Nama : Faizal Rahman
NIM : 20230040262
Kelas : TI23F

Percobaan 1

Jalankan program dibawah ini, amati error yang terjadi. Kemudian perbaiki seperti pada program dibawahnya.

```
public class Exception {

    public static void main(String[] args) {        int a[]=new int[5];
a[5]=100;
    }
}
```

Pembetulan program:

```
public class Exception {

    public static void main(String[] args)
{         int a[]=new int[5];         try
{
            a[5]=100;
        }
        catch(Exception e)
        {
            System.out.println("Terjadi pelanggaran memory");
        }
    }

}
```

Pengamatan eror:

Percobaan 2

Jalankan program dibawah ini, amati error yang terjadi. Kemudian perbaiki seperti pada program dibawahnya.

```
public class Exception2 {
    public static void main(String[] args) {
       int i=0;
       String greeting[]={
           "Hello World!",
           "No, I mean it!",
           "Hello World"
       };        while(i<4)
       {
           System.out.println(greeting[i]);            i++;
```

```
        }
    }
}
```

Pembetulan program:

```
public class Exception2 {
    public static void main(String[] args) {
int i=0;
        String greetings[]={
"Hello World!",
            "No,I mean it!",
            "HELLO WORLD!"
        };
        while(i<4)
        {
try
            {

System.out.println(greetings[i]);
i++;             }
            catch(ArrayIndexOutOfBoundsException e)
            {
                System.out.println("Resetting index value");
i=0;
            }
        }
    }
}
```

Pengamatan eror:

Error terjadi karena program mencoba mengakses elemen array dengan indeks 3, padahal array hanya memiliki indeks 0 sampai 2. Hal ini menyebabkan ArrayIndexOutOfBoundsException. Lalu, ditambahkan blok try-catch untuk menangani error ArrayIndexOutOfBoundsException. Pada saat indeks keluar dari batas array, program akan menampilkan pesan dan mereset indeks ke 0 sehingga program tidak berhenti dan terus menampilkan elemen array secara berulang.

Percobaan 3

Jalankan program dibawah ini, amati error yang terjadi. Kemudian perbaiki seperti pada program dibawahnya.

```
public class Exception3 {

    public static void main(String[] args) {         int bil=10;
        System.out.println(bil/0);
    }
}
```

Pembetulan Program:

```
public class Exception3 {

    public static void main(String[] args)
{         int bil=10;         try         {
            System.out.println(bil/0);
        }
        catch(Exception e)
        {
            System.out.println("Ini menghandle error yang terjadi");
        }
    }
}
```

Pembetulan Program:

```
public class CobaException3 {

    public static void main(String[] args)
{          int bil=10;          try          {
             System.out.println(bil/0);
         }
catch(ArithmeticException e)
         {
             System.out.println("Terjadi Aritmatika error");
         }
catch(Exception e)
         {
             System.out.println("Ini menghandle error yang terjadi");
         }
    }
}
```

Pengamatan Eror:

Program mengalami error karena mencoba melakukan pembagian bilangan dengan nol. Hal ini menyebabkan ArithmeticException dengan pesan "/ by zero". Lalu,
ditambahkan blok try-catch dengan Exception untuk menangani semua jenis error, dan menampilkan pesan “Ini menghandle error yang terjadi”.
Ditambahkan blok try-catch khusus dengan ArithmeticException agar lebih spesifik menangani error pembagian dengan nol, dengan pesan “Terjadi Aritmatika error”.

