Bab 4
Tour dari Package *java.lang*

4.1 Tujuan

Java datang dengan beberapa class built-in yang bermanfaat. Mari kita membahas class-class tersebut.

Setelah melengkapi pelajaran ini, Anda diharapkan dapat:

1. Menggunakan class-class Java yang telah ada
   - *Math*
   - *String*
   - *StringBuffer*
   - Wrapper
   - *Process*
   - *System*

4.2 Class *Math*

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

Method-Method Math
`public static double abs(double a)`
Menghasilkan nilai mutlak a. Sebuah method 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 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 method 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 method 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 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 keatas ke *long* terdekat. Sebuah method 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 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.

Tabel 1.1: Beberapa method dari class Math

Di bawah ini adalah program yang menunjukkan bagaimana method-method tersebut digunakan.

```
class MathDemo {
   public static void main(String args[]) {
      System.out.println("absolute value of -5: " +
                                                Math.abs(-5));
      System.out.println("absolute value of 5: " +
                                                Math.abs(-5));
      System.out.println("random number(max value is 10): " +
                                              Math.random()*10);
      System.out.println("max of 3.5 and 1.2: " +
                                             Math.max(3.5, 1.2));
      System.out.println("min of 3.5 and 1.2: " +
                                             Math.min(3.5, 1.2));
      System.out.println("ceiling of 3.5: " + Math.ceil(3.5));
      System.out.println("floor of 3.5: " + Math.floor(3.5));
      System.out.println("e raised to 1: " + Math.exp(1));
      System.out.println("log 10: " + Math.log(10));
      System.out.println("10 raised to 3: " + Math.pow(10,3));
      System.out.println("rounded off value of pi: " +
                                           Math.round(Math.PI));
      System.out.println("square root of 5 = " + Math.sqrt(5));
      System.out.println("10 radian = " + Math.toDegrees(10) +
                                              " degrees");
      System.out.println("sin(90): " +
                                  Math.sin(Math.toRadians(90)));
   }
}
```

Ini adalah contoh output dari program yang dibuat. Coba jalankan program dan bereksperimenlah secara bebas dengan memberikan argument.

```
absolute value of -5: 5
absolute value of 5: 5
random number(max value is 10): 4.0855332335477605
max of 3.5 and 1.2: 3.5
min of 3.5 and 1.2: 1.2
ceiling of 3.5: 4.0
floor of 3.5: 3.0
e raised to 1: 2.7182818284590455
log 10: 2.302585092994046
10 raised to 3: 1000.0
rounded off value of pi: 3
square root of 5 = 2.23606797749979
10 radian = 572.9577951308232 degrees
sin(90): 1.0
```

4.3 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 character atau disederhanakan dengan menggunakan class *String*. Sebagai catatan, bahwa sebuah objek *String* berbeda dari sebuah array dari character.

4.3.1 Constructor String

Class *String* mempunyai 11 constructor. Untuk melihat bagaimana constructor-constructor ini, perhatikan contoh berikut.

```
/* Contoh ini diambil dari catatan Dr. Encarnacion. */
class StringConstructorsDemo {
   public static void main(String args[]) {
      String s1 = new String();  // creates an empty string
      char chars[] = { 'h', 'e', 'l', 'l', 'o'};
      String s2 = new String(chars); // s2 = "hello";
      byte bytes[] = { 'w', 'o', 'r', 'l', 'd' };
      String s3 = new String(bytes);      // s3 = "world"
      String s4 = new String(chars, 1, 3);
      String s5 = new String(s2);
      String s6 = s2;
      System.out.println(s1);
      System.out.println(s2);
      System.out.println(s3);
      System.out.println(s4);
      System.out.println(s5);
      System.out.println(s6);
   }
}
```

4.3.2 Method-method String

Di bawah ini adalah daftar dari method-method *String*.

