Modul Pemrograman Berbasis Objek (PBO)
File Stream Dan Exception Part 2

Setelah kita membahas exception di part 1, kali ini kita akan membahas file stream.

File stream adalah proses bagaimana kita melakukan input dan output pada file diluar java. Mudahnya, biasanya kita melakukan input dan output secara internal pada terminal dan kode yang kita buat. Nah sekarang, bayangkan kita dapat membuat file non java melalui java. Atau membaca file text melalui kode di java java.

Mari kita langsung saja ke kodingan, buatlah folder Files, lalu didalam Files buatlah 4 folder, create, read, delete dan output

Folder create nantinya akan berisi kodingan agar kita dapat membuat file txt melalui java

Folder read akan berisi kodingan agar kita dapat membaca file txt yang telah kita buat

Folder delete mempunyai isi kodingan yang akan digunakan untuk menghapus file txt yang kita buat

Folder output berisi file txt yang kita buat, baca, dan akan di hapus

Mari kita ke folder create terlebih dahulu, di dalam folder create buat lah file bernama BuatFile.java, lalu masukan kode dibawah ini

```
import java.io.File;  // Import the File class
import java.io.IOException;  // Import the IOException class to handle
errors

public class BuatFile {
 public static void main(String[] args) {
   try {
     File buatFile = new File("../File/filebaru.txt");
     if (buatFile.createNewFile()) {
       System.out.println("Nama file adalah = " + buatFile.getName());
     } else {
       System.out.println("File sudah dibuat");
     }
   } catch (IOException e) {
     System.out.println("Terdapat error pada file");
     e.printStackTrace();
   }
 }
}
```

Mari kita bedah kode diatas

```
import java.io.File;  // Import the File class
import java.io.IOException;  // Import the IOException class to handle
errors
```

Pertama, kita akan menggunakan library java bernama File yang akan kita gunakan untuk membuat file. Method yang terdapat pada library File membutuhkan exception untuk menangani error saat pemanggilan method.

Kita juga menggunakan library java bernama IOException yang berisi berbagai parameter yang nantinya dapat kita gunakan di try catch

```
File buatFile = new File("../Output/filebaru.txt");
```

Kita membuat objek dari library File bernama buatFile. Instansiasi Library file membutuhkan parameter constructor yaitu alamat dan nama file yang akan di akses oleh objek buatFile. Karena kita akan menggunakan folder output, maka kita mengarahkan ke dalam folder output, lalu kita akan menggunakan file bernama filebaru.txt

Notes : Kalian tetap dapat membuat file didalam folder yang sama dengan kode java nya

Contoh nya
File createFile = new File(“namafile.txt”)

```
if (buatFile.createNewFile()) {
        System.out.println("Nama file adalah = " + buatFile.getName());
} else {
        System.out.println("File sudah dibuat");
}
```

Methot createNewFile kita gunakan untuk membuat file baru. Lokasi file dan nama file akan sesuai dengan yang kita berikan saat pembuatan objek buatFile. Selain membuat file yang kita inginkan juga akan me return nilai boolean. Jika file yang ingin dibuat sudah ada, maka file tidak akan dibuat dan me return false. Tetapi jika file belum dibuat, maka file akan dibuat dan me return true.

Kita memanfaatkan return dari hasil method createNewFile untuk membuat kondisi , jika hasil return createNewFile true maka akan menampilkan nama filenya, sedangkan jika me return false, maka akan memberi tahu bahwa file sudah dibuat.

Method getName digunakan untuk menampilkan nama file yang kita akses.

```java
try {
    File buatFile = new File("../File/filebaru.txt");
    if (buatFile.createNewFile()) {
      System.out.println("Nama file adalah = " + buatFile.getName());
    } else {
      System.out.println("File sudah dibuat");
    }
  } catch (IOException e) {
    System.out.println("Terdapat error pada file");
    e.printStackTrace();
  }
```

Sedangkan untuk try catch diatas, jika terdapat error saat pemanggilan method pada objek buatFile, maka catch nya akan dijalankan.

