

Pertemuan VI

METODE PADA *PACKAGE JAVA.LANG*

6.1. Metode untuk *Class String* dan *StringBuffer*

Class String disediakan oleh Java SDK dengan menggunakan kombinasi *character literals*. Tidak seperti bahasa pemrograman lainnya, seperti C atau C++, *strings* dapat digunakan menggunakan array dari karakter atau disederhanakan dengan menggunakan *class String*. Sebagai catatan, bahwa sebuah objek *String* berbeda dari sebuah array dari karakter.

Di bahasa java, *string* direpresentasikan dalam dua kelas, yaitu :

- a. *String*, untuk *string* konstan, yang tidak berubah isinya setelah penciptaannya.
- b. *StringBuffer*, untuk *string* yang memerlukan banyak manipulasi.

6.1.1. Metode untuk *Class String*

6.1.1.1. *Constructor String*

Ada beberapa *constructor* untuk kelas *string*, antara lain :

- a. *String()*
- b. *String(String Value)*
- c. *String(char value[])*
- d. *String(char value[], int offset, int count)*
- e. *String(byte ascii[], int hibyte, int offset, int count)*
- f. *String(byte ascii[], int hibyte)*
- g. *String(StringBuffer buffer)*

Contoh penggunaan *constructor* kelas *String* dalam program adalah sebagai berikut :

```
public class ConstructorString{
    public static void main(String args[]){
        String st1 = new String();
        st1 = "Constructor";
        String st2 = new String("Constructor");
        char ch[] = {'C','o','n','s','t','r','u','c','t','o','r'};
        String st3 = new String(ch);
        String st4 = new String(ch,0,ch.length);
        byte bytes[] = {'C','o','n','s','t','r','u','c','t','o','r'};
        String st5 = new String(bytes,0,bytes.length);
        String st6 = new String(bytes,0);
        String st7 = new String(new StringBuffer("Constructor
StringBuffer"));

        System.out.println(st1);
        System.out.println(st2);
        System.out.println(st3);
        System.out.println(st4);
        System.out.println(st5);
        System.out.println(st6);
        System.out.println(st7);
    }
}
```

6.1.1.2. Metode-metode pada String

Setelah kita dapat menciptakan obyek *string*, maka kita siap bekerja dengan menggunakan metode-metode di kelas *string*. Beberapa metode yang bermanfaat di kelas *string* adalah :

- `int length()`
- `char charAt(int index)`
- `boolean startsWith(String prefix)`
- `boolean startsWith(String prefix, int toffset)`
- `boolean endsWith(String suffix)`
- `int indexOf(int ch)`
- `int indexOf(int ch, int fromIndex)`
- `int indexOf(String str)`
- `int indexOf(String str, int fromIndex)`
- `int lastIndexOf(int ch)`
- `int lastIndexOf(int ch, int fromIndex)`
- `int lastIndexOf(String str)`
- `int lastIndexOf(String str, int fromIndex)`
- `String substring(int beginIndex)`
- `String substring(int beginIndex, int endIndex)`
- `boolean equals(Object anObject)`
- `boolean equalsIgnoreCase(String anotherString)`
- `int compareTo(String anotherString)`
- `String concat(String str)`
- `String replace(char oldChar, char newChar)`
- `String trim()`
- `String toLowerCase()`
- `String toUpperCase()`
- `static String valueOf(Object obj)`
- `static String valueOf(char data[])`
- `static String valueOf(char data[], int offset, int count)`
- `static String valueOf(boolean b)`
- `static String valueOf(char c)`
- `static String valueOf(int i)`
- `static String valueOf(long l)`
- `static String valueOf(float f)`
- `static String valueOf(double d)`
- `static Double Double.valueOf(String st).doubleValue()`

