

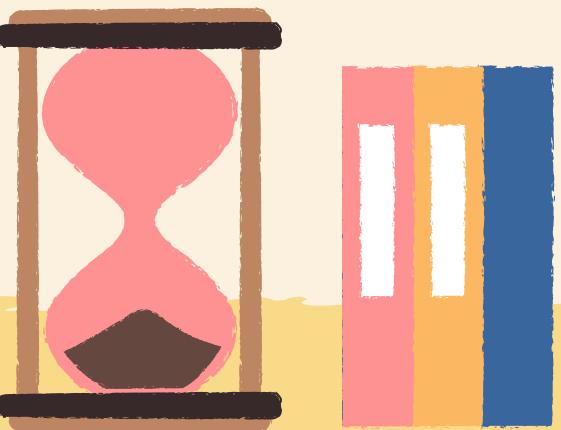
Presentasi PRAKTIKUM 7





anggota kelompok

INTAN KIRANI FEBIOLA



MUHAMMAD ARVA NUR IKHSAN SOPIAN

ANDRIANSYAH



A.TUGAS PENDAHULUAN

1. Apa yang anda ketahui tentang exception handling?
Dan apa fungsinya?

1. Exception handling adalah mekanisme untuk menangani kesalahan (errors) atau kondisi tak terduga yang terjadi selama eksekusi program. Ini memungkinkan program untuk terus berjalan tanpa crash atau berhenti tiba-tiba.

Fungsinya:

- 1) Menangani Error: Mencegah program crash saat terjadi kesalahan.
- 2) Meningkatkan Keandalan: Memastikan program tetap berfungsi dengan baik meski ada error.
- 3) Debugging Lebih Mudah: Memberikan pesan yang jelas tentang jenis kesalahan dan di mana terjadi.
- 4) Flow Control: Mengarahkan program ke jalur yang aman ketika error muncul.

Contoh di berbagai bahasa pemrograman sering menggunakan blok `try-catch` atau `try-except` untuk menangkap dan menangani exception ini.\



??

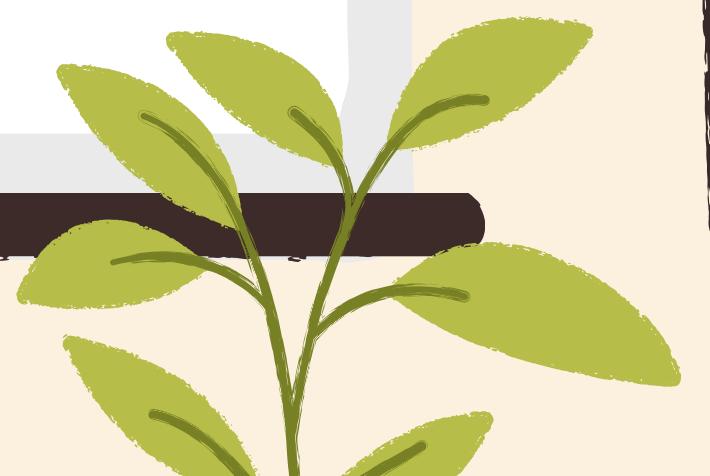
- 2. Jelaskan bentuk umum dari exception handling! Dan sebutkan syarat yang harus dipenuhi untuk membuat sebuah exception handling.



2. Bentuk umum exception handling terdiri dari blok `try`, `catch` (atau `except`), dan sering kali juga `finally`. Berikut struktur dasarnya:
syarat membuat exception handling:

- 1) Blok `try`: Harus ada bagian kode yang rawan error, dimasukkan dalam blok `try`.
- 2) Blok `catch`/`except`: Digunakan untuk menangkap dan menangani jenis exception tertentu.
- 3) Jenis Exception: Jika diperlukan, tentukan jenis exception yang ingin ditangani (misalnya, `NullPointerException`, `IOException`).
- 4) Blok `finally` (opsional): Kode ini dijalankan baik ada error maupun tidak, biasanya untuk cleanup atau penutupan resource (seperti file atau koneksi). Tanpa exception handling, program akan berhenti saat terjadi error, namun dengan mekanisme ini, error dapat ditangani dengan baik.:

**3. Tuliskan sebuah
contoh program
exception handling
sederhana! Dengan
menggunakan 3 blok
yaitu
try-catch-finally.**



code:

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license.
3  */
4
5  package com.mycompany.pendahuluan3praktikum7;
6
7  /**
8  *
9  * @author intan
10 */
11 public class Pendahuluan3praktikum7 {
12     public static void main(String[] args) {
13         try {
14             // Blok try: mencoba menjalankan kode yang mungkin menimbulkan error
15             int number = 10;
16             int result = number / 0; // Akan menghasilkan ArithmeticException (pembagian oleh nol)
17             System.out.println("Hasil: " + result);
18         } catch (ArithmaticException e) {
19             // Blok catch: menangkap dan menangani error
20             System.out.println("Terjadi kesalahan: " + e.getMessage());
21         } finally {
22             // Blok finally: selalu dijalankan, baik ada error atau tidak
23             System.out.println("Blok finally dijalankan.");
24         }
25     }
26 }
27
```

output:

```
Output - Run (pendahuluan3praktikum7) ×
Nothing to compile - all classes are up to date
--- exec:3.1.0:exec (default-cli) @ pendahuluan3praktikum7 ---
Terjadi kesalahan: / by zero
Blok finally dijalankan.

BUILD SUCCESS
Total time: 1.000 s
Finished at: 2024-09-24T16:44:40+07:00
```

Penjelasan:

Pada blok try, program mencoba membagi angka 10 dengan 0, yang menghasilkan ArithmaticException.

Blok catch menangkap exception tersebut dan menampilkan pesan error.

Blok finally akan dijalankan selalu, terlepas dari apakah ada error atau tidak.

4. Apa sebenarnya method itu? Method ada yang dinamakan dengan method rekursif, apa method rekursif itu? Jelaskan!



```
public static int faktorial(int n) {  
    if (n == 1) { // Base case  
        return 1;  
    } else {  
        return n * faktorial(n - 1); // Rekursi  
    }  
}
```

Method adalah blok kode yang memiliki nama dan dapat dipanggil untuk menjalankan tugas tertentu. Method biasanya menerima input (parameter) dan bisa mengembalikan output (return value). Method membantu mengorganisir kode agar lebih modular dan bisa digunakan ulang.

Method rekursif adalah method yang memanggil dirinya sendiri secara langsung atau tidak langsung untuk menyelesaikan suatu tugas. Setiap pemanggilan rekursif mendekatkan problem ke kondisi dasar (base case), yaitu kondisi di mana rekursi berhenti. Contoh method rekursif sederhana adalah menghitung factorial:

DEF
KLM
RST
XYZ

5. Ada berapa macam tipe method di java? Sebutkan dan berikan contoh! Maksimal 5 baris kode untuk contoh setiap tipe method! Serta panggil method tersebut dalam main programnya!

1) Method tanpa Return dan Parameter

Method ini tidak mengembalikan nilai dan tidak menerima parameter.

Berikut code dan outputnya:



```
* @author intan
*/
public class Method1java {
    // Method tanpa return dan tanpa parameter
    public static void greet() {
        System.out.println("Hello, Selamat Datang!");
    }

