

# **Laporan Pengerjaan Praktikum Pertemuan 10**

## **Defensive Programming**



**Disusun oleh :**

**Astria Rizka Latifahsary      231524037**

**Kelas :**

**D4 – 1B Teknik Informatika**

**SARJANA TERAPAN PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA POLITEKNIK  
NEGERI BANDUNG**

**Tahun Ajaran 2023 – 2024**

## Table of Contents

Bab 1 Exceptions Aren't Always Errors .....	4
1.1 Soal .....	4
1.2 Perubahan Kode .....	5
1.3 Hasil Output .....	6
1.4 Penjelasan Kode .....	6
Bab 2 Placing Exception Handlers .....	7
2.1 Soal .....	7
2.2 Perubahan Kode .....	8
2.3 Hasil Output .....	8
2.4 Penjelasan Kode .....	8
Bab 3 Throwing Exceptions .....	10
3.1 Soal .....	10
2.2 Perubahan Kode .....	11
4.3 Hasil Output .....	12
4.4 Penjelasan Kode .....	12
Link Github .....	14

## Daftar Gambar

Gambar 1 Perubahan Kode Case 1.....	5
Gambar 2 Ketika memasukan letters .....	6
Gambar 3 Menampilkan Non Letter (disini spasi) .....	6
Gambar 4 Perubahan Kode Case 2.....	8
Gambar 5 Hasil Output Case 2 .....	8
Gambar 6 Perubahan Pada Factorials.java .....	11
Gambar 7 perubahan pada MathUtils.java .....	12
Gambar 8 Hasil Output Case 3 .....	12

# Bab 1

## Exceptions Aren't Always Errors

### 1.1 Soal

File CountLetters.java contains a program that reads a word from the user and prints the number of occurrences of each letter in the word. Save it to your directory and study it, then compile and run it to see how it works. In reading the code, note that the word is converted to all upper case first, then each letter is translated to a number in the range 0..25 (by subtracting 'A') for use as an index. No test is done to ensure that the characters are in fact letters.

1. Run CountLetters and enter a phrase, that is, more than one word with spaces or other punctuation in between. It should throw an `ArrayIndexOutOfBoundsException`, because a non-letter will generate an index that is not between 0 and 25. It might be desirable to allow non-letter characters, but not count them. Of course, you could explicitly test the value of the character to see if it is between 'A' and 'Z'. However, an alternative is to go ahead and use the translated character as an index, and catch an `ArrayIndexOutOfBoundsException` if it occurs. Since you want don't want to do anything when a non-letter occurs, the handler will be empty. Modify this method to do this as follows: • Put the body of the first for loop in a try. • Add a catch that catches the exception, but don't do anything with it. Compile and run your program.
2. Now modify the body of the catch so that it prints a useful message (e.g., "Not a letter") followed by the exception. Compile and run the program. Although it's useful to print the exception for debugging, when you're trying to smoothly handle a condition that you don't consider erroneous you often don't want to. In your print statement, replace the exception with the character that created the out of bounds index. Run the program again; much nicer!

```
//*****  
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++)
            counts[word.charAt(i)-'A']++;

        //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]);
    }
}

```

## 1.2 Perubahan Kode



```

1  package Case1;
2  import java.util.Scanner;
3
4  public class CountLetters {
5      public static void main(String[] args) {
6          int[] counts = new int[26];
7          Scanner scan = new Scanner(System.in);
8
9          try {
10             // Get word from user
11             System.out.print("Enter a single word (letters only, please): ");
12             String word = scan.nextLine();
13
14             // Convert to all upper case
15             word = word.toUpperCase();
16
17             // Count frequency of each letter in string
18             for (int i = 0; i < word.length(); i++)
19                 counts[word.charAt(i) - 'A']++;
20
21             // Print frequencies
22             System.out.println();
23             for (int i = 0; i < counts.length; i++)
24                 if (counts[i] != 0)
25                     System.out.println((char) (i + 'A') + ": " + counts[i]);
26         } catch (ArrayIndexOutOfBoundsException e) {
27             char invalidChar = findInvalidCharacter(e.getMessage());
28             System.out.println("Not a letter: " + invalidChar);
29         } catch (Exception e) {
30             System.out.println("An error occurred: " + e.getMessage());
31         } finally {
32             scan.close();
33         }
34     }
35
36     private static char findInvalidCharacter(String message) {
37         for (char c : message.toCharArray()) {
38             if (!Character.isLetter(c)) {
39                 return c;
40             }
41         }
42         return '?'; // Default if no invalid character found
43     }
44 }
45