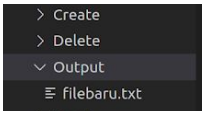
Parameter IOException sendiri digunakan untuk error yang berkaitan dengan input output. Hal ini sesuai dengan method yang kita gunakan pada objekFile, kita menggunakan method createNewFile untuk membuat file, sehingga jika terdapat error maka IOException akan dijalankan pada catch.

Method printStackTrace pada catch digunakan untuk memberi tahu error apa yang terjadi pada kodingan.

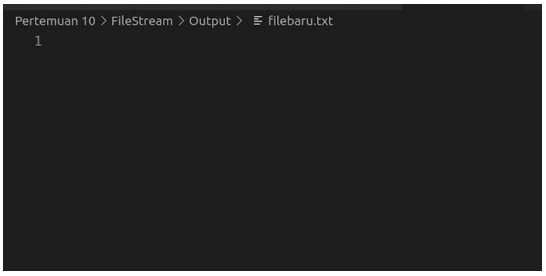
Kita telah membuat file txt didalam folder Output. Mari kita coba jalankan kode diatas pada terminal

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/FileStream
/Create$ javac BuatFile.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/FileStream
/Create$ java BuatFile
Nama file adalah = filebaru.txt
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/FileStream
/Create$
```

filebaru.txt pada folder output



Kita telah berhasil membuat file txt dengan java, tetapi file tersebut tidak ada tulisan, hanya file saja



Lalu, bagaimana cara kita menulis pada file txt yang baru saja kita buat ? Mari kita coba

Masih didalam folder create, buatlah file bernama TulisFile.java, lalu masukan kode dibawah ini

```java
import java.io.FileWriter;   // Import the FileWriter class
import java.io.IOException;  // Import the IOException class to handle errors

public class TulisFile {
 public static void main(String[] args) {
   try {
     FileWriter tulisFile = new FileWriter("../File/filebaru.txt");
     tulisFile.write("File ini ditulis dari java");
     tulisFile.close();
     System.out.println("File berhasil ditulis.");
   } catch (IOException e) {
     System.out.println("Terdapat error saat menulis file");
     e.printStackTrace();
   }
 }
}
```

Mari kita bedah kode diatas

```
import java.io.FileWriter;   // Import the FileWriter class
import java.io.IOException;  // Import the IOException class to handle
errors
```

Kali ini kita menggunakan library FileWriter untuk menulis file txt yang telah kita buat

```
FileWriter tulisFile = new FileWriter("../Output/filebaru.txt");
tulisFile.write("File ini ditulis dari java");
tulisFile.close();
System.out.println("File berhasil ditulis.");
```

Lalu, kita membuat objek tulisFile dan menginstasiasi objek tersebut dengan mengisi folder dan nama file yang akan kita isi tulisan.

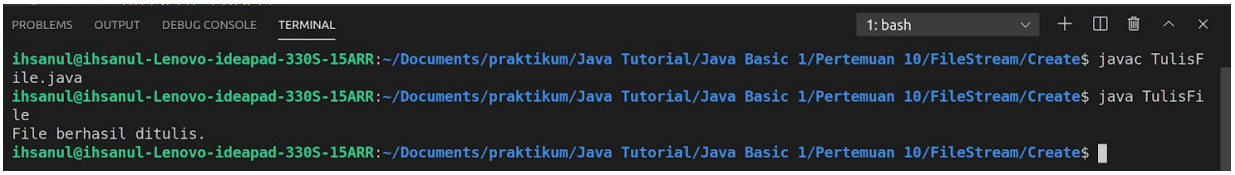
Kita menggunakan method write() pada objek tulisFile untuk mengisi file txt yang telah kita buat.

Setelah kita mengisi file txt dengan tulisan, kita harus menutup file txt tersebut dari kode java yang kita buat. Untuk menutup file txt yang kita akses dapat menggunakan method close().

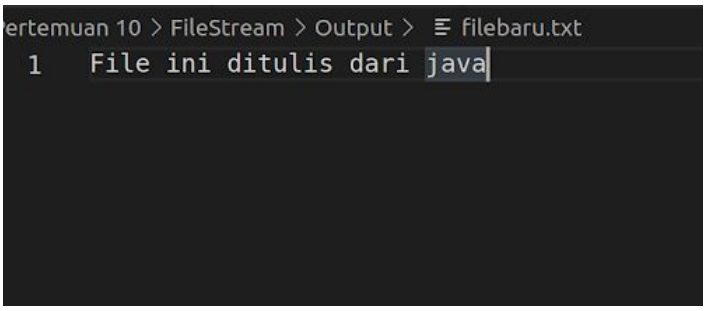
Dan itulah cara kita menulis file txt yang telah kita buat menggunakan java.

Untuk kode try and catch nya tidak dijelaskan, karena fungsinya sama saja dengan penjelasan yang diatas, bedanya try catch yang pertama digunakan untuk membuat file, sedangkan try catch pada kode yang baru saja kita buat digunakan untuk menulis file.

Mari kita coba jalankan di terminal



Kita telah berhasil menjalankan diterminal, mari kita buka filebaru.txt



Kita telah berhasil menulis di filebaru.txt melalui kode dijava.

Kita telah berhasil membuat dan menulis file, mari kita masuk ke langkah berikut nya yaitu membaca file yang kita buat melalui kode dijava.

Masuk ke folder read yang sudah kita buat, buatlah file file bernama BacaFile.java, lalu masukan kode dibawah ini

