**LAPORAN PRAKTIKUM**

**PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL VIII**

***EXCEPTION* DAN I/O**

****

**DOSEN PENGAMPU :**

Agus Priyanto, S. Kom., M.Kom.

**DISUSUN OLEH :**

Filfimo Yulfiz Ahsanul Hulqi

19102143

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**FAKULTAS INFORMATIKA**

**PRODI S1 INFORMATIKA**

**2020**

**BAB I**

**Tujuan Praktikum**

Mahasiswa diharapkan mampu untuk :

1. Mengerti dan memahami tentang konsep *Exception* dan I/O.
2. Mampu mengimplementasikan konsep dari Exception dan I/O dalam sebuah pemrograman sederhana.
3. Dapat memahami konsep dari *Exception Handling*
4. Dapat memilih dan menggunakan *Exception* dan I/O dengan tepat.

**BAB II**

**Dasar Teori**

1. **Exception**

*Exception* adalah kondisi *abnormal*/tidak wajar yang terjadi pada saat melakukan pengeksekusian suatu perintah program. *Exception* merupakan suatu mekanisme untuk menyampaian kepada program dan *user* bahwa sesuatu yang tidak diinginkan telah terjadi pada saat program sedang dijalankan/*runtime.*

Berikut adalah beberapa *Exception* yang ada didalam Java.

1. **AritmaticException**, *Exception* ini terjadi ketika kondisi yang tidak biasa telah terjadi dalam melakukan operasi aritmatika.
2. **ArrayIndexOutBoundsException**, *Exception* ini terjadi dikaernakan untuk menujukkan bahwa *array* telah diakses dengan *indeks* *ilegal*. *Indeks­*nya negatif atau lebih besar dari atau sama dengan ukuran *array*.
3. **ClassNotFoundException**, *Exception* ini muncul jika program mengakses kelas yang definisinya tidak dapat ditemukan.
4. **FileNotfoundException**, *Exception* ini akan muncul jika file tidak dapat diakses atau tidak dapat dibuka.
5. **IOException**, *Exeption* ini terjadi ketika operasi *input*-*output* gagal atau terputus.

Dalam Java ada lima *keywords* yang dapat digunakan untuk meenangani *Exception*, yaitu: *try, catch, finally, throw,* dan *throws*. (Semua class *Exception* ini terdapat didalam *package java.lang*).

1. *Try*-*Catch*

Digunakan untuk menangani *Exception* yang terjadi dalam program, cukup meletakkan kode yang ingin di*Exception* terhadapt kemungkinan *Exception* di dalam blok *try*, diikuti dengan blok *catch* yang menentukan jenis dari *Exception* apa yang akan ditangani.

1. *Throw*

Mekanisme *throw* memungkinkan program untuk mengirim/melempar *Exception* untuk kemudian program bereaksi atau *Exception* tersebut. Objek *Exception* adalah semua objek yang merupakan turunan dari *classThrowable*. Salah satu contoh objek *Exception* adalah *ArrayIndexOutOfBounds*.

1. *Finally*

Penggunaan blok *try*-*catch* terkadang membingungkan karena tidak dapat menentukan dengan pasti alur mana yang akan dieksekusi. Apalagi penggunaan *throw* mengakibatkan kode setelah *throw* tidak dapat dieksekusi atau justru terjadi kesalahan pada blok *catch*, menyebabkan program akan terhenti. Untuk mengatasi masalah ini, Java mempunyai solusi dengan *keyword finally*. Dimana semeua kode yang terdapat didalam blok *finally* ”pasti” dapat/akan dieksekusi apapun yang terjadi di dalam blok *try*-*catch*.

1. **I/O (*Input and Output*)**

*Stream* adalah aliran proses informasi data yang direpresentasikan secara abstark untuk menulis atau menghasilkan (*output*) dan membaca atau mendapatkan suatu informasi (*input*). Semua *streams* mempunyai sifat yang sama meskipun peralatan fisik yang berhubungan dengannya berbeda-beda, misalnya *keyboard*, jaringan, dll.

Paket yang menangani pembacaan dan penulisan info untuk pemrograman Java I/O, ada 2 tipe, yaitu *Byte Stream* dan *Character Stream*.

