LAPORAN PRAKTIKUM TEKNIK PEMROGRAMAN
PERTEMUAN KEENAM: DEFENSIVE PROGRAMMING

Oleh:

Ihsan Fauzi 241524048

PROGRAM STUDI D4-TEKNIK INFORMATIKA

JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2025

DAFTAR ISI

BAB I
PENDAHULUAN

1. Latar belakang
   Exception handling digunakan untuk mengatasi kesalahan atau kondisi tidak terduga yang dapat terjadi selama eksekusi program. Exception memberikan cara yang terstruktur untuk mendeteksi dan menangani error runtime tanpa menyebabkan program berhenti secara tidak wajar.

2. Tujuan
   Laporan ini bertujuan untuk menganalisis dan mendemonstrasikan implementasi penanganan exception dalam tiga kasus yang berbeda:
   - Menangani karakter non-huruf dalam program penghitung frekuensi huruf
   - Menangani token non-integer dalam program penjumlahan integer
   - Melempar exception untuk kondisi input yang tidak valid dalam penghitungan faktorial

3. Jenis-jenis Exception di java
   - Checked Exception: Exception yang wajib ditangani dengan try-catch atau dideklarasikan dengan throws. Contoh: IOException, SQLException
   - Unchecked Exception: Exception yang tidak wajib ditangani secara eksplisit. Contoh: ArrayIndexOutOfBoundsException, NullPointerException, IllegalArgumentException

4. Penanganan Exception

   - **try**: Menandai blok kode yang berpotensi menyebabkan kesalahan.
   - **catch**: Menangani exception yang terjadi dalam blok try.
   - **throw**: Digunakan untuk melemparkan exception secara manual.
   - **finally**: Selalu dieksekusi, digunakan untuk membersihkan sumber daya.

BAB II
CASE 1

1. Tujuan
   Case ini bertujuan menganalisis implementasi penanganan exception pada program CountLetters.java, yang menghitung frekuensi huruf dalam kata yang dimasukkan pengguna. Program asli mengalami masalah ketika karakter non-huruf dimasukkan, menyebabkan ArrayIndexOutOfBoundsException. Case ini menunjukkan cara menangani exception tersebut secara efektif.
2. Analisis
   Program asli bekerja sebagai berikut:
   - Meminta pengguna memasukkan sebuah kata
   - Mengubah kata menjadi huruf kapital
   - Menghitung kemunculan setiap huruf dengan menggunakan kode karakter sebagai indeks
   - Menampilkan frekuensi setiap huruf

   Masalah utama: Ketika karakter non-huruf (spasi, tanda baca, dll.) dimasukkan, ekspresi word.charAt(i)-'A' menghasilkan indeks di luar rentang valid (0-25) untuk array counts.
3. Hasil modifikasi sourcecode

```
//************************************************************
// CountLetters.java
//
// Reads a words from the standard input and prints the number of
// occurrences of each letter in that word.
//
//************************************************************
import java.util.Scanner;
public class CountLetters{
    public static void main(String[] args){
        int[] counts = new int[26];
        Scanner scan = new Scanner(System.in);
//get word from user
        System.out.print("Enter a single word (letters only,
please): ");
        String word = scan.nextLine();
//convert to all upper case
        word = word.toUpperCase();
//count frequency of each letter in string
        for (int i=0; i < word.length(); i++) {
            try {
                counts[word.charAt(i)-'A']++;
            } catch (ArrayIndexOutOfBoundsException e) {
                System.out.println("Not a letter: " +
word.charAt(i));
            }
        }
        //print frequencies
        System.out.println();
        for (int i=0; i < counts.length; i++)
            if (counts [i] != 0)
                System.out.println((char)(i +'A') + ": " +
counts[i]);
    }
}
```