```
import java.io.File;  // Import the File class
import java.io.FileNotFoundException;  // Import this class to handle
errors
import java.util.Scanner; // Import the Scanner class to read text
files

public class BacaFile {
 public static void main(String[] args) {
   try {
     File myObj = new File("../Output/filebaru.txt");
     Scanner myReader = new Scanner(myObj);
     while (myReader.hasNextLine()) {
       String data = myReader.nextLine();
       System.out.println(data);
     }
     myReader.close();
   } catch (FileNotFoundException e) {
     System.out.println("An error occurred.");
     e.printStackTrace();
   }
 }
}
```

Mari kita bedah kode diatas

```
import java.io.File;  // Import the File class
import java.io.FileNotFoundException;  // Import this class to handle
errors
import java.util.Scanner; // Import the Scanner class to read text
files
```

Pertama kita menggunakan library File kembali untuk mengakses file txt yang sudah kita buat

Lalu, kita kali ini menggunakan library FileNotFoundException sebagai exception yang akan kita digunakan pada try catch.

FileNotFoundException kita gunakan sebagai error, jika file yang ingin dibaca tidak ditemukan.

Lalu, kita menggunakan library Scanner. Sebelum nya, kita menggunakan library Scanner untuk menginput melalui terminal, kali ini kita menggunakan library Scanner untuk menerima input dari file txt yang kita akses untuk dibaca

```
File myObj = new File("../Output/filebaru.txt");
Scanner myReader = new Scanner(myObj);
while (myReader.hasNextLine()) {
    String data = myReader.nextLine();
    System.out.println(data);
}
```

Kita membuat objek dari library File bernama myObj yang parameter nya diisi dengan folder dan nama file yang akan kita akses

Lalu, kita membuat objek dari library Scanner bernama myReader. Kita mengisi parameter objek myReader dengan objek myObj. Hal ini membuat myReader dapat membaca txt yang telah kita buat

Untuk membaca file txt yang kita buat, kita menggunakan nextLine. Jika kalian ingat, method nextLine biasanya kita gunakan untuk menerima

input dari terminal. Tapi, kali ini kita gunakan untuk membaca file secara perbaris

Kita fokus kepada perulangan yang kita buat

