Praktikum Pertemuan 9

Disusun untuk memenuhi tugas mata kuliah Pemrograman Berorientasi Objek (Praktek)



**Disusun Oleh:**

**Dwika Ali Ramdhan (231511042) 2B – D3**

**Jurusan Teknik Komputer dan Informatika**

**Program Studi D-3 Teknik Informatika Politeknik Negeri Bandung**

**2024**

Contents

[1. Case 1 : Exceptions Aren't Always Errors 3](#_Toc180878913)

[1.1. Solusi 1 3](#_Toc180878914)

[1.2. Solusi 2 3](#_Toc180878915)

[2. Case 2 : Placing Exception Handlers 4](#_Toc180878916)

[2.1. Output 5](#_Toc180878917)

[2.2. Penjelasan 5](#_Toc180878918)

[3. Case 3 : Throwing Exceptions 5](#_Toc180878919)

[3.1. Factorials 5](#_Toc180878920)

[3.2. MathUtils 6](#_Toc180878921)

[3.3. Output 6](#_Toc180878922)

[3.4. Penjelasan 7](#_Toc180878923)

[LAMPIRAN 8](#_Toc180878924)

## Case 1 : Exceptions Aren't Always Errors

### Solusi 1

import java.util.Scanner;  
  
public class CountLettersno1 {  
 public static void main(String[] args){  
 int[] counts = new int[26];  
 Scanner scan = new Scanner(System.*in*);  
  
 System.*out*.print("Masukkan suatu kata (hanya boleh huruf) : ");  
 String word = scan.nextLine();  
  
 word = word.toUpperCase();  
 for (int i = 0; i < word.length(); i++) {  
 try {  
 counts[word.charAt(i)- 'A']++;  
 } catch (ArrayIndexOutOfBoundsException e){  
 return;  
 }  
 }  
  
 System.*out*.println();  
 for (int i = 0; i < counts.length; i++){  
 if (counts[i] != 0) {  
 System.*out*.println("Huruf " + (char)(i + 'A') + ": " + counts[i]);  
 }  
 }  
 }  
}

#### Output

A black background with white text

Description automatically generated

#### Penjelasan :

penanganan exception dengan try-catch yang kosong (silent exception handling). Ketika program menemui karakter non-huruf, exception akan ditangkap (catch) dan program langsung return (berhenti). Kelemahannya adalah program berhenti total saat menemui karakter non-huruf pertama yang ditemukan.

### Solusi 2

import java.util.Scanner;  
  
public class CountLettersno2 {  
 public static void main(String[] args) {  
 int[] counts = new int[26];  
 Scanner scan = new Scanner(System.*in*);  
  
 System.*out*.print("Masukkan suatu kata (hanya boleh huruf) : ");  
 String word = scan.nextLine();  
  
 word = word.toUpperCase();  
  
 for (int i = 0; i < word.length(); i++) {  
 try {  
 counts[word.charAt(i)-'A']++;  
 } catch (ArrayIndexOutOfBoundsException e) {  
 System.*out*.println("Bukan Merupaak Huruf : " + word.charAt(i));  
 }  
 }  
  
 System.*out*.println("Frekuensi Huruf:");  
 for (int i = 0; i < counts.length; i++) {  
 if (counts[i] != 0) {  
 System.*out*.println("Huruf " + (char)(i + 'A') + ": " + counts[i]);  
 }  
 }  
 }  
}

#### Output

A screenshot of a computer

Description automatically generated

#### Penjelasan :

Pengembangan dari solusi pertama dengan penanganan exception yang lebih baik. Program tidak berhenti saat menemui karakter non-huruf Tetap melanjutkan perhitungan untuk semua huruf yang valid

