MODUL PRAKTIKUM

I/O Stream Lanjutan

Versi 1.3

JARDIKNAS
Indonesian Education Network

1. Tujuan

- Tahu tipe-tipe stream yang umum digunakan
- Menggunakan class File dan methodnya
  - Karakter dan Stream byte
  - Input dan Output Streams
  - Node dan Filter Streams
- Menggunakan class-class Input/Output yang berbeda
  - *Reader*
  - *Writer*
  - *InputStream*
  - *OutputStream*
- Memahami konsep dari stream chaining
- Mendefinisikan serialisasi
- Memahami penggunaan dari kata kunci *transient*
- Menulis dan membaca dari sebuah object stream

2. Latar Belakang

Dalam module sebelumnya, Anda telah mempelajari bagaimana untuk mendapatkan input user dan memanipulasi file-file menggunakan strea. Kini Anda akan mempelajari lebih banyak tentang stream dan class-class stream yang lain.

Tipe-Tipe Stream yang Umum Digunakan

Stream Karakter dan Byte

Seperti yang elah disebutkan sebelumnya, secara umum ada dua tipe dari stream, yaitu stream karakter dan byte. Kita hanya mengulang perbedaan mendasar antara keduanya. Stream byte adalah abstraksi file atau alat untuk data biner sedangkan stream karakter adalah untuk karakter Unicode.

Class *InputStream* adalah abstraksi class root untuk semua input stream byte sedangkan class *OutputStream* adalah class root abstraksi dari semua output stream byte. Untuk stream karakter, superclasss yang sesuai dari semua class-class secara berturut-turut
adalah class *Reader* dan the *Writer.* Kedua class-class ini adalah abstraksi class-class untuk membaca dan menulis stream karakter.

Input dan Output Stream

Stream juga dikategorikan berdasarkan apakah mereka digunakan untuk membaca atau menulis stream.Walaupun ini sudah cukup nyata, izinkan saya untuk mendefinisikan tipe stream ini. Anda diijinan untuk membaca dari input stream tapi tidak menulisnya. Di lain pihak, Anda diijinkan untuk menulis output streams tapi tidak membacanya. Class *InputStream* dan class *Reader* adalah superclass-superclass dari semua input stream. Class *OutputStream* dan class *Writer* adalah class-class root dari semua output stream. Input stream

juga dikenal sebagai stream sumber (source stream) sejak kita memperoleh informasi dari stream ini. sementara itu output stream disebut juga stream hasil(sink stream).

Node dan Stream Filter

Kini package *java.io* membedakan antara node dan stream filter. Sebuah stream node adalah sebuah stream dengan fungsi dasar berupa fungsi membaca atau menulis dari sebuah lokasi khusus seperti pada disk atau dari jaringan. Tipe-tipe dari stream node terdiri atas file, memory dan jalur data. Stream filter, di lain pihak, diletakkan pada layer stream node diantara threads atau proses untuk menyediakan fungsi tambahan yang tidak dapat ditemukan dalam stream node oleh stream node itu sendiri. Penambahan lapisan pada sebuah stream node disebut dengan stream chaining. Sesi ini berturut-turut mempunyai sebuah tujuan dariclass-class stream yang berbeda. Untuk melengkapi daftar dari class-class ini, silahkan melihat dokumentasi Java's API.

3. Percobaan

Percobaan 1 Informasi File atau Folder:

```
import java.io.*;

public class IO01 {

	public static void main(String args[]) {

		String fileName = args[0];

		File fn = new File(fileName);

		System.out.println("Name: " + fn.getName());

		if (!fn.exists()) {

			System.out.println(fileName + " does not exists.");

			/* membuat sebuah temporary directory . */

			System.out.println("Creating temp directory...");

			fileName = "temp";

			fn = new File(fileName);

			fn.mkdir();

			System.out.println(fileName + (fn.exists()? "exists": "does
			not exist"));
```

```
                System.out.println("Deleting temp directory...");

                fn.delete();

                System.out.println(fileName + (fn.exists()? "exists": "does
                not exist"));

                return;

          }

          System.out.println(fileName + " is a " + (fn.isFile()? "file."
    :"directory."));

          if (fn.isDirectory()) {

                String content[] = fn.list();

                System.out.println("The content of this directory:");

                for (int i = 0; i < content.length; i++) {

                      System.out.println(content[i]);

                }

          }

          if (!fn.canRead()) {

                System.out.println(fileName + " is not readable.");

                return;

          }

          System.out.println(fileName + " is " + fn.length() + " bytes
          long.");

          System.out.println(fileName + " is " + fn.lastModified() + "
          bytes long.");

          if (!fn.canWrite()) {

                System.out.println(fileName + " is not writable.");

          }

    }

}
```

