PEMROGRAMAN 1
PEMROGRAMAN JAVA
LABSHEET

Topik

Mengakses File Dengan Java

Tujuan

Setelah selesai praktikum, mahasiswa diharapkan mampu membuat program untuk mengakses file (membuat file dan membaca file) dengan benar.

Alat dan Bahan

Komputer dengan software OS, pengolah kata, dan **Java Development Kit (JDK)**

Teori

Dalam program java, file terdapat dalam paket io. Terdapat 4 kelas utama untuk mengakses file, yaitu File, File streaming, File reader writer, dan File zip. Kelas file merupakan kelas yang menghubungkan file dalam disk dengan file sebagai *object*. Kelas file streaming berguna untuk mengakses file binary (file yang datanya tidak dapat dibaca secara langsung karena datanya diubah dalam bentuk biner oleh program). Kelas filereader writer berguna untuk mengakses file text (file yang datanya dapat dibaca secara langsung menggunakan notepad). File zip merupakan file kompresi yang paling banyak digunakan. Terdapat 2 jenis zip yang dapat diakses, yaitu gzip dan zip.

Class File untuk mengakses file di sistem operasi. Cara mendeklarasikannya sebagai berikut :

```
// file yang ada di direktori posisi sekarang
File file = new File("in.txt");
// file dengan absolute path File direktori/folder
File file = new File("d:\\myproject\\java\\Hello.java");
File file = new File("c:\\temp");
```

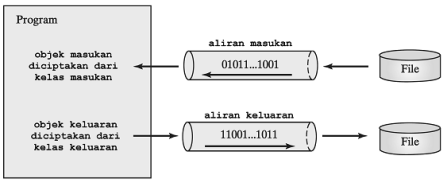
Method dari File :

a. public boolean exists() // untuk memeriksa file/folder ada
b. public long length() // menghitung panjang dari file
c. public boolean isDirectory() // memeriksa apakah folder/bukan
d. public boolean isFile() // memeriksa apakah file/bukan
e. public boolean canRead() // memeriksa apakah file dapat dibaca
f. public boolean canWrite() // memeriksa apakah file dapat ditulis
g. public boolean delete() // menghapus file/folder
h. public void deleteOnExit() // menghapus file ini setelah program keluar
i. public boolean renameTo(File dest) // mengubah nama file ini
j. public boolean mkdir() // membuat folder
k. public String[] list() // mengambil isi folder ini dalam array of string

l. public File[] listFiles() // mengambil isi folder ini dalam array of File
m. public String[] list(FilenameFilter filter) //mengfilter isi folder
n. public File[] listFiles(FilenameFilter filter) //mengfilter isi folder
o. public File[] listFiles(FileFilter filter) //mengfilter isi folder
p. public boolean accept(File dir, String file) //untuk menerima file/folder yang sudahdifilter

FileWriter Dan FileReader

FileWriter digunakan hanya untuk menuliskan text (String atau char) kedalam file dan kemudian isi file dapat dibaca kembali dengan menggunakan FileReader.



Untuk menulis teks ke suatu file bernama **contohfile.txt**, perlu membuat objek menggunakan kelas **PrintWriter** sebagai berikut:

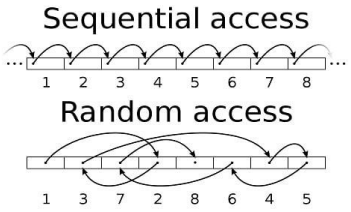
```
PrintWriter keluaran = new PrintWriter("contohfile.txt");
```

metode print dari objek keluaran dipakai untuk menulis suatu string ke dalam file. Contoh, pernyataan untuk menulis "**Selamat datang di Pemrograman Java**" ke dalam file:

```
keluaran.print("Selamat datang di Pemrograman Java");
```

Statemen berikut ini menutup file tersebut:

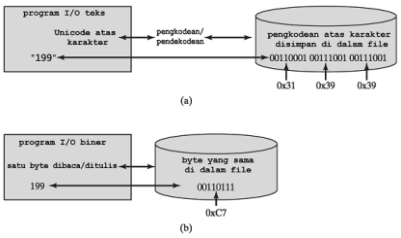
```
keluaran.close();
```



Terdapat banyak kelas I/O untuk berbagai tujuan. Pada umumnya, dapat diklasifikasikan menjadi kelas masukan dan kelas keluaran. Kelas masukan memuat metode-metode untuk membaca data, dan kelas keluaran mamuat metode-metode untuk menulis data. **PrintWriter** merupakan suatu contoh kelas keluaran, dan **Scanner** adalah suatu contoh kelas masukan. Kode berikut membuat suatu objek masukan untuk file **contohfile.txt** dan membaca data dari file itu:

```
Scanner masukan = new Scanner(new File("contohfile.txt"));
System.out.println(masukan.nextLine());
```

