

LAPORAN PRAKTIKUM 9
PEMROGRAMAN BERORIENTASI OBJEK (PBO)

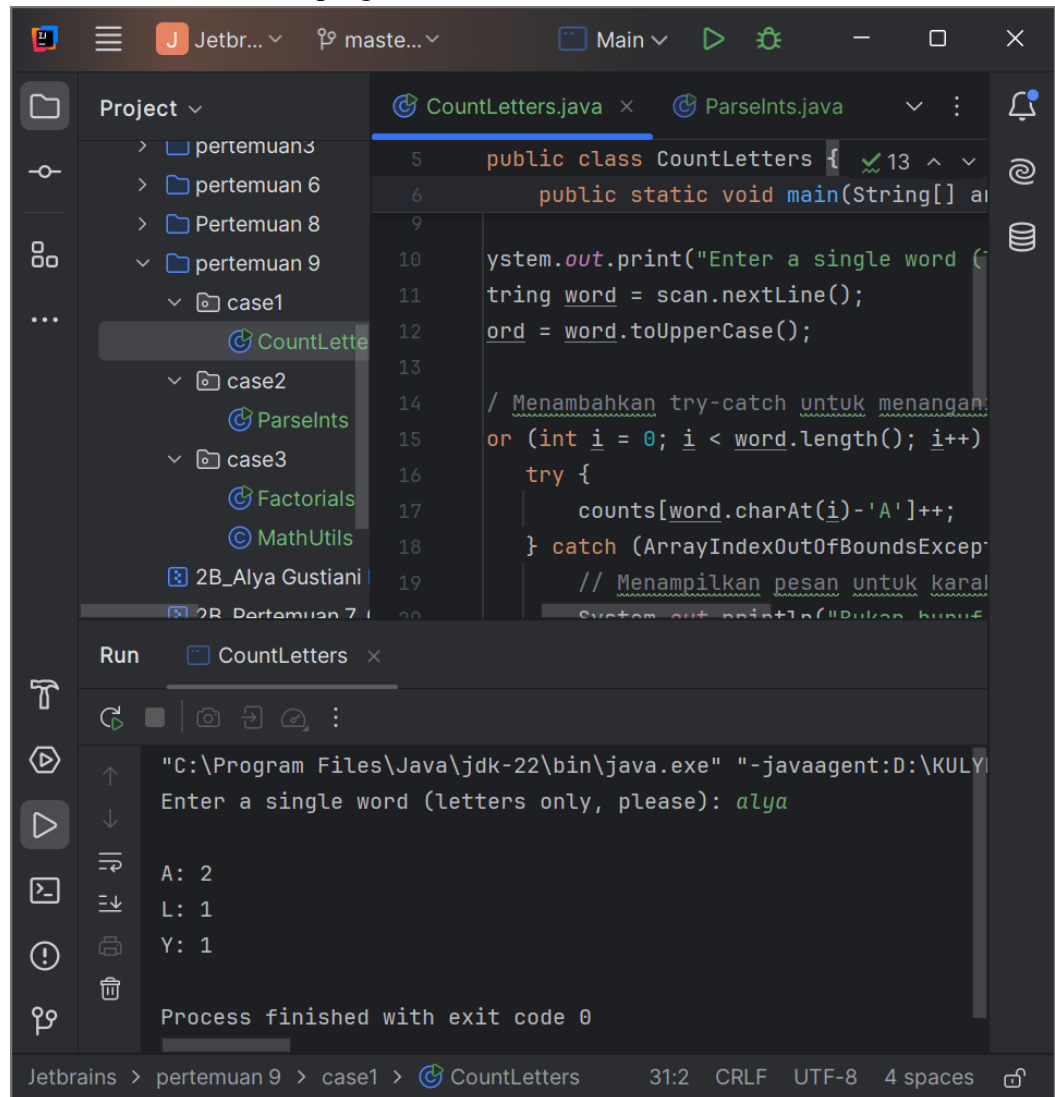


Nama : Alya Gustiani Nur 'Afifah
NIM : 231511035
Kelas/Prodi : 2B/D3 Teknik Informatika