```

Gambar 1 Perubahan Kode Case 1

### 1.3 Hasil Output

```
Enter a single word (letters only, please): astria
A: 2
I: 1
R: 1
S: 1
T: 1
```

*Gambar 2 Ketika memasukan letters*

```
Enter a single word (letters only, please): as tria
Not a letter:
```

*Gambar 3 Menampilkan Non Letter (disini spasi)*

### 1.4 Penjelasan Kode

#### 1. Handling Non-Letter Characters:

- Modifikasi metode **main** untuk menangkap **ArrayIndexOutOfBoundsException** yang mungkin terjadi saat mengakses indeks di luar rentang karena karakter yang bukan huruf. Namun, dalam blok **catch**, tidak ada tindakan yang diambil karena karakter non-huruf tidak ingin dihitung.
- Kompleksitas kode dapat dikurangi dengan menempatkan bagian dari loop pertama di dalam blok **try**, dan menambahkan blok **catch** yang tidak melakukan apa pun dengan pengecualian yang ditangkap.

#### 2. Print Useful Message for Non-Letter Characters:

- Modifikasi isi blok **catch** untuk mencetak pesan yang berguna (misalnya, "Not a letter") diikuti dengan karakter yang menyebabkan indeks keluar dari batas. Dengan cara ini, pesan yang lebih informatif akan dicetak ketika karakter non-huruf ditemukan, yang lebih baik daripada hanya mencetak pengecualian.
- Pesan yang dihasilkan akan memberikan informasi yang lebih berguna kepada pengguna daripada hanya mencetak pengecualian.

## Bab 2

### Placing Exception Handlers

#### 2.1 Soal

File ParseInts.java contains a program that does the following:

- Prompts for and reads in a line of input
- Uses a second Scanner to take the input line one token at a time and parses an integer from each token as it is extracted.
- Sums the integers.
- Prints the sum.

Save ParseInts to your directory and compile and run it. If you give it the input

10 20 30 40

it should print

The sum of the integers on the line is 100.

Try some other inputs as well. Now try a line that contains both integers and other values, e.g.,

We have 2 dogs and 1 cat.

You should get a `NumberFormatException` when it tries to call `Integer.parseInt` on "We", which is not an integer. One way around this is to put the loop that reads inside a try and catch the `NumberFormatException` but not do anything with it. This way if it's not an integer it doesn't cause an error; it goes to the exception handler, which does nothing. Do this as follows:

- Modify the program to add a try statement that encompasses the entire while loop. The try and opening `{` should go before the while, and the catch after the loop body. Catch a `NumberFormatException` and have an empty body for the catch.
- Compile and run the program and enter a line with mixed integers and other values. You should find that it stops summing at the first non-integer, so the line above will produce a sum of 0, and the line "1 fish 2 fish" will produce a sum of 1. This is because the entire loop is inside the try, so when an exception is thrown the loop is terminated. To make it continue, move the try and catch inside the loop. Now when an exception is thrown, the next statement is the next iteration of the loop, so the entire line is processed. The dogs-and-cats input should now give a sum of 3, as should the fish input.

```
//*****  
// 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()){
            val = Integer.parseInt(scanLine.next());
            sum += val;
        }
        System.out.println("The sum of the integers on this line is " + sum);
    }
}

