

PRAKTIKUM 7

DASAR INPUT OUTPUT

A. TUJUAN

1. Memahami dasar-dasar input-ouput dalam pemrograman Java
2. Memahami prinsip kerja stream
3. Memahami penggunaan console dalam proses input-output

B. DASAR TEORI

Pada program-program yang membutuhkan data-data eksternal, maka diperlukan suatu proses *input* dan *ouput* (I/O), dimana pada Java dukungan proses I/O ini sudah disediakan dalam paket `java.io`. Di dalam paket tersebut tersimpan banyak kelas dan *interface* siap pakai yang akan memudahkan *programmer* dalam pengambilan dan penyimpanan informasi dari/ke media lain (misalnya file).

Program Java melakukan proses I/O melalui *stream*, yaitu sebuah abstraksi yang dapat memberikan atau mendapatkan informasi. *Stream* dapat dihubungkan dengan peralatan fisik yang terdapat dalam sistem I/O Java, seperti *keyboard*, file, layar *console*, soket jaringan, dan lainnya. Walaupun dihubungkan dengan peralatan fisik yang berbeda, cara kerja *stream* selalu sama, sehingga kode program yang ditulis juga sama untuk masing-masing peralatan fisik. Misalnya, untuk melakukan penulisan sebuah teks ke layar *console* maupun ke dalam file, maka dapat digunakan kelas dan *method* yang sama.

Stream ada dua jenis, yaitu *stream byte* dan *stream karakter*. *Stream byte* digunakan untuk memberikan atau menyimpan informasi data dalam bentuk *byte*, misalnya untuk menulis dan membaca file biner. Sedangkan *stream karakter* pada proses I/O yang melibatkan data-data berbentuk karakter, misalnya proses baca/tulis ke suatu file teks, dengan menggunakan karakter *Unicode*.

Pendefinisian *stream* dilakukan dengan menggunakan empat kelas abstrak, yaitu `InputStream` dan `OutputStream`, sebagai *superclass* untuk kelas-kelas dalam kategori *stream byte*, dan kelas abstrak `Reader` dan `Writer` untuk kategori *stream*

karakter. Melalui proses pewarisan (*inheritance*), semua kelas yang diturunkan dari `InputStream` maupun `Reader` akan memiliki *method* `read()`, yang digunakan dalam proses pembacaan data. Adapun untuk proses penulisan data digunakan *method* `write()` dalam semua kelas yang diturunkan dari `OutputStream` maupun `Writer`. Daftar beberapa kelas dalam paket `java.io` yang termasuk dalam kategori *stream byte* ditunjukkan pada Tabel 1., sedangkan kelas-kelas dalam kategori *stream karakter* ditunjukkan pada Tabel 2.

Tabel 1. Kelas-kelas *stream byte*

Nama Kelas	Keterangan
<code>BufferedInputStream</code>	<i>Stream input</i> yang telah terbuffer
<code>BufferedOutputStream</code>	<i>Stream output</i> yang telah terbuffer
<code>ByteArrayInputStream</code>	<i>Stream input</i> yang membaca dari <i>array byte</i>
<code>ByteArrayOutputStream</code>	<i>Stream input</i> yang menulis ke <i>array byte</i>
<code>DataInputStream</code>	<i>Stream input</i> yang berisi <i>method-method</i> untuk membaca tipe data standar
<code>DataOutputStream</code>	<i>Stream output</i> yang berisi <i>method-method</i> untuk menuliskan tipe data standar
<code>FileInputStream</code>	<i>Stream input</i> yang membaca dari sebuah file
<code>FileOutputStream</code>	<i>Stream output</i> yang menulis ke sebuah file
<code>FilterInputStream</code>	Mengimplementasikan <code>InputStream</code>
<code>FilterOutputStream</code>	Mengimplementasikan <code>OutputStream</code>
<code>InputStream</code>	Kelas abstrak yang menjelaskan <i>stream input</i>
<code>OutputStream</code>	Kelas abstrak yang menjelaskan <i>stream output</i>
<code>PipedInputStream</code>	Penyalur <i>input</i>
<code>PipedOutputStream</code>	Penyalur <i>output</i>
<code>PrintStream</code>	<i>Stream output</i> yang berisi <i>method</i> <code>println()</code> dan <code>print()</code>

