

LAPORAN PRAKTIKUM

Exception Handling



Disusun Oleh:
Dzakir Tsabit Asy Syafiq (241511071)
Jurusan Teknik Komputer dan Informatika

Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung
22/10/2025

Pendahuluan

Praktikum ini bertujuan untuk memahami dan mengimplementasikan mekanisme penanganan eksepsi (exception handling) dalam bahasa pemrograman Java. Fokusnya adalah pada penggunaan try-catch untuk menangani eksepsi yang umum terjadi seperti `ArrayIndexOutOfBoundsException` dan `NumberFormatException`, serta cara melempar eksepsi secara manual (throw) menggunakan `IllegalArgumentException` untuk validasi input.

Pembahasan

Kasus 1: CountLetters.java

Tujuan: Memodifikasi program CountLetters agar dapat menangani input yang bukan huruf (seperti spasi atau tanda baca) tanpa menghentikan program. Program seharusnya mengabaikan karakter non-huruf dan, pada langkah akhir, melaporkan karakter apa yang diabaikan.

Implementasi (Kode Akhir): Berikut adalah kode CountLetters.java yang telah dimodifikasi sesuai instruksi akhir (langkah 1.3):

```
// **  
// 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 class CountLetters {  
    // public static void main(String[] args) {  
    public static void main(String[] args) {  
        // int[] counts = new int[26];  
        int[] counts = new int[26];
```

```

// Scanner scan = new Scanner(System.in);
Scanner scan = new Scanner(System.in);

// System.out.print("Enter a single word (letters only,
please): ");

// Modifikasi untuk mencerminkan instruksi agar bisa
memasukkan frasa
System.out.print("Enter a phrase: ");
// String word = scan.nextLine();
String word = scan.nextLine();

// //convert to all upper case
// word = word.toUpperCase();
word = word.toUpperCase();

// //count frequency of each letter in string
// for (int i=0; i < word.length(); i++) {
for (int i=0; i < word.length(); i++) {
    // Put the body of the first for loop in a try.
    try {
        // counts[word.charAt(i)-'A']++;
        counts[word.charAt(i)-'A']++;
        // Add a catch that catches the exception
    } catch (ArrayIndexOutOfBoundsException e) {
        // In your print statement, replace the exception with
the character that
        // created the out of bounds index.
        System.out.println("'" + word.charAt(i) + "' is not a
letter.");
    }
}

// //print frequencies

```

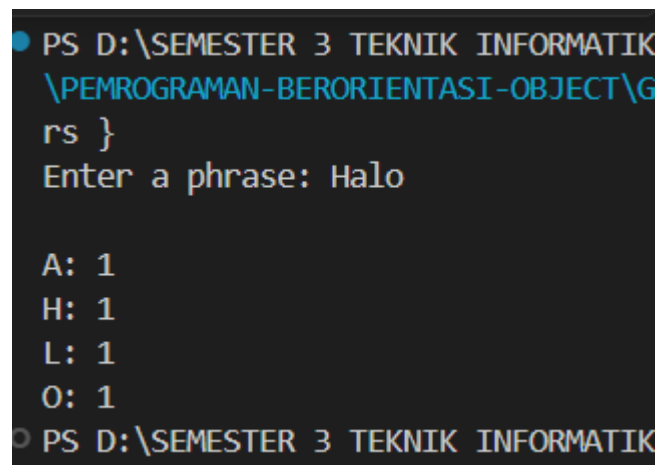
```

        // System.out.println();
        System.out.println();
        // for (int i=0; i < counts.length; i++)
        for (int i=0; i < counts.length; i++)
            // if (counts[i] != 0)
            if (counts[i] != 0)
                // System.out.println((char)(i+'A') + ": " +
counts[i]);
                System.out.println((char)(i+'A') + ": " + counts[i]);

        // }
    }
// }
}

```

Output :