1. *Byte* *stream*, *Byte stream* digunakan untuk operasi I/O yang menggunakan data biner (*byte)*. Pada Java *Byte* stream mempunyai 2 buah *superclass* yaitu *InputStream* dan *OutputStream* yang merupakan *class abstract*.
2. *Character stream*, *Character stream* digunakan untuk menangani operasi I/O yang menggunakan *character* dan merupakan sebuah objek yang dapat membaca dan menuliskan *byte stream*. Jadi, *character stream* adalah seebuah *byte* yang diteruskan oleh *class Reader* dan *Writter* yang merupakan *class abstract*.

**BAB III**

**PENJELASAN**

**GUIDED**

1. Buat program untuk menampilkan dan melakukan *single try*-*catch*.

package cobaexception;

import java.util.Scanner;

public class CobaException {

public static void main(String[] args) {

int[] array = {1,3,5,7,9};

Scanner Input = new Scanner(System.in);

System.out.println("Nilai array ke : ");

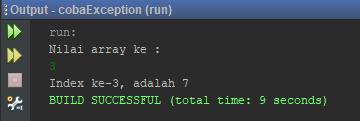
int index = Input.nextInt();

System.out.printf("Index ke-%d, adalah %d\n", index, array[index]);

}

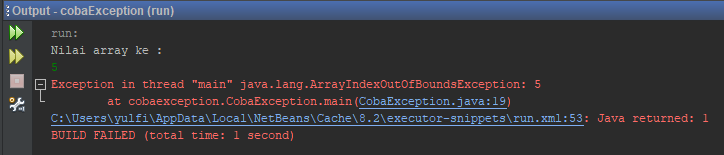
}

*Source code class* CobaException



*Running program* tanpa adanya *Exception*

Running program diatas adalah program mencari nilai *array*, ketika *array* yang di*input*kan sesuai dengan panjang data di*index*, maka program akan berjalan lancar tanpa adanya *exception*.



*Running program* terdapat *Exception*

Lalu, jika *input* *array* tidak sesuai dengan panjang data di*index*, maka program akan mengalami *exception*, dan berhenti (*build failed*). Untuk menangani hal tersebut dapat menggunakan *try*-*catch*.

*Source* *code* *class* CobaException dengan *try*-*catch*

package cobaexception;

import java.util.Scanner;

public class CobaException {

public static void main(String[] args) {

int[] array = {1,3,5,7,9};

Scanner Input = new Scanner(System.in);

System.out.println("Nilai array ke : ");

int index = Input.nextInt();

**//Exeption (Single Try-Catch)**

try{

System.out.printf("Index ke-%d, adalah %d\n", index, array[index]);

} catch(Exception a){

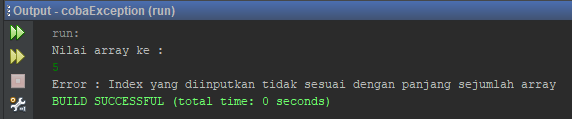
System.out.println("Error : Index yang diinputkan tidak sesuai "

+ "dengan panjang sejumlah array");

}

}

}



*Running program* terdapat *Exception* tetapi ditangani dengan *try*-*catch*

Pada program kali ini ditambah dengan argumen *keyword* *try*-*catch*. Program mempunyai 4 variabel didalam *array* yang bertipe *integer*. Kemudian *user* dapat memberikan *input* untuk mecari *array* didalam *index*. Program akan “mencoba” untuk mencari *array* tersebut disini mencari ke-5, tetapi karena panjang dari datatersebut hanya berjumlah 4, maka akan muncul pesan *error* dan program tetap berjalan secara normal (*build succes*), tidak seperti sebelumnya tanpa menggunakan *try*-*catch* program akan mengalami *exception* dan terhenti.

1. Buat program untuk menampilkan dan melakukan *multiple try*-*catch.*

package cobaexceptionmulti;

import java.util.Scanner;

import java.util.InputMismatchException;

public class CobaExceptionMulti {

public static void main(String[] args) {

Scanner Input = new Scanner(System.in);

try{

System.out.print("Masukan Bilangan : ");

int bil1 = Input.nextInt();

System.out.print("Masukan Pembagi : ");

int bil2 = Input.nextInt();

int hasil = bil1/bil2;

System.out.println(bil1 + " : " + bil2 + " = " + hasil + (" dibulatkan"));

} catch(ArithmeticException e){

System.out.println("Error : Tidak dapat melakukan kalkulasi");