Contoh penggunaan metode-metode *String* dalam program adalah sebagai berikut :

```
public class StringMethod{
    public static void main(String args[]){
        int indeks;
        String st = new String("ini untuk mencoba Metode string!");
        /* menghitung jumlah karakter dari st */
        int pj = st.length();
        /* membaca karakter ke-5 dari st */
```

```
char c = st.charAt(5);
/* apakah st dimulai dengan "ini" */
boolean b1 = st.startsWith("ini");
/* apakah st dimulai dengan "st" mulai karakter ke-4 */
boolean b2 = st.startsWith("st",4);
/* apakah st diakhiri dengan "string." */
boolean b3 = st.endsWith("string.");
/* apakah st diakhiri dengan "g!" */
boolean b4 = st.endsWith("g!");

System.out.println(st);
System.out.println(pj);
System.out.println(c);
System.out.println(b1);
System.out.println(b2);
System.out.println(b3);
System.out.println(b4);

/* pencarian karakter dari depan */
System.out.print("Karakter 'n' berada pada indeks : ");
indeks = st.indexOf('n');
while (indeks>=0){
    System.out.print(indeks + " ");
    indeks = st.indexOf('n',indeks+1);
}
System.out.println();

/* pencarian teks dari depan */
System.out.print("Teks \"in\" berada pada indeks : ");
indeks = st.indexOf("in");
while (indeks>=0){
    System.out.print(indeks + " ");
    indeks = st.indexOf("in",indeks+1);
}
System.out.println();

/* pencarian karakter dari belakang */
System.out.print("Karakter 'n' berada pada indeks : ");
indeks = st.lastIndexOf('n');
while (indeks>=0){
    System.out.print(indeks + " ");
    indeks = st.lastIndexOf('n',indeks-1);
}
System.out.println();

/* pencarian teks dari belakang */
System.out.print("Teks \"in\" berada pada indeks : ");
indeks = st.lastIndexOf("in");
while (indeks>=0){
    System.out.print(indeks + " ");
    indeks = st.lastIndexOf("in",indeks-1);
}
System.out.println();

/* menampilkan substring dari st mulai karakter ke-6 sampai terakhir
*/
System.out.println(st.substring(6));
/* menampilkan substring dari st mulai karakter ke-6 sampai ke-17 */
System.out.println(st.substring(6,17));

String st2 = new String("ini untuk mencoba Metode string!");
```

```

        System.out.println("st == st2 : " + st==st2);
        System.out.println("st.equals(st2) : " + st.equals(st2));
        System.out.println("st.equalsIgnoreCase(st2.toUpperCase()) : " +
st.equalsIgnoreCase(st2.toUpperCase()));

        System.out.println("st.compareTo(st2) : " + st.compareTo(st2));
        System.out.println("st.compareTo(st2.toLowerCase()) : " +
st.compareTo(st2.toLowerCase()));
        System.out.println("st.compareTo(st2.toUpperCase()) : " +
st.compareTo(st2.toUpperCase()));
    }
}

```

6.1.2. Metode untuk *Class StringBuffer*

6.1.2.1. Constructor *StringBuffer*

Ada beberapa constructor untuk kelas *StringBuffer*, yaitu :

- StringBuffer()*
Digunakan untuk menciptakan obyek *StringBuffer* kosong.
- StringBuffer(int length)*
Untuk menciptakan obyek *StringBuffer* yang panjangnya sesuai nilai *Length* dan diinisialisasi dengan spasi.
- StringBuffer(String str)*
Untuk menciptakan *StringBuffer* dari obyek *string*. Berguna ketika perlu melakukan modifikasi obyek *string* konstan.