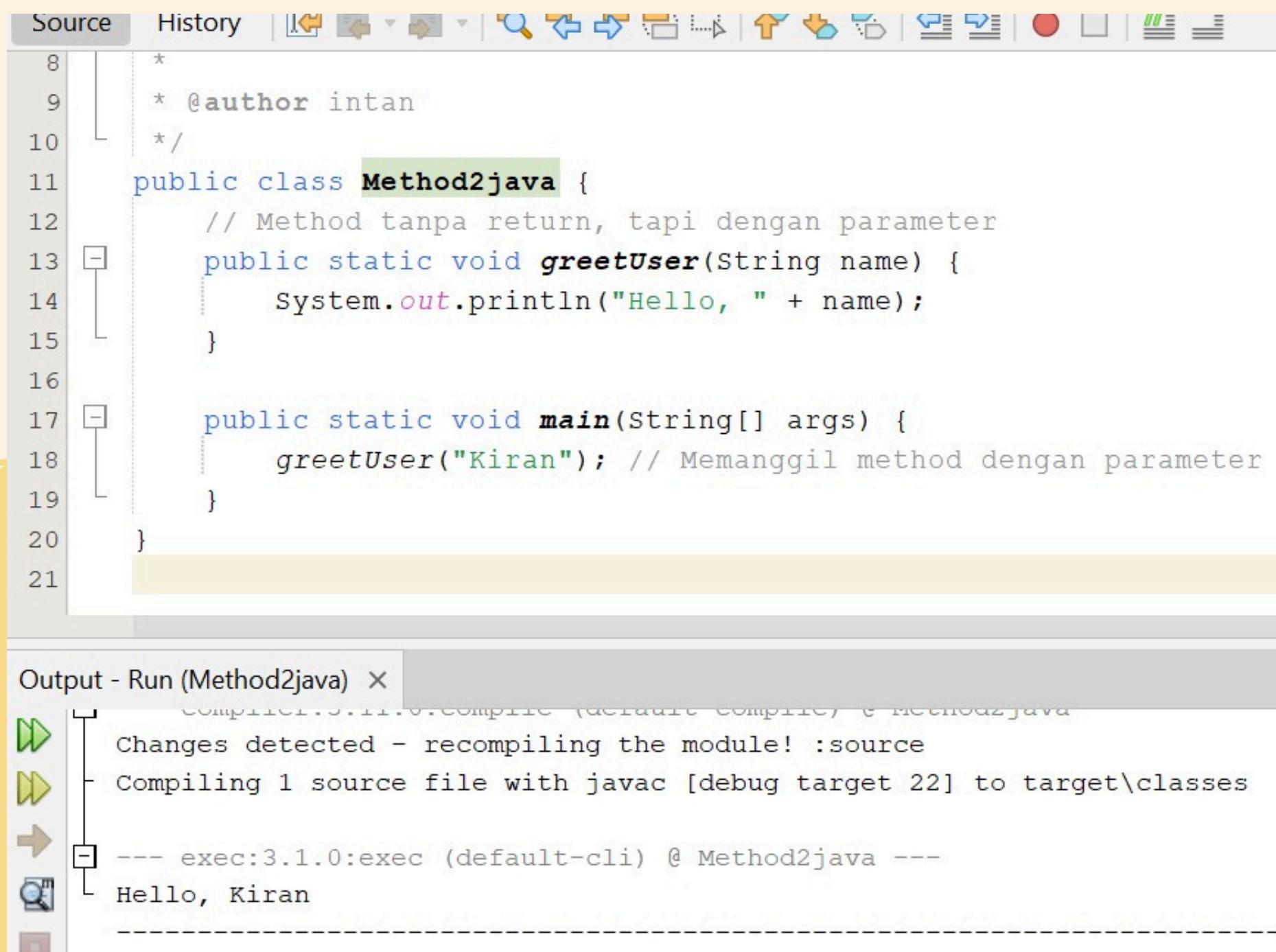
    public static void main(String[] args) {
        greet(); // Memanggil method tanpa return dan parameter
    }
}
```

```
out - Run (method1java) ×
  Compiling 1 source file with javac [debug target 22] to target\classes
Changes detected - recompiling the module! :source
Compiling 1 source file with javac [debug target 22] to target\classes
--- exec:3.1.0:exec (default-cli) @ method1java ---
Hello, Selamat Datang!
```

2) Method tanpa return, tapi dengan parameter

Method ini tidak mengembalikan nilai, tetapi menerima parameter.

Berikut code dan outputnya:



The screenshot shows an IDE interface with a toolbar at the top and two code editors below. The top editor is titled 'Source' and contains the following Java code:

```
8  *
9  * @author intan
10 */
11 public class Method2java {
12     // Method tanpa return, tapi dengan parameter
13     public static void greetUser(String name) {
14         System.out.println("Hello, " + name);
15     }
16
17     public static void main(String[] args) {
18         greetUser("Kiran"); // Memanggil method dengan parameter
19     }
20 }
21
```

The bottom editor is titled 'Output - Run (Method2java)' and shows the terminal output of the program execution:

```
Changes detected - recompiling the module! :source
Compiling 1 source file with javac [debug target 22] to target\classes
--- exec:3.1.0:exec (default-cli) @ Method2java ---
Hello, Kiran
```

3) Method dengan return, tapi tanpa parameter.

Method ini mengembalikan nilai,
tetapi tidak menerima parameter.

Berikut Code dan Output:

Output - Run (Method3.java)

```
Changes detected - recompiling the module! :source
Compiling 1 source file with javac [debug target 22] to target\classes
--- exec:3.1.0:exec (default-cli) @ Method3java ---
Number: 12
-----
BUILD SUCCESS
```

4) Method dengan return dan parameter

Method ini mengembalikan nilai dan menerima parameter.

The screenshot shows an IDE interface with several tabs at the top: "...va", "Method1java.java", "Method2java.java", "Method3java.java", and "Method4java.java". The "Method4java.java" tab is active. Below the tabs is a toolbar with various icons. The main area displays the Java code:

```
9  * @author intan
10 */
11 public class Method4java {
12     // Method dengan return dan parameter
13     public static int sum(int a, int b) {
14         return a + b;
15     }
16
17     public static void main(String[] args) {
18         int result = sum(5, 6); // Menyimpan hasil penjumlahan
19         System.out.println("Sum: " + result);
20     }
21 }
```

The code defines a class named `Method4java` with a `sum` method that takes two integers as parameters and returns their sum. It also contains a `main` method that calls `sum` with arguments 5 and 6, and prints the result. The output window at the bottom shows the compilation process and the execution output:

```
Output - Run (Method4java) ×
Changes detected - recompiling the module! :source
Compiling 1 source file with javac [debug target 22] to target\classes
--- exec:3.1.0:exec (default-cli) @ Method4java ---
Sum: 11
-----
BUILD SUCCESS
```

Penjelasan:

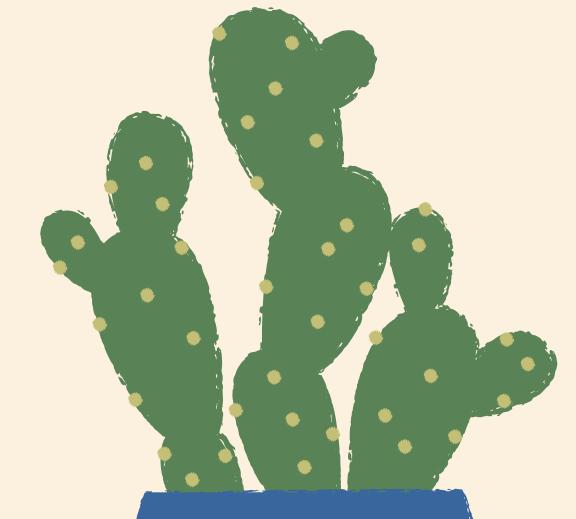
Method 1: Tidak menerima input dan tidak mengembalikan hasil.

Method 2: Menerima input (parameter), tapi tidak mengembalikan hasil.

Method 3: Tidak menerima input, tapi mengembalikan hasil.

- **Method 4: Menerima input dan mengembalikan hasil.**

Setiap method dipanggil dalam method main untuk mengeksekusi fungsionalitasnya



6. dibawah ini adalah contoh method rekursif (Code dan Output)

The screenshot shows an IDE interface with several tabs at the top: "...va", "Method1java.java", "Method2java.java", "Method3java.java", "Method4java.java", and "Rekursifpraktikum7.java". The "Source" tab is selected. Below it is the code for "Rekursifpraktikum7.java".