Politeknik Negeri Bandung

2024

1. Link Repository GitHub :
<https://github.com/alyagustiani/PBOSem2/tree/master/pertemuan%209>
2. Case 1
 - a. Screenshoot hasil akhir program



The screenshot shows an IDE window with the following components:

- Project Explorer:** Displays a project structure with folders 'pertemuan 3', 'pertemuan 6', 'Pertemuan 8', 'pertemuan 9', and subfolders 'case1', 'case2', 'case3'. Under 'case1', the file 'CountLetters.java' is selected.
- Code Editor:** Shows the code for 'CountLetters.java'. The code includes a 'main' method that prompts the user to enter a word, converts it to uppercase, and counts the frequency of each letter. It uses a try-catch block to handle 'ArrayIndexOutOfBoundsException'.
- Run Console:** Shows the execution output. The prompt 'Enter a single word (letters only, please):' is followed by the input 'alya'. The output shows the letter counts: 'A: 2', 'L: 1', and 'Y: 1'. The console also indicates 'Process finished with exit code 0'.

- b. Screenshoot setiap jawaban soal yang dipertanyakan.
 - Penjelasan Masalah: Program CountLetters akan melemparkan `ArrayIndexOutOfBoundsException` jika pengguna memasukkan lebih dari satu kata atau memasukkan karakter selain huruf (seperti spasi atau tanda baca). Karena indeks yang dihasilkan tidak berada dalam rentang 0 hingga 25.
 - Tugas:
 - Modifikasi program dengan menempatkan loop pertama dalam blok try dan tambahkan blok catch untuk menangani

ArrayIndexOutOfBoundsException. Blok catch ini kosong karena karakter non-huruf akan diabaikan.

```
// Menambahkan try-catch untuk menangani karakter non-huruf
for (int i = 0; i < word.length(); i++) {
    try {
        counts[word.charAt(i) - 'A']++;
    } catch (ArrayIndexOutOfBoundsException e) {
        // Menampilkan pesan untuk karakter yang bukan huruf
        System.out.println("Bukan huruf: " + word.charAt(i));
    }
}
```

- Modifikasi blok catch untuk mencetak pesan yang informatif (misalnya, "Not a letter") diikuti dengan pengecualian. Setelah itu, ubah program agar mencetak karakter yang menyebabkan pengecualian, bukan pengecualiannya. Ini dilakukan agar pesan lebih informatif.

```
} catch (ArrayIndexOutOfBoundsException e) {
    // Menampilkan pesan untuk karakter yang bukan huruf
    System.out.println("Bukan huruf: " + word.charAt(i));
}
}
```

c. Permasalahan yang dihadapi.

- Program crash saat menerima input non-huruf
- ArrayIndexOutOfBoundsException muncul karena mencoba mengakses index array di luar range 0-25
- Program tidak bisa menangani spasi dan tanda baca
- Program berhenti total saat menemui karakter pertama yang bukan huruf

d. Solusi

- Solusi dengan empty catch block kurang informatif bagi user
- Pesan exception bisa membuat output terlihat berantakan jika ada banyak karakter non-huruf
- Pendekatan ini menggunakan exception untuk flow control, yang bisa dianggap tidak ideal dari segi performa
- Memori tambahan digunakan untuk membuat objek exception