} catch(InputMismatchException e){

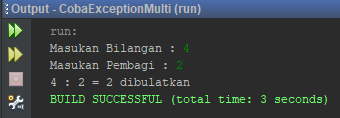
System.out.println("Error : Tipe data bukan integer");

}

}

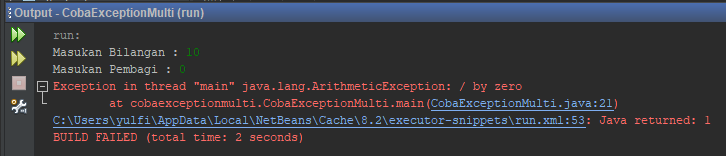
}

*Source code class* Coba*ExceptionMulti*



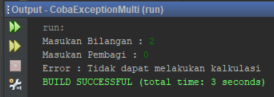
*Running program* tanpa adanya *Exception*

Running program diatas adalah program sederhana melakukan pembagian yang dibulatkan, dimana *user* diminta untuk memasukan bilangan yang ingin dibagi kemudian memasukan pembagi, lalu akan muncul hasil dari pembagian yang sudah dibulatkan.



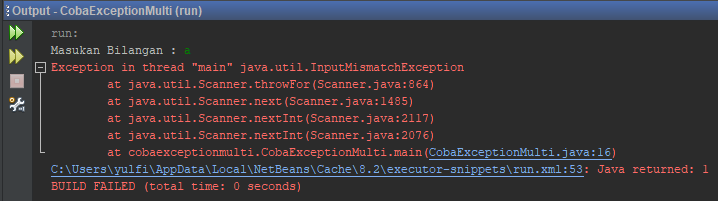
*Running program* terdapat *Exception Aritmetic*

*Exception* ini terjadi karena adanya kondisi yang tidak biasa dalam operasi aritmatika, yaitu pembagi memiliki nilai 0, yang padahal bilangan bulat tidak bisa dibagi dengan 0.



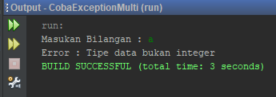
*Running program* terdapat *Exception* tetapi ditangani dengan *multiple try-catch*

Pada program kali ini ditambah dengan argumen *keyword* *try*-*catch* tetapi memiliki lebih dari satu *try*-*catch* (*multiple try*-*catch*). Program sebelumnya akan terhenti jika pembagi memunyai nilai 0, tetapi setelah dilakukan *try*-*catch* program akan memunculkan pesan error tetapi masih berjalan dengan baik.



*Running program* terdapat *Exception Missmatch*

*Exception* ini terjadi karena adanya kesalahan tipe data yang di*input*kan oleh *user*, didalam program ini seharusnya memasukan tipe data *integer*.



*Running program* terdapat *Exception* tetapi ditangani dengan *try-catch*

Masih sama, disini adalah bagian *try*-*catch* kedua. Program sebelumnya akan terhenti jika *user* memasukan *input* selain *integer*, tetapi setelah dilakukan *try*-*catch* program akan memunculkan pesan error tetapi masih berjalan dengan baik, jika diperhatikan stukturnya hampir mirip seperti *if else*.

1. Buat program untuk menampilkan dan melakukan *try*-*catch finally.*

package cobaexceptionmulti;

import java.util.Scanner;

import java.util.InputMismatchException;

public class CobaExceptionMulti {

public static void main(String[] args) {

Scanner Input = new Scanner(System.in);

try{

System.out.print("Masukan Bilangan : ");

int bil1 = Input.nextInt();

System.out.print("Masukan Pembagi : ");

int bil2 = Input.nextInt();

int hasil = bil1/bil2;

System.out.println(bil1 + " : " + bil2 + " = " + hasil + (" dibulatkan"));

} catch(ArithmeticException e){

System.out.println("Error : Tidak dapat melakukan kalkulasi");

} catch(InputMismatchException e){

System.out.println("Error : Tipe data bukan integer");

} finally{

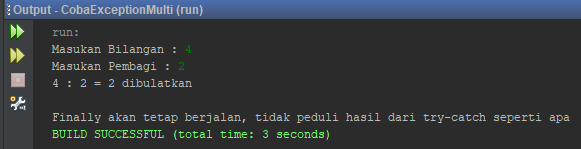
System.out.println("\nFinally akan tetap berjalan, tidak peduli hasil dari try-catch seperti apa");