Method-Method String
`public char charAt(int index)`
Mengirim karakter di indeks yang dispesifikasikan oleh parameter *index*.
`public int compareTo(String anotherString)`
Membandingkan dua *String* dan mengirim bilangan int yang menspesifikasikan apakah objek string pemanggil kurang dari atau sama dengan *anotherString.* Bernilai negatif jika objek yang dilewatkan (*passed* string) lebih besar, 0 jika kedua string sama, dan bernilai positif jika objek string pemanggil (calling string) lebih besar.
`public int compareToIgnoreCase(String str)`
Serupa dengan compareTo tetapi *case insensitivity*.
`public boolean equals(Object anObject)`

Method-Method String
Menghasilkan nilai *true* jika parameter tunggalnya tersusun dari karakter yang sama dengan objek tempat Anda memanggil *equals*. Sedangkan jika parameter yang dispesifikkan bukan sebuah objek *String* atau jika tidak cocok dengan urutan simbol pada string, method akan dikembalikan dengan nilai false.
`public boolean equalsIgnoreCase(String anotherString)`
Serupa dengan equals tetapi *case insensitivity*.
`public void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)`
Mendapatkan characters dari string yang dimulai pada index *srcBegin* hingga index *srcEnd* dan mengkopi character-character tersebut pada array *dst* dimulai pada index *dstBegin*.
`public int length()`
Menghasilkan panjang String.
`public String replace(char oldChar, char newChar)`
Mengganti karakter, semua yang kemunculan *oldChar* diganti *newChar*.
`public String substring(int beginIndex, int endIndex)`
Mengirim substring dimulai dari indeks yang dispesifikasikan *beginIndex* dan berakhir dengan indeks yang dispesifikasikan *endIndex.*
`public char[] toCharArray()`
Returns the character array equivalent of this string.
`public String trim()`
Menghilangkan *whitespace* di awal dan akhir objek String.
`public static String valueOf(-)`
Dapat menggunakan tipe data sederhana seperti boolean, integer atau character, atau juga menggunakan sebuah objek sebagai parameter. Mengirim objek String yang merepresentasikan tipe tertentu yang dilewatkan sebagai parameter.

Tabel 1.2.1: Beberapa method dari class String

Perhatikan bagaimana method-method tersebut digunakan dalam program di bawah ini.

```
class StringDemo {
   public static void main(String args[]) {
      String name = "Jonathan";
      System.out.println("name: " + name);
      System.out.println("3rd character of name: " +
                                                name.charAt(2));
      /* character yang pertama nampak secara berurutan
      mempunyai nilai unicode lebih kecil */
      System.out.println("Jonathan compared to Solomon: " +
                                        name.compareTo("Solomon"));
      System.out.println("Solomon compared to Jonathan: " +
                                "Solomon".compareTo("Jonathan"));
      /* 'J' mempunyai nilai unicode yang lebih kecil dibanding
      'j' */
      System.out.println("Jonathan compared to jonathan: " +
```

```
                                        name.compareTo("jonathan"));
            System.out.println("Jonathan compared to jonathan (ignore
                    case): " + name.compareToIgnoreCase("jonathan"));
            System.out.println("Is Jonathan equal to Jonathan? " +
                                        name.equals("Jonathan"));
            System.out.println("Is Jonathan equal to jonathan? " +
                                        name.equals("jonathan"));
            System.out.println("Is Jonathan equal to jonathan (ignore
                      case)? " + name.equalsIgnoreCase("jonathan"));
            char charArr[] = "Hi XX".toCharArray();
            /* Membutuhkan tambahan 1 untuk indeks endSrc dari
            getChars */
            "Jonathan".getChars(0, 2, charArr, 3);
            System.out.print("getChars method: ");
            System.out.println(charArr);
            System.out.println("Length of name: " + name.length());
            System.out.println("Replace a's with e's in name: " +
                                         name.replace('a', 'e'));
            /* Membutuhkan tambahan 1 untuk parameter endIndex dari
            substring*/
            System.out.println("A substring of name: " +
                                           name.substring(0, 2));
            System.out.println("Trim \"  a b c d e f   \": \"" +
                                "  a b c d e f   ".trim() + "\"");
            System.out.println("String representation of boolean
                        expression 10>10: " + String.valueOf(10>10));
            /* method toString secara implisit dipanggil method
            println */
            System.out.println("String representation of boolean
                        expression 10<10: " + (10<10));
            /* Catatan, tidak ada perubahan pada nama objek String
            meskipun setelah penggunaan semua method. */
            System.out.println("name: " + name);
        }
    }
```

Ini adalah output dari program yang dibuat.

```
name: Jonathan
3rd character of name: n
Jonathan compared to Solomon: -9
Solomon compared to Jonathan: 9
Jonathan compared to jonathan: -32
Jonathan compared to jonathan (ignore case): 0
Is Jonathan equal to Jonathan? true
Is Jonathan equal to jonathan? false
Is Jonathan equal to jonathan (ignore case)? true
content of charArr after getChars method: Hi Jo
Length of name: 8
Replace a's with e's in name: Jonethen
A substring of name: Jo
Trim "  a b c d e f   ": "a b c d e f"
String representation of boolean expression 10>10: false
String representation of boolean expression 10<10: false
name: Jonathan
```

