Bab 5
Aplikasi Berbasis Teks

5.1 Tujuan

Pembahasan kali ini akan menitikberatkan pada bahasan penggunaan argument command-line. Selebihnya, Anda akan mempelajari mengenai penggunaan streams untuk mendapatkan nilai input dari user pada saat runtime, sekaligus dalam proses manipulasi file.

Setelah menyelesaikan pembahasan ini, Anda diharapkan dapat :

1. Mendapatkan input dari comman-line
2. Mengetahui cara untuk memanipulasi properties dari sistem
3. Membaca standart input
4. Membaca dan menulis file

5.2 Argument Command-Line dan System Properties

Seperti yang telah Anda ketahui pada pembahasan sebelumnya, JAVA mengijinkan user untuk memasukkan data dari command-line. Sebagai contoh, untuk meneruskan argument 1 dan 2 kepada program Java bernama Calculate, anda dapat menuliskan baris berikut pada command prompt

```
java Calculate 1 2
```

Pada contoh berikut ini, data 1 disimpan pada variabel *args[0]*, begitu pula dengan data 2 yang disimpan pada *args[1]*. Sehingga, tujuan dari deklarasi *String args[]* sebagai sebuah parameter pada *method* utama menjadi jelas.

Selain melewatkan argument menuju *method* utama, Anda juga dapat memanipulasi system properties dari command-line.

System properties hampir menyamai environment variables, namun tidak memiliki ketergantungan pada spesifikasi platform yang digunakan. Sebuah property secara sederhana berupa pemetaan antara property name dan value yang dimilikinya. Hal ini ditunjukkan pada Java dalam class Properties. Class System menyediakan sebuah method untuk menentukan system properties yang digunakan, method *getProperties* yang menghasilkan sebuah object *Properties.* Class yang sama juga menyediakan method *getProperty* yang memiliki dua buah bentuk.

public static String getProperty(String key)
Bentuk ini menghasilkan nilai String dari System Properties yang ditunjukkan oleh key yang ditentukan. Jika hasil menunjukkan nilai null, berarti tidak terdapat property dengan key yang ditentukan.
public static String getProperty(String key, String def)
Bentuk ini juga menghasilkan nilai String dari System Properties sesuai key yang ditentukan. Akan menghasilkan nilai *def*, sebuah nilai default, jika tidak terdapat property dengan key yang sesuai.

Tabel 1.1: getProperty() method dari class System

Kita tidak dapat cukup berhenti pada detail dari system properties, namun dilanjutkan dengan memanipulasi system properties yang digunakan. Jika Anda tertarik mempelajari lebih lanjut tentang system properties, Anda dapat menelusuri dokumentasi API yang disediakan.

Anda dapat menggunakan argument opsional –D pada perintah Java dalam command-line untuk menambahkan property baru.

```
java -D<name>=value
```

Sebagai contoh, untuk mengatur system property dengan nama *user.home* bernilai *phillipines*, gunakan perintah berikut :

```
java -Duser.home=philippines
```

Untuk menampilkan daftar system properties yang tersedia pada sistem Anda, gunakan method *getProperties* seperti yang ditunjukkan sebagai berikut :

```
System.getProperties().list(System.out);
```

5.3 Membaca Standard Input

Dibandingkan dengan mendapatkan masukan user dari command-line, sebagian user lebih memilih untuk memasukkan data bilamana diminta oleh program pada saat eksekusi. Satu cara dalam melakukan hal ini adalah dengan menggunakan stream. Sebuah stream adalah abstraksi dari sebuah file atau sebuah perangkat yang mengijinkan beberapa set item untuk dibaca atau ditulisi. Streams terhubung dengan physical devices seperti keyboards, consoles dan files. Terdapat dua bentuk umum dari streams, byte streams dan character streams. Byte streams digunakan pada data biner, sedangkan character streams digunakan pada karakter Unicode. *System.in* dan *System.out* adalah dua contoh dari byte streams yang digunakan dalam Java. Contoh pertama mereferensikan pada keyboard, kemudian contoh kedua mereferensikan pada console.

Untuk membaca karakter dari keyboard, Anda dapat menggunakan byte stream *System.in* yang terdapat pada object BufferedReader. Baris berikut menunjukkan bagaimana untuk melakukan hal tersebut :

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

Method read dari object BufferedReader selanjutnya digunakan untuk membaca nilai input dari perangkat input.

```
ch=(int)br.read(); //method read menghasilkan nilai integer
```

Cobalah contoh kode berikut :

```
import java.io.*;

class FavoriteCharacter {
	public static void main(String args[]) throws IOException {
		System.out.println("Hi, what's your favorite character?");
		char favChar;
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		favChar = (char) br.read();
		System.out.println(favChar + " is a good choice!");
	}
}
```

Jika Anda lebih memilih untuk membaca keseluruhan baris daripada membaca satu karakter tiap waktu, gunakan method readLine :

```
str = br.readLine();
```

Berikut ini sebuah program yang hampir menyerupai contoh sebelumnya, namun membaca keseluruhan string, bukan satu karakter.

```
import java.io.*;

class GreetUser {
        public static void main(String args[]) throws IOException {
                System.out.println("Hi, what's your name?");
                String name;
                BufferedReader br = new BufferedReader(new
                InputStreamReader(System.in));
                name = br.readLine();
                System.out.println("Nice to meet you, " + name + "! :)");
        }
}
```

Pada saat menggunakan streams, jangan lupa untuk mengimport package *java.io* seperti yang ditunjukkan dibawah ini :

```
        import java.io.*;
```

Satu hal lagi yang perlu untuk diingat, pembacaan dari streams dapat menyebabkan terjadinya exception. Jangan lupa untuk menangani exception tersebut menggunakan perintah try-catch atau dengan mengindikasikan exception pada klausa throws dalam method.