```

## 2.2 Peubahan Kode



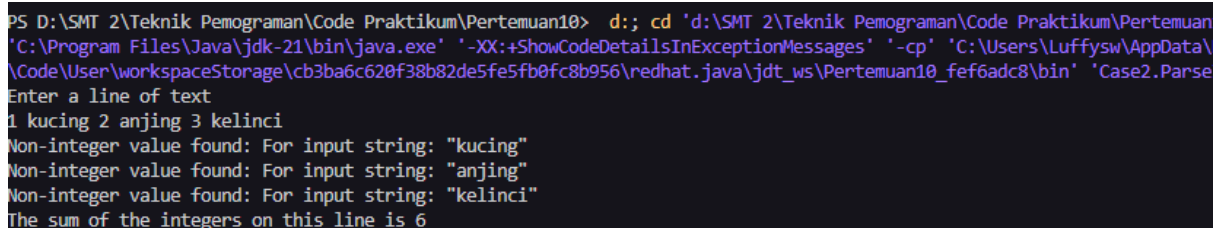
```

1  package Case2;
2  import java.util.Scanner;
3
4  // *****
5  // ParseInts.java
6  //
7  // Reads a line of text and prints the integers in the line.
8  //
9  // *****
10 public class ParseInts {
11     public static void main(String[] args) {
12         int val, sum = 0;
13         Scanner scan = new Scanner(System.in);
14         String line;
15
16         System.out.println("Enter a line of text");
17         Scanner scanLine = new Scanner(scan.nextLine());
18         while (scanLine.hasNext()) {
19             try {
20                 val = Integer.parseInt(scanLine.next());
21                 sum += val;
22             } catch (NumberFormatException e) {
23                 System.out.println("Non-integer value found: " + e.getMessage());
24             }
25         }
26         scan.close();
27         scanLine.close();
28
29         System.out.println("The sum of the integers on this line is " + sum);
30     }
31 }

```

Gambar 4 Perubahan Kode Case 2

## 2.3 Hasil Output



```

PS D:\SMT 2\Teknik Pemograman\Code Praktikum\Pertemuan10> d.; cd 'd:\SMT 2\Teknik Pemograman\Code Praktikum\Pertemuan
'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Luffysw\AppData\
\Code\User\workspaceStorage\cb3ba6c620f38b82de5fe5fb0fc8b956\redhat.java\jdt_ws\Pertemuan10_fef6adc8\bin' 'Case2.Parse
Enter a line of text
1 kucing 2 anjing 3 kelinci
Non-integer value found: For input string: "kucing"
Non-integer value found: For input string: "anjing"
Non-integer value found: For input string: "kelinci"
The sum of the integers on this line is 6

```

Gambar 5 Hasil Output Case 2

## 2.4 Penjelasan Kode



Kode ini merupakan program sederhana yang membaca baris teks yang dimasukkan oleh pengguna, kemudian mencari dan menjumlahkan semua bilangan bulat yang ada dalam baris tersebut.

1. Program dimulai dengan mengimpor kelas `Scanner` untuk membaca input dari pengguna.
2. Di dalam metode **main**, kita menginisialisasi variabel **val** untuk menyimpan setiap bilangan bulat yang diproses, dan **sum** untuk mengakumulasi jumlah dari semua bilangan bulat.
3. Pengguna diminta untuk memasukkan sebuah baris teks.
4. Objek **Scanner** yang disebut **scanLine** dibuat untuk membaca token-token dari baris input.
5. Masuk ke dalam loop **while** yang akan terus berjalan selama masih ada token tersedia dalam baris input.
6. Di dalam loop, kita mencoba untuk mengurai setiap token sebagai bilangan bulat menggunakan **Integer.parseInt(scanLine.next())**.
7. Jika penguraian berhasil, kita menambahkan bilangan bulat yang terurai ke dalam **sum**. Jika penguraian gagal (karena menemukan token yang bukan bilangan bulat), sebuah **NumberFormatException** akan dilemparkan.
8. Kita menangkap **NumberFormatException** dan mencetak pesan yang menunjukkan bahwa sebuah nilai yang bukan bilangan bulat ditemukan.
9. Setelah loop selesai, kita menutup scanner untuk melepaskan sumber daya.
10. Terakhir, kita mencetak jumlah dari semua bilangan bulat yang diurai dari baris input.

Untuk memodifikasi program sesuai instruksi:

- Pindahkan blok **try** dan **catch** ke dalam loop **while** untuk memastikan bahwa setiap token diproses secara individu.
- Hapus isi dari blok **catch**, sehingga loop dapat melanjutkan pemrosesan token bahkan jika terjadi **NumberFormatException**.

Modifikasi ini memastikan bahwa program terus menjumlahkan bilangan bulat bahkan jika menemui token yang bukan bilangan bulat, sesuai dengan instruksi yang diberikan.

