan

`new` dan *method constructor*-nya

```
Seharusnya seperti ini :
 String str = new String("ini contoh
string");
Cukup dituliskan  ->
 String str = “ini contoh string”;
```

Method string

- startWith()
- endWith()
- length()

Contoh

```
public class StringCoba {
 static String Makan="saya makan";
 static String Nasi="Nasi Kucing";
 static String nama="fadil";


    public static void main(String[] args) {

        StringCoba s = new StringCoba();
        System.out.println(Makan+" "+Nasi);
        System.out.println(nama.toUpperCase());
        System.out.println(nama.length());
```

Contoh (lanj)

```
String str = new String("ini contoh string");
        System.out.println(str.toUpperCase());
        boolean pembuktian =
str.startsWith("ini");
        boolean pembuktian2 =
str.endsWith("ini");
        System.out.println(pembuktian);
        System.out.println(pembuktian2);

    }

}
```

Class Math

Math adalah kelas yang terdapat paket java.lang yang berisi fungsi-fungsi matematika dan konstanta penting di matematika

METHOD MATH

```
y = Math.cos(pi/2);
y = Math.sqrt(16);
y = Math.pow(3,2);
y = Math.abs(-3);
```

Contoh

```
public class Main {
static void test(){
     double d1= 100;
     double d2 = Math.sqrt(d1);
     double d3 = Math.pow(2,3);
     System.out.println(" d2 = "+d2);
     System.out.println(" d3 = "+d3);
}
     public static void main(String[] args) {
          test();
     }
}
```

Class Scanner

Class Scanner adalah Class yang digunakan untuk menginputkan data dari keyboard melalui layar console

Method Scanner

1. nextInt();
2. nextShort();
3. nextByte();
4. next();
5. nextBoolean();
6. nextDouble()
7. nextFloat()
8. nextLong();
9. nextLine();
10. nextBigDesimal();

Contoh

```
public class Scan {

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("masukkan bilangan bulat");
        int bil = s.nextInt();

        System.out.println("Masukkan nama anda ");
        String nama = s.next();

        System.out.println("masukkan kondisi");
        boolean kondisi = s.nextBoolean();
```

Contoh (lanj)

```
System.out.println("masukkan bilangan desimal");
        float bill = s.nextFloat();

      System.out.println("tampilan masukan :");
        System.out.println(bil);
        System.out.println(nama);
        System.out.println(kondisi);
        System.out.println(bill);

    }
}
```

USER DEFINED CLASS

Pengertian

Merupakan konsep pembuatan Class pada pemograman berorientasi object yang mana dilakukan oleh user sendiri dari awal proses hingga akhir.

Mengapa *user-defined class*

Tipe data primitif dan bult in class tidak cukup untuk menangani pemograman *real word aplication* . Dalam dunia nyata, kita memiliki benda-benda yang jauh lebih rumit. Pemrograman berorientasi objek memungkinkan kita untuk memodelkan object yang ada pada duinia nyata

Keuntungan dan kerugian

Kentungan

1. Terjaminnya Validitas data.
2. Rincian informasi dapat disembunyikan.
3. *Class* dapat digunakan kembali.
4. *Class* dapat dibuat dan dimodifikasi sesuai dengan apa yang diinginkan oleh user

Kekurangan

1. Pembuatan Kelas Dilakukan Dari Awal.
2. Dibutuhkan pemahaman lebih tentangpengelompokan objek-objek yang ada di dunia nyata

Contoh User Define Class

```
public class Manusia{

    static String berdarah= "panas";
    String reproduksi = "melahirkan";

    public static String bernafas(){
        Scanner sc = new Scanner(System.in);
        System.out.println("masukkan jenis
  pernafasan");
        String nafas=sc.next();
        return nafas;
    }
```

Contoh User Define Class

```
public static void main(String[] args) {
        Manusia m = new Manusia();
        String darah = Manusia.berdarah;
        String nafas = Manusia.bernafas();
        System.out.println("manusia bereproduksi secara
  "+m.reproduksi);

        System.out.println("manusia berdarah "+darah);
        System.out.println("manusia bernafas dengan
  "+nafas);
    }

}
```

Implementasi dari kelas Manusia

Ini juga Implementasi dari kelas Manusia

Keyword Static

Keyword yang digunakan ketika kita ingin mengakses suatu komponen (atribut/method) pada suatu class tanpa harus melakukan instansiasi object dan pemanggilan tanpa harus menggunakan object reference nya

```
public class Keong {

    static String bernafas;
    static String golongan;
    static String berkembangbiak;
    String darah;

    public static String getGolongan() {
        return golongan;
    }
    public void setGolongan(String golongan) {
        this.golongan = golongan;
    }
```

```
public static void main(String[] args) {
    Keong k = new Keong();
    bernafas = "kulit";
    berkembangbiak="bertetelur";
    k.darah = "dingin";
    k.setGolongan("molusca dan avertebrata");
    System.out.println("keong termasuk golongan "+getGolongan());
    System.out.println("keong bernafas dengan "+bernafas);
    System.out.println("keong berkembang biak dengan jalan
  "+berkembangbiak);
    System.out.println("keong berdarah "+k.darah);
  }
}
```

Tanpa menggunakan object reference

Menggunakan object reference

Postest

1. Buatlah sebuah kelas dengan Nama Mahasiswa.
2. Setelah kelas Mahasiswa dibuat definisikan atributnya diantaranya nama, nim
3. definisikan atribut-atribut (variable-variabel) tadi dengan data yang bertipe String.
4. Cetaklah masing- masig variable tadi yang sudah anda buat pada sebuah method dengan nama cetak.
5. Panggil method tadi di method utama.
6. Jika menurut anda variabel kurang bisa anda tambahkan sendiri misal fakultas, jurusan, angkatan (POINT++)