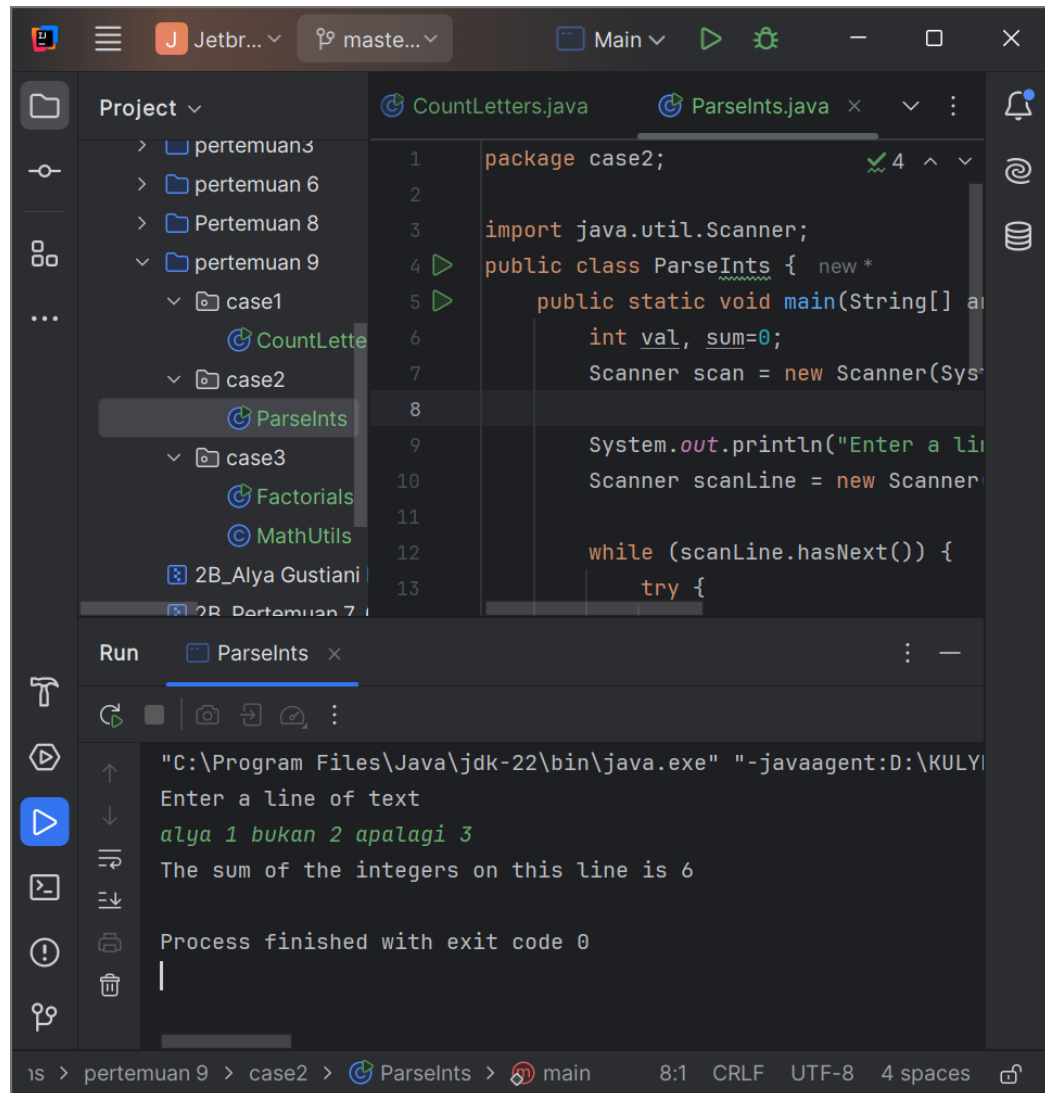
e. Lesson learnt

- Penempatan try-catch sangat penting:
 - Jika di luar loop: program berhenti total saat exception
 - Jika di dalam loop: program bisa lanjut ke token berikutnya
- Empty catch block bisa berguna untuk mengabaikan error yang expected
- Exception handling bisa digunakan sebagai cara alternatif filtering data

f. Nama teman yang membantu memecahkan permasalahan di persoalan ini:

3. Case 2

a. Screenshoot hasil akhir program



The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project structure with folders 'pertemuan 3', 'pertemuan 6', 'Pertemuan 8', and 'pertemuan 9'. Under 'pertemuan 9', there are subfolders 'case1', 'case2', and 'case3'. The 'case2' folder contains the file 'ParseInts.java', which is currently selected.
- Code Editor:** Displays the code for 'ParseInts.java'. The code is as follows:

```
1 package case2;
2
3 import java.util.Scanner;
4 public class ParseInts {
5     public static void main(String[] args) {
6         int val, sum=0;
7         Scanner scan = new Scanner(System.in);
8
9         System.out.println("Enter a line of text");
10        Scanner scanLine = new Scanner(scan.nextLine());
11
12        while (scanLine.hasNext()) {
13            try {
```

- Run Console:** Shows the execution of the program. The output is:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:D:\KULYI
Enter a line of text
alya 1 bukan 2 apalagi 3
The sum of the integers on this line is 6
Process finished with exit code 0
```
- Bottom Bar:** Shows the current file path: '1s > pertemuan 9 > case2 > ParseInts > main', along with settings like '8:1 CRLF UTF-8 4 spaces'.

b. Screenshoot setiap jawaban soal yang dipertanyakan.

- **Penjelasan Masalah:** Program ParseInts akan melemparkan `NumberFormatException` jika pengguna memasukkan baris yang berisi kombinasi angka dan kata (seperti "We have 2 dogs and 1 cat"). Kesalahan ini terjadi saat `Integer.parseInt` mencoba mengonversi token yang bukan angka.
- **Tugas:**
- **Modifikasi program** dengan menempatkan loop yang membaca dan mengonversi angka dalam blok `try` dan tambahkan blok `catch` setelah loop untuk menangani `NumberFormatException` dengan tubuh yang kosong. Ini akan menghindari error saat token bukan angka ditemukan.

```

while (scanLine.hasNext()) {
    try {
        val = Integer.parseInt(scanLine.next());
        sum += val;
    } catch (NumberFormatException e) {
        // Abaikan token yang bukan angka
    }
}

```

- Pindahkan blok try dan catch ke dalam loop, sehingga ketika pengecualian terjadi, loop akan melanjutkan iterasi berikutnya. Dengan cara ini, semua token dalam baris akan diproses, meskipun terdapat token yang bukan angka.