## Bab 3

### Throwing Exceptions

#### 3.1 Soal

File `Factorials.java` contains a program that calls the `factorial` method of the `MathUtils` class to compute the factorials of integers entered by the user. Save these files to your directory and study the code in both, then compile and run `Factorials` to see how it works. Try several positive integers, then try a negative number. You should find that it works for small positive integers (values  $< 17$ ), but that it returns a large negative value for larger integers and that it always returns 1 for negative integers.

1. Returning 1 as the factorial of any negative integer is not correct—mathematically, the factorial function is not defined for negative integers. To correct this, you could modify your `factorial` method to check if the argument is negative, but then what? The method must return a value, and even if it prints an error message, whatever value is returned could be misconstrued. Instead it should throw an exception indicating that something went wrong so it could not complete its calculation. You could define your own exception class, but there is already an exception appropriate for this situation—`IllegalArgumentException`, which extends `RuntimeException`. Modify your program as follows:
  - Modify the header of the `factorial` method to indicate that `factorial` can throw an `IllegalArgumentException`.
  - Modify the body of `factorial` to check the value of the argument and, if it is negative, throw an `IllegalArgumentException`. Note that what you pass to throw is actually an instance of the `IllegalArgumentException` class, and that the constructor takes a `String` parameter. Use this parameter to be specific about what the problem is.
  - Compile and run your `Factorials` program after making these changes. Now when you enter a negative number an exception will be thrown, terminating the program. The program ends because the exception is not caught, so it is thrown by the main method, causing a runtime error.
  - Modify the main method in your `Factorials` class to catch the exception thrown by `factorial` and print an appropriate message, but then continue with the loop. Think carefully about where you will need to put the `try` and `catch`.
2. Returning a negative number for values over 16 also is not correct. The problem is arithmetic overflow—the factorial is bigger than can be represented by an `int`. This can also be thought of as an `IllegalArgumentException`—this `factorial` method is only defined for arguments up to 16. Modify your code in `factorial` to check for an argument over 16 as well as for a negative argument. You should throw an `IllegalArgumentException` in either case, but pass different messages to the constructor so that the problem is clear.

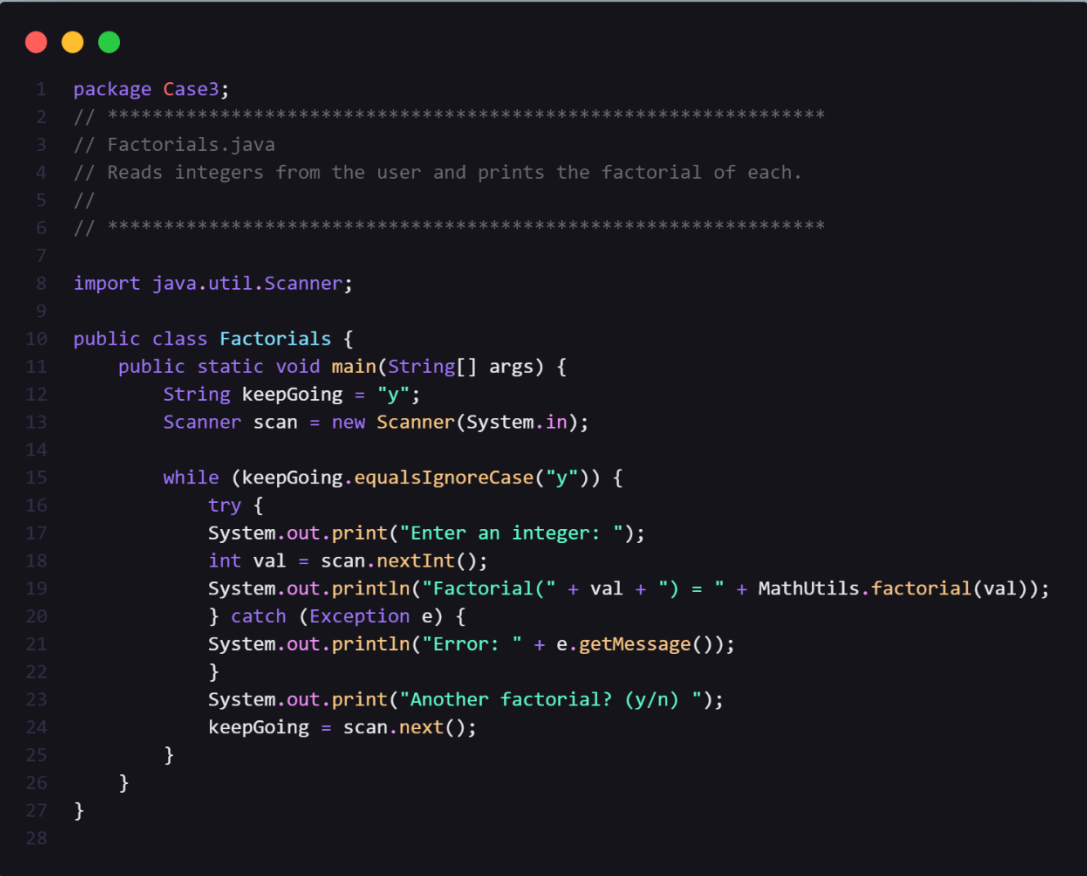
```
// *****  
// Factorials.java  
// Reads integers from the user and prints the factorial of each.  
// *****  
import java.util.Scanner;  
public class Factorials{  
    public static void main(String[] args){  
        String keepGoing = "y";  
        Scanner scan = new Scanner(System.in);  
 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