```
11 public class Rekursifpraktikum7 {
12
13     // Method rekursif untuk menghitung faktorial
14     public static int faktorial(int n) {
15         if (n == 1) { // Base case: jika n sama dengan 1
16             return 1;
17         } else {
18             return n * faktorial(n - 1); // Memanggil method rekursif dengan n-1
19         }
20     }
21
22     public static void main(String[] args) {
23         int number = 5;
24         int result = faktorial(number); // Memanggil method rekursif
25         System.out.println("Faktorial dari " + number + " adalah: " + result);
26     }
27 }
```

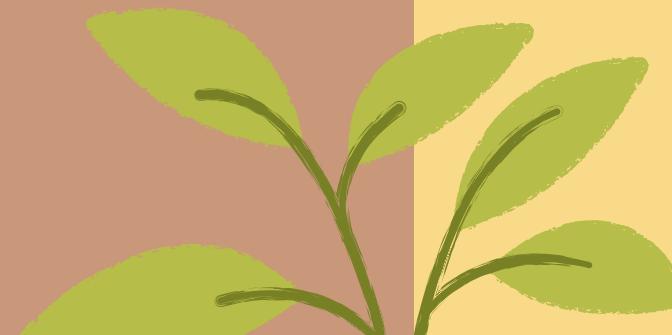
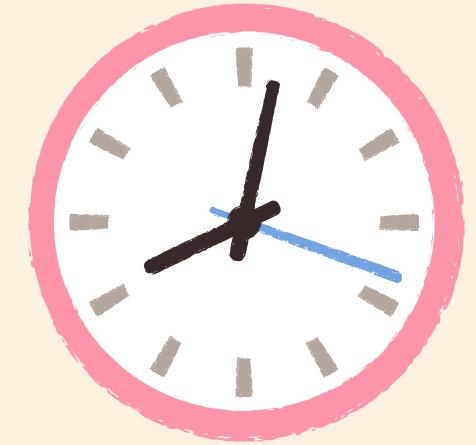
Below the code editor is the "Output - Run (Rekursifpraktikum7)" window. It displays the following log:

```
Changes detected - recompiling the module! :source
Compiling 1 source file with javac [debug target 22] to target\classes
--- exec:3.1.0:exec (default-cli) @ Rekursifpraktikum7 ---
Faktorial dari 5 adalah: 120
-----
BUILD SUCCESS
-----
Total time: 1.442 s
```



7. Akses Modifier di Java adalah kata kunci yang menentukan tingkat aksesibilitas suatu variabel, method, atau class. Dengan akses modifier, kita bisa mengontrol siapa saja yang bisa mengakses atau memodifikasi data dan method dalam suatu class. Ada empat jenis akses modifier di Java, masing-masing memiliki tingkat akses yang berbeda.

Macam-Macam Akses Modifier:



MACAM-MACAM AKSES MODIFIER:

1.)Public

Elemen dengan akses modifier public dapat diakses dari mana saja, baik dari dalam maupun luar class, bahkan dari package yang berbeda.

```
public class PublicModifier {  
    public int data = 10; // Variabel public  
  
    public void display() { // Method public  
        System.out.println("Ini method public");  
    }  
}
```

2.)Private

Elemen dengan akses modifier private hanya bisa diakses dari dalam class itu sendiri. Elemen private tidak bisa diakses dari luar class, bahkan dari subclass.

```
/*
public class Privatemodifier {

    private int data = 20; // Variabel private

    private void display() { // Method private
        System.out.println("Ini method private");
    }

    public void showData() {
        System.out.println("Data: " + data); // Bisa diakses dalam class
    }
}
```

3.)Protected

Elemen dengan akses modifier protected bisa diakses dari dalam class, subclass (baik dalam package yang sama atau berbeda), dan class lain dalam package yang sama.

```
public class Protectedmodifier {  
  
    protected int data = 30; // Variabel protected  
  
    protected void display() { // Method protected  
        System.out.println("Ini method protected");  
    }  
}
```

4.)Default (Package-Private)

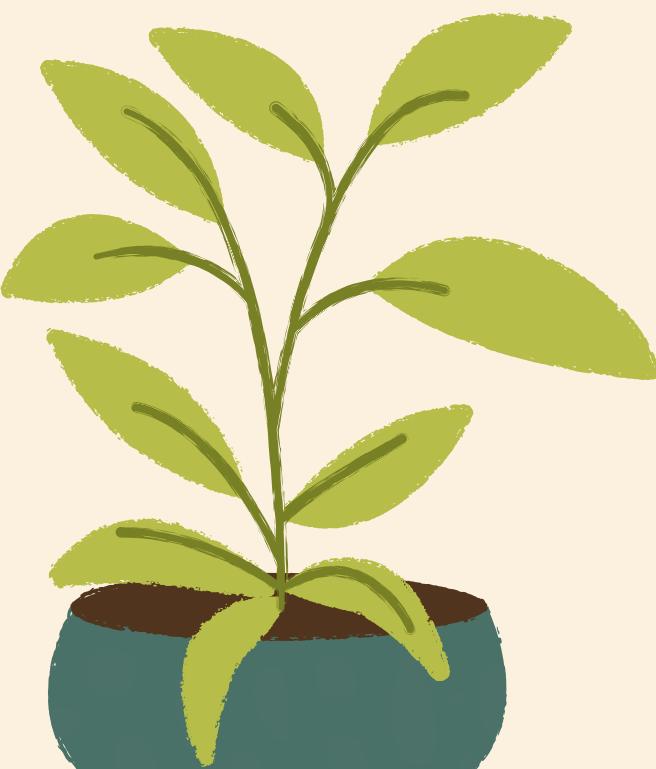
Jika tidak ada akses modifier yang ditentukan, maka elemen tersebut memiliki akses default.

Akses ini memungkinkan elemen diakses hanya dari class dalam package yang sama.

```
class DefaultClass { // Class default (tanpa modifier)