4. Penjelasan perubahan (ArrayIndexOutOfBoundsException)
    A. Definisi
    ArrayIndexOutOfBoundsException adalah exception runtime yang terjadi ketika mencoba mengakses elemen array dengan indeks yang berada di luar rentang valid untuk array tersebut (baik negatif atau lebih besar dari atau sama dengan panjang array).
    B. Penggunaan
    Dalam kode yang dimodifikasi, kita menempatkan operasi akses array (counts[word.charAt(i)-'A']++) di dalam blok try dan menangkap ArrayIndexOutOfBoundsException yang dihasilkan ketika karakter non-huruf diproses. Ini memungkinkan kita untuk:
        - Mendeteksi karakter non-huruf tanpa validasi karakter eksplisit
        - Menanganinya dengan baik dengan menampilkan pesan tanpa menyebabkan program crash
        - Melanjutkan pemrosesan karakter-karakter lain dalam input
5. Test Case
    A. Test Case 1

```
Enter a single word (letters only, please): Hello

E: 1
H: 1
L: 2
O: 1
```

    B. Test Case 2

```
Enter a single word (letters only, please): Hello World
Not a letter:

D: 1
E: 1
H: 1
L: 3
O: 2
R: 1
W: 1
```

BAB III
CASE 2

1. Tujuan
   Case ini bertujuan untuk membaca baris teks dan menjumlahkan semua integer yang ada di dalamnya. Masalah muncul ketika input mengandung token yang bukan integer, menyebabkan NumberFormatException.
2. Analisis
   Program asli bekerja sebagai berikut:
   - Meminta dan membaca baris input dari pengguna
   - Menggunakan Scanner kedua untuk memproses baris tersebut token per token
   - Mengurai (parse) integer dari setiap token dan menjumlahkannya
   - Mencetak jumlah tersebut

   Masalah utama: Ketika token bukan integer (misalnya kata), Integer.parseInt() akan melempar NumberFormatException.

3. Hasil modifikasi sourcecode

```
//
*****************************************************************
// ParseInts.java
//
// Reads a line of text and prints the integers in the line.
//
//
*****************************************************************
import java.util.Scanner;
public class ParseInts{
    public static void main(String[] args){
        int val, sum=0;
        Scanner scan = new Scanner(System.in);
        String line;
        System.out.println("Enter a line of text");
        Scanner scanLine = new Scanner(scan.nextLine());
        while (scanLine.hasNext()) {
            try{
                val = Integer.parseInt(scanLine.next());
                sum += val;
            } catch (NumberFormatException e) {  // Lanjutkan ke
token berikutnya tanpa melakukan apa-apa
            }
        }
        System.out.println("The sum of the integers on this line
is " + sum);
    }
}
```

4. Penjelasan perubahan (NumberFormatException)
   A. Definisi
      NumberFormatException adalah exception runtime yang dilempar ketika mencoba mengonversi string ke tipe numerik, tetapi string tersebut tidak memiliki format yang sesuai.
   B. Penggunaan

Dalam kode yang dimodifikasi, operasi parsing integer (Integer.parseInt(scanLine.next())) ditempatkan di dalam blok try dan menangkap NumberFormatException yang dihasilkan ketika token bukan integer. Yang memungkinkan kita untuk:

- Mendeteksi token non-integer tanpa menghentikan program
- Melewatkan token non-integer dan melanjutkan ke token berikutnya
- Mengumpulkan semua integer yang valid dalam baris input

5. Test Case
   A. Test Case 1 (hanya integer)

```
Enter a line of text
10 20 30 40
The sum of the integers on this line is 100
```

   B. Test Case 2

```
Enter a line of text
5 kelinci dan 10 kancil
The sum of the integers on this line is 15
```

BAB IV
CASE 3

1. Tujuan
   Case ini menghitung faktorial dari integer yang dimasukkan oleh pengguna menggunakan metode faktorial dari kelas MathUtils. Program ini memiliki dua masalah utama: mengembalikan 1 untuk input negatif (yang secara matematis tidak tepat) dan mengembalikan nilai negatif untuk input besar karena overflow (<17).