```

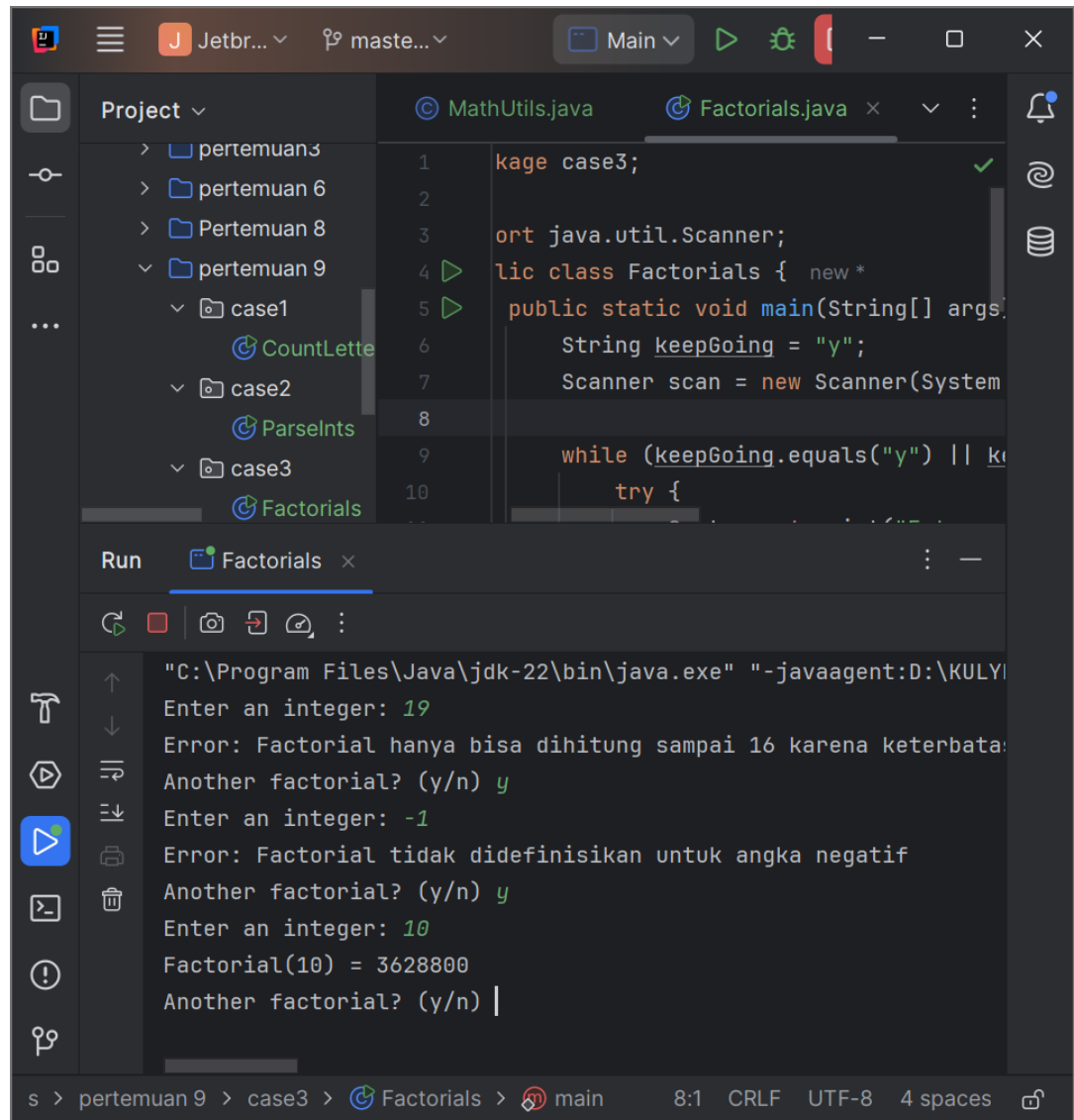
while (scanLine.hasNext()) {
    try {
        val = Integer.parseInt(scanLine.next());
        sum += val;
    } catch (NumberFormatException e) {
        // Abaikan token yang bukan angka
    }
}

```

- Permasalahan yang dihadapi.
 - Program crash ketika menemui kata-kata non-angka
 - Berhenti total saat menemui exception pertama
- Solusi dari permasalahan yang dihadapi.
 - Try-catch ditempatkan di dalam loop while
 - NumberFormatException ditangkap untuk setiap token
 - Program melanjutkan ke token berikutnya ketika menemui non-angka
 - Sum tetap dihitung untuk semua angka valid yang ditemukan
- Lesson learnt
 - Penempatan try-catch sangat penting:
 - Jika di luar loop: program berhenti total saat exception
 - Jika di dalam loop: program bisa lanjut ke token berikutnya
 - Empty catch block bisa berguna untuk mengabaikan error yang expected
 - Exception handling bisa digunakan sebagai cara alternatif filtering data
- Nama teman yang membantu memecahkan permasalahan di persoalan ini:

4. Case 3

a. Screenshoot hasil akhir program



The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project structure with folders 'pertemuan 3', 'pertemuan 6', 'Pertemuan 8', and 'pertemuan 9'. Under 'pertemuan 9', there are subfolders 'case1', 'case2', and 'case3'. The 'case3' folder contains files 'CountLetter', 'ParseInts', and 'Factorials'.
- Code Editor:** Displays the code for 'Factorials.java'. The code includes imports for 'Math' and 'java.util.Scanner', a class declaration 'class Factorials', and a 'main' method. The 'main' method uses a 'Scanner' to read input, checks for negative values, and uses a 'while' loop to calculate factorials. It also includes a check for the value 16, which triggers an error message.
- Run Console:** Shows the output of the program. It displays the command to run the program, followed by user input and the program's response. The output shows that for input 19, an error is thrown because the factorial is too large. For input -1, an error is thrown because the factorial is not defined for negative numbers. For input 10, the factorial is calculated as 3628800.

