

MODUL 7

I/O Stream

Tujuan:

Mahasiswa dapat mengenal dan memahami konsep I/O Stream, serta penerapan I/O Stream dalam konsep OOP

Materi:

- | | |
|-----------------------|-----------------------|
| ✓ Pengantar | ✓ <i>OutputStream</i> |
| ✓ <i>OutputStream</i> | ✓ Soal Latihan |

Referensi:

- ❖ Fikri, Rijalul. 2005. *Pemrograman Java*. Yogyakarta: Penerbit Andi
- ❖ Hermawan, Benny. 2004. *Menguasai Java 2 & Object Oriented Programming*. Yogyakarta: Penerbit Andi

1.1. Pengantar

Ilustrasi 1

Langkah – langkah apa saja yang akan anda lakukan ketika anda disuruh menghapuskan 3 buah kalimat sama persis seperti yang diucapkan oleh dosen, tanpa ada pengurangan maupun penambahan kalimat?

Ilustrasi 2

Menurut anda, apa kelebihan CD-RW/DVD-RW dibandingkan dengan CD-R/DVD-R?

Pada ilustrasi 1, **menulis** pada sebuah kertas dan **membaca** kembali isi pada kertas kemudian dihapuskan merupakan salah satu cara termudah dalam meniru perkataan dosen. Sedangkan pada ilustrasi 2, penggunaan CD-RW/DVD-RW merupakan salah satu penyimpanan yang lebih baik dibandingkan dengan CD-R/DVD-R. Hal ini dikarenakan pada CD-RW dapat melakukan proses **read** dan **write** berulang kali, meskipun antara CD-R dengan CD-RW memiliki ukuran dan kecepatan menulis yang sama. Demikian juga untuk membandingkan antara DVD-R dengan DVD-RW.

Read atau/dan Write yang tampak pada ilustrasi 1 dan ilustrasi 2, merupakan salah satu konsep Java yang biasa dikenal dengan nama I/O Stream. Di dalam Java, penerapan Read menggunakan Class `InputStream`, sedangkan penerapan Write menggunakan Class `OutputStream`. Penggunaan read/write digunakan untuk membantu anda dalam menyimpan hasil keluaran (output) program yang telah anda inputkan ke dalam komputer pada sebuah file. Untuk lebih jelas mengenai I/O Stream, simak penjelasannya di bawah ini.

1.2. *OutputStream*

OutputStream merupakan class induk yang digunakan untuk menangani operasi output. **Class ini merupakan kelas abstrak**, dimana kelas ini tidak dapat digunakan secara langsung ke dalam kelas utama, melainkan harus diturunkan terlebih dahulu ke kelas turunannya. Berikut beberapa class turunan dari class *OutputStream* yang dapat digunakan:

- ✓ *ByteArrayOutputStream*
- ✓ *ObjectOutputStream*
- ✓ *FileOutputStream*
- ✓ *PipedOutputStream*
- ✓ *FilterOutputStream*

Algoritma dalam penulisan data ke dalam file:

- 🔗 Koneksi *OutputStream* ke dalam file
- 📄 Tulis data
- 🔒 Tutup file

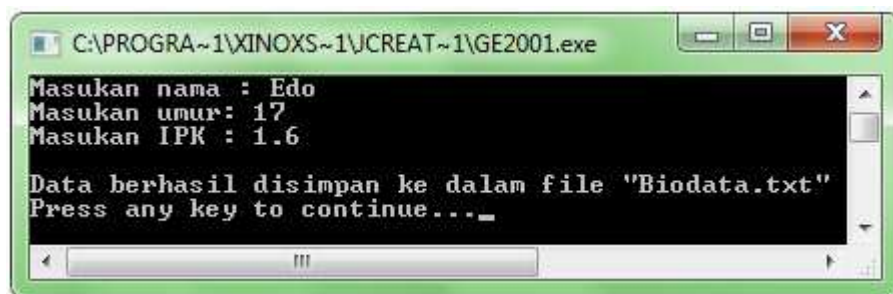
Contoh:

```
1 import java.io.*;
2 class TulisFile
3 {
4     public static void main (String [] args) throws Exception
5     {
6         BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8         //input data
9         System.out.print("Masukan nama : ");
10        String nama = br.readLine();
11        System.out.print("Masukan umur: ");
12        int umur = Integer.parseInt(br.readLine());
13        System.out.print("Masukan IPK : ");
14        double ipk = Double.parseDouble(br.readLine());
15
16        System.out.println();
17
18        //akses FileOutputStream
19
20        //buat file dengan nama "Biodata.txt"
21        FileOutputStream fos = new FileOutputStream("Biodata.txt");
22        //melakukan proses "write" dengan DataOutputStream
23        DataOutputStream dos = new DataOutputStream (fos);
24
25        //tampung data inputan ke dalam file "Biodata.txt"
26        dos.writeUTF(nama);
27        dos.writeInt(umur);
28        dos.writeDouble(ipk);
29
30        //menutup DataOutputStream
31        dos.close();
32
33        //cetak keterangan
34        System.out.println("Data berhasil disimpan ke dalam file \"Biodata.txt\"");
35    }
36 }
```

Keterangan:

- Line 9-14 = inputan user yang ditampung ke dalam variable
- Line 21 = membuat file “Biodata.txt” dengan menggunakan kelas `FileOutputStream`
- Line 23 = memberikan kemampuan kepada file tersebut dalam menulis data (*write*) ke dalam file menggunakan kelas `DataOutputStream`
- Line 26 = memasukkan data yang bertipe **String** ke dalam file “Biodata.txt” menggunakan method **`writeUTF(nama_file)`**
- Line 27 = memasukkan data yang bertipe **Integer** ke dalam file “Biodata.txt” menggunakan method **`writeInt(nama_file)`**
- Line 28 = memasukkan data yang bertipe **Double** ke dalam file “Biodata.txt” menggunakan method **`writeDouble(nama_file)`**
- Line 31 = menutup file “Biodata.txt” sehingga tidak dapat dilakukan proses “write” kembali
- Line 34 = mencetak keterangan