Percobaan 4

Jalankan program dibawah ini, amati error yang terjadi. Kemudian perbaiki seperti pada program dibawahnya.

```
public class CobaException4 {
    public static void main(String[] args) {
int bil=10;
       String
b[]={"a","b","c"};        try
{
           System.out.println(b[3]);
           System.out.println(bil/0);
       }
       catch(ArithmeticException e)
       {
           System.out.println("Terjadi Aritmatika error");
       }
       catch(ArrayIndexOutOfBoundsException e)
       {
           System.out.println("Melebihi jumlah array");
       }
       catch(Exception e)
       {
           System.out.println("Ini menghandle error yang terjadi");
       }
       }
}
```

Pembetulan Program:

```
public class CobaException4 {
    public static void main(String[] args) {
int bil=10;
       String
b[]={"a","b","c"};        try
{
           System.out.println(bil/0);
           System.out.println(b[3]);
       }
       catch(ArithmeticException e)
       {
           System.out.println("Terjadi Aritmatika error");
       }
       catch(ArrayIndexOutOfBoundsException e)
       {
           System.out.println("Melebihi jumlah array");
       }
       catch(Exception e)
       {
           System.out.println("Ini menghandle error yang terjadi");
       }
       }
}
```

Pengamatan Eror:

Program error karena mencoba mengakses elemen array pada indeks 3, sedangkan array hanya memiliki indeks 0 sampai 2. Hal ini menimbulkan ArrayIndexOutOfBoundsException. Lalu,
Urutan perintah di dalam blok try diubah, sehingga operasi pembagian (bil/0) dilakukan lebih dulu sebelum mengakses elemen array yang salah. Dengan urutan baru ini, error ArithmeticException muncul lebih dulu dan ditangani, sehingga program menampilkan pesan “Terjadi Aritmatika error”.

Percobaan 5

Jalankan program dibawah ini, berikan analisa penggunaan try dan catch pada program dibawah ini.

```
public class Exception5 {

    public static void main(String[] args)
{         int bil=10;         try         {
            System.out.println(bil/0);
        }
        catch(ArithmeticException e)
        {
            System.out.println("Pesan error:  ");
            System.out.println(e.getMessage());
System.out.println("Info stack erase");
            e.printStackTrace();
            e.printStackTrace(System.out);
}
        catch(Exception e)
        {
            System.out.println("Ini menghandle error yang terjadi");
        }
    }

}
```

Analisa penggunaan Try dan Catch:

Program ini menggunakan blok try untuk mengeksekusi operasi pembagian bilangan dengan nol yang berpotensi menyebabkan error. Saat error ArithmeticException terjadi, blok catch yang sesuai akan menangkapnya dan menampilkan pesan error menggunakan getMessage serta menampilkan jejak error melalui printStackTrace. Selain itu, disiapkan juga blok catch(Exception e) sebagai cadangan untuk menangani jenis error lainnya, meskipun dalam kasus ini tidak terpakai karena error sudah ditangani oleh ArithmeticException. Dengan adanya try-catch, program dapat menampilkan informasi detail error tanpa terhenti secara mendadak.

Percobaan 6

Jalankan program dibawah ini, berikan analisa penggunaan try dan catch pada program dibawah ini.

```
public class ThrowExample {

    static void demo()
    {
        NullPointerException t;
        t=new NullPointerException("Coba Throw");         throw t;
```

```
        // Baris ini tidak lagi dikerjakan;
        System.out.println("Ini tidak lagi dicetak");
    }

    public static void main(String[] args)
{         try         {
            demo();
            System.out.println("Selesai");
        }
        catch(NullPointerException e)
        {
            System.out.println("Ada pesan error: "+e);
        }
    }

}
```

Analisa penggunaan Try dan Catch:
Program ini menggunakan blok try untuk menjalankan metode demo() yang secara sengaja melempar (throw) sebuah objek NullPointerException. Blok catch dengan tipe NullPointerException disiapkan untuk menangkap exception yang dilempar tersebut dan menampilkan pesan error yang berisi informasi exception. Dengan mekanisme try-catch ini, program mampu menangani exception yang dibuat sendiri tanpa menghentikan jalannya program secara mendadak, dan memberikan pesan yang jelas kepada pengguna.