```
1 package kage case3;
2
3 import java.util.Scanner;
4 public class Factorials {
5     public static void main(String[] args) {
6         Scanner scan = new Scanner(System.in);
7         String keepGoing = "y";
8         while (keepGoing.equals("y") || keepGoing.equals("Y")) {
9             try {
10                 int n = scan.nextInt();
11                 if (n < 0) {
12                     System.out.println("Error: Factorial tidak didefinisikan untuk angka negatif");
13                 } else if (n > 16) {
14                     System.out.println("Error: Factorial hanya bisa dihitung sampai 16 karena keterbatasan tipe int");
15                 } else {
16                     System.out.println("Factorial(" + n + ") = " + factorial(n));
17                 }
18                 System.out.println("Another factorial? (y/n) ");
19                 keepGoing = scan.next();
20             } catch (Exception e) {
21                 System.out.println("Error: " + e.getMessage());
22             }
23         }
24     }
25 }
```

Run Console Output:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:D:\KULY..."
Enter an integer: 19
Error: Factorial hanya bisa dihitung sampai 16 karena keterbatasan tipe int
Another factorial? (y/n) y
Enter an integer: -1
Error: Factorial tidak didefinisikan untuk angka negatif
Another factorial? (y/n) y
Enter an integer: 10
Factorial(10) = 3628800
Another factorial? (y/n) |
```

b. Screenshoot setiap jawaban soal yang dipertanyakan.

- **Penjelasan Masalah:** Program Factorials mengembalikan nilai yang salah untuk input negatif (mengembalikan 1) dan untuk angka lebih besar dari 16 (mengembalikan angka negatif karena overflow). Secara matematis, faktorial tidak terdefinisi untuk bilangan negatif, dan overflow terjadi karena nilai faktorial terlalu besar untuk diwakili oleh tipe int.
- **Tugas:**
- **Modifikasi metode factorial di kelas MathUtils** untuk mengecek apakah argumen negatif. Jika iya, lempar `IllegalArgumentException` dengan pesan yang spesifik.

```

    if (n < 0) {
        throw new IllegalArgumentException("Factorial tidak didefinisikan untuk angka negatif");
    }
}

```

- Modifikasi metode main di Factorials untuk menangkap pengecualian yang dilempar oleh factorial dan mencetak pesan yang sesuai, namun tetap melanjutkan loop.

```

try {
    System.out.print("Enter an integer: ");
    int val = scan.nextInt();
    System.out.println("Factorial(" + val + ") = " + MathUtils.factorial(val));
} catch (IllegalArgumentException e) {
    System.out.println("Error: " + e.getMessage());
}

```

- Modifikasi metode factorial untuk juga mengecek apakah argumen lebih besar dari 16 dan lempar IllegalArgumentException dengan pesan berbeda untuk memperjelas masalah.

```

    if (n > 16) {
        throw new IllegalArgumentException("Factorial hanya bisa dihitung sampai 16 karena keterbatasan tipe data int");
    }
}

```

c. Permasalahan yang dihadapi.

- Mengembalikan 1 untuk angka negatif (seharusnya undefined)
- Mengembalikan nilai negatif untuk input > 16 (overflow)

d. Solusi dari permasalahan yang dihadapi.

- Validasi input negatif dengan throw IllegalArgumentException
- Validasi input > 16 untuk mencegah overflow
- Pesan error yang spesifik untuk setiap jenis kesalahan
- Exception handling di main method untuk mencegah program crash
- Tambahan InputMismatchException handling untuk input non-numerik

e. Lesson learnt

- Gunakan exceptions untuk menangani kasus invalid daripada mengembalikan nilai default yang menyesatkan
- IllegalArgumentException cocok untuk parameter yang tidak valid
- Manfaatkan message dalam exception untuk memberikan informasi spesifik tentang error
- Perhatikan batasan tipe data (dalam kasus ini int overflow)
- Exception handling membantu program tetap berjalan meski ada error

f. Nama teman yang membantu memecahkan permasalahan di persoalan ini: