PRAKTIKUM 8
FILE INPUT OUTPUT

A. TUJUAN

1. Memahami cara memasukkan *Input* dan menampilkan *Output*
2. Memahami dasar membaca dan menulis file

B. DASAR TEORI

Dasar Baca / Tulis File

Terdapat dua buah *stream* yang sering digunakan untuk melakukan proses pembacaan/penulisan data dari/ke dalam file, yaitu FileInputStream (untuk membaca data) dan FileOutputStream (untuk menulis data). Keduanya akan membentuk *stream byte* yang terhubung ke sebuah file. Untuk membuka file, harus membentuk *objek* dari salah satu kelas *stream* tersebut dengan menyertakan nama file sebagai *argument* pada *constructor*nya. *Constructor* dari kedua kelas tersebut di*overload* menjadi beberapa *constructor* sebagai berikut:

```
FileInputStream(String fileName) throws FileNotFoundException
FileOutputStream(String fileName) throws FileNotFoundException
```

Dalam hal ini fileName adalah nama file yang akan dibuka. Bila file tidak ditemukan pada saat menggunakan *stream input*, maka kedua *constructor* di atas akan membangkitkan eksepsi FileNotFoundException, sedangkan eksepsi saat menggunakan *stream output* akan muncul bila file *output* tidak dapat terbentuk/terbuat. Apabila terdapat file dengan nama yang sama pada direktori tempat file dibuat/dibuka, maka file lama akan ditumpuki.

Setelah selesai menggunakan *stream* yang terhubung dengan file, maka *stream* tersebut harus ditutup dengan menggunakan *method* close() dengan bentuk umum deklarasi sebagai berikut:

```
void close() throws IOException
```

Untuk membaca data dari file, perlu memanggil *method* read(). Setiap kali *method* tersebut dipanggil, maka program akan membaca *byte* tunggal yang terdapat

dalam file dan mengembalikan nilai *byte* tersebut dalam bentuk nilai *integer*. Apabila data terakhir dari file yang dibaca (*end-of-file*, EOF) telah ditemukan, maka *method* read() akan menghasilkan nilai -1. *Method* tersebut juga dapat membangkitkan eksepsi IOException apabila terdapat kegagalan pada proses pembacaan datanya.

Proses penulisan data ke dalam file menggunakan *method* write(). Karena System.out adalah *objek* dari tipe PrintStream, dan PrintStream itu sendiri merupakan turunan dari kelas OutputStream, maka System.out dapat menggunakan *method* write() yang sebenarnya didefinisikan pada kelas OutputStream. Bentuk umum dari *method* write() yang telah di *override* oleh kelas PrintWriter adalah sebagai berikut:

```
void write(int nilaiByte)
```

Data yang dituliskan harus dalam bentuk *integer*, sehingga harus diyakinkan dulu bahwa data telah dikonversi ke tipe int dulu sebelum memasukkan data tersebut ke dalam *stream*.

Kelas File

Kelas File dalam paket java.io tidak beroperasi dengan menggunakan *stream*, tetapi terhubung langsung dengan file dan sistem file yang ada, sehingga *objek* dari kelas File digunakan untuk memperoleh dan memanipulasi informasi yang berkaitan dengan file, seperti hak akses (*permission*), waktu dan tanggal pembuatan atau modifikasi, lokasi direktori yang ditempatinya, dan sebagainya.

Beberapa *constructor* yang dapat digunakan untuk membuat *objek* dari kelas File dan contoh kodenya sebagai berikut:

```
File(String path) → File file1 = new File("/java");
File(String path, String namaFile)
                → File file2 = new File("/java", "contoh.java");
File(File objFile, String namaFile) → File file3 = new File(file1, "contoh.java");
```

dengan path adalah lokasi tempat file berada dan namaFile adalah nama dari file yang akan diakses, sedangkan objFile adalah *objek* dari kelas File yang akan digunakan untuk menunjukkan direktori dimana file berada.

Beberapa *method* dalam kelas File yang digunakan untuk memanipulasi file adalah sebagai berikut:

- Memperoleh informasi file → sesuai Tabel 8.1

Tabel 8.1 Daftar method untuk memperoleh informasi file

Nama Method	Keterangan
Exist()	Mengembalikan nilai true apabila file ada
getCanonicalpath()	Mengembalikan nama lengkap
getName()	Mengembalikan nama file relatif
getParent()	Mengembalikan directory yang ditempatinya
canRead()	Mengembalikan nilai true bila file dapat dibaca
canWrite()	Mengembalikan nilai true bila file dapat ditulis
lasModified()	Mengembalikan waktu modifikasi yang dilakukan terhadap file
length()	Mengembalikan ukuran file
isFile()	Mengembalikan nilai true bila file yang diakses oleh objek File berupa file (bukan direktori)
isDirectory	Mengembalikan nilai true bila file yang diakses oleh objek File berupa direktori

- Membuat file → createNewFile()
- Mengubah nama file → renameTo()
- Menghapus file → delete(), yang mengembalikan nilai true bila proses penghapusan berhasil dan false bila gagal.
- Menampilkan daftar file dan direktori →list(), daftar file dan direktori yang diperoleh akan disimpan dalam variabel bertipe array dari tipe string.
- Membuat direktori baru → mkdirs()