Contoh penggunaan *constructor StringBuffer* untuk menciptakan obyek *StringBuffer* :

```

String st = new String("Constructor StringBuffer");
StringBuffer sb1 = new StringBuffer();
StringBuffer sb2 = new StringBuffer(32);
StringBuffer sb3 = new StringBuffer(st);

```

6.1.2.2. Metode-Metode Pada *StringBuffer*

Beberapa metode di kelas *StringBuffer* adalah sebagai berikut :

- *int length()*
- *int capacity()*
- *synchronized void setLength(int newLength)*
- *synchronized char charAt(int index)*
- *synchronized void setCharAt(int index, char ch)*
- *synchronized StringBuffer append(Object obj)*
- *synchronized StringBuffer append(String str)*
- *synchronized StringBuffer append(char c)*
- *synchronized StringBuffer append(char str[])*
- *synchronized StringBuffer append(char str[] , int offset, int len)*
- *StringBuffer append (boolean b)*
- *StringBuffer append(int i)*
- *StringBuffer append(long l)*

- `StringBuffer append(float f)`
- `StringBuffer append(double d)`
- `synchronized StringBuffer insert(int offset, Object obj)`
- `synchronized StringBuffer insert(int offset, String str)`
- `synchronized StringBuffer insert(int offset, char c)`
- `synchronized StringBuffer insert(int offset, char str[])`
- `StringBuffer insert(int offset, boolean b)`
- `StringBuffer insert(int offset, int i)`
- `StringBuffer insert(int offset, long l)`
- `StringBuffer insert(int offset, float f)`
- `StringBuffer insert(int offset, double d)`
- `String toString()`

Contoh penggunaan metode-metode *StringBuffer* dalam program adalah sebagai berikut :

```
public class StringBufferMethod{
    public static void main(String args[]){
        StringBuffer sb = new StringBuffer("Mencoba metode StringBuffer");

        System.out.println("Nilai sb : \""+sb+"\"");
        System.out.println(sb.length());
        System.out.println(sb.capacity());

        sb.setLength(28);
        System.out.println("Nilai sb : \""+sb+"\"");
        System.out.println(sb.length());
        System.out.println(sb.capacity());

        System.out.println("Karakter ke-6 : "+sb.charAt(6));
        sb.setCharAt(14, '-');
        System.out.println("Nilai sb : \""+sb+"\"");

        String st = new String("di java ");
        sb.append(st);
        sb.append(2f);
        System.out.println("Nilai sb : \""+sb+"\"");

        sb.insert(15, "metode ");
        System.out.println("Nilai sb : \""+sb+"\"");
        st=sb.toString();
        System.out.println("Nilai st : \""+st+"\"");
    }
}
```

6.2. Metode untuk *Class Math*

Java juga menyediakan konstanta dan metode untuk menunjukkan perbedaan operasi matematika seperti fungsi trigonometri dan logaritma. Selama metode-metode ini semua *static*, kita dapat menggunakannya tanpa memerlukan sebuah objek *Math*. Untuk melengkapi daftar konstanta dan metode-metode ini, lihatlah acuan pada dokumentasi Java API. Dibawah ini beberapa metode-metode umum yang sering digunakan.