Hasilnya adalah sebagai berikut:



Untuk menambah file yang telah ada isinya, maka diperlukan diperlukan nilai `true` pada parameter `FileOutputStream`, sehingga baris pada class `TulisFile.java`:

```
FileOutputStream fos = new FileOutputStream("Biodata.txt");
```

Diubah menjadi:




```
FileOutputStream fos = new FileOutputStream("Biodata.txt", true);
```

1.3. *InputStream*

InputStream merupakan class induk yang digunakan untuk menangani operasi input. **Class ini merupakan kelas abstrak**, dimana kelas ini tidak dapat digunakan secara langsung ke dalam kelas utama, melainkan harus diturunkan terlebih dahulu ke kelas turunannya. Berikut beberapa class turunan dari class `InputStream` yang dapat digunakan:

- | | |
|-------------------------------------|----------------------------------|
| ✓ <code>ByteArrayInputStream</code> | ✓ <code>ObjectInputStream</code> |
| ✓ <code>FileInputStream</code> | ✓ <code>PipedInputStream</code> |
| ✓ <code>FilterInputStream</code> | |

Algoritma dalam membaca data dalam sebuah file:

-  Koneksi `InputStream` ke dalam file
-  Baca data
-  Tutup file

Contoh:

```
1 import java.io.*;
2 class BacaFile
3 {
4     public static void main (String [] args) throws Exception
5     {
6         BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8         //akses FileInputStream
9
10        //baca file dengan nama "Biodata.txt"
11        FileInputStream fis = new FileInputStream("Biodata.txt");
12        //melakukan proses "read" dengan DataInputStream
13        DataInputStream dis = new DataInputStream (fis);
14
15        //cetak data
16        System.out.println("Masukan nama : "+dis.readUTF());
17        System.out.println("Masukan umur : "+dis.readInt());
18        System.out.println("Masukan IPK : "+dis.readDouble());
19
20        //menutup DataInputStream
21        dis.close();
22    }
23 }
```

Keterangan:

- Line 11 = mencari file “Biodata.txt” dengan menggunakan kelas
FileInputStream
- Line 13 = memberikan kemampuan kepada file tersebut dalam
membaca data (*read*) ke dalam file menggunakan kelas
DataInputStream
- Line 16 = mencetak data yang bertipe **String** pada file
“Biodata.txt” menggunakan method
readUTF(nama_file)
- Line 17 = mencetak data yang bertipe **Integer** pada file
“Biodata.txt” menggunakan method
readInt(nama_file)
- Line 18 = mencetak data yang bertipe **Double** pada file
“Biodata.txt” menggunakan method
readDouble(nama_file)
- Line 20 = menutup file “Biodata.txt” sehingga tidak dapat dilakukan
proses “read” kembali

Hasilnya adalah sebagai berikut:



Lalu bagaimana jika anda ingin membaca data dalam file yang terdapat 2 record atau lebih? Cobalah untuk beresckperimen sendiri... ☺

1.4. Soal Latihan

- Seperti pada soal latihan modul 6, buatlah inputan user untuk memasukkan bilangan 1 dan bilangan 2 pada class Utama.
- Data bilangan 1 dan bilangan 2 kemudian ditampung ke dalam file bernama "latihan7.txt".
- Lakukan pembacaan file tersebut sehingga dapat diketahui hasil penjumlahan, pengurangan, perkalian, dan pembagian.

Jawabannya...

Berikut adalah script kelas utama kalkulator:

```

1 import java.io.*;
2 class Utama
3 {
4     public static void main (String [] args) throws Exception
5     {
6         BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7         //instance of class
8         Kalkulator k = new Kalkulator();
9
10        //input
11        System.out.print("Masukan bilangan pertama : ");
12        double a= Double.parseDouble(br.readLine());
13        System.out.print("Masukan bilangan Kedua : ");
14        double b= Double.parseDouble(br.readLine());
15
16        //akses FileOutputStream
17        //buat file dengan nama "Latihan7.txt"
18        FileOutputStream fos = new FileOutputStream("Latihan7.txt");
19        //melakukan proses "write" dengan DataOutputStream
20        DataOutputStream dos = new DataOutputStream (fos);
21        //tampung data inputan ke dalam file "Latihan7.txt"
22        dos.writeDouble(a);
23        dos.writeDouble(b);
24        //menutup DataOutputStream
25        dos.close();
26
27        //baca file dengan nama "Latihan7.txt"
28        FileInputStream fis = new FileInputStream("Latihan7.txt");
29        //melakukan proses "read" dengan DataInputStream
30        DataInputStream dis = new DataInputStream (fis);
31        //baca data pada file "Latihan7.txt" dan tampung ke dalam variabel
32        double c = dis.readDouble();
33        double d = dis.readDouble();
34        //menutup DataInputStream
35        dis.close();
36
37        k= new Kalkulator (c,d);
38
39        System.out.println();

```

```

40        //output
41        System.out.print ("Hasil Penjumlahan = ");
42        k.Penjumlahan();
43        System.out.print ("Hasil Pengurangan = ");
44        k.Pengurangan();
45        System.out.println("Hasil Perkalian = "+k.Perkalian());
46        System.out.println("Hasil Pembagian = "+k.Pembagian());
47    }
48 }

```