    int data = 40; // Variabel default

    void display() { // Method default
        System.out.println("Ini method default");
    }
}
```



B.TUGAS PRAKTIKUM

1. Ketik kembali kode sederhana berikut ini :

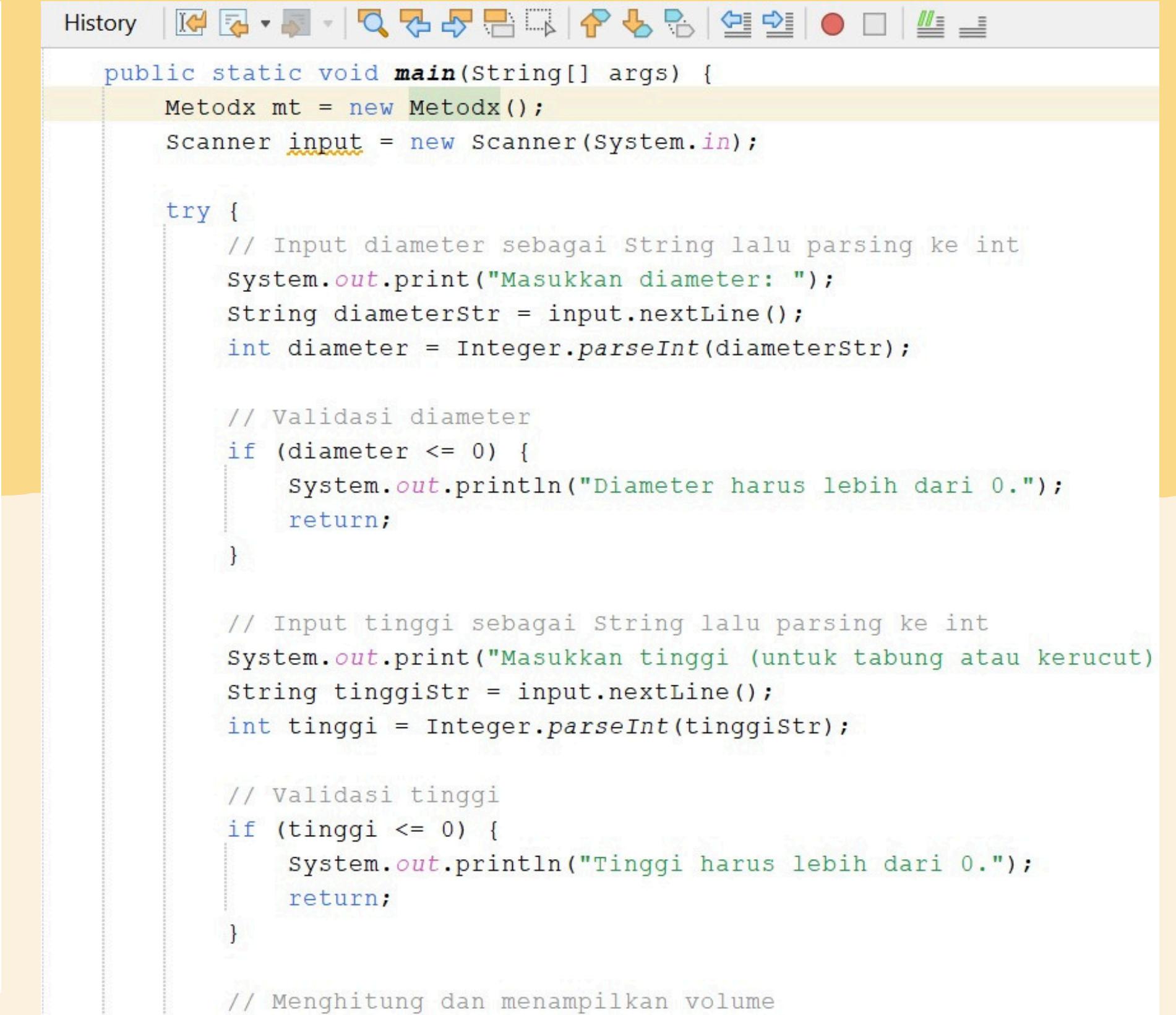
```
public class metodx {  
    public double luas_lingkaran(int diameter) {  
        int jari2=diameter/2;  
        double luas=Math.PI * Math.pow(jari2,2);  
        return luas;  
    }  
    public static void main(String[]args) {  
        metodx mt=new metodx();  
        System.out.print(mt.luas_lingkaran(20));  
        System.exit(0);  
    }  
}
```

Instruksi yang harus dijalankan :

- Ketik kembali kode sederhana diatas, Compile dan jalankan programnya!
- Buat method baru yang menangani proses menghitung volume bangun yang mempunyai lingkaran misal tabung, kerucut, bola. Nilai diameter dimasukkan lewat console!

CODE:

```
6
7 import java.util.Scanner;
8
9 public class Metodx {
10
11     // Method untuk menghitung luas lingkaran
12     public double luas_lingkaran(int diameter) {
13         int jari2 = diameter / 2;
14         double luas = Math.PI * Math.pow(jari2, 2);
15         return luas;
16     }
17
18     // Method untuk menghitung volume tabung
19     public double volume_tabung(int diameter, int tinggi) {
20         double luasAlas = luas_lingkaran(diameter);
21         return luasAlas * tinggi;
22     }
23
24     // Method untuk menghitung volume kerucut
25     public double volume_kerucut(int diameter, int tinggi) {
26         double luasAlas = luas_lingkaran(diameter);
27         return (1.0 / 3.0) * luasAlas * tinggi;
28     }
29
30     // Method untuk menghitung volume bola
31     public double volume_bola(int diameter) {
32         int jari2 = diameter / 2;
33         return (4.0 / 3.0) * Math.PI * Math.pow(jari2, 3);
34     }
```



The screenshot shows a Java code editor window with the following code:

```
History | Back | Forward | Stop | Refresh | Home | Search | Find | Replace | Open | Save | Print | Copy | Paste | Cut | Select All | Undo | Redo | Minimize | Maximize | Close | Minimize | Maximize | Close
```

```
public static void main(String[] args) {
    Metodx mt = new Metodx();
    Scanner input = new Scanner(System.in);

    try {
        // Input diameter sebagai String lalu parsing ke int
        System.out.print("Masukkan diameter: ");
        String diameterStr = input.nextLine();
        int diameter = Integer.parseInt(diameterStr);

        // Validasi diameter
        if (diameter <= 0) {
            System.out.println("Diameter harus lebih dari 0.");
            return;
        }

        // Input tinggi sebagai String lalu parsing ke int
        System.out.print("Masukkan tinggi (untuk tabung atau kerucut)");
        String tinggiStr = input.nextLine();
        int tinggi = Integer.parseInt(tinggiStr);

        // Validasi tinggi
        if (tinggi <= 0) {
            System.out.println("Tinggi harus lebih dari 0.");
            return;
        }

        // Menghitung dan menampilkan volume
```

```
63 // Menghitung dan menampilkan volume
64 System.out.println("Volume Tabung: " + mt.volume_tabung(diameter, tinggi));
65 System.out.println("Volume Kerucut: " + mt.volume_kerucut(diameter, tinggi));
66 System.out.println("Volume Bola: " + mt.volume_bola(diameter));
67
68 } catch (NumberFormatException e) {
69     System.out.println("Input tidak valid. Harap masukkan angka.");
70 } finally {
71     input.close(); // Menutup scanner
72 }
73
74     System.exit(0);
75 }
76 }
77 }
```

output:

Output

Run (metodx) X Run (metodx) X

- resources:3.3.1:resources (default-resources) @ metodx ---
- skip non existing resourceDirectory C:\Users\intan\OneDrive\Documents\NetBea
- compiler:3.11.0:compile (default-compile) @ metodx ---
- Nothing to compile - all classes are up to date
- exec:3.1.0:exec (default-cli) @ metodx ---
- Masukkan diameter: 20
- Masukkan tinggi (untuk tabung atau kerucut): 30
- Volume Tabung: 9424.77796076938
- Volume Kerucut: 3141.592653589793
- Volume Bola: 4188.790204786391

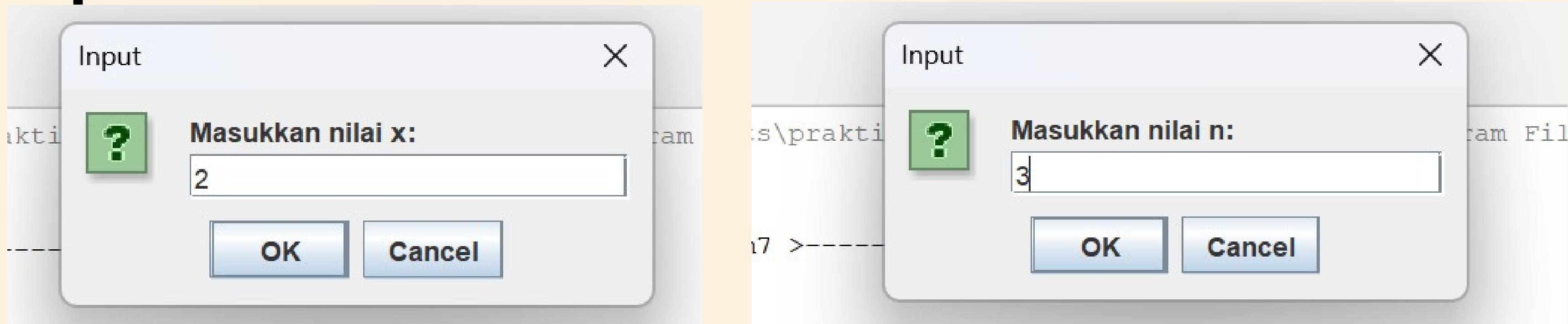
BUILD SUCCESS

**2. Buat sebuah fungsi rekursif yang bisa menghitung nilai :
Dimana nilai parameter n dan x dimasukkan melalui
JOptionPane!**

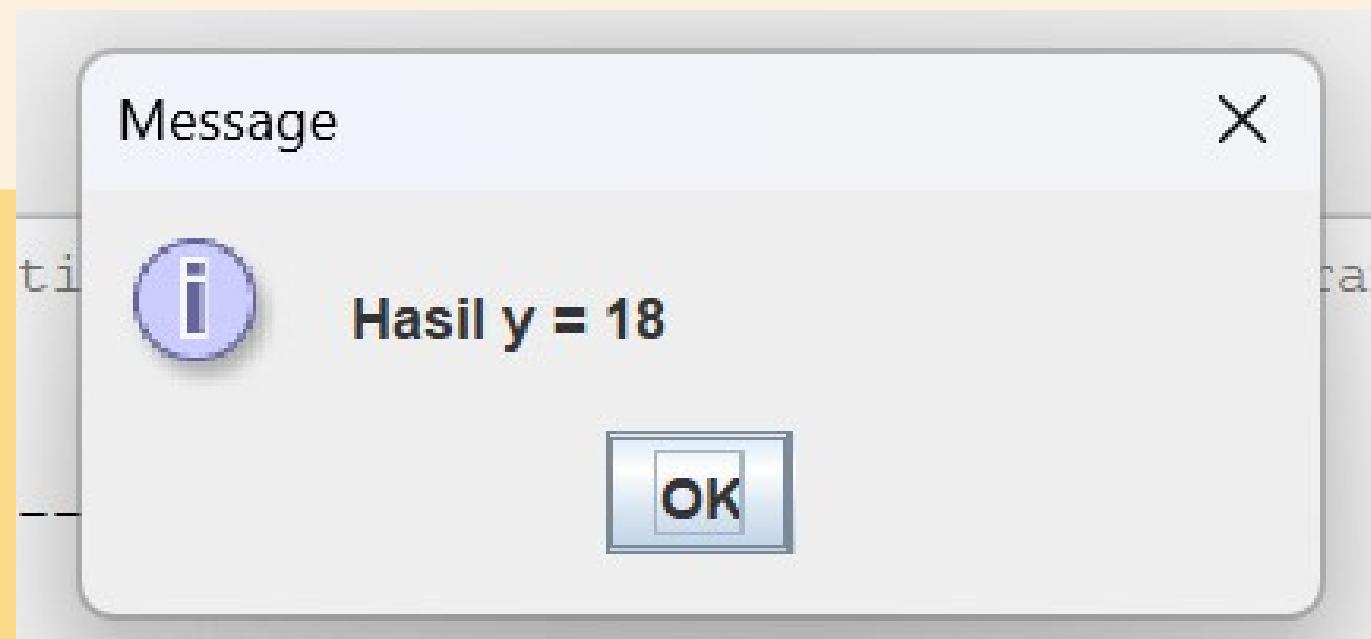
CODE:

```
public class Praktikum2praktikum7 {  
  
    // Fungsi rekursif untuk menghitung y  
    public static int hitungY(int x, int n) {  
        // Base case  
        if (n == 1) {  
            return x + 2 * 1;  
        } else {  
            // Rekursi  
            return (x + 2 * n) + hitungY(x, n - 1);  
        }  
    }  
  
    public static void main(String[] args) {  
        // Mengambil input dari JOptionPane  
        String inputX = JOptionPane.showInputDialog("Masukkan nilai x:");  
        String inputN = JOptionPane.showInputDialog("Masukkan nilai n:");  
  
        // Konversi input menjadi integer  
        int x = Integer.parseInt(inputX);  
        int n = Integer.parseInt(inputN);  
  
        // Panggil fungsi rekursif untuk menghitung hasil  
        int hasil = hitungY(x, n);  
  
        // Tampilkan hasilnya  
        JOptionPane.showMessageDialog(null, "Hasil y = " + hasil);  
    }  
}
```

input:



hasil:



Contoh Eksekusi:

Jika pengguna memasukkan $x = 2$ dan $n = 3$, maka perhitungannya akan seperti ini:

$$y = (2+2\times 1) + (2+2\times 2) + (2+2\times 3)$$
$$y = (2 + 2 \times 1) + (2 + 2 \times 2) + (2 + 2 \times 3)$$
$$y = 4 + 6 + 8 = 18$$
$$y = 4 + 6 + 8 = 18$$

Jadi, program akan menampilkan hasil $y = 18$.

3. Ketik kembali kode berikut ini :

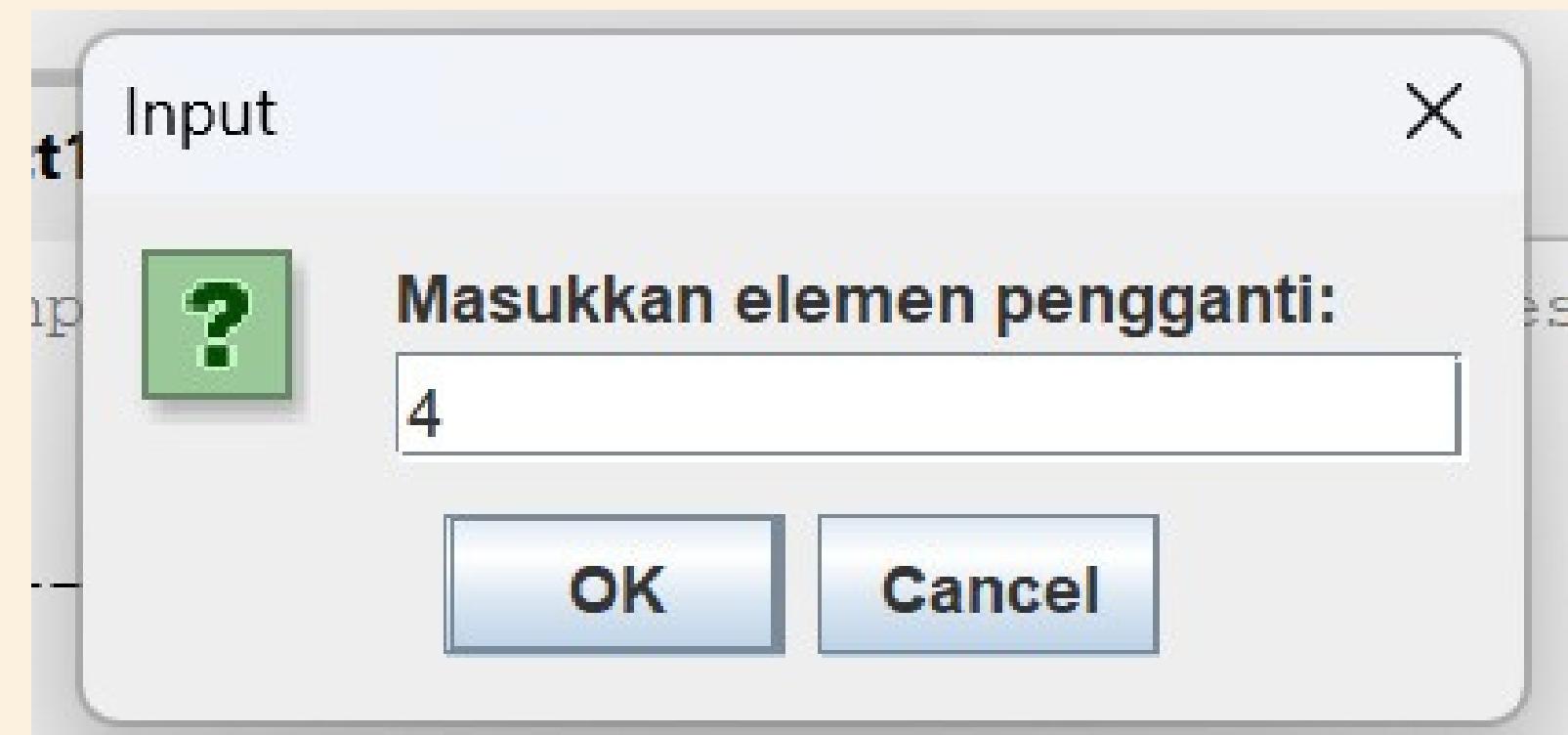
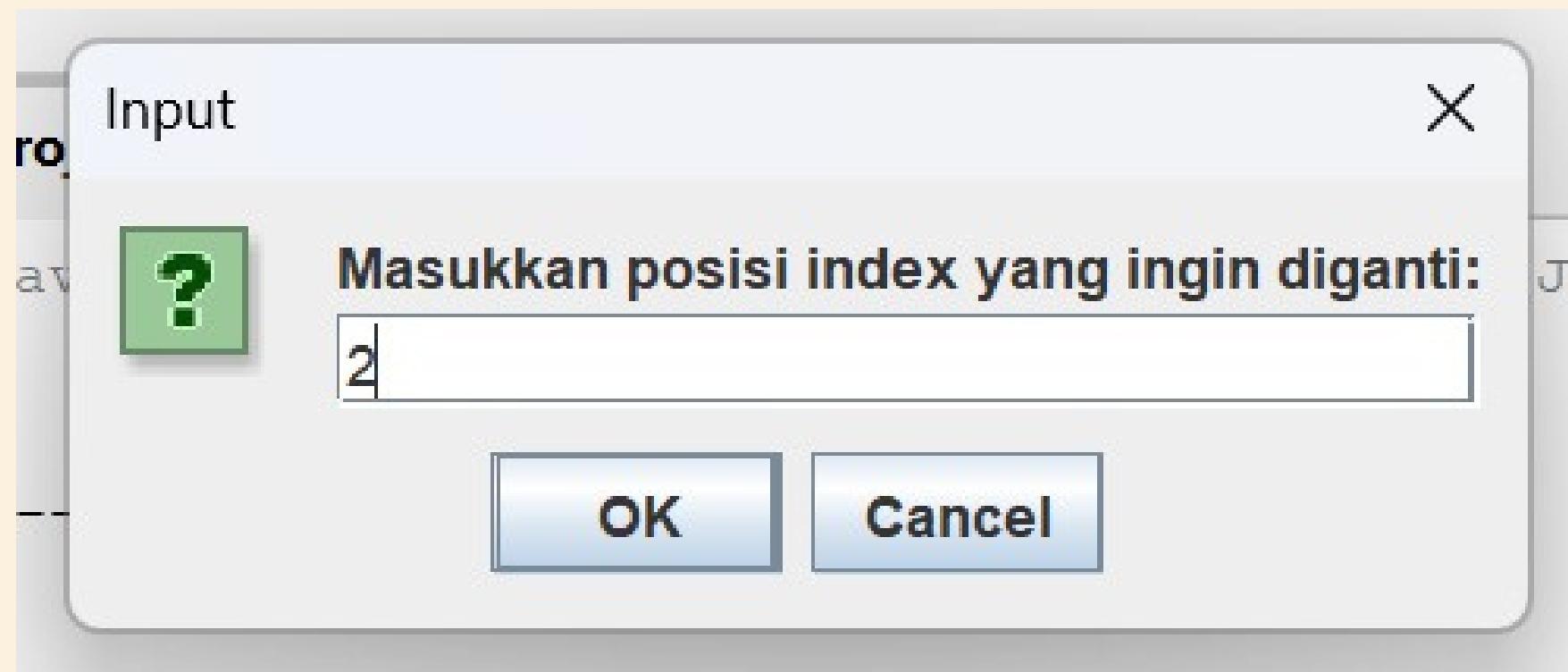
```
public class cobaException3 {  
    public static void main(String args[]) {  
        int bil = 10;  
        String b[]={ "a", "b", "c" };  
        try {  
            System.out.println(bil/0);  
            System.out.println(b[3]);  
        }  
        catch(ArithmeticException ai) {  
            System.out.println("Error Aritmetik"); System.out.println(ai.getMessage());  
        }  
        catch(ArrayIndexOutOfBoundsException n){  