}

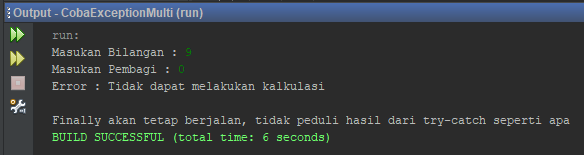
}

}

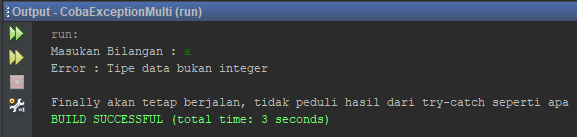
Masih menggunakan *Source code class* Coba*ExceptionMulti*



*Running program* tanpa adanya melakukan *Exception*



*Running program* terdapat *Exception* *Arithmetic* teratasi



*Running program* terdapat *Exception Missmatch* teratasi

Pada program kali ini ditambah dengan argumen *keyword* *finally* setelah melakukan *try*-*catch*. Jika program dijalankan maka isi dari *finally* akan tetap dieksekusi dan muncul apapun yang terjadi pada blok *try*-*catch*. *Finally* akan dijalankan setelah *try*-*catch* dilakukan, fungsi dari *finally* lebih digunakan untuk membuka file atau mengedit file maka *finally* akan tetap dijalankan.

1. Buat program untuk menampilkan dan melakukan *Throw*.

package cobathrow;

public class CobaThrow {

public static void methodLain(){

try{

throw new ArrayIndexOutOfBoundsException(1);

} catch(ArrayIndexOutOfBoundsException e){

System.out.println("Penangganan exception dalam methodLain()");

throw e;

}

}

public static void main(String[] args) {

try{

methodLain();

} catch(ArrayIndexOutOfBoundsException e){

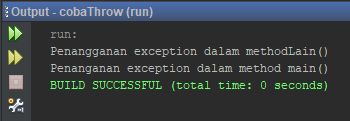
System.out.println("Penanganan exception dalam method main()");

}

}

}

*Source code class* Coba*Throw*



*Running program Exception* dengan *Throw*

Pada program kali ini menggunakan *keyword* *throw* untuk mengatasi *Exception* yaitu pada *method*Lain dan didalam *main*. Mekanisme dari *throw* sendiri memungkinkan program untuk mengirim/melempar *exception* untuk kemudian bereaksi atas *exception* tersebut.

**BAB III**

**PENJELASAN**

**UNGUIDED**

Cari minimal 3 Exception lainnya yang ada didalam Java, buatlah pembahasan mengenai *Exception* tersebut. Sertakan nama *Exception*, penyebab dilemparkannya *Exception* tersebut, *source code* implementasi *try*-*catch* yang menangkap *Exception*, dan lampirkan *screenshot running program*.

1. NullPointerException

package nullexception;

public class Barang {

private String kode;

private String nama;

public void setKode(String Kode) {

try {

kode = Kode;

if (kode == null) {

throw new NullPointerException();

}

} catch (NullPointerException npe) {

System.out.println("ERROR: " + "Kode barang tidak boleh null");

}

}

public String getKode() {

return kode;

}

public void setNama(String Nama) {

try {

nama = Nama;

if (nama == null) {

throw new NullPointerException();

}

} catch (NullPointerException npe) {

System.out.println("ERROR: " +"Nama barang tidak boleh null");

}

}

public String getNama() {

return nama;

}

}

*Source code class* Barang

*Source code class NullExcception* untuk *running program*

package nullexception;

public class NullException {

public static void main(String[] args) {

Barang obj = new Barang();

obj.setKode(null);

obj.setNama("Novel Bumi");

System.out.println("Kode : " + obj.getKode());

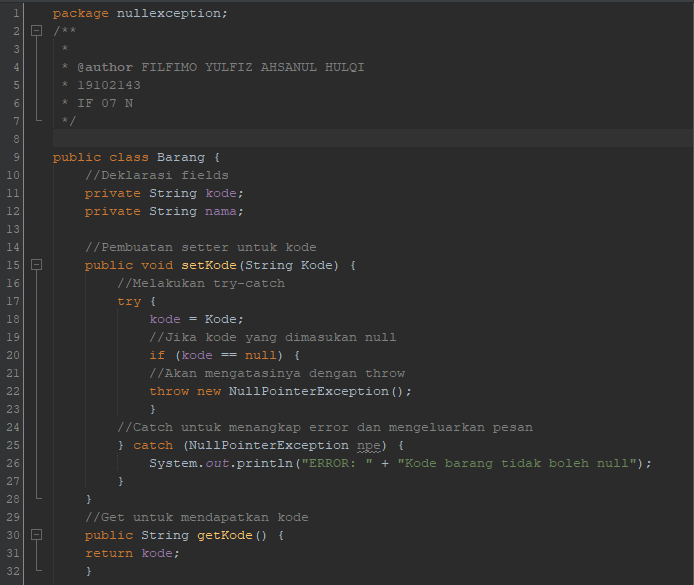
System.out.println("Nama : " + obj.getNama());

