Materi II

Nama : Roy Agus Martin Marbun

Kelas : XII – RPL

Mata Pelajaran : Pemrograman Berorientasi Objek

BAB I

Data Bentuk String dan Pelbagai Propertinya

A. StringBuffer Class

StringBuffer class adalah class yang terdapat pada *package java.lang* dan digunakan untuk membuat objek *string* dapat dirubah (*mutable*) dan tidak seperti *class String* yang tidak dapat dirubah (*immutable*).

StringBuffer lebih dianjurkan untuk digunakan ketika memerlukan banyak operasi pada *string*, seperti memasukkan, menghapus atau memodifikasi *string*. *StringBuffer* dapat juga disisipi karakter dan *substring* di tengahnya, atau ditambah di belakangnya.

StringBuffer memiliki kapasitas *default* sebesar 16 karakter, tetapi biasanya ukuran dapat diatur sendiri dengan mendefinisikan kapasitas pada saat pembuatan.
Contoh :

StringBuffer baru = new StringBuffer(50);

Contoh di atas merupakan *StringBuffer* kosong dengan kapasitas 50 karakter.

1. Cara Mendefinisikan *StringBuffer*.
   Ada tiga acara untuk mendefinisikan *StringBuffer* antara lain.
   - StringBuffer nama = new StringBuffer();
     Secara tidak langsung variabel nama akan menjadi objek *StringBuffer* dengan ukuran 16 karakter karena *default*-nya adalah 16 karakter.
   - StringBuffer kelas = new StringBuffer(50);
     Objek kelas merupakan sebuah *StringBuffer* dengan panjang karakter sebesar 50 karakter.
   - StringBuffer mapel = new StringBuffer(String);
     Objek mapel merupakan sebuah *StringBuffer* dengan panjang karakter *string* ditambah dengan 16 karakter.

Contoh deklarasi *StringBuffer* dalam *Netbeans*.

```
public class ContohDasar {

    public static void main(String[] args) {

        String nama = "Grafika";
        StringBuffer kelas = new StringBuffer();
        StringBuffer mapel = new StringBuffer(50);
        StringBuffer sekolah = new StringBuffer(nama);

        System.out.println("Kelas : "+kelas.capacity());
        System.out.println("Mapel : "+mapel.capacity());
        System.out.println("Sekolah :"+sekolah.capacity());

    }
}
```

Outputnya

```
run:
Kelas : 16
Mapel : 50
Sekolah :23
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. ***Object String*.**

String adalah suatu objek yang merupakan kumpulan dari elemen karaker-karakter. Objek string dalam Java dapat dibuat dengan 2 cara, yaitu :

- Penulisan sesuatu di dalam tanda antara petik ganda (*literal String*). Cara ini digunakan untuk mengakomodasi kebiasaan dari Bahasa C/C++.
  Contohnya :

  String s = “Hello Word”;

  System.out.println(“Hello World”);

- Pembuatan objek *string* dengan *keyword new*,
  Contohnya :

  String s = new String(“Halo”);

3. ***Class String*.**

Kelas *String* di Java mempunyai cukup banyak konstruktor dan method. Untuk mewakili sebuah obyek *string* di Java, dapat menggunakan kelas *String*. Kelas *String* terletak di paket *java.lang* dan tidak perlu lagi mengimpor kelas tersebut saat digunakan di dalam program Java. Anda cukup membuat obyek dari kelas *String* dengan menggunakan salah satu konstruktor yang dimiliki.

String merupakan urutan dari karakter-karakter. Seperti misalnya “Java” adalah *string* yang terdiri dari 4 karakter. *String* adalah *class* yang bersifat *immutable*, artinya ketika

string itu dibuat maka kontennya tidak bisa diubah.Terdapat 2 cara untuk membuat *String* pada pemrograman Java, diantaranya adalah:

- Menggunakan Literal,
  Contoh :

  String nama = “Roy”;
  String Mapel = “PBO”;

- Menggunakan *keyword new*,
  Contoh :

  String nama = new String(“Roy”);
  String mapel = new String(“PBO”);

4. Konstruktor.

Konstruktor pada Java merupakan method khusus yang dipakai oleh Java untuk membuat sebuah objek di dalam kelas dan tiap kelas boleh memiliki lebih dari satu konstruktor. Konstruktor juga merupakan method yang akan dieksekusi pada saat pembuatan objek (*instance*).
Contoh Konstruktor :

```
public ContohDasar(){
    System.out.println("Ini adalah contoh Konstruktor");
}
```

Karakteristik daripada sebuah konstruktor yaitu antara lain :

a) Nama Konstruktor = nama Kelas.
b) Tidak mengembalikan nilai atau *return value* termasuk *void*.
c) Cara menggunakan konstruktor yaitu dengan menggunakan kata kunci *new*.

5. *Modifier*.

Class dalam program Java dapat saling berhubungan dengan cara memberikan akses terhadap member mereka. Salah satu hubungan class yang pernah dipelajari yaitu *inheritance* (pewarisan).

Pada hubungan *inheritance*, semua method ataupun atribut yang di dalam *class* induk akan bisa diakes oleh *class* anak atau subclassnya, kecuali method atau atribut tersebut diberikan modifier *private*.

Secara Umum ada 3 macam modifier yang digunakan dalam Java yaitu : *Public*, *Private* dan *Protected*. Jika tidak menggunakan salah satu dari *modifier* tersebut, maka isi daripada *class* tidak menggunakan *modifier* atau *no modifier*.

Wilayah Akes	Public	Protected	Default	Private
Di Kelas yang sama	V	V	V	V
Beda kelas dan di package yang sama	V	V	V	X
Beda kelas, beda package dan berada di kelas turunan	V	V	X	X
Beda kelas, beda package dan tidak di kelas turunan.	V	X	X	X

Deskripsi :

- *Public* : *Modifier* yang membuat kelas/method/atribut dapat diakses dari mana saja.
- *Protected* : *Modifier* yang membuat kelas/method/atribut hanya bisa diakses dari *class* itu sendiri, *subclass* atau class anak dan berada dalam satu *package*.
- *Private* : *Modifier* yang membuat kelas/method/atribut hanya bisa diakses oleh dari dalam *class* itu sendiri.

6. *Method* pada *Class String*.

Method merupakan sebutan untuk *behavioral*/*function* di Java. Method selalu memiliki kurung lengkung atau "()", kurung lengkung tersebut bisa jug disemati suatu variabel atau parameter. Parameter sendiri adalah sebutan dari variabel yang terletak dalam kurung lengkung suatu *method*. Aturan penamaan method sama dengan aturan penamaan variabel. *Method* sendiri ada dua jenis yaitu :

a. *Void Method*
Void Method adalah *method* yang tidak mengembalikan suatu nilai. Dilihat sekilas, void method dapat diidentifikasi dengan adanya kata kunci "*void*" di depan nama method.
Contoh :

```
void eat(){
 System.out.println("anjing makan");
 }
```

b. *Return Method*
Return Method adalah *method* yang mengembalikan nilai. *Method* ini bisa diidentifikasi dengan adanya data *type* di depan nama *method*-nya dan kata kunci return di dalam *method*-nya.
Contoh :

```
int getAge(){
 return 3;
 }
```

Method memiliki beberapa karakteristik diantaranya sebagai berikut.

- Dapat mengembalikan satu nilai (*return*) atau tidak sama sekali (*void*).
- Dapat diterima beberapa parameter yang dibutuhkan atau tidak ada parameter sama sekali. Parameter bisa juga disebut sebagai argumen fungsi.
- Setelah *method* telah selesai dieksekusi, *method* akan kembali pada *method* yang memanggilnya.