## Case 2 : Placing Exception Handlers

import java.util.Scanner;  
  
public class ParseInts1 {  
 public static void main(String[] args) {  
 int val, sum = 0;  
 Scanner scan = new Scanner(System.*in*);  
 String line;  
 System.*out*.println("\nEnter another line of text");  
 Scanner scanLine = new Scanner(scan.nextLine());  
  
 try {  
 while (scanLine.hasNext()) {  
 val = Integer.*parseInt*(scanLine.next());  
 sum += val;  
 }  
 } catch (NumberFormatException e) {  
 //Kosong  
 }  
 System.*out*.println("Version 1 - The sum of the integers on this line is " + sum);  
 scanLine.close();  
  
 scanLine = new Scanner(scan.nextLine());  
 sum = 0;  
  
 while (scanLine.hasNext()) {  
 try {  
 val = Integer.*parseInt*(scanLine.next());  
 sum += val;  
 } catch (NumberFormatException e) {  
 //Kosong  
 }  
 }  
  
 System.*out*.println("Version 2 - The sum of the integers on this line is " + sum);  
 scanLine.close();  
 scan.close();  
 }  
}

### Output

A screenshot of a computer program

Description automatically generated

### Penjelasan

Terdapat 2 versi pada program ini :

Versi 1 :Try-catch block berada di luar loop while, Ketika menemui token non-integer, program akan keluar dari loop. Total penjumlahan hanya mencakup angka sebelum token non-integer pertama

Versi 2 : Try-catch block berada di dalam loop while, Setiap token non-integer akan dilewati tanpa menghentikan program, Program tetap melanjutkan pemrosesan untuk mencari angka-angka lain, Total penjumlahan bersaldari angka angka yang telah di lewati.

## Case 3 : Throwing Exceptions

### Factorials

import java.util.Scanner;  
  
public class ParseInts1 {  
 public static void main(String[] args) {  
 int val, sum = 0;  
 Scanner scan = new Scanner(System.*in*);  
 String line;  
 System.*out*.println("\nEnter another line of text");  
 Scanner scanLine = new Scanner(scan.nextLine());  
  
 try {  
 while (scanLine.hasNext()) {  
 val = Integer.*parseInt*(scanLine.next());  
 sum += val;  
 }  
 } catch (NumberFormatException e) {  
 //Kosong  
 }  
 System.*out*.println("Version 1 - The sum of the integers on this line is " + sum);  
 scanLine.close();  
  
 scanLine = new Scanner(scan.nextLine());  
 sum = 0;  
  
 while (scanLine.hasNext()) {  
 try {  
 val = Integer.*parseInt*(scanLine.next());  
 sum += val;  
 } catch (NumberFormatException e) {  
 //Kosong  
 }  
 }  
  
 System.*out*.println("Version 2 - The sum of the integers on this line is " + sum);  
 scanLine.close();  
 scan.close();  
 }  
}

### MathUtils

public class MathUtils1 {  
 public static int factorial(int n) throws IllegalArgumentException {  
 if (n < 0) {  
 throw new IllegalArgumentException("Factorial is not defined for negative numbers: " + n);  
 }  
  
 if (n > 16) {  
 throw new IllegalArgumentException("Factorial overflow: This method can't compute factorial for numbers greater than 16");  
 }  
  
 int fac = 1;  
 for (int i = n; i > 0; i--) {  
 fac \*= i;  
 }  
 return fac;  
 }  
}

### Output

A screen shot of a computer

Description automatically generated

### Penjelasan

MathUtils1:

* Menambahkan throws IllegalArgumentException pada method factorial
* Melakukan dua validasi input:
  1. Menolak angka negatif dengan exception khusus
  2. Menolak angka > 16 untuk mencegah integer overflow
* Memberikan pesan error yang spesifik untuk setiap jenis kesalahan.

Factorials:

* Menggunakan try-catch untuk menangkap IllegalArgumentException
* Menampilkan pesan error yang informatif ke pengguna
* Program tetap berjalan meski terjadi error
* Memungkinkan pengguna mencoba input lain
* Mencegah program crash dengan menangani exception dengan baik

# LAMPIRAN

LINK GITHUB: <https://github.com/DAliRIJTK/PBO_Dwika_231511042.git>