C. TUGAS PENDAHULUAN

1. Jelaskan perbedaan pemakaian di dalam program untuk konstruktor berikut
    a. File(String path)
    b. File(String path, String namaFile)
    c. File(File objFile, String namaFile)

2. RandomAccessFile adalah kelas yang memungkinkan membaca dan menulis data tanpa melalui pembacaan secara sekuensial. Kelas ini memiliki konstruktor dengan bentuk sebagai berikut:

   RandomAccessFile(String nama, String mode)

   RandomAccessFile(File file, String mode)

   Jelaskan pemakaian konstruktor-konstruktor tersebut!

D. PERCOBAAN

1. Eksekusi program berikut, dan analisalah.

```
import java.io.*;

public class DemoBacaFile {
   public static void main(String() args) {
      FileInputStream finput = null;
      int data;

      try {
         finput = new FileInputStream("D:/file1.txt");
      }
      catch (FileNotFoundException fnfe) {
         System.out.println("File tidak ditemukan.");
         return;
      }

      try {
         while ((data = finput.read())!= -1) {
            System.out.println((char)data);
         }
      }
      catch (IOException ioe) {
         System.out.println(ioe.getMessage());
         return;
      }

      try {
         finput.close();
      }
      catch(IOException ioe){}
   }
}
```

2. Eksekusi program berikut, dan analisalah.

```
import java.io.*;

public class DemoTulisFile {
   public static void main(String() args) {

      FileOutputStream foutput = null;
      String data = "Baris pertama \nBaris kedua \nBaris ketiga";

      try {
         foutput = new FileOutputStream("d:/output.txt");
      }
      catch (FileNotFoundException fnfe) {
         System.out.println("File tidak dapat terbentuk.");
         return;
      }

      try {
         for (int i=0; i<data.length(); i++) {
            foutput.write((int)data.charAt(i));
         }
      }
      catch (IOException ioe) {
         System.out.println(ioe.getMessage());
         return;
      }

      try {
         foutput.close();
      }
      catch (IOException ioe) {}
   }
}
```

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

```
import java.io.*;

public class DemoSalinFile {
   public static void main(String() args) {
      FileInputStream finput = null;
      FileOutputStream foutput = null;

      int data;

      try{
```

```
            finput = new FileInputStream("d:/file1.txt");
        }
        catch (FileNotFoundException fnfe) {
            System.out.println("File input tidak ditemukan");
            return;
        }

        try{
            foutput = new FileOutputStream("d:/file2.txt");
        }
        catch (FileNotFoundException fnfe) {
            System.out.println("File output tidak dapat terbentuk");
            return;
        }

        try {
            while ((data = finput.read()) != -1) {
                foutput.write(data);
            }
        }
        catch (IOException ioe) {
            System.out.println(ioe.getMessage());
            return;
        }

        try {
            finput.close();
            foutput.close();
        }
        catch (IOException ioe) {}
    }
}
```

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

```
import java.io.File;

public class DemoListDirektori {
    public static void main(String() args) {
        String dir = "d:/hanif";
        File f = new File(dir);

        String() daftar = f.list();
        java.util.Arrays.sort(daftar);

        System.out.println("File dan direktori dalam D:\\hanif");
        System.out.println();
```

```
        for(int i=0; i<daftar.length; i++) {
            File fTemp = new File(dir + "/" + daftar(i));
            if (fTemp.isDirectory()) {
                System.out.println(daftar(i) + "\t\t<DIR>");
            }
            else {
                System.out.println(daftar(i));
            }
        }
    }
}
```

5. Eksekusi program berikut, dan analisalah.

```
import java.io.*;

public class AksesRandom {
    public static void main(String() args) throws IOException {
        RandomAccessFile berkas = null;
        berkas = new RandomAccessFile("abc.dat", "rw");
        berkas.writeBytes("ABCDEFGHIJKLMNOPQRSTUVW");
        char kar = ' ';

        berkas.seek(0);
        System.out.println("isi berkas: ");
        while (berkas.getFilePointer()<berkas.length()) {
            kar = (char) berkas.readByte();
            System.out.print(kar);
        }

        System.out.println();

        berkas.seek(3);
        berkas.writeByte((int) 'Z');

        System.out.println("Sesudah penggantian");

        berkas.seek(0);
        System.out.println("isi berkas ");
        while (berkas.getFilePointer()<berkas.length()) {
            kar = (char) berkas.readByte();
            System.out.print(kar);
        }
    }
}
```

6. Buat program yang mengubah urutan string dari belakang. Misalnya, bila diinputkan ABCDEFGHIJ maka outputnya JIHGFEDCBA.

E. TUGAS LAPORAN RESMI

1. Buatlah program untuk membuat suatu file baru pada disk.
2. Buatlah program untuk menghapus file3.txt yang ada pada suatu disk.
3. Buatlah program untuk membuat sebuah direktori baru.
4. Dari percobaan yang telah dilakukan, buat analisa program-program tersebut.