Percobaan 7

Jalankan program dibawah ini, berikan analisa penggunaan try dan catch pada program dibawah ini.

```
public class ThrowExample2 {

    public static void main(String[] args)
{         try         {
            throw new Exception("Here's my Exception");
        }
        catch(Exception e)
        {
            System.out.println("Caught Exception");
            System.out.println("e.getMessage():"+e.getMessage());
            System.out.println("e.toString():"+e.toString());
System.out.println("e.printStackTrace():");
            e.printStackTrace();
        }
    }
}
```

Analisa penggunaan Try dan Catch:
Program ini menggunakan blok try untuk secara sengaja melempar (throw) sebuah objek Exception dengan pesan tertentu. Blok catch dengan tipe Exception menangkap exception tersebut dan menampilkan berbagai informasi terkait error, seperti pesan error dengan getMessage(), deskripsi lengkap exception dengan toString(), serta detail jejak error menggunakan printStackTrace(). Dengan adanya try-catch, program dapat memproses dan melaporkan error secara lengkap tanpa terhenti mendadak saat exception dilempar.

Percobaan 8

Jalankan program dibawah ini, berikan analisa penggunaan throws pada program dibawah ini.

```
import java.io.*;

public class Test3 {
public void methodA(){
        System.out.println("Method A");
    }
    public void methodB() throws IOException
    {
        System.out.println(20/0);
        System.out.println("Method B");
    }
}
```

```
class Utama
{
     public static void main(String[] args) throws IOException
     {
        Test3 c=new Test3();
        c.methodA();
        c.methodB();
     }
}
```

Kemudian coba ubah class utama diatas dengan yang program baru di bawah ini:

```
class Utama
{
     public static void main(String[] args)
     {
            Test3 o=new Test3();
            o.methodA();
try
            {
                 o.methodB();
}
            catch(Exception e)
            {
                System.out.println("Error di Method B");
            }
finally
            {
                System.out.println("Ini selalu dicetak");
            };
     }
}
```

Analisa penggunaan Throw:

Pada program ini sebenarnya tidak ada penggunaan eksplisit throw untuk melempar exception secara manual, melainkan error muncul akibat operasi pembagian dengan nol pada methodB. Meskipun methodB mendeklarasikan throws IOException, exception yang benar-benar terjadi adalah ArithmeticException yang dilempar secara otomatis oleh Java saat program mencoba melakukan 20/0. Pada versi awal program, karena tidak ada blok try-catch, exception ini langsung menyebabkan program berhenti dan menampilkan error. Pada perbaikan, exception ditangani menggunakan blok try-catch di dalam main, sehingga error dapat diantisipasi dengan menampilkan pesan “Error di Method B” tanpa menghentikan program, dan blok finally memastikan pesan “Ini

selalu dicetak” tetap dieksekusi. Dengan demikian, program tidak melempar exception menggunakan throw secara manual, tetapi mendemonstrasikan bagaimana exception yang dilempar otomatis bisa diantisipasi.

Percobaan 9

Jalankan program membalik string (reverse) berikut ini, coba hapus isi dari method reverse (`”This is a string”)` kemudian jalankan kembali.

```
class Propagate {

    public static void main(String[] args)
    {
try
{
            System.out.println(reverse("This is a string"));
        }
        catch(Exception e)
        {
            System.out.println("The String was blank");
        }
finally
        {
            System.out.println("All done");
        }
    }

    public static String reverse(String s) throws Exception
    {
         if(s.length()==0)
         {
              throw new Exception();
         }
         String reverseStr = "";          for(int i=s.length()-1 ; i>=0 ; --i){
reverseStr+=s.charAt(i);
         }
         return reverseStr;
    }

}
```

Analisa hasil menjalankan program:
Program menjalankan method reverse untuk membalik string. Saat diberikan input string "This is a string", program berhasil membalik string tersebut dan menampilkannya. Jika input string dikosongkan atau dihapus dari pemanggilan method, program tidak bisa dijalankan karena terjadi error di tahap kompilasi: reverse cannot be resolved to a variable. Ini karena method reverse memerlukan argumen bertipe String dan pemanggilan tanpa argumen dianggap tidak valid oleh Java, sehingga error muncul sebelum program sempat dieksekusi.