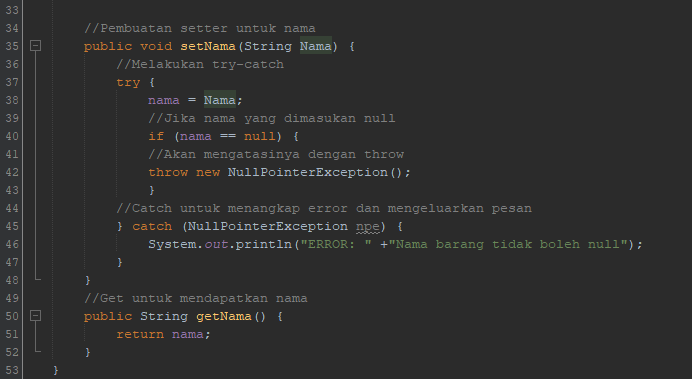
}

}

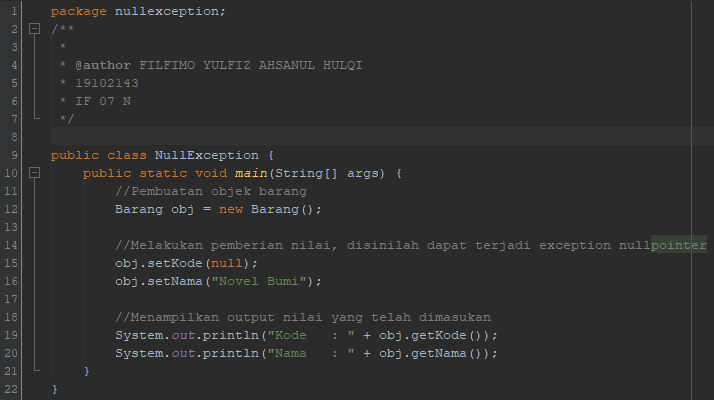
Screenshot program

* *Class* Barang

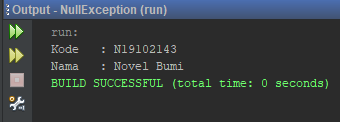




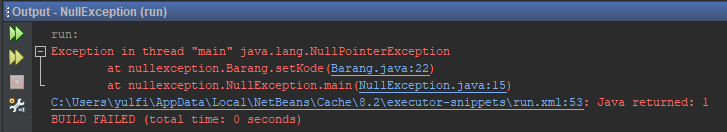
* *Class NullException running program*



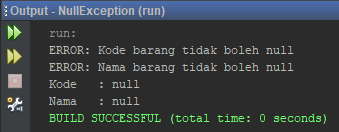
* *Ouput*



*Running program* tanpa *Exception*



*Running program* terdapat *NullPointerException*



*Running program* terdapat *Exception NullPointerException* teratasi

Penjelasan program

Pada program diatas adalah contoh *NullPointerException*, *Exception* ini terjadi ketika program berusaha untuk mengakses suatu atribut atau method dari sebuah objek menggunakan variabel yang tidak merujuk ke sebuah objek. Objek disini bisa berupa data *string, integer,* dan tipe data lainnya atau bahkan dapat berupa *class*.

Dapat dilihat pada *screenshot running* pertama jika *input* sesuai tidak bernilai *null* maka program akan dapat dijalankan, tetapi jika nilai yang di*input* *null*, maka program akan berhenti, kemudian *screenshot running* ketiga adalah hasil dari menggunakan *try-catch* untuk mengatasi *Exception*.