Semua program Java otomatis akan mengimpor paket `java.lang` yang mendefinisikan sebuah kelas bernama `System` yang berkaitan dengan lingkungan *runtime*. Kelas `System` juga mendefinisikan tiga variabel *stream*, yaitu `in`, `out`, dan `err`, dimana data-datanya dideklarasikan sebagai *public* atau *static*. Dengan demikian variabel-variabel tersebut dapat digunakan tanpa harus membentuk objek dari kelas `System` terlebih dahulu. `System.out` adalah *stream output* standar dengan *default* layar *console*. `System.in` adalah objek dari tipe `PrintStream`, *defaultnya* berupa *keyboard* dan mengacu pada *stream input* standar. Sedangkan pada *stream error* standar terdapat `System.err` yang secara *default* berupa layar *console*. `System.out`

dan `System.err` adalah *objek* dari tipe `PrintStream` dan mempunyai cara kerja yang sama, yaitu mengeluarkan pesan kesalahan ke layar *console*.

Tabel 2. Kelas-kelas *stream* karakter

Nama Kelas	Keterangan
<code>BufferedReader</code>	<i>Stream</i> karekter <i>input</i> yang telah terbuffer
<code>BufferedWriter</code>	<i>Stream</i> karakter <i>output</i> yang telah terbuffer
<code>CharArrayReader</code>	<i>Stream input</i> yang membaca dari <i>array</i> karakter
<code>CharArrayWriter</code>	<i>Stream input</i> yang menulis ke <i>array</i> karakter
<code>FileReader</code>	<i>Stream input</i> yang membaca dari file
<code>FileWriter</code>	<i>Stream output</i> yang menulis ke file
<code>FilterReader</code>	<i>Reader</i> yang terfilter
<code>FilterWriter</code>	<i>Writer</i> yang terfilter
<code>InputStreamReader</code>	<i>Stream input</i> yang menerjemahkan <i>byte</i> ke karakter
<code>LineNumberReader</code>	<i>Stream input</i> yang menghitung jumlah baris
<code>OutputStreamWriter</code>	<i>Stream input</i> yang menerjemahkan karakter ke <i>byte</i>
<code>PipedReader</code>	Penyalur <i>input</i>
<code>PipedWriter</code>	Penyalur <i>output</i>
<code>PrintWriter</code>	<i>Stream output</i> yang berisi <i>method</i> <code>println()</code> dan <code>print()</code>
<code>Reader</code>	Kelas abstrak yang menjelaskan <i>stream</i> karakter <i>input</i>
<code>StringReader</code>	<i>Stream input</i> yang membaca dari sebuah <i>string</i>
<code>StringWriter</code>	<i>Stream output</i> yang menulis ke sebuah <i>string</i>
<code>Writer</code>	<i>Stream output</i>

Node Stream

Terdapat tiga tipe dasar *node*, yaitu:

- File
- Memori (misalnya *objek* `array` atau `String`)
- *Pipe* (suatu kanal dari satu proses)

Tipe dari *node stream* dapat dibagi lagi sesuai dengan Tabel 3.

Tabel 3. Tipe Node Stream

Tipe	Stream Karakter	Stream Byte
File	<code>FileReader</code>	<code>FileInputStream</code>
	<code>FileWriter</code>	<code>FileOutputStream</code>
Memory:	<code>CharArrayReader</code>	<code>ByteArrayInputStream</code>
<code>array</code>	<code>CharArrayWriter</code>	<code>ByteArrayOutputStream</code>
Memory:	<code>StringReader</code>	N/A
<code>String</code>	<code>StringWriter</code>	
Pipe	<code>PipeReader</code>	<code>PipedInputStream</code>
	<code>PipeWriter</code>	<code>PipedOutputStream</code>

