I/O STREAM

Pada program-program yang membutuhkan data-data eksternal, maka diperlukan suatu proses input dan output (I/O), dimana pada Java dukungan proses I/O ini sudah disediakan dalam paket java.io. 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*, socket 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.

BYTE STREAMS

Byte streams merupakan kelas interface yang digunakan untuk menangani data biner. Semua kelas byte stream merupakan turunan InputStream dan OutputStream. Berikut ini merupakan FileInputStream dan FileOutputStream dengan contoh program bernama CopyBytes, dimana menggunakan byte stream untuk menyalin xanadu.txt.

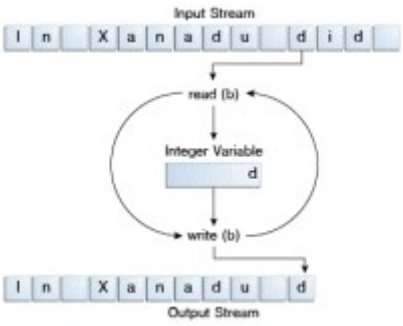
```
public class CopyBytes {
    public static void main(String[] args) throws IOException{
        FileInputStream in = null;
        FileOutputStream out = null;

        try {
            in = new FileInputStream("xanadu.txt");
            out = new FileOutputStream("outagain.txt");
            int c;

            while ((c = in.read()) != -1){
                out.write(c);
            }
        } finally{
            if (in != null){
                in.close();
            }
            if (out != null){
                out.close();
            }
        }
    }
}
```

Gambar 1. ***Copy Bytes***

CopyBytes menghabiskan sebagian besar waktu pada simple loop yang membaca input stream dan menulis output stream, satu byte pada satu waktu, seperti pada Gambar 2.



Gambar 2. Simple by stream input dan output

CHARACTER STREAMS

Character stream merupakan kelompok kelas yang digunakan untuk menangani proses baca tulis karakter Unicode. Karakter stream I/O secara otomatis menerjemahkan format internal ke dan dari lokal karakter set. Semua karakter stream merupakan turunan dari Reader dan Writer, seperti dengan byte, terdapat kelas karakter stream I/O yaitu FileReader dan FileWriter yang diilustrasikan pada Gambar 3.

```
public class CopyCharacter {
    public static void main(String[] args) throws IOException{
        FileReader inputStream = null;
        FileWriter outputStream = null;

        try{
            inputStream = new FileReader("xanadu.txt");
            outputStream = new FileWriter("characteroutput.txt");

            int c;
            while ((c = inputStream.read()) != -1){
                outputStream.write(c);
            }
        }
        finally{
            if(inputStream != null){
                inputStream.close();
            }
            if (outputStream != null){
                outputStream.close();
            }
        }
    }
}
```

Gambar 3. Copy Character

CopyCharacter sangat mirip dengan CopyBytes. Hal terpenting yang membedakannya adalah CopyCharacter menggunakan FileReader dan FileWriter untuk input dan output pada tempat FileInputStream dan FileOutputStream.

LINE ORIENTED I/O

Karakter I/O biasanya terjadi pada unit yang lebih besar daripada single character. Satu unit umum adalah *line*: sebuah string karakter dengan terminator garis di bagian akhir. Terminator garis bisa menjadi *carriage-return/line-feed sequence* (“\r\n”), *single carriage-return* (“\r”), atau *single line-feed* (“\n”). Untuk mendukung semua kemungkinan terminator garis mengikuti program untuk membaca file teks yang dibuat pada sistem operasi, kita dapat memodifikasi contoh CopyCharacters menggunakan *line-oriented* I/O. Untuk melakukannya maka

dibutuhkan dua kelas, yaitu: BufferedReader dan PrintWriter. Gambar 4 merupakan contoh pemanggilan BufferReader.readline oleh CopyLines dan PrintWrite.println untuk melakukan input dan output satu baris satu waktu.

```
public class CopyLines {
    public static void main(String[] args) throws IOException{
        BufferedReader inputStream = null;
        PrintWriter outputStream = null;
        try{
            inputStream = new BufferedReader(new FileReader("xanadu.txt"));
            outputStream = new PrintWriter(new FileWriter("characteroutput.txt"));
            String l;
            while ((l = inputStream.readLine()) !=null){
                outputStream.println(l);
            }

        }
        finally{
            if(inputStream != null){
                inputStream.close();
            }
            if (outputStream != null){
                outputStream.close();
            }
        }
    }
}
```

Gambar 4. ***Copy Lines***

Untuk praktikum selanjutnya, *sourcecode* yang diberikan sengaja memiliki kesalahan (*compile error* dan *logic error*). Jika terdapat tanda Warning, silahkan menemukan error pada program terlebih dahulu sebelum dikompilasi.

READ CHARACTER FROM THE CONSOLE

Program pada Gambar 5 mendemonstrasikan penggunaan *method* read() untuk membaca karakter dari *keyboard* sampai *user* mengetikkan karakter ‘q’.

```
class ReadCharacters {
    public static void main(String[] args) throws IOException {
        char c;
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter character, 'q' to quit.");
        do {
            c = (char) br.read();
            System.out.println(c);
        } while (c != 'q');
    }
}
```

Gambar 5. Read Character

READ STRING FROM CONSOLE INPUT

[*Warning! Logic error here*] Program berikut ini mendemonstrasikan penggunaan kelas *method* readLine() dari kelas BufferReader untuk membaca *input* berupa teks dari *user* dan menampilkan teks tersebut di layar, sampai *user* mengetikkan "stop".

```
class ReadStrings {
    public static void main(String[] args) {
        String str;
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter lines of text. ");
        System.out.println("Enter 'stop' to quit. ");
        try {
            do {
                str = br.readLine();
                System.out.println(str);
            } while (str.equals("stop"));
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
        finally{
            try {
                if (br!=null) br.close();
            } catch (IOException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

Gambar 6. ***Read String***

FILE INPUT STREAM

Program berikut ini mendemonstrasikan bagaimana membaca *single byte, array of byte,* dan *sub range array of byte.* Program ini juga menunjukkan cara menggunakan *method* available() untuk mengetahui jumlah *byte* yang tersisa dan bagaimana menggunakan *method* skip() untuk melompat (tidak membaca) sejumlah *byte* yang tidak diinginkan.

```
public class FileInputStreamDemo {
    public static void main(String[] args) throws FileNotFoundException, IOException {
        int size;
        InputStream f = new FileInputStream("FileInputStreamDemo.java");
        System.out.println("Total Available Bytes : " + (size = f.available()));
        int n = size / 40;
        System.out.println("First " + n + " bytes of the file one read() at a time");
        for (int i = 0; i < n; i++) {
            System.out.println((char) f.read());
        }
        System.out.println("\nStill Available: " + f.available());
        System.out.println("Reading the next " + n + "with one read(b[])");
        byte b[] = new byte[n];
        if (f.read(b) != n) {
            System.err.println("couldn't read" + n + "bytes.");
        }
        System.out.println(new String(b, 0, n));
        System.out.println("\nStill Available: " + (size = f.available()));
        System.out.println("Skipping half of remaining bytes with skip()");
        f.skip(size / 2);
        System.out.println("Still Available: " + f.available());
        System.out.println("Reading " + n / 2 + "into the end of array");
        if (f.read(b, n / 2, n / 2) != n / 2) {
            System.err.println("couldn't read" + n / 2 + "bytes.");
        }
        System.out.println(new String(b, 0, b.length));
        System.out.println("\nStill Available: " + f.available());
        f.close();
    }
}
```

Gambar 7. File Input Stream

Pada Gambar 7 program membaca *file* yang berada pada direktori yang sama dengan program.

READING FILE

Program menggunakan *method* read() untuk membaca isi *file* satu karakter setiap waktunya, setiap karakter yang dibaca ditampilkan pada monitor. Nama *file* yang akan dibaca oleh program diperoleh dari *command line argument.*

```
public class ShowFile {
    public static void main(String[] args) throws IOException {
        int i;
        FileInputStream in;
        try {
            in = new FileInputStream (args[0]);
        } catch (FileNotFoundException e) {
            System.out.println("File not found");
            return;
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Usage: Showfile file");
            return;
        }
        do {
            i = in.read();
            if (i!=-1) System.out.println((char) i );
        } while (i!=-1);
        in.close();
    }
}
```

Gambar 8. Show File

WRITING FILE

Program menggunakan *method* write() untuk menyalin isi suatu *file* ke dalam *file* lain.

```
public class CopyFile {
    public static void main(String[] args) throws IOException {
        int i;
        FileReader fin;
        FileWriter fout;
        try {
            fin = new FileReader(args[0]);
            fout = new FileWriter(args[1]);
        } catch (FileNotFoundException e) {
            System.out.println("Error opening output file.");
            return;
        } catch (IndexOutOfBoundsException e) {
            System.out.println("Usage: CopyFile from .. to ..");
            return;
        }
        try {
            do {
                i = fin.read();
                if (i != 1) {
                    fout.write(i);
                }
            } while (i != 1);
        } catch (IOException e) {
            System.out.println("File error.");
        } finally {
            try {
                if (fin != null) {
                    fin.close();
                }

                if (fout != null) {
                    fout.close();
                }
            } catch (IOException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

Gambar 9. *Copy File*

TUGAS

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

   Input:

   ```
   Masukan Nama Anda : Putri Kahyangan
   Masukan Alamat Anda : Langit Gg 5 No. 7 Angkasa
   Masukann Nomor Telepon : 08211345678
   ```

   Output:

   ```
   Halo Putri Kahyangan, alamatmu di Langit Gg 5 No. & Angkasa
   Nomor teleponmu adalah 08211345678
   ```

2. Buatlah program yang menghitung nilai rata-rata seorang mahasiswa dengan input nama mahasiswa, jumlah mata kuliah dan nilai. Tampilan input dan outputnya sebagai berikut:

   Input:

   ```
   Masukkan Nama : Putri Kahyangan
   Jumlah mata kuliah : 4
   MK1 : 80
   MK2 : 75
   MK3 : 90
   MK4 : 85
   ```

   Output:

   ```
   Putri Kahyangan, nilai rata-rata dari 4 mata kuliah yang kamu masukkan adalah 82.5.
   ```