Jika file **contohfile.txt** memuat "**Selamat datang di Pemrograman Java**", maka masukan.nextLine() menghasilkan string "**Selamat datang di Pemrograman Java**"



Cara menuliskan suatu text yang disimpan dalam variabel write dalam file misal "contohfile.txt" dengan menggunakan FileWriter. Perhatikan method write() pada FileWriter hanya menerima parameter String atau array dari char, karena itulah FileWriter hanya bisa menuliskan data berupa text kedalam suatu file.

```
// String yang akan ditulis kedalam file
String write = "Mengisi file dengan menggunakan Java";
// Mengisi file dengan FileWriter
FileWriter fw = new FileWriter("contohfile.txt");
fw.write(write);
fw.flush();
```

```
fw.close();
```

Cara membaca isi file “contohfile.txt” dengan menggunakan FileReader. Method read() pada FileReader akan membaca isi text dalam file huruf per huruf satu persatu dari posisi pertama sampai posisi terakhir. Huruf yang berhasil dibaca akan dimasukkan kedalam variabel i dalam bentuk int tapi sebenarnya berisi data huruf (char). Karena itulah method read() ini harus dipanggil berulang kali didalam while loop. Apabila huruf yang dibaca sudah habis (sudah berada setelah posisi terakhir) maka method read() akan mengembalikan -1, dan saat itulah while loop harus dihentikan. Data huruf yang dihasilkan (dalam variabel i) dapat diganti menjadi bentuk char dan dicetak melalui method System.out.println.

```
// Membaca file dengan
FileReader fr = new FileReader("contohfile.txt");
int i;
while ((i = fr.read()) != -1) {
   System.out.print((char) i);
}
fr.close();
```

Contoh 1. FileWriterReader.java

```
import java.util.Scanner;
import java.io.*;
public class FileWriterReader {
 public static void main(String[] args)throws IOException{
   Scanner scn = new Scanner(System.in);
   String nama; int jumlah; double harga; char tanya; File file=new File("contohfile.txt");
   FileWriter fw=new FileWriter(file);
   BufferedWriter bw=new BufferedWriter(fw);
   System.out.printf("Nama : ");
   nama=scn.next();
   while(!nama.equalsIgnoreCase("x")){
     System.out.printf("Jumlah : ");
     jumlah=scn.nextInt();
     System.out.printf("Harga : ");
     harga=scn.nextDouble();
     System.out.printf("Disimpan(Y/T)? ");
     tanya=scn.next().toUpperCase().charAt(0);
     if(tanya=='Y'){
      bw.write(nama+" ");
      bw.write(String.valueOf(jumlah)+" ");
      bw.write(String.valueOf(harga)+" ");
      bw.flush();
     }
     System.out.println();
     System.out.printf("Nama : ");nama=scn.next();
    }
    if(fw!=null) fw.close();
    FileReader fr=new FileReader("contohfile.txt");
    Scanner fscn=new Scanner(fr);
    while(fscn.hasNext()){
      nama=fscn.next();
      jumlah=fscn.nextInt();
```

```
        harga=fscn.nextDouble();
        System.out.printf("%s %d %f %f\n",nama,jumlah,harga,jumlah*harga);
      }
      if(fr!=null) fr.close();
    }
}
```

Untuk membuka file dan membaca isinya digunakan class Scanner dan class FileReader. Class FileReader digunakan untuk membuka dan membaca file, sedangkan class Scanner digunakan untuk membaca/mengambil isi file.
Untuk menggunakan class Scanner dan class FileReader kita harus melakukan import class pada awal pemrograman seperti berikut:

```
import java.util.Scanner;
import java.io.FileReader;
import java.io.FileNotFoundException;
```

class FileNotFoundException digunakan untuk melakukan pemeriksaan, apakah file yang akan dibuka ada atau tidak ada. Selanjutnya di dalam tubuh program buat object dengan type Scanner seperti berikut:

```
Scanner inputFile = new Scanner(new FileReader(<namaFile));
```

Parameter yang digunakan pada Scanner adalah **new FileReader(<namaFile>)** (menggantikan System.in), hal ini untuk tujuan memberitahu Scanner bahwa yang dibaca adalah dari file (melalui FileReader). Instruksi FileReader memiliki exception/pengecualian kalau file yang ingin dibuka tidak ditemukan, untuk itu maka instruksi tersebut harus diletakkan di dalam block **try … catch** dengan menggunakan **FileNotFoundException** sebagai pendeteksi kejadian (even handler).