```
while (myReader.hasNextLine()) {
    String data = myReader.nextLine();
    System.out.println(data);
}
```

Seperti yang dijelaskan diatas, file akan dibaca baris perbaris. Jadi perulangan diatas, akan terus menampilkan isi dari file tersebut secara baris perbaris. Perulangan akan berhenti saat tidak ada baris lagi yang mempunyai text.

Notes : Method nextLine akan membaca file secara perbaris

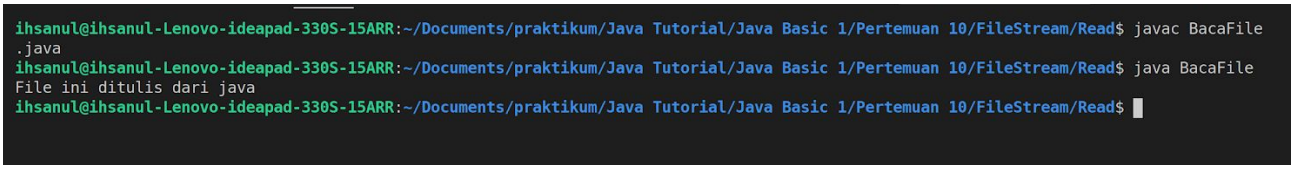
Jadi, ketika menjalan pertama kali method nextLine, maka yang akan dibaca adalah baris pertama, ketika memanggil kembali method nextLine maka yang akan dibaca adalah baris kedua, begitu seterusnya. Tetapi jika tidak ada lagi baris yang bisa dibaca, maka file akan error.

Oleh karena itu kita menggunakan method hasNextLine untuk mengecek apakah masih ada baris selanjutnya yang dapat dibaca

```
  try {
    File myObj = new File("../Output/filebaru.txt");
    Scanner myReader = new Scanner(myObj);
    while (myReader.hasNextLine()) {
      String data = myReader.nextLine();
      System.out.println(data);
    }
    myReader.close();
  } catch (FileNotFoundException e) {
    System.out.println("An error occurred.");
    e.printStackTrace();
  }
```

Try catch yang kita gunakan untuk mengechek apakah file yang ingin dibaca tersedia. Parameter FileNotFoundException pada catch kita gunakan untuk error yang terjadi jika File yang ingin dibaca tidak ditemukan

Mari kita coba jalankan pada terminal



Kita berhasil membaca tulisan didalam filebaru.txt

Nah, ada pertanyaan, gimana kita dapat mengetahui informasi mengenai file txt yang kita buat, contoh nya sizenya berapa, lokasi file nya dimana, nama file nya apa

Mari kita coba langsung di kodingan, masih di folder read, buatlah file bernama InformasiFile.java, lalu masukan kode dibawah ini pada file tersebut

```
import java.io.File;  // Import the File class
public class InformasiFile {
 public static void main(String[] args) {
   File informasiFile = new File("../Output/filebaru.txt");
   if (informasiFile.exists()) {
     System.out.println("Nama file = " + informasiFile.getName());
     System.out.println("Alamat file = " +
                        informasiFile.getAbsolutePath());
     System.out.println("Apakah file boleh ditulis ? " +
                        informasiFile.canWrite());
     System.out.println("Apakah file data dibaca ? " +
                        informasiFile.canRead());
     System.out.println("Ukuran file " + informasiFile.length());
   } else {
     System.out.println("File tidak ada");
   }
 }
}
```

Kita menggunakan kembali library File untuk mengakses file txt yang kita buat.

Mari kita bahas method library file yang kita gunakan satu persatu

Method exist() digunakan untuk mengechek apakah file yang ingin kita baca tersedia. Jika ada maka return true, jika tidak ada maka return false

Method getName() digunakan untuk mengechek nama dari file yang kita akses

Method getAbsolutePath() digunakan untuk mengecheck lokasi dari file yang kita akses

Method canWrite() digunakan untuk mengecheck apakah file txt yang kita akses dapat kita isi

Method canRead digunakan untuk mengechek apakah file txt yang kita akses dapat dibaca oleh program

Method length digunakan untuk mengechek size dari file txt yang kita akses

Mari kita coba jalankan kode diatas, melalui terminal

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/FileStream/Read$ javac Informas
iFile.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/FileStream/Read$ java Informasi
File
Nama file = filebaru.txt
Alamat file = /home/ihsanul/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/FileStream/Read/../Output/filebaru.txt
Apakah file boleh ditulis ? true
Apakah file data dibaca ? true
Ukuran file 26
```