- `public static double abs(double a)`
Menghasilkan nilai mutlak `a`. Sebuah metode yang di-overload. Dapat juga menggunakan nilai float atau integer atau juga long integer sebagai parameter, dengan kondisi tipe kembaliannya sesuai tipe data `a`, yaitu :
`public static float abs(float a)`
`public static long abs(long a)`
`public static int abs(int a)`
- `public static double random()`
Menghasilkan nilai positif bilangan acak (random) yang lebih besar atau sama dengan 0.0 tetapi kurang dari 1.0.
- `public static double max(double a, double b)`
Menghasilkan nilai maksimum, diantara dua nilai double, `a` and `b`. Sebuah metode yang di-overload. Dapat juga menggunakan nilai float atau integer atau juga long integer sebagai parameter, dengan kondisi tipe kembalinya juga menggunakan float atau integer atau long integer, secara berturut-turut.
- `public static double min(double a, double b)`
Menghasilkan nilai minimum diantara dua nilai double, `a` and `b`. Sebuah metode yang di-overload. Dapat juga menggunakan nilai float atau integer atau juga long integer sebagai parameter, dengan kondisi tipe kembaliannya juga menggunakan float atau integer atau long integer, secara berturut-turut.
- `public static double ceil(double a)`
Menghasilkan bilangan bulat terkecil yang lebih besar atau sama dengan `a`.
- `public static double floor(double a)`
Menghasilkan bilangan bulat terbesar yang lebih kecil atau sama dengan `a`.
- `public static double exp(double a)`
Menghasilkan angka Euler, `e` pangkat `a`.
- `public static double log(double a)`
Menghasilkan logaritma natural dari `a`.
- `public static double pow(double a, double b)`
Menghasilkan `a` pangkat `b`.
- `public static long round(double a)`
Menghasilkan pembulatan ke atas ke long terdekat. Sebuah metode yang di-overload. Dapat juga menggunakan float pada argument dan akan menghasilkan pembulatan ke atas ke int terdekat.
- `public static double sqrt(double a)`
Menghasilkan akar kuadrat `a`.
- `public static double sin(double a)`
Menghasilkan sinus sudut `a` dalam radian.
- `public static double cos(double a)`
Menghasilkan cosinus sudut `a` dalam radian.
- `public static double tan(double a)`
Menghasilkan tangen sudut `a` dalam radian.
- `public static double asin(double a)`
Menghasilkan anti-sinus sudut `a` dalam radian.
- `public static double acos(double a)`
Menghasilkan anti-cosinus sudut `a` dalam radian.

- `public static double atan(double a)`
Menghasilkan anti-tangen sudut a dalam radian.
- `public static double toDegrees(double angrad)`
Menghasilkan nilai derajat yang kira-kira setara dengan nilai radian yang diberikan.
- `public static double toRadians(double angdeg)`
Menghasilkan nilai radian yang kira-kira setara dengan nilai derajat yang diberikan.

Selain metode-metode diatas, ada nilai konstanta yang sering digunakan, yaitu E dan π . Konstanta E merepresentasikan nilai simbol e pada operasi ekponensial, nilainya adalah 2.7182..... Sedangkan konstanta π merepresentasikan simbol Phi, yang nilainya 3.1415.....

Contoh penggunaan metode-metode *Math* dalam program adalah sebagai berikut :

```
public class MathDemo {
    public static void main(String args[]) {
        System.out.println("Nilai absolut dari -5 : " + Math.abs(-5));
        System.out.println("Nilai absolut dari 5 : " + Math.abs(5));
        System.out.println("Nialai random(nilai max : 10) : " +
Math.random()*10);
        System.out.println("max dari 3.5 dan 1.2 : " + Math.max(3.5, 1.2));
        System.out.println("min dari 3.5 dan 1.2 : " + Math.min(3.5, 1.2));
        System.out.println("Pembulatan keatas dari 3.3: " + Math.ceil(3.3));
        System.out.println("Pembulatan keatas dari -3.3: " + Math.ceil(-
3.3));
        System.out.println("Pembulatan kebawah dari 3.6: " +
Math.floor(3.6));
        System.out.println("Pembulatan kebawah dari -3.3: " + Math.floor(-
3.3));
        System.out.println("Nilai e : " + Math.E);
        System.out.println("e pangkat 1 : " + Math.exp(1));
        System.out.println("log 10 : " + Math.log(10));
        System.out.println("10 pangkat 3 : " + Math.pow(10,3));
        System.out.println("Nilai pi : " + Math.PI);
        System.out.println("Nilai pembulatan dari pi : " +
Math.round(Math.PI));
        System.out.println("Akar kuadrat dari 5 : " + Math.sqrt(5));
        System.out.println("10 radian : " + Math.toDegrees(10) + " degrees");
        System.out.println("sin(90) : " + Math.sin(Math.toRadians(90)));
        System.out.println("cos(180) : " + Math.cos(Math.toRadians(180)));
    }
}
```