4.3.3 Class StringBuffer

Ketika objek *String* diciptakan, objek *String* tidak bisa lagi dimodifikasi. Objek *StringBuffer* serupa dengan objek *String*, kecuali kenyataan bahwa objek *StringBuffer* bersifat dapat berubah atau dapat dimodifikasi, sedangkan pada object *String* bersifat konstan. Panjang dan isi dapat diubah hingga beberapa pemanggilan method.

Ini adalah beberapa method pada class *StringBuffer*. Lihatlah acuan pada dokumentasi Java API.

Method-Method StringBuffer
`public int capacity()`
Mengirim jumlah memori yang dialokasikan untuk *StringBuffer*.
`public StringBuffer append(-)`
Appends merepresentasikan string dari argument untuk objek *StringBuffer*. Menggunakan parameter tunggal seperti tipe-tipe data berikut: *boolean, char, char [], double, float, int, long, Object, String and StringBuffer*. Masih mempunyai versi yang di-overload lainnya.
`public char charAt(int index)`
Mengirim character di lokasi tertentu di *StringBuffer* yang dispesifikasikan parameter *index*.
`public void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)`
Mendapatkan characters dari objek yang dimulai pada indeks *srcBegin* hingga indeks *srcEnd* dan mengkopi character- character tersebut pada array *dst* dimulai pada indeks *dstBegin*.
`public StringBuffer delete(int start, int end)`
Menghapus character-character pada range yang ditentukan.
`public StringBuffer insert(int offset, -)`
Menyisipkan beragam tipe data di *offset* spesifik di StringBuffer. Sebuah method yang di-overload. Tipe data yang mungkin digunakan: *boolean, char, char [], double, float, int, long, Object and String*. Masih mempunyai versi yang di-overload lainnya.
`public int length()`
Memperoleh panjang atau jumlah character di objek *StringBuffer*.
`public StringBuffer replace(int start, int end, String str)`
Mengganti bagian dari objek, seperti yang dispesifikasikan oleh argument satu dua, dengan spesifikasi string *str*.
`public String substring(int start, int end)`
Substring menyaring bagian tertentu dari string, dimulai pada pengspesifikasian indeks *start* hingga indeks the *end*.
`public String toString()`
Mengkonversi objek ke representasi string.

Tabel 1.2.2: Beberapa method dari class StringBuffer

Program di bawah ini menunjukkan bagaimana menggunakan method-method tersebut.

```
class StringBufferDemo {
   public static void main(String args[]) {
      StringBuffer sb = new StringBuffer("Jonathan");
      System.out.println("sb = " + sb);
      /* initial capacity is 16 */
      System.out.println("capacity of sb: " + sb.capacity());
      System.out.println("append \'O\' to sb: " +
                                          sb.append("O"));
      System.out.println("sb = " + sb);
      System.out.println("3rd character of sb: " +
                                           sb.charAt(2));
      char charArr[] = "Hi XX".toCharArray();
      /* Need to add 1 to the endSrc index of getChars */
      sb.getChars(0, 2, charArr, 3);
      System.out.print("getChars method: ");
      System.out.println(charArr);
      System.out.println("Insert \'jo\' at the 3rd cell: " +
                                         sb.insert(2, "jo"));
      System.out.println("Delete \'jo\' at the 3rd cell: " +
                                            sb.delete(2,4));
      System.out.println("length of sb: " + sb.length());
      System.out.println("replace: " +
                                    sb.replace(3, 9, " Ong"));
      /* Need to add 1 to the endIndex parameter of substring*/
      System.out.println("substring (1st two characters): " +
                                         sb.substring(0, 3));
      System.out.println("implicit toString(): " + sb);
   }
}
```

Ini adalah output dari program yang telah dibuat di atas. Sekali lagi, bereksperimen secara bebas dengan code-code merupakan cara terbaik mempelajari sintaks-sintaks yang ada.

```
sb = Jonathan
capacity of sb: 24
append 'O' to sb: JonathanO
sb = JonathanO
3rd character of sb: n
getChars method: Hi Jo
Insert 'jo' at the 3rd cell: JojonathanO
Delete 'jo' at the 3rd cell: JonathanO
length of sb: 9
replace: Jon Ong
substring (1st two characters): Jon
implicit toString(): Jon Ong
```

