Bab 9

Pengenalan Exception pada OOP.

Definisi Exception

Exception merupakan kondisi yang tidak diinginkan yang dapat terjadi selama eksekusi program. Exception muncul ketika program menghadapi situasi yang **tidak terduga**, seperti kesalahan pembagian nol, maupun akses data yang tidak valid. Exception dapat menyebabkan program keluar dari jalur normalnya, menghentikan proses eksekusi, dan memerlukan penanganan khusus agar program dapat berjalan dengan baik. Oleh karenanya diperlukan mekanisme yang membantu menangani error tersebut atau kesalahan yang terjadi, mekanisme ini dikenal dengan **Exception Handling**.

Definisi Exception Handling

Exception Handling merupakan mekanisme yang diperlukan dalam menangani error yang terjadi pada saat **run-time** (program berjalan) atau yang lebih dikenal dengan sebutan runtime error. Secara umum, adanya kesalahan/ error yang terjadi pada program pada saat runtime dapat menyebabkan program berhenti atau hang. Untuk itulah diperlukan mekanisme untuk memastikan bahwa program tetap dapat berjalan meskipun terdapat kesalahan yang terjadi.

Keuntungan penggunaan Exception Handling :

1. Exception bisa membawa sangat banyak informasi tentang kesalahan yang terjadi (deskripsi, stack trace(jejak error), lokasi baris kode dll).
2. Exception sudah didukung oleh banyak IDE modern yang ada di pasar.
3. Mekanisme Exception terintegrasi secara baik dengan OOP.

Jenis-jenis Exception

Terdapat 3 jenis Exception :

1. ***Checked Exception***, adalah exception yang terjadi saat **compile time**. jika suatu metode dapat menimbulkan checked exception, maka kode tersebut harus menyertakan blok try-catch untuk menangani exception tersebut atau mendeklarasikan penggunaan throws pada metodenya.

No	*Exception*	Deskripsi
1	*classNotFoundException*	Kelas tidak ditemukan
2	*CloneNotSupportedException*	Melakukan clone objek yang tidak mengimplementasikan interface cloneable
3	*IllegalAccessException*	Pengaksesan ke kelas ditolak
4	*InstantiationException*	Menciptakan oobbjek darikelas abstract ataupun interface

Jenis-jenis Exception

2. Uncecked Exception,

   Uncecked Exception, adalah exception yang terjadi saat **run-time** (execution time). Error ini terjadi dalam lingkup internal dari aplikasi, biasanya terjadi karena salah penulisan kode. Contoh: NullPointerException.

3. Error,

   Error, adalah exception yang diluar **kendali user** atau **programmer**. Eror ini terjadi di lingkup eksternal dari aplikasi. Ketika exception ini terjadi, maka tidak ada yang bisa dilakukan untuk mengatasinya. Contoh: ketika perangkat kerasnya rusak saat ingin membaca data.

Penanganan Exception

1. Try-Catch

try-catch adalah mekanisme yang digunakan dalam pemrograman untuk menangani *exception* (kesalahan) yang mungkin terjadi selama eksekusi kode. Pernyataan *try* digunakan untuk mengelompokkan kode yang dapat menyebabkan *exception*, sementara pernyataan *catch* digunakan untuk menangkap dan menangani exception tersebut. Kode yang dimasukkan dalam block try-catch biasa disebut sebagai protected code.

Berikut bentuk penulisan *Try-Catch*:

```
try {
      //Protected code
} catch (ExceptionType nama_variabel) {
       //Catch Block
}
```

Penanganan Exception

2. Multiple Catch

Dengan menggunakan multiple catch dapat menangani **lebih dari 1 exception**. Ini memberikan kemampuan untuk merespons secara khusus terhadap berbagai kondisi kesalahan yang mungkin terjadi selama eksekusi program.

Berikut bentuk penulisan *Multiple catch*:

```
try {
      //Protected code
} catch (ExceptionType nama variabel) {
       //Catch Block
} catch (ExceptionType nama variabel) {
       //Catch Block
}
```

Penanganan Exception

3. Finally

Block finally adalah block yang di tambahkan di akhir block try-catch. Finally **akan selalu dijalankan** setelah try-catch baik terjadi exception atau tidak.

Berikut bentuk penulisan finally:

```
try {
      //Protected code
} catch (ExceptionType nama_variabel){
      //Catch Block
} finally {
      //finally Block
}
```

Implementasi Try-catch , finally

```
15  public class Lat_Exception {
16      public static void main(String[] args) {
17          try{
18              int a ,b ,hasil;
19              // membuat scanner baru
20              Scanner keyboard = new Scanner( source:System.in);
21              System.out.println( x:"======Progam Pembagian=====");
22              System.out.println( x:"Masukkan agka 1 = ");
23              a = Integer.parseInt( s:keyboard.next());
24              System.out.println( x:"Masukkan angka ke 2 = ");
25              b = Integer.parseInt( s:keyboard.next());
26              hasil = a/b;
27              System.out.println( x:Integer.toString( i:hasil));
28          }catch(ArithmeticException c){
29              JOptionPane.showMessageDialog( parentComponent:null,  message:"Nilai pembagi tidak boleh 0 !!",  title:"Warning", messageType:2);
30          }catch(NumberFormatException d){
31              JOptionPane.showMessageDialog( parentComponent:null,  message:"Input yang anda masukkan huruf bukan angka !!",  title:"Warning", messageType:2);
32          }finally{
33              System.out.println( x:"Trimakasi sudah menjalankan program");
34          }
35      }
36  }
```

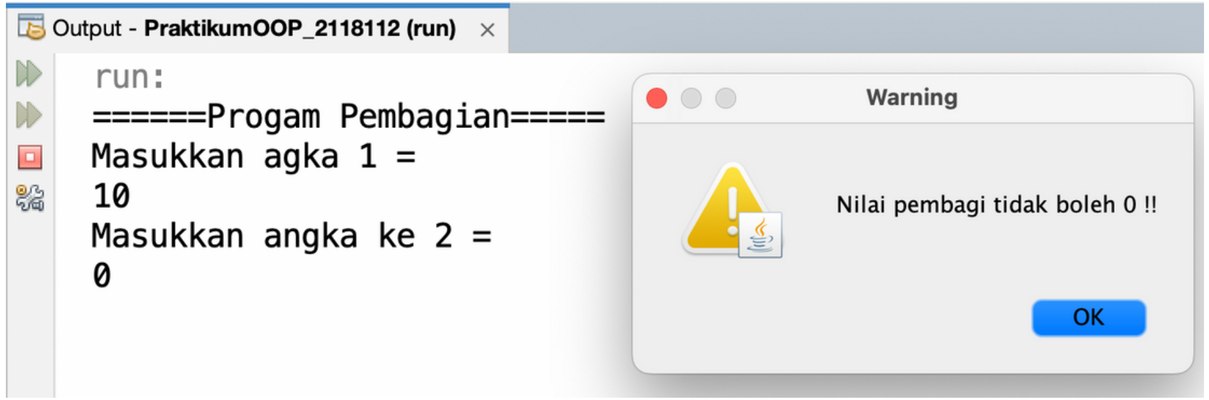
Implementasi Try-catch , finally

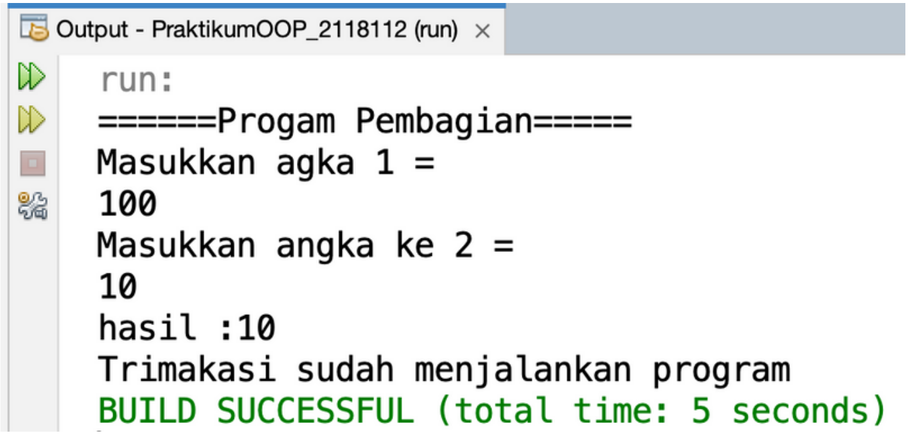
Masukkan Library JoptionPane dan Library Scanner di bawah nama package class yang dibuat. **Scanner** adalah kelas dalam Java yang digunakan untuk membaca input dari berbagai sumber, seperti keyboard (System.in), file, atau string.

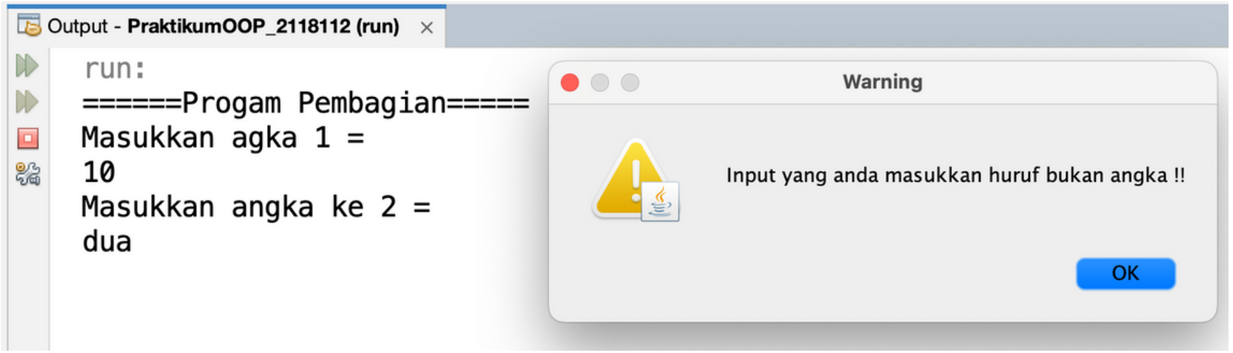
```
package Bab8.Exception;

// mengimpor Scanner dan JOption ke program
import javax.swing.JOptionPane;
import java.util.Scanner;
```

Hasil Try-catch , finally







Keyword Throw dan Throws

1. Throw

   Keyword **throw** digunakan untuk **melempar exception atau bug** yang dibuat secara manual, misalnya saat kita membuat sebuah program dan user salah memasukan input, tetapi program tidak menandakan bahwa itu terjadi error, karena input yang dimasukan tidak berpotensi akan terjadi error. Dengan menggunakan kata kunci throw, kita dapat melempar exception pada kondisi yang telah kita tentukan.Tujuannya memberikan sinyal kepada program bahwa suatu kondisi khusus telah terjadi yang memerlukan pemrosesan khusus.

```
1  if (obj == null) {
2    throw new NullPointerException("Objek tidak diinisialisasi.");
3  }
```

Keyword Throw dan Throws

2. Throws

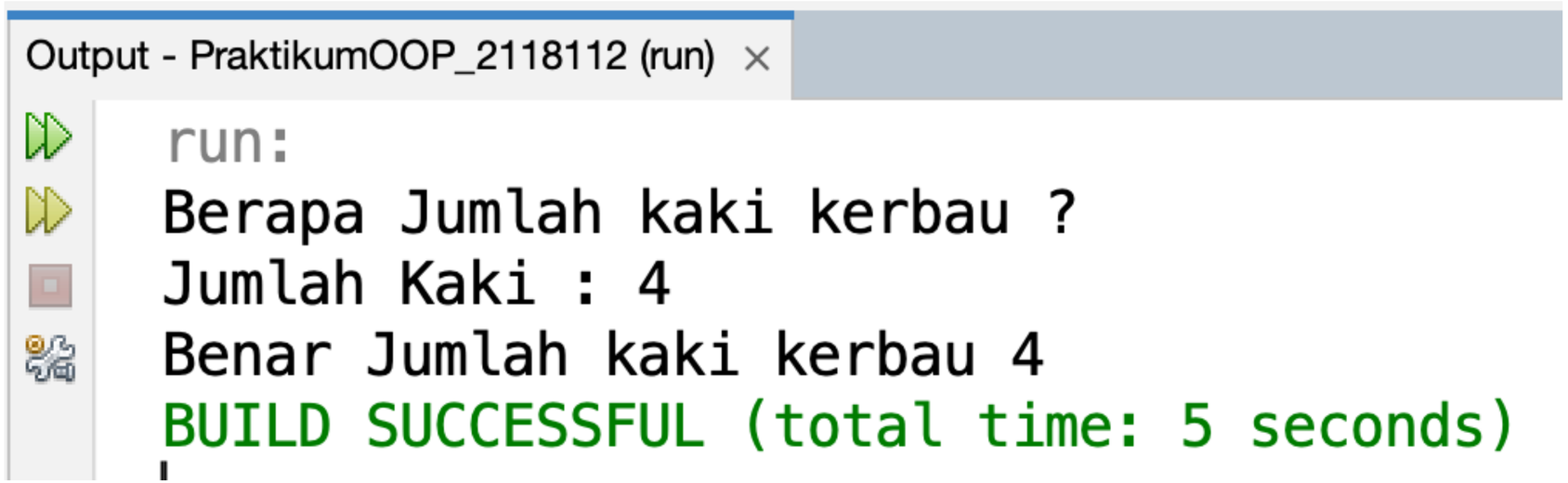
Keyword throws digunakan pada sebuah method yang kemungkinkan berpotensi suatu exception(error), sehingga perlu ditangkap exception/errornya.

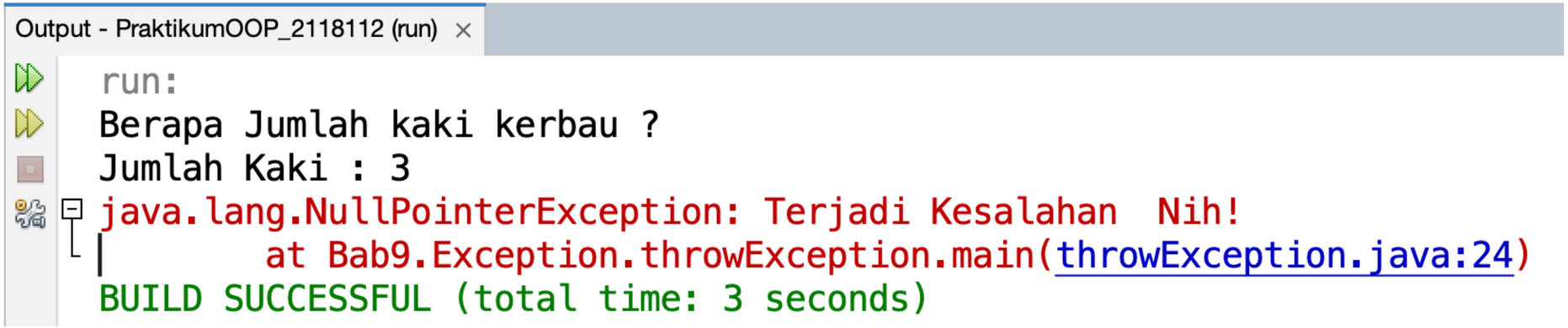
```
try {
  public void metodeDenganException() throws ExceptionType1, ExceptionType2 {
    // Potongan kode yang dapat menyebabkan exception
    // ...
  }
} catch (ExceptionType1 e1) {
  // Tangani exception tipe 1
} catch (ExceptionType2 e2) {
  // Tangani exception tipe 2
}
```

Implementasi Throw

```java
import java.util.Scanner;
public class throwException {
    public static void main(String[] args) {
        Scanner input = new Scanner( source:System.in);
        int jmlKaki;
        System.out.println( x:"Berapa Jumlah kaki kerbau ?");
        try{
            System.out.print( s:"Jumlah Kaki : ");
            jmlKaki = input.nextInt(); //Mendapatkan Input Dari User
            if(jmlKaki != 4){
                //Jika jumlah kaki kerbau lebih atau kurang dari 4, maka akan terjad error
                throw new NullPointerException( s:"Terjadi Kesalahan  Nih!");
            }else{
                System.out.println( x:"Benar Jumlah kaki kerbau 4");
            }
        }catch(NullPointerException e){
            //Menampilkan Pesan Kesalahan
            e.printStackTrace();
        }
    }
}
```

Hasil Throw


Output - PraktikumOOP_2118112 (run)
run:
Berapa Jumlah kaki kerbau ?
Jumlah Kaki : 4
Benar Jumlah kaki kerbau 4
BUILD SUCCESSFUL (total time: 5 seconds)


Output - PraktikumOOP_2118112 (run)
run:
Berapa Jumlah kaki kerbau ?
Jumlah Kaki : 3
java.lang.NullPointerException: Terjadi Kesalahan Nih!
at Bab9.Exception.throwException.main(throwException.java:24)
BUILD SUCCESSFUL (total time: 3 seconds)

Implementasi Throws

```
11  public class throwsException {
12      static void Error() throws ClassNotFoundException{
13          System.out.println( x:"Ada Yang Error Ni!");
14          throw new ClassNotFoundException( s:"Error sudah Saya Tangkap");
15      }
16      public static void main(String[] args) {
17          try{
18              throwsException.Error();
            }catch(Exception e){
                e.printStackTrace();
21          }
22      }
23  }
```

Hasil Throws

Latihan Login

```
public class Login {
    private String username, password;
    public String nama;
    public Login() {
        nama = "Firman Frezy Pradana";
        username = "Firman";
        password = "Firm4n123";
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    boolean CekLogin(String Username , String password){
        if(username.equals( anObject:getUsername()) && password.equals( anObject:getPassword())){
            return true;
        }
        return false;
    }
}
```

Latihan Login

Latihan Login

```
99      private void btn_loginActionPerformed(java.awt.event.ActionEvent evt) {
100         // TODO add your handling code here:
101         Login login = new Login();
102         String uname = null;
103         String password = null;
104         if (txtUsername.getText().isBlank() == false) {
105             uname = txtUsername.getText();
106         }
107
108         if (txtPassword.getText().isBlank() == false) {
109             password = txtPassword.getText();
110         }
111
112         try {
113
114             boolean isAuthenticated = login.CekLogin( Username: uname, password);
115             if (isAuthenticated) {
116                 JOptionPane.showMessageDialog( parentComponent: rootPane, "LOGIN BERHASIL. Selamat datang, " + login.nama + "!");
117             }
118         } catch (Exception a) {
119             System.out.println( x: a.getMessage());
120             JOptionPane.showMessageDialog( parentComponent: rootPane,  message: "Login Gagal . Username/Password tidak boleh kosong");
121         }
122     }
```

"Open NetBeans, and let's write code Java"