1. NumberFormatException

package cobanumberformat;

import java.util.Scanner;

public class CobaNumberFormat {

public static void main(String[] args) {

int number;

Scanner input = new Scanner(System.in);

while (true) {

System.out.println("Masukan angka Integer: ");

try {

number = Integer.parseInt(input.next());

System.out.println("Anda memasukan angka: " + number);

break;

}

catch (NumberFormatException e) {

System.out.println("NumberFormatException occured");

}

}

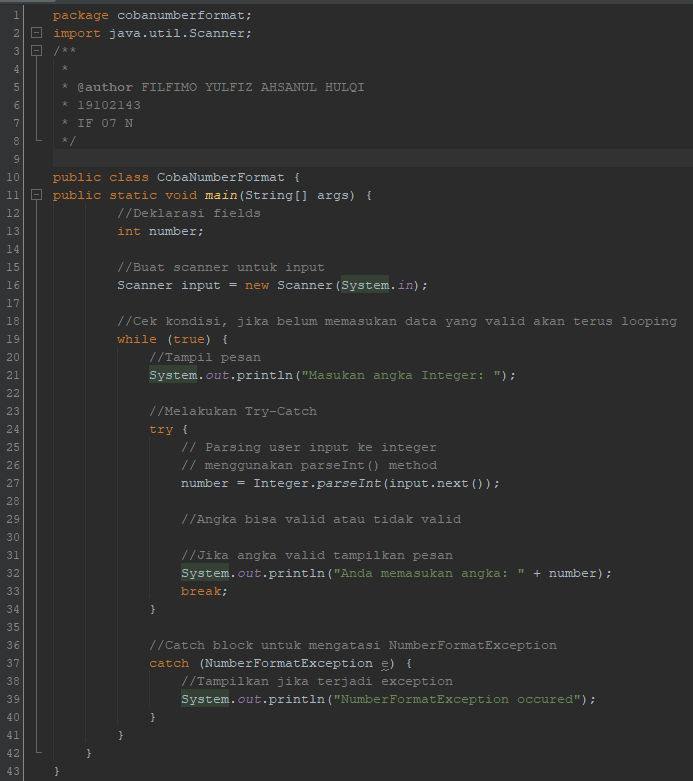
}

}

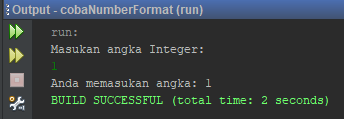
*Source code class* Coba*NumberFormat*

Screenshot program

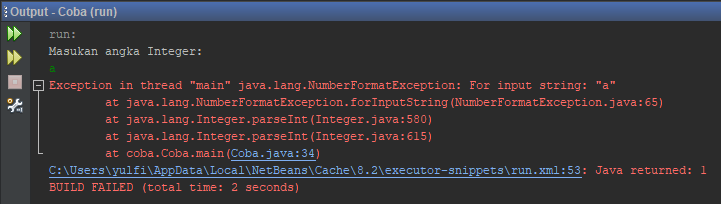
* *Class NumberFormat*



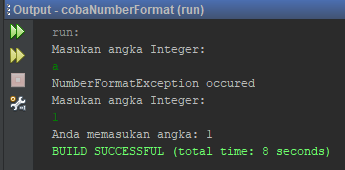
* *Output*



*Running program* tanpa *Exception*



*Running program* terdapat *NumberFormatException*



*Running program* terdapat *Exception NumberFormatException* teratasi

Penjelasan program

Pada program diatas adalah contoh *NumberFormatException*, *Exception* ini terjadi ketika program mencoba untuk mengkoversikan *string* dengan format yang tidak sesuai ke nilai angka, yang berarti, ketika mungkin untuk mengkonversikan *string* ke tipe data numerik (*float, int, double*, dsb), *Exception* ini akan dilempar.

Dapat dilihat pada *screenshot running* pertama jika *input* sesuai maka program akan berjalan seperti biasa, jika *input* tidak sesuai maka program terhenti, kemudian untuk *screenshot running* ketiga adalah hasil dari menggunakan *try-catch* untuk mengatasi *Exception*.