Percobaan 10

Program berikut ini menerapkan IOException ketika membuat objek dari class RandomAccessFile (java.io) yang menghasilkan file txt.

```
class RandomAccessRevisi
{
    public static void main(String[] args) {

        String
bookList[]={"Satu","Dua","Tiga"};         int
yearList[]={1920,1230,1940};         try
{
            RandomAccessFile books = new RandomAccessFile
("books.txt","rw");

            for(int i=0;i<3;i++)
            {
                 books.writeUTF(bookList[i]);
books.writeInt(yearList[i]);
            }
            books.seek(0);
            System.out.println(books.readUTF()+" "+books.readInt());
            System.out.println(books.readUTF()+ " "+books.readInt());
books.close();

}
        catch(IOException e)
        {
            System.out.println("Indeks melebihi batas");
        }
        System.out.println("test");
    }

}
```

Analisa hasil menjalankan program:
Program ini membuat file books.txt menggunakan class RandomAccessFile dari package java.io. Di dalam blok try, program menulis tiga data string dan integer ke file, masing-masing mewakili nama buku dan tahun. Setelah penulisan, program mengatur pointer file kembali ke awal (seek(0)) lalu membaca dan menampilkan dua data pertama: "Satu 1920" dan "Dua 1230". Program berjalan lancar tanpa error sehingga blok catch tidak dieksekusi. Setelah selesai, program menutup file dan menampilkan "test" di akhir sebagai tanda program berakhir. File books.txt berhasil dibuat dan berisi data sesuai yang ditulis.

Percobaan 11

Program berikut ini menunjukkan proses throw and catch pada class yang extends Throwable.

```
class RangeErrorException extends Throwable
{
    public RangeErrorException(String s)
    {
         super(s);     }
```

```
    public static void main(String[] args)
    {
         int
position=1;
try          {
             if(position>0)
             {
                 throw new RangeErrorException("Position " +position);
             }
}
         catch(RangeErrorException e)
         {
             System.out.println("Range  error:  "  +e.getMessage());
}
         System.out.println("This is the last program.");
    }
}
```

Analisa hasil menjalankan program:
Program ini membuat class RangeErrorException yang mewarisi Throwable dan digunakan untuk melempar exception secara eksplisit. Pada main, program memeriksa nilai position, dan jika lebih dari 0, exception RangeErrorException dilempar dengan pesan tertentu. Blok catch menangkap exception tersebut dan mencetak pesan: "Range error: Position 1". Setelah penanganan exception selesai, program tetap melanjutkan eksekusi dan mencetak "This is the last program.". Program menunjukkan bagaimana custom exception dapat dibuat dengan mewarisi Throwable, lalu dilempar dan ditangani tanpa menghentikan seluruh program.

Percobaan 12

Program berikut ini menunjukkan proses throw and catch pada class yang extends Exception.

```
class MyException extends Exception{

    private String Teks;
MyException(String s)
    {
        Teks="Exception generated by: "+s;
        System.out.println(Teks);
    }
}
class Eksepsi
{
    static void tampil(String s)throws Exception
    {
         System.out.println("Tampil");
if(s.equals("amir"))
         {
               throw new MyException(s);
         }
         System.out.println("OK!");
    }

    public static void main(String[] args)throws Exception
    {
try
{
            tampil("ali");
tampil("amir");
        }
        catch(MyException ex)
        {
            System.out.println("Tangkap:"+ex);
        }
    }
}
```

Analisa hasil menjalankan program:

Program ini mendemonstrasikan bagaimana custom exception dibuat dengan mewarisi Exception. Class MyException menampilkan pesan saat objek exception dibuat. Pada main, method tampil dipanggil dua kali: pertama dengan "ali", kedua dengan "amir". Saat "ali" diproses, tidak terjadi error sehingga tampil "Tampil" dan "OK!". Saat "amir" diproses, program membuat dan melempar MyException, menampilkan pesan "Exception generated by: amir". Exception ini ditangkap blok catch, yang kemudian mencetak "Tangkap:MyException". Program berhenti setelah menangani exception, menunjukkan proses throw dan catch berjalan dengan baik pada custom exception yang mewarisi Exception.