            System.out.println("Error karena melebihi kapasitas array");  
            System.out.println(n.getMessage());  
        }  
        catch(Exception e){  
            System.out.println("Ada error"); System.out.println(e.getMessage());  
        }  
    }  
}
```

Instruksi yang harus dijalankan :

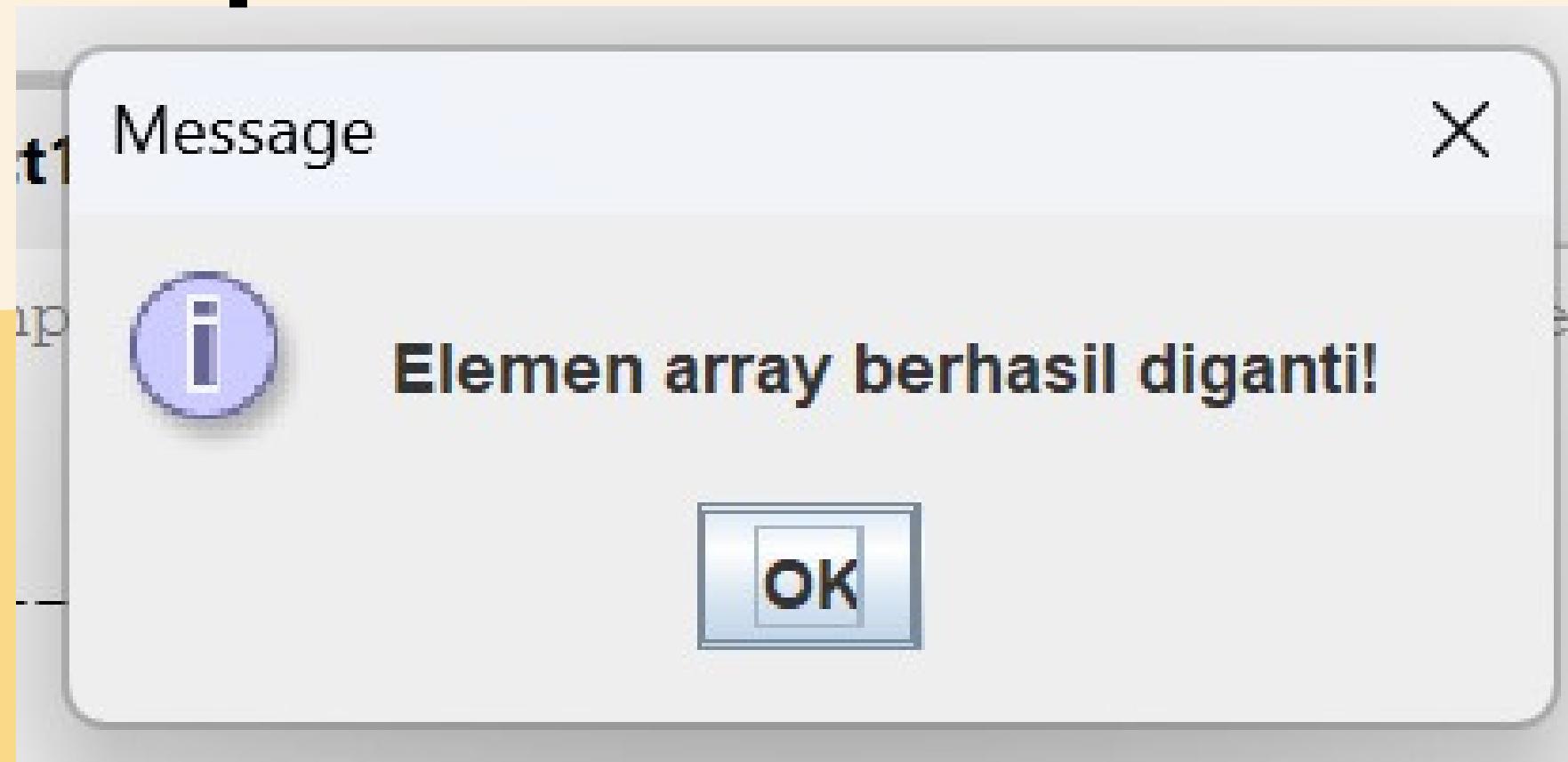
- Ketik kembali kode sederhana diatas, Compile dan jalankan programnya!
- Apa yang dihasilkan dari program tersebut?
- Lanjutkan dengan membuat sebuah method yang berfungsi untuk mengubah nilai element array yang diinginkan berdasarkan posisi indexnya! posisi index yg ingin diganti dan element pengganti dimasukkan lewat JOptionPane.

CODE:

input:



output:



Hasil Program:

Saat Anda menjalankan kode di atas, dua kesalahan akan muncul:

ArithmetricException: Terjadi karena mencoba membagi angka dengan nol pada baris `System.out.println(bil / 0);`.

ArrayIndexOutOfBoundsException: Terjadi ketika mencoba mengakses elemen array di luar batasnya (yaitu `b[3]`, sementara array `b` hanya memiliki indeks 0, 1, dan 2).

Proses dihentikan setelah **ArithmetricException** karena exception ini sudah tertangkap di blok `catch`. Blok berikutnya yang menangani **ArrayIndexOutOfBoundsException** tidak dijalankan.

4. Masih ingat dengan PR Budi ini ?

Bangun Rumus

Panjang * Lebar

Panjang :20 , Lebar :30.

$1/2 * \text{Alas} * \text{Tinggi}$

Alas : 20 , Tinggi : 10

$\text{Phi} * r * r$

Jari-jari : 10

Pertanyaannya :

- Kali ini bantu Budi menyelesaikan permasalahannya dengan membuatkan sebuah program

sederhana untuk menghitung luas bangun tersebut jika nilai dari variable yang digunakan

diinputkan menggunakan JOptionPane dan perhitungan masing masing bangun ditangani oleh sebuah method. Gunakan method dengan parameter !

code:

The screenshot shows a Java code editor with the following code:

```
36     double luasPersegi = luasPersegiPanjang(panjang, lebar);
37
38     // Menampilkan hasil luas persegi panjang
39     JOptionPane.showMessageDialog(null, "Luas Persegi Panjang: " + luasPersegi);
40
41     // Input untuk segitiga
42     String inputAlas = JOptionPane.showInputDialog("Masukkan alas segitiga:");
43     String inputTinggi = JOptionPane.showInputDialog("Masukkan tinggi segitiga:");
44     double alas = Double.parseDouble(inputAlas);
45     double tinggi = Double.parseDouble(inputTinggi);
46     double luasSegitiga = luasSegitiga(alas, tinggi);
47
48     // Menampilkan hasil luas segitiga
49     JOptionPane.showMessageDialog(null, "Luas Segitiga: " + luasSegitiga);
50
51     // Input untuk lingkaran
52     String inputJariJari = JOptionPane.showInputDialog("Masukkan jari-jari lingkaran:");
53     double jariJari = Double.parseDouble(inputJariJari);
54     double luasLingkaran = luasLingkaran(jariJari);
55
56     // Menampilkan hasil luas lingkaran
57     JOptionPane.showMessageDialog(null, "Luas Lingkaran: " + luasLingkaran);
58 }
59 }
```

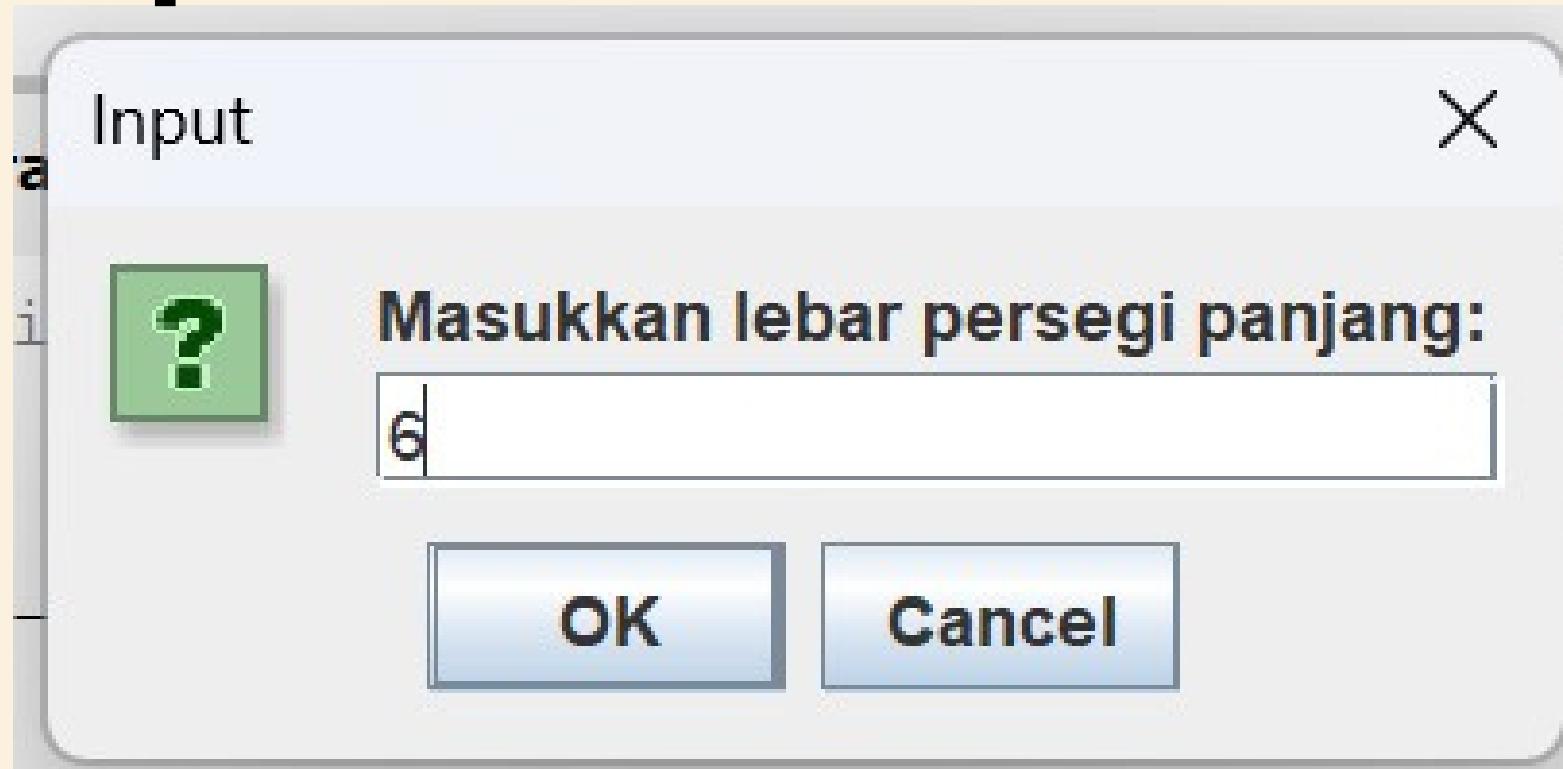
input:

Input

Masukkan lebar persegi panjang:

6

OK Cancel

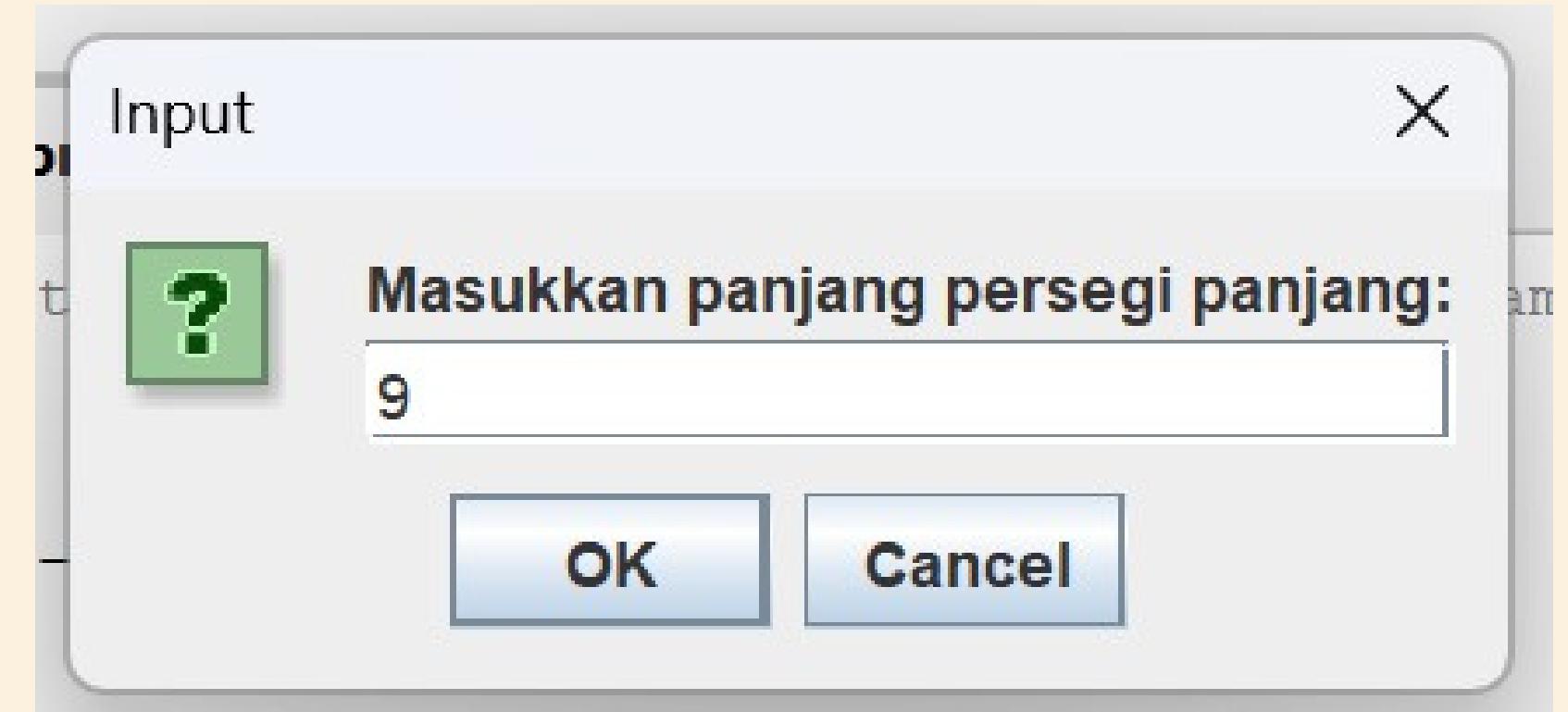


Input

Masukkan panjang persegi panjang:

9

OK Cancel

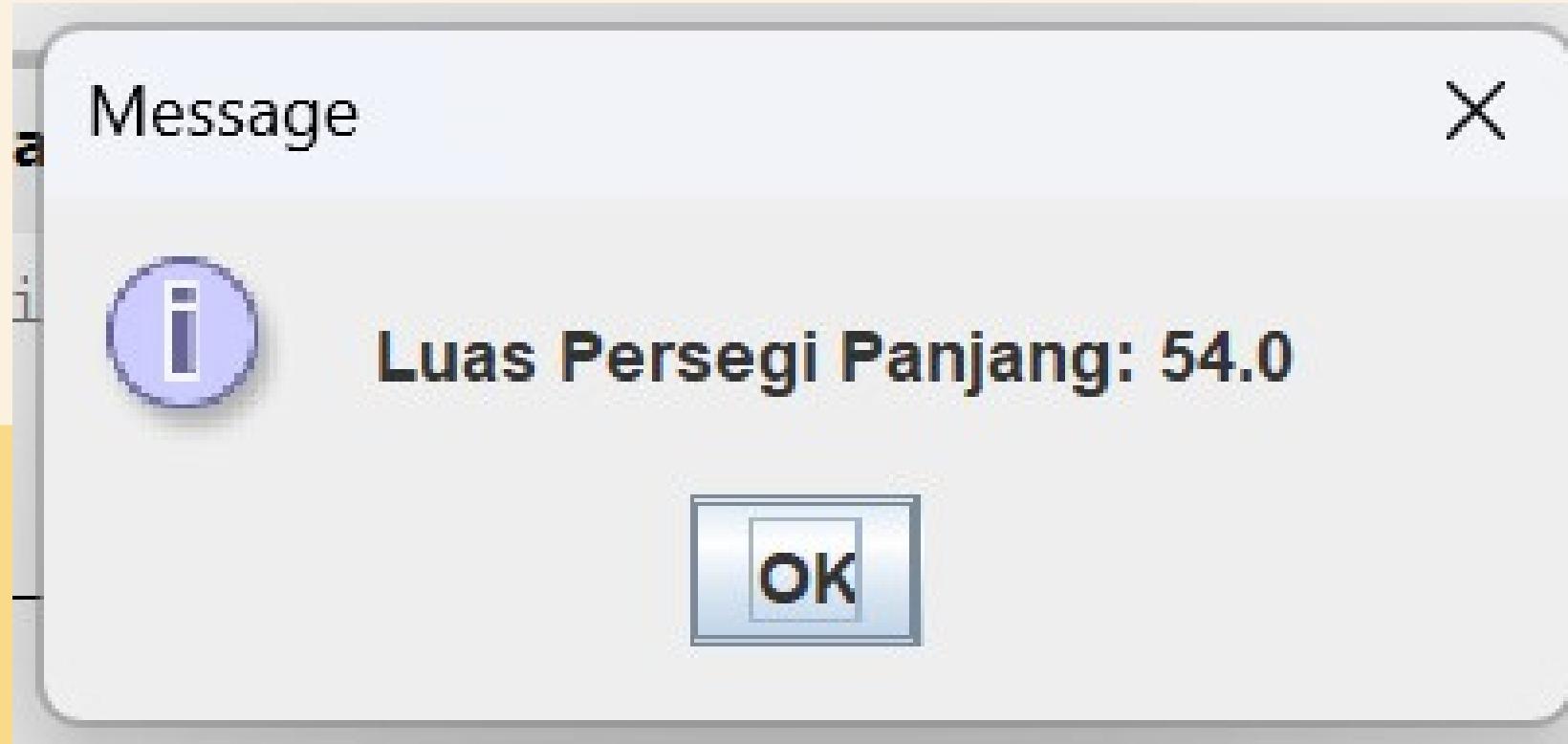


hasil:

Message

Luas Persegi Panjang: 54.0

OK



KESIMPULAN