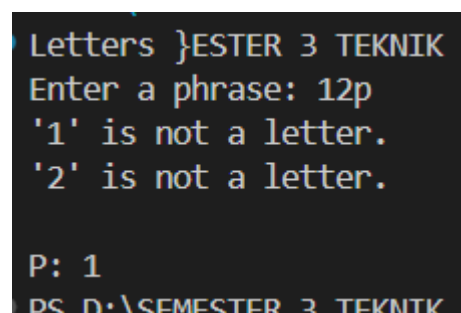


```

PS D:\SEMESTER 3 TEKNIK INFORMATIK
\PEMROGRAMAN-BERORIENTASI-OBJECT\G
rs }
Enter a phrase: Halo

A: 1
H: 1
L: 1
O: 1
PS D:\SEMESTER 3 TEKNIK INFORMATIK

```



```

Letters }ESTER 3 TEKNIK
Enter a phrase: 12p
'1' is not a letter.
'2' is not a letter.

P: 1
PS D:\SEMESTER 3 TEKNIK

```

Penjelasan:

1. Blok try ditempatkan di dalam *looping* for, mengelilingi pernyataan `counts[word.charAt(i)-'A']++`.

2. Ini adalah baris yang berpotensi menimbulkan `ArrayIndexOutOfBoundsException` jika `word.charAt(i)` bukan huruf, yang akan menghasilkan indeks di luar rentang 0-25.
3. Blok catch ditambahkan untuk menangkap `ArrayIndexOutOfBoundsException`.
4. Sesuai instruksi akhir, blok catch diisi dengan pernyataan `println` yang mencetak karakter spesifik yang menyebabkan *exception*, sehingga memberikan umpan balik yang lebih ramah pengguna.

Kasus 2: ParseInts.java

Tujuan: Memodifikasi program `ParseInts` agar dapat membaca satu baris teks yang berisi campuran angka dan kata, kemudian menjumlahkan semua angka yang ada dan mengabaikan token yang bukan angka (non-integer).

Implementasi (Kode Akhir): Berikut adalah kode `ParseInts.java` yang telah dimodifikasi sesuai instruksi akhir (langkah 2.2):

```
// // ParseInts.java
// // Reads a line of text and prints the integers in the line.
// import java.util.Scanner;

// public class ParseInts {
public class ParseInts {
    // public static void main(String[] args) {
    public static void main(String[] args) {
        // int val, sum=0;
        int val, sum=0;

        // Scanner scan = new Scanner(System.in);
        Scanner scan = new Scanner(System.in);
        // String line;
        // System.out.println("Enter a line of text");
        System.out.println("Enter a line of text");
        // Scanner scanLine = new Scanner(scan.nextLine());
        Scanner scanLine = new Scanner(scan.nextLine());

        // while (scanLine.hasNext()) {
        while (scanLine.hasNext()) {
            // To make it continue, move the try and catch inside the loop.
            try {
                // val = Integer.parseInt(scanLine.next());
```

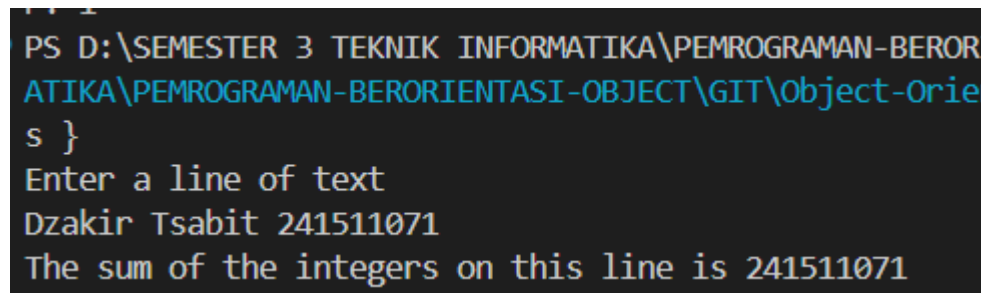
```

        val = Integer.parseInt(scanLine.next());
        // sum += val;
        sum += val;
    // Catch a NumberFormatException
    } catch (NumberFormatException e) {
        // and have an empty body for the catch.
        // Blok catch sengaja dikosongkan untuk mengabaikan token non-integer
        // dan melanjutkan ke iterasi loop berikutnya
    }
// }
}

// System.out.println("The sum of the integers on this line is " + sum);
System.out.println("The sum of the integers on this line is " + sum);
// }
}
}