Melakukan Input

Dalam Java, *input console* dilakukan melalui pembacaan terhadap *stream* System.in. Untuk mendapatkan karakter-karakter yang dimasukkan melalui *keyboard* ke dalam *layer console*, diperlukan membungkus System.in di dalam *objek* BufferedReader. Hal ini dilakukan untuk membentuk *stream* karakter karena System.in sebenarnya merupakan *stream byte*. Adapun bentuk *constructor* dari BufferedReader sebagai berikut,

```
BufferedReader(Reader inputReader)
```

inputReader adalah *stream* yang akan dihubungkan dengan *instance* atau *objek* dari kelas BufferedReader yang dibuat. Karena Reader merupakan kelas abstrak, maka perlu dicari kelas turunannya yang berupa kelas konkrit. Salah satunya adalah kelas InputStreamReader, yang dapat mengonversi *byte* ke karakter. Sedangkan agar *objek* dari InputStreamReader dapat dihubungkan dengan System.in, perlu digunakan bentuk *constructor* seperti berikut,

```
InputStreamReader(InputStream inputStream)
```

Dalam hal ini, inputStream dapat diisi dengan System.in. Sehingga untuk membuat *objek* BufferedReader yang dapat terhubung dengan *keyboard*, perlu digunakan kode berikut:

```
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
```

Atau bisa juga ditulis:

```
InputStreamReader isr = new InputStreamReader(System.in);  
BufferedReader br = new BufferedReader(isr);
```

Pada tahap ini *objek* br sudah siap digunakan untuk melakukan proses *input*, yaitu dengan melakukan pemanggilan terhadap *method* read() maupun readLine().

Membaca Input Data Karakter

Untuk membaca *input* berupa karakter, digunakan *method* read() yang terdapat pada kelas BufferedReader, dengan pendeklarasian sebagai berikut:

```
int read() throws IOException
```

Membaca Input Data String

Untuk melakukan *input* berupa *string* harus digunakan *method* `readLine()`, bukan `read()`, dengan deklarasi *method* sebagai berikut:

```
String readLine() throws IOException
```

Proses *input* data *string* di dalam Java juga berlaku untuk karakter spasi.

Membaca Input Data Numerik

Untuk *input* berupa data numerik, maka caranya sama dengan melakukan *input* terhadap data *string*. Selanjutnya *string* hasil *input* tersebut dikonversi ke tipe numerik dengan memanggil *method* `parseInt()` yang terdapat pada kelas `Integer` untuk bilangan bulat, atau `parseDouble()` yang terdapat pada kelas `Double` untuk bilangan riil. Parameter yang dilewatkan ke dalam *method* tersebut harus bertipe *string*. Sebaiknya proses konversi berada di dalam blok `try-catch` untuk menghindari masukan *string* yang tidak dapat dikonversi ke bilangan bulat, misalnya *string* “abc”.

Menampilkan Output

Untuk menampilkan *output* ke layar *console*, dapat dengan mudah dilakukan melalui *method* `print()` maupun `println()`. Untuk media *output* lainnya (misalnya: file), *method* yang digunakan untuk melakukan proses ini adalah `write()`.

Selain menggunakan `System.out` untuk menulis ke layar *console*, Java juga menyediakan kelas `PrintWriter` untuk keperluan yang sama. `PrintWriter` adalah salah satu kelas *stream* yang berbasis pada data karakter (*character-based*). Bentuk *constructor* dari kelas `PrintWriter` adalah sebagai berikut:

```
PrintWriter(OutputStream outputStream, Boolean flushOnNewLine)
```

Dimana `outputStream` adalah *objek* dari `OutputStream` dan `flushOnNewLine` adalah parameter Boolean yang menyatakan apakah *stream output* akan dibuang atau tidak setiap kali *method* `println()` dipanggil.

