Input Stream dan Output Stream

Yup kita telah belajar tentang input output sederhana. Kali ini kita belajar akan proses input output yang agak sedikit kompleks dengan menggunakan package java.io. Package java.io berisi komponen yang dapat kita gunakan untuk proses input dan output (I/O) di dalam Java. Input adalah proses untuk membaca dari berkas ke dalam program, sedangkan output adalah proses untuk menuliskan ke berkas.

Sebelum terjun lebih jauh kita harus mengetahui istilah stream terlebih dahulu. Stream adalah proses secara sekuensial elemen data dari waktu ke waktu. Stream bisa diibaratkan seperti barang-barang yang diproses satu per satu pada ban berjalan (*conveyor belt*).

Kenapa stream? Karena proses input output akan dijalankan secara stream, di mana prosesnya akan berjalan secara sekuensial.

Di Java ada 2 macam stream yaitu inputStream dan outputStream.

1. InputStream, digunakan membaca data dari sumber.
2. OutputStream, digunakan untuk menuliskan data ke suatu destination (target sumber).

Ada beberapa io stream yang akan kita pelajari seperti byte streams, character streams, standar streams, dan lain-lain.

Byte Streams

Byte streams digunakan untuk proses input output dengan ukuran 8-bit bytes. Contoh komponen byte stream adalah `FileInputStream` dan `FileOutputStream`. Ada banyak komponen byte streams lainnya tapi kurang lebih penggunaanya sama.

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

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

            while ((c = in.read()) != -1) {
                out.write(c);
            }
        } catch (IOException e) {
            e.printStackTrace();
```

Di dalam block finally kita tidak boleh lupa untuk menutup stream. Lupa untuk menutup stream akan menyebabkan kebocoran pada resource (resource leaks).

Character Streams

Java menggunakan Unicode conventions untuk menyimpan data characternya. Character stream digunakan untuk memproses input output dari 16-bit unicode. Ada banyak komponen character streams tapi yang sering digunakan adalah `FileReader` dan `FileWriter`. Contoh kodenya seperti ini.

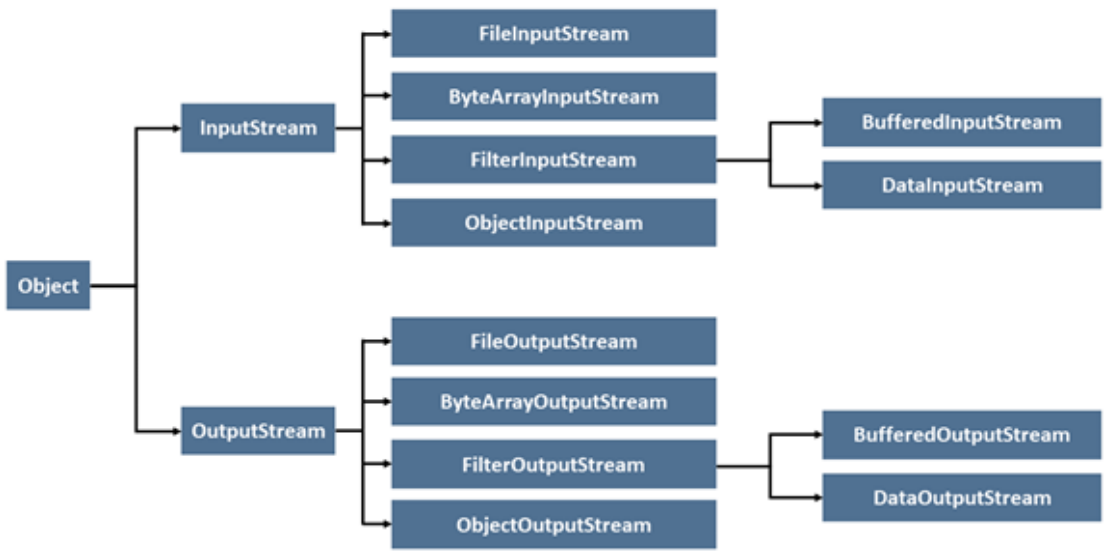
```java
package com.dicoding.javafundamental.inputoutput;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        FileReader in = null;
        FileWriter out = null;

        try {
            in = new FileReader("latihan_input.txt");
            out = new FileWriter("latihan_ouput.txt");
            int c;
```

Penggunaan komponen byte streams `FileInputStream` dan `FileOutputStream` tidak jauh beda. Sebabnya, di dalam `FileReader` dan `FileWriter` terdapat komponen byte streams tersebut. Yang membedakan adalah `FileReader` dan `FileWriter` dapat membaca data 2 bytes dalam satu waktu.

Lalu apakah kita harus menggunakan byte streams atau character streams? Byte streams termasuk low-level I/O, jadi gunakanlah ketika ingin memproses data primitive. Ketika data yang ingin kita proses memiliki character dengan Unicode conventions maka sebaiknya gunakan character streams.

Hierarki Input Output Streams

Hierarki dari kelas input dan output streams bisa dilihat pada gambar di bawah ini.



Bisa dilihat bahwa ada banyak macam kelas streams yang bisa kita gunakan di Java. Tidak perlu belajar semua komponen yang ada. Akan tetapi yang perlu kita pahami adalah perbedaan antara `InputStream` dan `OutputStream`.

File Navigation

Navigation input output tidak kalah penting dengan proses input output. Kita juga harus memahami directories di dalam Java. Directories dalam Java adalah File yang dapat memuat daftar berkas dan direktori. Kita menggunakan objek File untuk membuat directories lalu untuk menampilkan daftar berkas yang tersedia di dalam direktori tertentu.

Untuk navigasi, kita bisa menggunakan `objekFile` yang bisa menampilkan daftar berkas dan direktori. Dengan menggunakan objek File, kita bisa membuat directory dengan menggunakan fungsi `mkdir()` atau `mkdirs`.

1. `mkdir()`, metode untuk membuat directory. Nilainya true ketika sukses dan false ketika gagal. Gagal bisa disebabkan oleh path directory yang sudah yang ada, atau karena keseluruhan path nya tidak ada.
2. `mkdirs()`, metode yang digunakan untuk membuat directory dan parent directory-nya.

Misalnya seperti ini untuk membuat suatu directory.

```java
package com.dicoding.javafundamental.inputoutput;

import java.io.File;

public class Main {
    public static void main(String[] args) {
        String dirname = "/java/latihan1";
        File file = new File(dirname);

        // Buat directory
        file.mkdirs();
    }
}
```

Karena menggunakan `mkdirs`, kode di atas akan membuat directory `latihan1` dan parent directory java.

Kemudian kita bisa menampilkan list file dari directories dengan memanggil metode `list()`. Misalnya seperti ini.

```java
package com.dicoding.javafundamental.inputoutput;

import java.io.File;

public class Main {
    public static void main(String args[]) {
        String dirname = "/java/latihan1";
        File file = null;
        String[] paths;

        try {
            // Instansiasi objek File
            file = new File(dirname);

            // Ambil list files dan masukkan ke string paths
```

Dengan memanfaatkan `mkdir` dan `list`, kita bisa membuat program yang dapat melakukan navigasi ke directories di dalam storage.

Sebelumnya

Selanjutnya