1. FileNotFoundException

package cobafilenotfound;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class CobaFileNotFound {

public static void main(String args[]) {

try {

File file = new File("D://!KULIAH!//!Kuliah!//!KULIAH!//4 SEMESTER "

+ "4//PBO//LP8ProjectFinal//file.txt");

Scanner sc = new Scanner(file);

while (sc.hasNextLine())

System.out.println(sc.nextLine());

} catch (FileNotFoundException e) {

System.out.println("File tidak ditemukan");

}

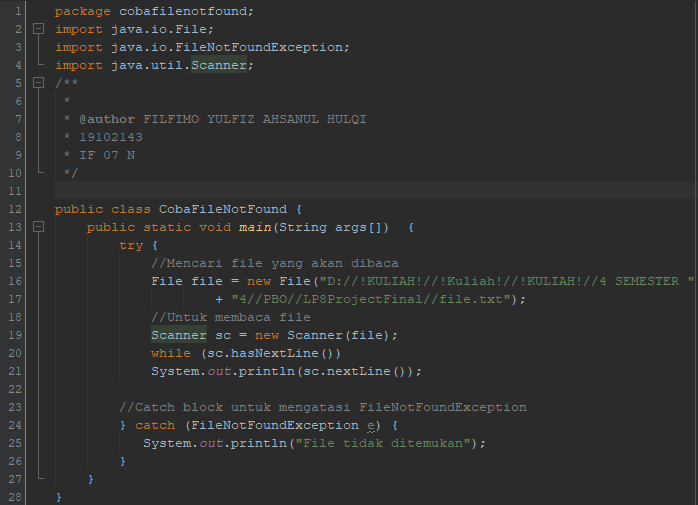
}

}

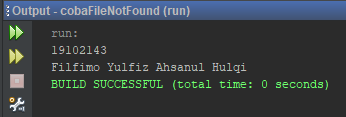
*Source code class* Coba*FileNotFound*

Screenshot program

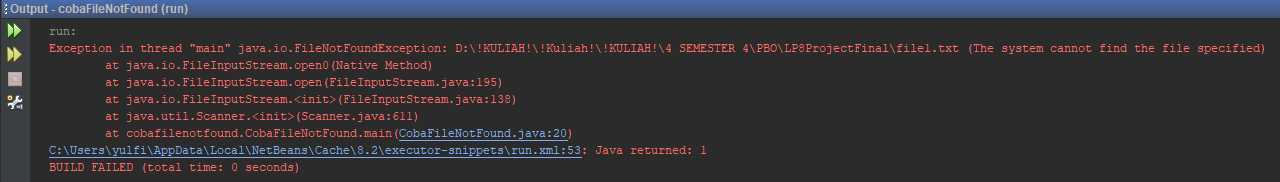
* *Class FileNotFound*



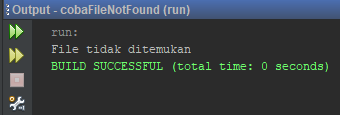
* *Output*



*Running program* tanpa *Exception*



Running program terdapat FileNotFoundException



*Running program* terdapat *Exception FileNotFoundException* teratasi

Penjelasan program

Pada program diatas adalah contoh *FileNotFoundException*, *Exception* ini terjadi ketika program mencoba untuk membuka suatu *file*, tetapi program tersebut tidak dapat membuka *file* tersebut atau *file* tidak dapat ditemukan, maka *Exception* ini akan dilempar.

Dapat dilihat pada *screenshot running* pertama adalah jika *file* dapat ditemukan maka program akan berjalan seperti biasa, jika *file* tidak dapat dibuka atau tidak ditemukan maka program terhenti, kemudian untuk *screenshot running* ketiga adalah hasil dari menggunakan *try-catch* untuk mengatasi *Exception*.

**BAB IV**

**Kesimpulan**

Pada praktikum kali ini dapat diambil beberapa kesimpulan :

1. *Exception* adalah kondisi yang tidak wajar pada program, yang diberitahukan oleh *compiler*.
2. *Exception Handling* adalah teknik yang sangat diperlukan untuk menangani *error* yang terjadi pada saat *runtime* untuk tidak mengganggu seluruh kode pada program.
3. Ada bermacam-macam *keyword* untuk *Exception Handling* yang terdapat dalam bahasa pemrograman Java.

**BAB V**

**DAFTAR PUSTAKA**

1. https://www.geeksforgeeks.org/ (2021, 28 Juni). “Types of Exception in Java with Examples”. Diakses pada 06 Juli 2021 dari <https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/>.
2. https://www.geeksforgeeks.org/ (2021, 25 Mei). “NumberFormatException in Java with Examples”. Diakses pada 06 Juli 2021 dari <https://www.geeksforgeeks.org/numberformatexception-in-java-with-examples/>.
3. http://mooryku.blogspot.com. (2013, 11 November). “Jenis Exception NullPointerException”. Diakses pada 06 Juli 2021 dari http://mooryku.blogspot.com /2013/11/jenis-exception-nullpointerexception.html.