Output atau keluaran dari program diatas adalah :

```
Nilai absolut dari -5 : 5
Nilai absolut dari 5 : 5
Nialai random(nilai max : 10) : 8.511990856790277
max dari 3.5 dan 1.2 : 3.5
min dari 3.5 dan 1.2 : 1.2
Pembulatan keatas dari 3.3: 4.0
Pembulatan keatas dari -3.3: -3.0
Pembulatan kebawah dari 3.6: 3.0
Pembulatan kebawah dari -3.3: -4.0
Nilai e : 2.718281828459045
```

```
e pangkat 1 : 2.7182818284590455
log 10 : 2.302585092994046
10 pangkat 3 : 1000.0
Nilai pi : 3.141592653589793
Nilai pembulatan dari pi : 3
Akar kuadrat dari 5 : 2.23606797749979
10 radian : 572.9577951308232 degrees
sin(90) : 1.0
cos(180) : -1.0
```

6.3. Metode untuk *Class Data Type Wrapper*

Sesungguhnya, tipe data *primitive* seperti *int*, *char* and *long* bukanlah sebuah objek. Sehingga, variabel-variabel tipe data ini tidak dapat mengakses metode-metode dari *class Object*. Hanya objek-objek nyata, yang dideklarasikan menjadi referensi tipe data, dapat mengakses metode-metode dari *class Object*. Ada suatu keadaan, bagaimanapun, ketika kita membutuhkan sebuah representasi objek untuk variabel-variabel tipe *primitive* dalam rangka menggunakan metode-metode Java *built-in*. Sebagai contoh, kita boleh menambahkan variabel tipe *primitive* pada objek *Collection*. Disinilah *class wrapper* masuk. *Class wrapper* adalah representasi objek sederhana dari variabel-variabel non-objek yang sederhana. Demikian daftar dari *class wrapper*.

Tabel 6.1. Tipe data primitive dan *class wrapper*-nya yang sesuai

<i>Tipe Data Primitive</i>	<i>Class Wrapper yang Sesuai</i>
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Nama-nama *class wrapper* cukup mudah untuk diingat selama nama-nama itu sama dengan tipe data *primitive*. Dan juga sebagai catatan, bahwa *class-class wrapper* diawali dengan huruf besar dan versi yang ditunjukkan dari tipe data *primitive*.

Di bawah ini contoh penggunaan *class wrapper* untuk *boolean*.

```
class BooleanWrapper {
    public static void main(String args[]) {
        boolean booleanVar = 1>2;
        Boolean booleanObj = new Boolean("TRUE");
        /* primitif ke objek; dapat juga menggunakan method valueOf */
        Boolean booleanObj2 = new Boolean(booleanVar);
        System.out.println("booleanVar = " + booleanVar);
        System.out.println("booleanObj = " + booleanObj);
        System.out.println("booleanObj2 = " + booleanObj2);
        System.out.println("compare 2 wrapper objects: " +
booleanObj.equals(booleanObj2));
        /* objek ke primitif */
        booleanVar = booleanObj.booleanValue();
        System.out.println("booleanVar = " + booleanVar);
    }
}
```

6.4. Metode untuk *Class Process dan Runtime*

6.4.1. *Class Process*

Class Process menyediakan metode-metode untuk memanipulasi proses-proses, seperti mematikan proses, menjalankan proses dan mengecek status proses. *Class* ini merepresentasikan program-program yang berjalan. Di bawah ini beberapa metode pada *class Process*.

- `public abstract void destroy()`
Mengakhiri proses.
- `public abstract int waitFor() throws InterruptedException`
Tidak mengirim sampai proses yang dipanggil berakhir.

6.4.2. *Class Runtime*

Di sisi lain, *class Runtime* merepresentasikan lingkungan *runtime*. Dua metode penting pada *class Runtime* adalah metode *getRuntime* dan *exec*.