Seperti halnya `System.out`, `PrintWriter` juga mendukung *method* `print()` dan `println()` untuk semua tipe data, termasuk yang berjenis *objek*; cara kerjanya pun sama seperti pada saat menggunakan `System.out`. Apabila argumen yang dilewatkan berupa *objek*, maka `PrintWriter` akan memanggil *method* `toString()` dari *objek*

bersangkutan, kemudian menampilkan hasilnya. Cara pembentukan *objek* `PrintWriter` untuk menampilkan *output* adalah sebagai berikut:

```
PrintWriter pw = new PrintWriter(System.out, true);
```

C. TUGAS PENDAHULUAN

1. Jelaskan fungsi *method-method* berikut ini.
 - a. `int read()`
 - b. `int read(byte[] buffer)`
 - c. `int read(byte[] buffer, int offset, int length)`
2. Jelaskan perbedaan antara 2 *method* berikut:
 - a. `int read(byte[] buffer)`
 - b. `int read(byte[] cbuf)`

D. PERCOBAAN

1. Tulislah program berikut, lakukan kompilasi dan amati hasilnya.

```
import java.io.*;

public class InputKarakter {
    public static void main(String[] args) throws IOException {

        System.out.print("Masukkan sembarang karakter : ");

        char ch;

        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);

        ch = (char) br.read();

        System.out.println("Karakter yang dimasukkan adalah \' " + ch + " \' ");
    }
}
```

2. Tulislah program berikut, lakukan kompilasi dan amati hasilnya.

```
import java.io.*;

public class InputNumerik {
    public static void main(String[] args) throws IOException {
        System.out.print("Masukkan sebuah bilangan bulat: ");

        String temp;
        int bilangan= 0;

        InputStreamReader isr = new InputStreamReader (System.in);
        BufferedReader br = new BufferedReader(isr);

        temp = br.readLine();

        try {
            bilangan = Integer.parseInt(temp);
        } catch (NumberFormatException nfe) {
            System.out.println("Data yang dimasukkan " + "bukan bilangan bulat");
            System.exit(1);
        }

        System.out.println("Bilangan yang dimasukkan " + "adalah " + bilangan);
    }
}
```

3. Eksekusi dan amati program berikut.

```
public class Write {
    public static void main(String[] args) {
        int i = 'A';
        byte b = 65;
        char c = 'B';

        System.out.print (i);
        System.out.print (b);
        System.out.print (c);
    }
}
```

4. Tulislah program berikut, lakukan kompilasi dan amati hasilnya.

```
import java.io.PrintWriter;

public class DemoPrintWriter {
    public static void main(String[] args) {
```

```

PrintWriter pw = new PrintWriter(System.out,true);

double d = 3.1416;
int i = 123;
byte b= 65;
char c = 'A';
String s = "Ini adalah string";

pw.println("Data bertipe double : "+d);
pw.println("Data bertipe int   : "+i);
pw.println("Data bertipe byte  : "+b);
pw.println("Data bertipe char  : "+c);
pw.println("Data bertipe string : "+s);
    }
}

```

5. Buatlah program untuk menerima masukan data dan menampilkan hasilnya sesuai tampilan berikut:

```

Masukkan nama Anda : Putri Kahyangan
Masukkan Alamat Anda : Langit Gg 5 No. 7 Angkasa
Masukkan Nomer telepon: 081234567890
Halo Putri Kahyangan, alamatmu di Langit Gg 5 No. 7 Angkasa
Nomer teleponmu adalah 081234567890

```

E. TUGAS LAPORAN RESMI

1. Jelaskan perbedaan penggunaan dari print(), printf() dan println()! Berikan contoh program yang menunjukkan perbedaan tersebut.
2. Analisalah setiap program di atas.
3. Buatlah program yang menghitung nilai rata-rata seorang mahasiswa dengan *input* nama mahasiswa, jumlah mata kuliah dan nilai. Tampilan *input* dan *outputnya* sebagai berikut:

```

Masukkan Nama : Putri Kahyangan
Jumlah mata kuliah : 4
MK1 : 80
MK2 : 75
MK3 : 90
MK4 : 85
Putri Kahyangan, nilai rata-rata dari 4 mata kuliah yang kamu masukkan
adalah 82.5.

```