Nama : Ramanda Setiawan
NIM : 20230040054
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:
Karena program mencoba mengakses elemen array dengan indeks 3 meskipun array hanya memiliki indeks 0 hingga 2, terjadi error ArrayIndexOutOfBoundsException. Untuk memperbaikinya, blok try-catch ditambahkan. Program akan menampilkan pesan dan mereset indeks ke 0 saat indeks keluar dari batas array. Ini memungkinkan program untuk terus menampilkan elemen array secara berulang tanpa berhenti.

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");
         }
    }
}
```

- **PengamatanEror:**
  Program mengalami error saat mencoba membagi angka dengan nol. Ini menghasilkan ArithmeticException dengan pesan "by zero". Kemudian, ditambahkan blok try-catch dengan Exception untuk menangani semua jenis kesalahan dan menampilkan pesan "Ini menangani kesalahan yang terjadi." Selanjutnya, ditambahkan blok try-catch khusus dengan ArithmeticException untuk menangani kesalahan pembagian dengan nol dengan lebih tepat, dengan pesan "Terjadi Aritmatika kesalahan".

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 mengalami error karena mencoba mengakses elemen array pada indeks 3, sedangkan array hanya memiliki indeks 0 hingga 2. Hal ini menyebabkan Exception ArrayIndexOutOfBounds. Kemudian, urutan perintah di blok try diubah sehingga operasi pembagian (bil/0) dilakukan lebih dulu sebelum mengakses elemen array yang

salah . Dengan urutan baru ini, error ArithmeticException muncul dan ditangani lebih dulu. Akibatnya, 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 memakai blok try untuk menjalankan operasi pembagian dengan nol, yang berisiko menimbulkan error. Jika terjadi kesalahan berupa ArithmeticException, blok catch yang sesuai akan menangkapnya, lalu menampilkan pesan error melalui getMessage dan memperlihatkan jejak error dengan printStackTrace. Selain itu, tersedia juga blok catch(Exception e) sebagai antisipasi untuk menangani jenis error lain, walaupun dalam situasi ini tidak digunakan karena error sudah diatasi oleh ArithmeticException. Dengan mekanisme try-catch ini, program tetap bisa menampilkan informasi detail mengenai error tanpa berhenti secara tiba-tiba.

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:**

  Untuk menjalankan metode demo(), program ini menggunakan blok try 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 menggunakan mekanisme try-catch ini, program dapat menangani exception yang dibuat sendiri tanpa menghentikan program secara mendadak dan memberikan pesan yang jelas kepada penggunanya tentang bagaimana menangani exception tersebut.

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:**

Blok try digunakan dalam program untuk secara sengaja melempar (throw) sebuah objek Exception dengan pesan khusus. Blok catch tipe Exception menangkap exception tersebut dan menampilkan berbagai informasi terkait error, seperti pesan error dengan getMessage(), deskripsi lengkap exception dengan toString(), dan detail jejak error dengan printStackTrace. Adanya try-catch memungkinkan program untuk memproses dan melaporkan kesalahan secara lengkap tanpa terhenti secara 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 method B. Meskipun metode B mendeklarasikan pelemparan IOException, exception yang sebenarnya terjadi adalah ArithmeticException, yang dilempar secara otomatis oleh Java saat program mencoba melakukan 20/0. Tidak ada penggunaan eksplisit melempar untuk melempar exception secara manual dalam program ini. Pada versi awal program, kesalahan ini langsung menyebabkan program berhenti dan menampilkan kesalahan karena tidak ada blok try-catch. Namun, pada perbaikan, kesalahan ditangani di dalam main, sehingga error dapat diantisipasi dengan menampilkan pesan " "Kesalahan di Metode B" terus dieksekusi tanpa menghentikan program, dan blok akhirnya memastikan bahwa pesan "Kesalahan di Metode B" selalu dicetak. Oleh karena itu, program menunjukkan cara mengantisipasi exception yang dilempar secara otomatis daripada menggunakan throw untuk melempar exception secara manual.

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 ini menggunakan method reverse untuk membalikkan sebuah string. Ketika input string yang diberikan adalah "This is a string", program berhasil memproses dan menampilkan hasil string yang sudah terbalik. Namun, jika input string tidak diisi atau pemanggilan method dilakukan tanpa argumen, program akan gagal dikompilasi dan menghasilkan error: reverse cannot be resolved to a variable. Hal ini terjadi karena method reverse membutuhkan argumen bertipe String, sehingga pemanggilan tanpa argumen dianggap tidak sah oleh Java dan error muncul sebelum program dijalankan.

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 menggunakan class RandomAccessFile dari paket java.io untuk membuat file books.txt. Program menulis tiga data string dan integer ke file di blok try, masing-masing berisi nama buku dan tahun. Setelah penulisan, program mengembalikan pointer file ke awal (seek(0)), membaca dan menampilkan dua data pertama: "One 1920" dan "Two 1230". Blok catch tidak dieksekusi karena program berjalan tanpa

masalah. Program menutup file dan menampilkan "test" di akhir untuk menunjukkan bahwa program selesai. Berhasil dibuat, file books.txt 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 menciptakan class RangeErrorException yang mewarisi Throwable dan digunakan untuk melempar exception. Nilai position diperiksa oleh program saat main; jika nilainya lebih dari 0, exception RangeErrorException dikirim dengan pesan khusus. "Kesalahan jalur: Posisi 1" adalah pesan yang dicetak setelah blok catch menangkap kesalahan tersebut. Setelah penanganan exception selesai, program tetap melanjutkan eksekusi dan mencetak "This is the last program." Ini menunjukkan cara membuat exception khusus dengan mewarisi Throwable, melemparkannya, dan menanganinya tanpa menghentikan seluruh program.

Percobaan 12

```
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);
        }
    }
}
```