```

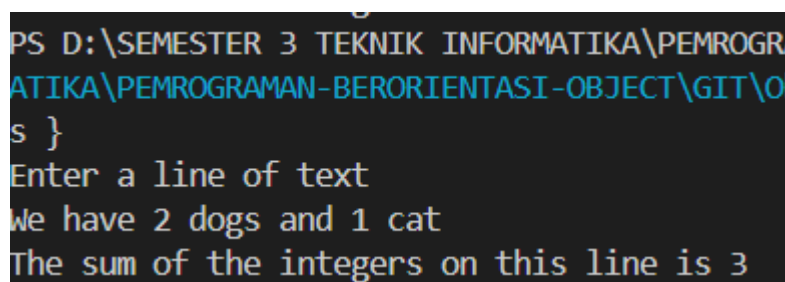
Output:



```

PS D:\SEMESTER 3 TEKNIK INFORMATIKA\PEMROGRAMAN-BERORIENTASI-OBJECT\GIT\Object-Oriented\src> java -cp .\bin\*.jar .\src\Main
Enter a line of text
Dzakir Tsabit 241511071
The sum of the integers on this line is 241511071

```



```

PS D:\SEMESTER 3 TEKNIK INFORMATIKA\PEMROGRAMAN-BERORIENTASI-OBJECT\GIT\Object-Oriented\src> java -cp .\bin\*.jar .\src\Main
Enter a line of text
We have 2 dogs and 1 cat
The sum of the integers on this line is 3

```

Penjelasan:

1. Instruksi awal meminta menempatkan seluruh while loop di dalam try-catch. Namun, ini menyebabkan *loop* berhenti saat `NumberFormatException` pertama kali terjadi.
2. Solusi akhir adalah memindahkan blok try-catch ke *dalam* while loop.
3. Dengan cara ini, jika `Integer.parseInt(scanLine.next())` gagal (misalnya saat membaca "We"), `NumberFormatException` akan ditangkap.

4. Blok catch yang kosong akan dieksekusi, dan *loop* akan melanjutkan ke iterasi berikutnya (`scanLine.hasNext()`). Ini memungkinkan program untuk memproses seluruh baris, menjumlahkan "2" dan "1" dari input "We have 2 dogs and 1 cat." untuk menghasilkan total 3.

Kasus 3: Factorials.java & MathUtils.java

Tujuan: Memodifikasi kelas `MathUtils` untuk melempar `IllegalArgumentException` ketika argumen yang diberikan ke metode `factorial` tidak valid (negatif atau terlalu besar sehingga menyebabkan *overflow*). Selanjutnya, memodifikasi kelas `Factorials` untuk menangani *exception* ini dengan baik tanpa menghentikan program.

Implementasi (Kode Akhir):

1. MathUtils.java (Dimodifikasi)

```
// // MathUtils.java
// // Provides static mathematical utility functions.
// public class MathUtils {
public class MathUtils {
    // // Returns the factorial of the argument given
    // public static int factorial(int n) {
    // Modify the header of the factorial method to indicate that factorial can throw
    public static int factorial(int n) throws IllegalArgumentException {

        // Modify the body of factorial to check the value of the argument and, if it
        is negative,
        if (n < 0) {
            // throw an IllegalArgumentException.
            // Use this parameter to be specific about what the problem is.
            throw new IllegalArgumentException("Factorial is not defined for negative
            integers.");
        }

        // Modify your code in factorial to check for an argument over 16
        if (n > 16) {
            // The problem is arithmetic overflow
            // throw an IllegalArgumentException in either case, but pass different
            messages
            throw new IllegalArgumentException("Argument too large (over 16); result
            will overflow int.");
        }
    }
}
```

```

        // int fac = 1;
        int fac = 1;
        // for (int i=n; i>0; i--)
        for (int i=n; i>0; i--)
            // fac *= i;
            fac *= i;
        // return fac;
        return fac;
    // }
}
// }
}