Percobaan 2 File Terkopi:

```
import java.io.*;
class IO02 {
	void copy(String input, String output) {
		FileReader reader;
		FileWriter writer;
		int data;
		try {
			reader = new FileReader(input);
			writer = new FileWriter(output);
			while ((data = reader.read()) != -1) {
				writer.write(data);
			}
			reader.close();
			writer.close();
		} catch (IOException ie) {
			ie.printStackTrace();
		}
	}
	public static void main(String args[]) {
		String inputFile = args[0];
		String outputFile = args[1];
		CopyFile cf = new CopyFile();
		cf.copy(inputFile, outputFile);
	}
}
```

Percobaan 3 File Terkopi :

```
import java.io.*;
class IO03 {
	void copy(String input, String output) {
		BufferedReader reader; BufferedWriter writer; String data;
		try {
			reader = new BufferedReader(new FileReader(input));
			writer = new BufferedWriter(new FileWriter(output));
			while ((data = reader.readLine()) != null) {
				writer.write(data, 0, data.length());
			}
			reader.close();
			writer.close();
		} catch (IOException ie) {
			ie.printStackTrace();
		}
	}
	public static void main(String args[]) {
		String inputFile = args[0];
		String outputFile = args[1];
		CopyFile cf = new CopyFile();
		cf.copy(inputFile, outputFile);
	}
}
```

Percobaan 4 File Terkopi:

```
import java.io.*;
class I004 {
	void copy(String input, String output) {
		FileInputStream inputStr;
		FileOutputStream outputStr;
		int data;
		try {
			inputStr = new FileInputStream(input);
			outputStr = new FileOutputStream(output);
			while ((data = inputStr.read()) != -1) {
				outputStr.write(data);
			}
			inputStr.close();
			outputStr.close();
		} catch (IOException ie) {
			ie.printStackTrace();
		}
	}
	public static void main(String args[]) {
		String inputFile = args[0];
		String outputFile = args[1];
		CopyFile cf = new CopyFile();
		cf.copy(inputFile, outputFile);
	} }
```

Percobaan 5 Character Dari File tercetak:

```
import java.io.*;
class CopyFile {
      void copy(String input) {
            PushbackInputStream inputStr;
            PrintStream outputStr;
            int data;
            try {
                  inputStr = new PushbackInputStream(new
                  FileInputStream(input));
                  outputStr = new PrintStream(System.out);
                  while ((data = inputStr.read()) != -1) {
                        outputStr.println("read data: " + (char) data);
                        inputStr.unread(data);
                        data = inputStr.read();
                        outputStr.println("unread data: " + (char) data);
                  }
                  inputStr.close();
                  outputStr.close();
            } catch (IOException ie) {
                  ie.printStackTrace();
            }
      }
      public static void main(String args[]) {
            String inputFile = args[0];
            CopyFile cf = new CopyFile();
            cf.copy(inputFile);
```

```
    }
}
```

Percobaan 6 Terbentuk File Boolean.ser:

```
import java.io.*;
public class SerializeBoolean {
    SerializeBoolean() {
        Boolean booleanData = new Boolean("true");
        try {
            FileOutputStream fos = new
            FileOutputStream("boolean.ser");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(booleanData);
            oos.close();
        } catch (IOException ie) {
            ie.printStackTrace();
        }
    }
    public static void main(String args[]) {
        SerializeBoolean sb = new SerializeBoolean();
    }
}
```

Percobaan 7 Unserialized Boolean:

```
import java.io.*;

public class UnserializeBoolean {

	UnserializeBoolean() {

		Boolean booleanData = null;

		try {

			FileInputStream fis = new

			FileInputStream("boolean.ser");

			ObjectInputStream ois = new ObjectInputStream(fis);

			booleanData = (Boolean) ois.readObject();

			ois.close();

		} catch (Exception e) {

			e.printStackTrace();

		}

		System.out.println("Unserialized Boolean from " + "boolean.ser");

		System.out.println("Boolean data: " + booleanData);

		System.out.println("Compare data with true: " +
		booleanData.equals(new Boolean("true")));

	}

	public static void main(String args[]) {

		UnserializeBoolean usb = new UnserializeBoolean();

	}

}
```

4. Latihan

4.1.Enkripsi Sederhana

Baca dari sebuah file khusus oleh user dan encrypt isi file menggunakan teknik penggeseran yang sederhana. Juga, tanyakan pada user untuk menginput ukuran pergeseran. Output dari pesan yang telah di encrypt pada file yang lain yang memiliki nama yang juga dibuat oleh user sendiri.

Sebagai contoh,
Ukuran pergeseran: 1
Pesan yang dibaca dari file: Hello
Pesan ter-encrypt: Ifmmp