```

        System.out.println("Factorial(" + val + ") = " +
        MathUtils.factorial(val));
        System.out.print("Another factorial? (y/n) ");
        keepGoing = scan.next();
    }
}

```

## 2.2 Perubahan Kode



```

1  package Case3;
2  // *****
3  // Factorials.java
4  // Reads integers from the user and prints the factorial of each.
5  //
6  // *****
7
8  import java.util.Scanner;
9
10 public class Factorials {
11     public static void main(String[] args) {
12         String keepGoing = "y";
13         Scanner scan = new Scanner(System.in);
14
15         while (keepGoing.equalsIgnoreCase("y")) {
16             try {
17                 System.out.print("Enter an integer: ");
18                 int val = scan.nextInt();
19                 System.out.println("Factorial(" + val + ") = " + MathUtils.factorial(val));
20             } catch (Exception e) {
21                 System.out.println("Error: " + e.getMessage());
22             }
23             System.out.print("Another factorial? (y/n) ");
24             keepGoing = scan.next();
25         }
26     }
27 }
28

```

Gambar 6 Perubahan Pada Factorials.java

```

1 package Case3;
2 // *****
3 // MathUtils.java
4 //
5 // Provides static mathematical utility functions.
6 //
7 // *****
8 public class MathUtils {
9     //-----
10    // Returns the factorial of the argument given
11    //-----
12    public static int factorial(int n) throws IllegalArgumentException {
13        int fac = 1;
14        if (n < 0) {
15            throw new IllegalArgumentException("Argument cannot be negative");
16        }
17        if (n > 16) {
18            throw new IllegalArgumentException("Argument is too large, it may cause arithmetic overflow");
19        }
20        for (int i = n; i > 0; i--) {
21            fac *= i;
22        }
23        return fac;
24    }
25 }

```

*Gambar 7 perubahan pada MathUtils.java*

### 4.3 Hasil Output

```

Enter an integer: -2
Error: Argument cannot be negative
Another factorial? (y/n) y
Enter an integer: 4
Factorial(4) = 24
Another factorial? (y/n) y
Enter an integer: 17
Error: Argument is too large, it may cause arithmetic overflow
Another factorial? (y/n) n

```

*Gambar 8 Hasil Output Case 3*

### 4.4 Penjelasan Kode

1. **Handling Negative Input:**
  - Modifikasi metode **factorial** pada kelas **MathUtils** untuk melempar pengecualian **IllegalArgumentException** jika argumen negatif.
  - Modifikasi metode **main** pada kelas **Factorials** untuk menangkap pengecualian yang dilempar oleh metode **factorial** dan mencetak pesan kesalahan yang sesuai.
2. **Handling Large Input:**

- Modifikasi metode **factorial** pada kelas **MathUtils** untuk melempar pengecualian **IllegalArgumentException** jika argumen lebih dari 16, karena akan menyebabkan overflow.
- Pesan kesalahan yang berbeda akan dilewatkan ke konstruktor pengecualian tergantung pada kondisi argumen.

## **Link Github**

[https://github.com/LuffySW/TekPro\\_Luffysw/tree/main/Pertemuan10](https://github.com/LuffySW/TekPro_Luffysw/tree/main/Pertemuan10)