```

2. Factorials.java (Dimodifikasi)

```

// // Factorials.java
// // Reads integers from the user and prints the factorial of each.
import java.util.Scanner;

// public class Factorials {
public class Factorials {
    // public static void main(String[] args) {
    public static void main(String[] args) {
        // String keepGoing = "y";
        String keepGoing = "y";
        // Scanner scan = new Scanner(System.in);
        Scanner scan = new Scanner(System.in);

        // while (keepGoing.equals("y") || keepGoing.equals("Y")) {
        while (keepGoing.equals("y") || keepGoing.equals("Y")) {
            // Think carefully about where you will need to put the try and catch.
            // Modify the main method... to catch the exception... but then continue
            with the loop.
            // try-catch ditempatkan di dalam loop agar loop tetap berlanjut
            try {
                // System.out.print("Enter an integer: ");

```



```

        System.out.print("Enter an integer: ");
        // int val = scan.nextInt();
        int val = scan.nextInt();

        // System.out.println("Factorial(\"" + val + "\") = " +
MathUtils.factorial(val));

        System.out.println("Factorial(\"" + val + "\") = " +
MathUtils.factorial(val));

        // catch the exception thrown by factorial
    } catch (IllegalArgumentException e) {
        // and print an appropriate message
        System.out.println(e.getMessage());
    }

    // System.out.print("Another factorial? (y/n) ");
    System.out.print("Another factorial? (y/n) ");
    // keepGoing = scan.next();
    keepGoing = scan.next();
    }
    // }
    }
}

```

Output:

```
als }
Enter an integer: 16
Factorial("16") = 2004189184
Another factorial? (y/n) y
Enter an integer: 10
Factorial("10") = 3628800
Another factorial? (y/n) y
Enter an integer: 2
Factorial("2") = 2
Another factorial? (y/n) y
Enter an integer: 1
Factorial("1") = 1
Another factorial? (y/n) y
Enter an integer: 5
Factorial("5") = 120
Another factorial? (y/n) █
```

Penjelasan:

1. **MathUtils.java:** Metode factorial sekarang memiliki dua pemeriksaan *guard clause* di bagian atas.
 - Satu untuk $n < 0$ dan satu lagi untuk $n > 16$ (karena faktorial dari 17 sudah melebihi nilai maksimum int).
 - Kedua kondisi ini akan throw new `IllegalArgumentException`, tetapi dengan pesan kesalahan yang berbeda untuk menjelaskan masalahnya secara spesifik.
2. **Factorials.java:**
 - Blok try-catch ditambahkan di dalam while loop.
 - Blok try berisi kode yang meminta input pengguna dan memanggil `MathUtils.factorial`.
 - Blok catch menangani `IllegalArgumentException` yang mungkin dilempar oleh factorial.
 - Alih-alih menghentikan program, blok catch hanya mencetak pesan dari *exception* (misalnya, "Factorial is not defined for negative integers."), dan program melanjutkan ke bagian akhir *loop* untuk bertanya "Another factorial? (y/n)".

Kesimpulan

Praktikum ini telah berhasil mendemonstrasikan tiga konsep inti dari penanganan eksepsi:

1. **Menangani Eksepsi Bawaan (Case 1):** Menggunakan try-catch untuk menangani `ArrayIndexOutOfBoundsException` sebagai bagian dari alur program normal, bukan sebagai kesalahan fatal.
2. **Penempatan try-catch (Case 2):** Memahami perbedaan kritis antara menempatkan try-catch di luar *loop* (menghentikan semua iterasi) dan di dalam *loop* (hanya mengabaikan iterasi saat ini dan melanjutkan).
3. **Melempar Eksepsi (Case 3):** Menggunakan throw untuk memberlakukan validasi input (aturan bisnis) dalam sebuah metode dan menangani eksepsi tersebut di level pemanggil (*caller*).