5.4 Menangani File

Pada beberapa kasus, masukan data disimpan pada sebuah file. Selanjutnya, terdapat beberapa cara jika Anda ingin menyimpan output dari program pada sebuah file. Pada sistem terkomputerisasi, data dari Siswa yang dapat digunakan sebagai input oleh sistem umumnya terseimpan pada sebuah file terpisah. Kemudian, salah satu kemungkinan output dari sistem adalah informasi tentang mata pelajaran yang diikuti oleh siswa. Sekali lagi, output dalam hal ini dapat disimpan dalam sebuah file. Seperti yang terlihat pada aplikasi, terdapat suatu kebutuhan untuk membaca dan menulis sebuah file. Anda akan mempelajari tentang file input dan output pada bagian ini.

5.4.1 Membaca sebuah File

Untuk membaca sebuah file, Anda dapat menggunakan class *FileInputStream*. Berikut ini adalah salah satu constructor dari class tersebut :

```
FileInputStream(String filename)
```

Constructor tersebut membuat sebuah koneksi terhadap file dimana nama dari file tersebut ditunjukkan sebagai sebuah argument. Exception berupa *FileNotFoundException* akan muncul jika file tidak ditemukan atau tidak dapat dibuka dan kemudian dibaca.

Setelah membuat sebuah input stream, Anda kemudian dapat menggunakannya untuk membaca sebuah file dengan menggunakan method *read*. Method *read* menghasilkan sebuah nilai integer, dan akan menunjukkan nilai 1 jika telah mencapai batas akhir file.

Berikut ini contohnya :

```
import java.io.*;

class ReadFile {
  public static void main(String args[]) throws IOException {
          System.out.println("What is the name of the file to read from?");
          String filename;
          BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
          filename = br.readLine();
          System.out.println("Now reading from " + filename + "...");
          FileInputStream fis = null;
          try {
                  fis = new FileInputStream(filename);
          } catch (FileNotFoundException ex) {
                  System.out.println("File not found.");
          }
          try {
                  char data;
                  int temp;
                  do {
                          temp = fis.read();
                          data = (char) temp;
                  if (temp != -1) {
                          System.out.print(data);
                  }
               } while (temp != -1);
          } catch (IOException ex) {
                  System.out.println("Problem in reading from the file.");
          }
  }
}
```

5.4.2 Menulis sebuah file

Untuk menuliskan sebuah file, Anda dapat menggunakan class *FileOutputStream*. Berikut ini salah satu constructor yang dapat Anda gunakan.

```
FileOutputStream(String filename)
```

Contructor tersebut menyediakan jalur output stream terhadap sebuah file yang akan ditulisi. Sebuah Exception berupa *FileNotFoundException* akan muncul jika file yang dimaksud tidak dapat dibuka untuk ditulisi.

Jika output stream telah dibuat, Anda dapat menggunakannya untuk menulisi file yang dituju menggunakan method *write*. Method tersebut menggunakan penandaan sebagai berikut :

```
void write(int b)
```

Parameter *b* mereferensikan data yang akan dituliskan pada file sesuai dengan hasil output stream.

Program berikut menunjukkan contoh penulisan terhadap file :

```
import java.io.*;

class WriteFile {
       public static void main(String args[]) throws IOException {
              System.out.println("What is the name of the file to be written
              to?");
              String filename;
              BufferedReader br = new BufferedReader(new
                  InputStreamReader(System.in));
              filename = br.readLine();
              System.out.println("Enter data to write to " + filename +
                  "...");
              System.out.println("Type q$ to end.");
              FileOutputStream fos = null;
              try {
                     fos = new FileOutputStream(filename);
              } catch (FileNotFoundException ex) {
                     System.out.println("File cannot be opened for
                          writing.");
              }
              try {
                     boolean done = false;
                     int data;
              do {
                     data = br.read();
                     if ((char)data == 'q') {
                            data = br.read();
                            if ((char)data == '$') {
                                   done = true;
                            } else {
                                   fos.write('q');
                                   fos.write(data);
                            }
                     } else {
                            fos.write(data);
                     }
                  } while (!done);
              } catch (IOException ex) {
                     System.out.println("Problem in reading from the file.");
              }
       }
}
```

5.5 Latihan

5.5.1 Spasi menjadi Underscore (_)

Buatlah sebuah program yang memuat dua String sebagai argument, sumber dan nama file tujuan. Kemudian, baca file sumber dan tuliskan isi dari file tersebut terhadap file tujuan, seluruh spasi yang ada (` `) diubah menjadi underscore (` _ `).