Kita berhasil membaca informasi mengenai filebaru.txt melalui java

Kita telah berhasil membuat file txt dan membaca file txt tersebut, sekarang kita akan mencoba menghapus file txt yang sudah kita buat

Buka folder delete yang sudah kita buat, lalu buat file bernama HapusFile.java dan masukan kode dibawah ini

```
import java.io.File;  // Import the File class

public class HapusFile {
 public static void main(String[] args) {
   File hapus = new File("../Output/filebaru.txt");
   if (hapus.delete()) {
     System.out.println("File yang dihapus: " + hapus.getName());
   } else {
     System.out.println("Gagal menghapus file");
   }
 }
}
```

Sama seperti sebelumnya, untuk mengakses file kita menggunakan library File kembali. Untuk menghapus file txt yang sudah kita buat, kita dapat menjalankan method delete().

Mari kita jalankan kode diatas pada terminal

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/FileStream/Delete$ javac HapusF
ile.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/FileStream/Delete$ java HapusFi
le
File yang dihapus: filebaru.txt
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/FileStream/Delete$
```

Kita telah berhasil menjalankan kode diatas pada terminal, mari kita check folder Output

filebaru.txt telah terhapus

Karena sudah tidak ada filebaru.txt, kita ingin menghapus folder output juga. Mari kita coba menghapus folder melalui java

Masih didalam folder delete, buatlah file baru bernama HapusFolder.java, lalu masukan kode dibawah ini

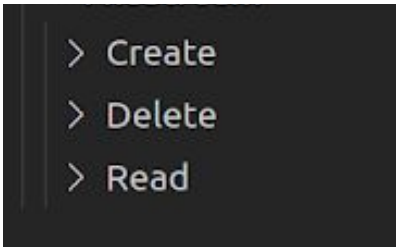
```java
import java.io.File;

public class HapusFolder {
 public static void main(String[] args) {
   File hapus = new File("../Output");
   if (hapus.delete()) {
     System.out.println("Deleted the folder: " + hapus.getName());
   } else {
     System.out.println("Failed to delete the folder.");
   }
 }
}
```

Dapat kita lihat, untuk menghapus folder, method yang digunakan sama saja seperti menghapus file txt.

Mari kita jalankan di terminal

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/FileStream/Delete$ javac HapusF
older.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/FileStream/Delete$ java HapusFo
lder
Deleted the folder: Output
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/FileStream/Delete$
```

Kita berhasil menjalankan kode diatas, mari kita lihat apakah folder output sudah terhapus



Yap, folder sudah terhapus

Notes : Pada modul ini, kita menggunakan file txt sebagai contoh, tetapi kita juga dapat menggunakan selain file txt

Berikut referensi untuk filestream

Library file : https://www.w3schools.com/java/java_files.asp

Create/Write file : https://www.w3schools.com/java/java_files_create.asp

Read file : https://www.w3schools.com/java/java_files_read.asp

Delete file : https://www.w3schools.com/java/java_files_delete.asp

Tugas
1. Buatlah studi kasus berbentuk kodingan yang berisi create file, read file, dan delete seperti pada contoh praktikum ini
2. Masukan hasil kodingan ke dalam laporan berikut output nya dan penjelasan kodingan tersebut
3. Upload kodingan tersebut bebas dimana saja, lalu masukan link kodingan tersebut didalam file ini

Tugas dikirim pada elen, dengan format sebagai berikut
1. Pada penamaan file gunakan format berikut
Praktikum keberapa_Judul Praktikum_Nama_Nim
Contoh :
Praktikum01_Perkenalan-Java_IhsanulFikriAbiyyu_0220318021
2. Format laporan bertipe pdf
3. Pada cover praktikum, masukan beberapa hal dibawah ini- Praktikum ke berapa
- Judul Praktikum
- Nama
- Nim
4. Tenggat waktu mengerjakan adalah 10 pekan semenjak tugas diberikan
5. Diperbolehkan berkelompok tetapi hanya antara yang tidak punya laptop/komputer