2. Analisis

   Program asli bekerja sebagai berikut:

   - Meminta pengguna memasukkan integer
   - Memanggil metode factorial untuk menghitung faktorial
   - Menampilkan hasilnya dan bertanya apakah pengguna ingin menghitung faktorial lain

   Masalah utama:

   - Faktorial untuk bilangan negatif seharusnya tidak didefinisikan
   - Faktorial untuk bilangan lebih besar dari 16 menyebabkan overflow pada tipe int

3. Hasil modifikasi sourcecode

```
//
*******************************************************************
// Factorials.java
// Reads integers from the user and prints the factorial of each.
//
//
*******************************************************************
import java.util.Scanner;
public class Factorials{
    public static void main(String[] args){
        String keepGoing = "y";
        Scanner scan = new Scanner(System.in);
        while (keepGoing.equals("y") || keepGoing.equals("Y")){
            System.out.print("Enter an integer: ");
            int val = scan.nextInt();

            try {
                System.out.println("Factorial(" + val + ") = " +
MathUtils.factorial(val));
            } catch (IllegalArgumentException e){
                System.out.println("Error: "+ e.getMessage());
            }
            System.out.print("Another factorial? (y/n) ");
            keepGoing = scan.next();
        }
    }
}
```

```
//
*****************************************************************
// MathUtils.java
//
// Provides static mathematical utility functions.
//
//
*****************************************************************
public class MathUtils{
    //-----------------------------------------------------------------
-
// Returns the factorial of the argument given
//-----------------------------------------------------------------
    public static int factorial(int n) throws
IllegalArgumentException{
        if (n<0){
            throw new IllegalArgumentException("Faktorial tidak
didefinisikan untuk bilangan negatif");
        }
        if (n>16){
            throw new IllegalArgumentException("Nilai terlalu
besar, akan menyebabkan overflow (maksimum adalah 16)");
        }
        int fac = 1;
        for (int i=n; i>0; i--)
            fac *= i;
        return fac;
    }
}
```

4. Penjelasan perubahan (IllegalArgumentException)
    A. Definisi
    IllegalArgumentException adalah exception runtime yang dilempar untuk menunjukkan bahwa metode telah diberikan argumen yang tidak sesuai atau tidak valid.
    B. Penggunaan
    Dalam kode yang dimodifikasi, kita memodifikasi metode factorial untuk memeriksa nilai argumen dan melempar IllegalArgumentException jika nilai tersebut negatif atau lebih besar dari 16. Ini memungkinkan kita untuk:
        - Menangani kondisi input yang tidak valid dengan tepat
        - Memberikan pesan yang jelas tentang masalah spesifik
        - Mencegah hasil yang tidak benar (seperti mengembalikan 1 untuk input negatif)
5. Test Case

```
Enter an integer: 10
Factorial(10) = 3628800
Another factorial? (y/n) y
Enter an integer: -5
Error: Faktorial tidak didefinisikan untuk bilangan negatif
Another factorial? (y/n) y
Enter an integer: 18
Error: Nilai terlalu besar, akan menyebabkan overflow (maksimum adalah 16)
Another factorial? (y/n) n

Process finished with exit code 0
```

BAB V
KESIMPULAN

Dari tiga kasus di atas, dapat disimpulkan bahwa defensive programming dengan exception handling sangat penting untuk meningkatkan ketahanan program terhadap error runtime. Setiap kasus menunjukkan pendekatan yang berbeda:

Case 1 menunjukkan cara menangani input yang tidak sesuai dengan menangkap ArrayIndexOutOfBoundsException, memungkinkan program tetap berjalan tanpa crash.

Case 2 menyoroti pentingnya menangani format data yang tidak valid dengan NumberFormatException, sehingga program tetap dapat memproses data yang benar.

Case 3 memperlihatkan bagaimana IllegalArgumentException dapat digunakan untuk mencegah hasil yang tidak valid akibat input yang tidak sesuai dalam perhitungan matematis.

Dengan menerapkan defensive programming yang baik, program dapat menjadi lebih andal, mencegah kesalahan yang tidak terduga, dan memberikan umpan balik yang lebih jelas kepada pengguna.

BAB VI
LINK GITHUB

https://github.com/isanzzi/Exception-error-java