4.4 Class-class Wrapper

Sesungguhnya, tipe data primitif seperti *int*, *char* and *long* bukanlah sebuah objek. Sehingga, variabel-variabel tipe data ini tidak dapat mengakses method-method dari class *Object*. Hanya objek-objek nyata, yang dideklarasikan menjadi referensi tipe data, dapat mengakses method-method dari class *Object*. Ada suatu keadaan, bagaimanapun, ketika Anda membutuhkan sebuah representasi objek untuk variabel-variabel tipe primitif dalam rangka menggunakan method- method Java built-in. Sebagai contoh, Anda boleh menambahkan variabel tipe primitif 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.

Tipe Data Primitif	***Class Wrapper yang Sesuai***
Boolean	Boolean
Char	Character
Byte	Byte
Short	Short
Int	Integer
Long	Long
Float	Float
Double	Double

Tabel 1.3: Tipe data primitif dan class wrappernya yang sesuai

Nama-nama class wrapper cukup mudah untuk diingat selama nama-nama itu sama dengan tipe data primitif. 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);
   }
}
```

4.5 Class *Process* dan *Runtime*

4.5.1 Class Process

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

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

Tabel 1.4.1: Beberapa method dari class Process

4.5.2 Class Runtime

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

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

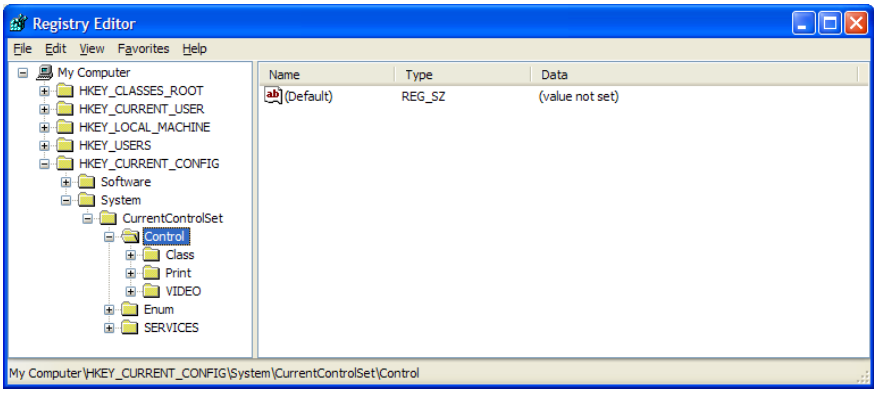
Tabel 1.4.2: Beberapa method dari class Runtime

4.5.3 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();      //try removing this line
      } catch (Exception e) {
         System.out.println("regedit is an unknown command.");
```

```
            }
        }
    }
```



Gambar 1.4.3: Membuka registry editor

4.6 Class *System*

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

Method-Method System
`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 code untuk Anda sendiri.
`Public static long currentTimeMillis()`
Waktu dispesifikasikan dalam GMT (Greenwich Mean Time) serta merupakan jumlah milidetik yang telah dilewati sejak tengan 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)`

Method-Method System
Mengubah stream yang berhubungan dengan *System.out*, yang mana standart mengacu pada console.

Tabel 1.5: Beberapa method dari class System

Ini adalah demo dari beberapa method-method tersebut.

```
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);
   }
}
```

4.7 Latihan

4.7.1 Evaluasi Ekspresi

Menggunakan method-method class built-in *Math*, buatlah sebuah program yang menggunakan nilai double *x* sebagai inputan dan evaluasilah nilai mutlak dari ekspresi yang mengikuti.
x^2 * cos(45derajat) + akar(e), e adalah angka Euler.

Input: 10
Output: 72.35939938935488
Input: 11
Output: 87.20864179427238

4.7.2 Palindrome

Palindrome adalah sebuah string yang membaca sama ketika mengarah ke depan atau sebaliknya. Beberapa contoh dari palindrome : hannah, ana, and bib. Menggunakan *String* atau class *StringBuffer*, buatlah sebuah program yang menggunakan satu string sebagai inputan dan tentukan jika ini sebuah palindrome atau bukan.

4.7.3 Notepad

Menggunakan class *Process* and *Runtime*, bukalah aplikasi notepad dari program java.