Contoh:

```
import java.util.Scanner;
import java.io.FileReader;
import java.io.FileNotFoundException;

public class membacaDariFile {
	public static void main(String[] args) {
		String barisIsi;
		try {
			Scanner inputFile = new Scanner(new FileReader("daftarNama.txt"));
			while(inputFile.hasNextLine()){
				barisIsi = inputFile.nextLine();
				System.out.println(barisIsi);
			}
		}
		catch(FileNotFoundException e) {
			System.out.println(e);
		}
	}
}
```

Random Access File

Cara kedua untuk menulis dan membaca isi file adalah dengan menggunakan RandomAccessFile. Perbedaan RandomAccessFile dengan FileWriter adalah RandomAccessFile dapat menuliskan data binary (byte[]) atau data text dan dapat membaca isi file yang dimulai dari posisi tertentu. Cara menuliskan data binary kedalam file "Cobarnd.txt" dengan menggunakan RandomAccessFile. Method write pada RandomAccessFile dapat menerima parameter array dari byte, sehingga kita dapat menuliskan data binary.

```
// Mengisi file dengan RandomAccessFile
RandomAccessFile rnd = new RandomAccessFile("Cobarnd.txt", "rw");
rnd.write(write.getBytes());
```

Contoh cara membaca file dengan menggunakan RandomAccessFile. Method seek dipakai untuk membaca isi file dari posisi tertentu, misal membaca isi file "Coba.txt" mulai dari posisi ke 13.

```
// Membaca file dengan RandomAccessFIle
System.out.println("");
rnd.seek(13);
while ((i = rnd.read()) != -1) {
   System.out.print((char) i);
}
rnd.close();
```

FILEOUTPUTSTREAM DAN FILEINPUTSTREAM

Cara ketiga memakai FileOutputStream untuk menuliskan data kedalam file dan FileInputStream untuk membaca data dari file. FileOutputStream seperti juga RandomAccessFile dapat menuliskan data binary kedalam file. Sedangkan dengan menggunakan FileInputStream kita dapat membaca panjang file dan membaca seluruh isi file sekaligus tanpa menggunakan while loop. Contoh cara memakai FileOutputStream untuk menuliskan data binary kedalam file "Cobafos.txt".

```
// Mengisi file dengan FileOutputStream
FileOutputStream fos = new FileOutputStream("Cobafos.txt");
fos.write(write.getBytes());
fos.flush();
fos.close();
```

Contoh cara memakai FileInputStream untuk membaca isi file. Dengan memakai method available() panjang isi file dapat dibaca, dengan byte array dibuat panjang yang sama. Pemakaian method read untuk membaca seluruh isi file secara bersamaan. Byte array yang diperoleh dapat diganti menjadi String (text) bila data dalam file berbentuk text.

```
FileInputStream fis = new FileInputStream("Cobafos.txt");
int length = fis.available();
byte[] bread = new byte[length];
fis.read(bread);
String read = new String(bread);
System.out.println(read);
fis.close();
```

Contoh 2. Akses File

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
```

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.RandomAccessFile;
public class AksesFile {
  public static void main(String[] args) {
    try {
     // String yang akan ditulis kedalam file
     String write = "Mengisi file dengan menggunakan Java";
     // Mengisi file dengan FileWriter
     FileWriter fw = new FileWriter("Cobafw.txt");
     fw.write(write);
     fw.flush();
     fw.close();
     // Membaca file dengan FileReader
     FileReader fr = new FileReader("Cobafw.txt");
     int i;
     while ((i = fr.read()) != -1) {
       System.out.print((char) i);
      }
      fr.close();
      // Mengisi file dengan RandomAccessFile
      RandomAccessFile rnd = new RandomAccessFile("Cobarnd.txt", "rw");
      rnd.write(write.getBytes());
       // Membaca file dengan RandomAccessFIle
       System.out.println("");
       rnd.seek(13);
       while ((i = rnd.read()) != -1) {
         System.out.print((char) i);
        }
        rnd.close();
        // Mengisi file dengan FileOutputStream
        FileOutputStream fos = new FileOutputStream("Cobafos.txt");
        fos.write(write.getBytes());
        fos.flush();
        fos.close();
         // Membaca file dengan FileInputStream
         System.out.println("");
         FileInputStream fis = new FileInputStream("Cobafos.txt");
         int length = fis.available();
         byte[] bread = new byte[length];
         fis.read(bread);
         String read = new String(bread);
         System.out.println(read);
         fis.close();
       } catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

Praktek

Buat Kelompok yang terdiri dari 3 orang (3 orang/kelompok)

1. Buat program seperti pada **Contoh 1. FileWriterReader.java**
2. Kompile dan jalankan program tersebut
3. Amati hasilnya dan pahami perintah-perintah yang ada
4. Ulangi langkah 1-3 diatas untuk **Contoh 2. Akses File**
5. Kembangkan program di atas sehingga :
   - ada menu buat file dan membaca file
   - file yang akan dibuat/dibaca folder dan namanya diinput dari keyboard

Analisa Hasil Praktek

Berdasarkan praktek yang telah Anda lakukan diskusikan dengan teman sekelompok, kemudian buatlah kesimpulan untuk disampaiakan/dipresentasikan di kelas anda.

Tugas

Buat program yang akan membaca isi file teks, kemudian program tersebut menghitungjumlah huruf tertentu (sesuai keinginan user).