- `public static Runtime getRuntime()`
Mengirim objek *runtime* yang merepresentasikan lingkungan *runtime* yang berhubungan dengan aplikasi Java saat itu.
- `public Process exec(String command) throws IOException`
Disebabkan *command* yang ditentukan untuk dieksekusi. Memperbolehkan Anda mengeksekusi proses baru.

6.4.3. Membuka *Registry Editor*

Berikut program untuk membuka registry editor tanpa harus mengetikkan perintah dari command prompt.

```
class RuntimeDemo {
    public static void main(String args[]) {
        Runtime rt = Runtime.getRuntime();
        Process proc;
        try {
            proc = rt.exec("regedit");
            proc.waitFor(); //cobalah menghapus baris ini
        }
        catch (Exception e) {
            System.out.println("regedit is an unknown command.");
        }
    }
}
```

6.5. Metode untuk *Class System*

Class System menyediakan beberapa *field* dan metode yang bermanfaat, seperti *standard input*, *standard output* dan sebuah metode yang berguna untuk mempercepat penyalinan bagian sebuah array. Di bawah ini beberapa metode menarik dari *class System*. Sebagai catatan, bahwa semua metode-metode *class* adalah *static*.

- `Public static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length)`
Mengkopi *length* elemen dari array *src* dimulai pada posisi *srcPos* ke *dest* yang dimulai pada indeks *destPos*. Lebih cepat daripada memprogram secara manual kode program sendiri.

- `Public static long currentTimeMillis()`
Waktu ditentukan dalam GMT (Greenwich Mean Time) serta merupakan jumlah milidetik yang telah dilewati sejak tengah malam 1 Januari 1970. Waktu dalam ukuran milidetik.
- `Public static void exit(int status)`
Mematikan Java Virtual Machine (JVM) yang sedang berjalan. Nilai bukan nol untuk status konvensi yang mengindikasikan keluar yang abnormal.
- `Public static void gc()`
Menjalankan garbage collector, yang mereklamasi space memori tak terpakai untuk digunakan kembali.
- `Public static void setIn(InputStream in)`
Mengubah stream yang berhubungan dengan `System.in`, yang mana standart mengacu pada keyboard.
- `Public static void setOut(PrintStream out)`
Mengubah stream yang berhubungan dengan `System.out`, yang mana standart mengacu pada console.

Ini adalah demo dari beberapa metode-metode diatas :

```
import java.io.*;
class SystemDemo {
    public static void main(String args[]) throws IOException {
        int arr1[] = new int[1050000];
        int arr2[] = new int[1050000];
        long startTime, endTime;

        /* menginisialisasi arr1 */
        for (int i = 0; i < arr1.length; i++) {
            arr1[i] = i + 1;
        }

        /* mengkopi secara manual */
        startTime = System.currentTimeMillis();
        for (int i = 0; i < arr1.length; i++) {
            arr2[i] = arr1[i];
        }

        endTime = System.currentTimeMillis();
        System.out.println("Time for manual copy: " + (endTime-startTime) +
            " ms.");
        /* menggunakan utilitas copy yang disediakan oleh java - yaitu
        method arraycopy */
        startTime = System.currentTimeMillis();
        System.arraycopy(arr1, 0, arr2, 0, arr1.length);
        endTime = System.currentTimeMillis();
        System.out.println("Time for manual copy: " + (endTime-startTime) +
            " ms.");
        System.gc(); //force garbage collector to work
        System.setIn(new FileInputStream("temp.txt"));
        System.exit(0);
    }
}
```

Referensi:

1. Hariyanto, Bambang, (2007), *Esensi-esensi Bahasa Pemrograman Java*, Edisi 2, Informatika Bandung, November 2007.
2. Utomo, Eko Priyo, (2009), *Panduan Mudah Mengenal Bahasa Java*, Yrama Widya, Juni 2009.
3. Tim Pengembang JENI, JENI 1-6, Depdiknas, 2007