Kesimpulannya, kita telah membahas beberapa konsep penting dalam pemrograman Java, seperti exception handling menggunakan blok `try-catch-finally`, method termasuk method rekursif, dan penggunaan akses modifier. Selain itu, kita telah membuat program sederhana untuk menghitung luas bangun datar (persegi panjang, segitiga, lingkaran) dengan input dari pengguna melalui JOptionPane, yang juga menggunakan method untuk menangani logika perhitungan. Program ini memperlihatkan cara kerja dasar method dan bagaimana memanfaatkan input/output GUI di Java.

Kesimpulan dari diskusi kita mencakup beberapa konsep dan aplikasi pemrograman dalam Java:

Penjelasan Kode:

Method luasPersegiPanjang: Menghitung luas persegi panjang dengan rumus panjang * lebar.

Method luasSegitiga: Menghitung luas segitiga dengan rumus $\frac{1}{2} * \text{alas} * \text{tinggi}$.

Method luasLingkaran: Menghitung luas lingkaran dengan rumus $\pi * r^2$ (di mana r adalah jari-jari).

Input menggunakan JOptionPane: Untuk setiap bangun, kita meminta pengguna memasukkan nilai panjang, lebar, alas, tinggi, dan jari-jari melalui `JOptionPane.showInputDialog`. Input ini kemudian dikonversi menjadi double menggunakan `Double.parseDouble()`.

Output menggunakan JOptionPane.showMessageDialog: Hasil perhitungan luas masing-masing bangun ditampilkan menggunakan `JOptionPane.showMessageDialog`.



Terima Kasih

