NAMA : **REVAN FATKHUREZI DESVIANSYAH**

NIM : **20230040097**

KELAS : **TI23E**

Percobaan 1 - ArrayIndexOutOfBoundsException

Masalah: Akses indeks array yang tidak ada.

```
public class Exception1 {
    public static void main(String[] args) {
        int a[] = new int[5];
        a[5] = 100;
    }
}
```

Solusi dengan Exception Handling:

```
public class Exception1 {
    public static void main(String[] args) {
        int a[] = new int[5];
        try {
            a[5] = 100; // Akses indeks yang tidak ada
        } catch (Exception e) {
            System.out.println("Terjadi pelanggaran memory"); //
Menangkap error
        }
    }
}
```

Penjelasan:

- **Error yang terjadi:** ArrayIndexOutOfBoundsException karena mencoba mengakses indeks ke-5 dari array yang hanya memiliki indeks 0-4.
- **Solusi:** Menggunakan blok try-catch untuk menangkap dan menangani error.

Percobaan 2 - ArrayIndexOutOfBoundsException dengan perulangan

Masalah: Perulangan yang mencoba mengakses indeks di luar batas array.

```
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++;
        }
    }
}
```

Solusi dengan Exception Handling:

```
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; // Mereset i agar tidak keluar dari array
            }
        }
    }
}
```

Penjelasan:

- **Error yang terjadi:** ArrayIndexOutOfBoundsException karena i mencapai 4 yang melebihi panjang array (indeks 0-2).
- **Solusi:** Tangani exception dengan catch dan reset indeks i jika terjadi kesalahan.
-

Percobaan 3 - ArithmeticException (Pembagian dengan 0)

Masalah: Pembagian dengan angka 0.

```
public class Exception3 {
    public static void main(String[] args) {
        int bil = 10;
        System.out.println(bil / 0); // Pembagian dengan 0
    }
}
```

Solusi dengan Exception Handling:

```
public class Exception3 {
    public static void main(String[] args) {
        int bil = 10;
        try {
            System.out.println(bil / 0); // Pembagian dengan 0
        } catch (Exception e) {
            System.out.println("Ini menghandle error yang terjadi"); //
Menangkap error
        }
    }
}
```

Penjelasan:

- **Error yang terjadi:** ArithmeticException karena pembagian dengan 0.
- **Solusi:** Gunakan blok try-catch untuk menangkap dan memberikan pesan yang sesuai.

Percobaan 4 - Menggunakan Multiple Catch

Masalah: Beberapa jenis error dalam satu program.

```
public class Exception4 {
    public static void main(String[] args) {
        int bil = 10;
        String b[] = { "a", "b", "c" };
        try {
            System.out.println(b[3]); // ArrayIndexOutOfBoundsException
            System.out.println(bil / 0); // ArithmeticException
        } 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");
        }
    }
}
```

Penjelasan:

- **Error yang terjadi:** ArrayIndexOutOfBoundsException dan ArithmeticException.
- **Solusi:** Menangani berbagai jenis error dengan beberapa blok catch.

Percobaan 5 - Menangkap dan Menampilkan Pesan Exception

Masalah: Menampilkan informasi lebih lanjut tentang error.

```java
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: " + e.getMessage());
            e.printStackTrace(); // Menampilkan detail stack trace
        } catch (Exception e) {
            System.out.println("Ini menghandle error yang terjadi");
        }
    }
}
```

Penjelasan:

- **Error yang terjadi:** ArithmeticException.
- **Solusi:** Menampilkan pesan error dan stack trace untuk analisis lebih lanjut.

Percobaan 6 - Throwing Custom Exception

Masalah: Melemparkan exception secara manual.

```
public class ThrowExample {
    static void demo() {
        NullPointerException t;
        t = new NullPointerException("Coba Throw");
        throw t; // Melemparkan exception secara manual
    }
    public static void main(String[] args) {
        try {
            demo();
            System.out.println("Selesai");
        } catch (NullPointerException e) {
            System.out.println("Ada pesan error: " + e);
        }
    }
}
```

Penjelasan:

- **Melemparkan exception:** Menggunakan throw untuk melempar exception secara manual.
- **Solusi:** Tangkap exception yang dilempar dengan catch.

Percobaan 7 - Exception dengan e.printStackTrace()

Masalah: Menangkap exception dan menampilkan detailnya.

```java
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");
            e.printStackTrace(); // Menampilkan detail stack trace
        }
    }
}
```

Penjelasan:

- **Error yang terjadi:** Exception dilempar secara manual.
- **Solusi:** Menangkap dan menampilkan detail exception menggunakan e.printStackTrace().

Percobaan 8 - Menggunakan Throws

Masalah: Menggunakan throws untuk menangani exception yang dilempar oleh method lain.

```java
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(); // Akan menyebabkan error
    }
}
```

Penjelasan:

- **Error yang terjadi:** ArithmeticException karena pembagian dengan 0 di methodB.
- **Solusi:** Menggunakan throws untuk melempar exception yang mungkin terjadi.

Percobaan 9 – Membalik String dengan Exception

Masalah: Menangani string kosong (blank) yang dilempar sebagai exception.

```
public class Propagate {

    public static void main(String[] args) {
        try {
            System.out.println(reverse("This is a string")); // akan
berhasil
            // System.out.println(reverse("")); // uncomment ini untuk
melihat exception
        } 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(); // Lempar error kalau string kosong
        }
        String reverseStr = "";
        for (int i = s.length() - 1; i >= 0; --i) {
            reverseStr += s.charAt(i);
        }
        return reverseStr;
    }
}
```

Penjelasan:

- Jika string kosong dikirim ke reverse(), maka method akan melempar exception.
- Exception ditangkap di main(), dan finally akan **selalu dijalankan**.

Percobaan 10 – IOException dan File

Masalah: Menulis dan membaca file menggunakan RandomAccessFile.

```
import java.io.*;

public 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); // Kembali ke awal file
            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");
    }
}
```

Penjelasan:

- RandomAccessFile digunakan untuk menulis dan membaca data dari file.
- Exception IOException ditangkap jika file tidak dapat dibuat/dibaca.

Percobaan 11 – Custom Exception dengan Throwable

Masalah: Membuat exception sendiri dari kelas 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.");
    }
}
```

Penjelasan:

- Kelas RangeErrorException dibuat sendiri dan **turunannya adalah Throwable**.
- Exception dilempar manual dengan throw dan ditangkap menggunakan catch.

Percobaan 12 – Custom Exception dengan Exception

Masalah: Membuat exception sendiri yang diturunkan dari 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");   // tidak melempar exception
            tampil("amir");  // melempar MyException
        } catch (MyException ex) {
            System.out.println("Tangkap: " + ex);
        }
    }
}
```

Penjelasan:

- MyException adalah kelas exception kustom yang memperluas Exception.
- Exception hanya dilempar jika input adalah "amir".