MODUL 7

EXCEPTION HANDLING

DESKRIPSI MATERI

Modul 7 yang berjudul *Exception Handling* ini memaparkan tentang bagaimana cara menangkap kesalahan saat program dijalankan agar jalannya program tidak keluar begitu saja dari alur yang ditentukan. Beberapa kata kunci yang digunakan diantaranya adalah *try*, *catch*, *finally*, *throw*, dan *throws*. Selain itu, di sini juga diperkenalkan penggunaan komponen Swing berupa *list* dan *combo box* serta cara menghubungkan beberapa *frame* dalam sebuah program.

PETUNJUK KHUSUS

Urutan jalannya program diatur dalam *method main* dan *exception handling* akan membaca pengecualian sesuai dengan urutan tersebut. Untuk menangkap kesalahan saat program dijalankan, maka Anda harus mengetahui bagian mana yang mungkin mengalami kesalahan dan memasukkannya dalam blok *try* sehingga saat kesalahan terjadi dapat ditangkap oleh blok *catch*.

MODUL 7

EXCEPTION HANDLING

A. Alokasi Waktu

Pertemuan : 13 dan 14

Jam Studi : 2 x 4 JS (8 x 50 menit)

B. Kompetensi Dasar

Mengaplikasikan *exception handling*.

C. Tujuan Praktikum

1. Menerapkan konsep *exception handling*.
2. Menangkap *exception*.
3. Membuat *catch* secara bertingkat.
4. Melemparkan *exception*.
5. Menggunakan klausa *throws*.
6. Membuat *class exception* sendiri.
7. Menggunakan *class* JList dan JComboBox.

D. Dasar Teori

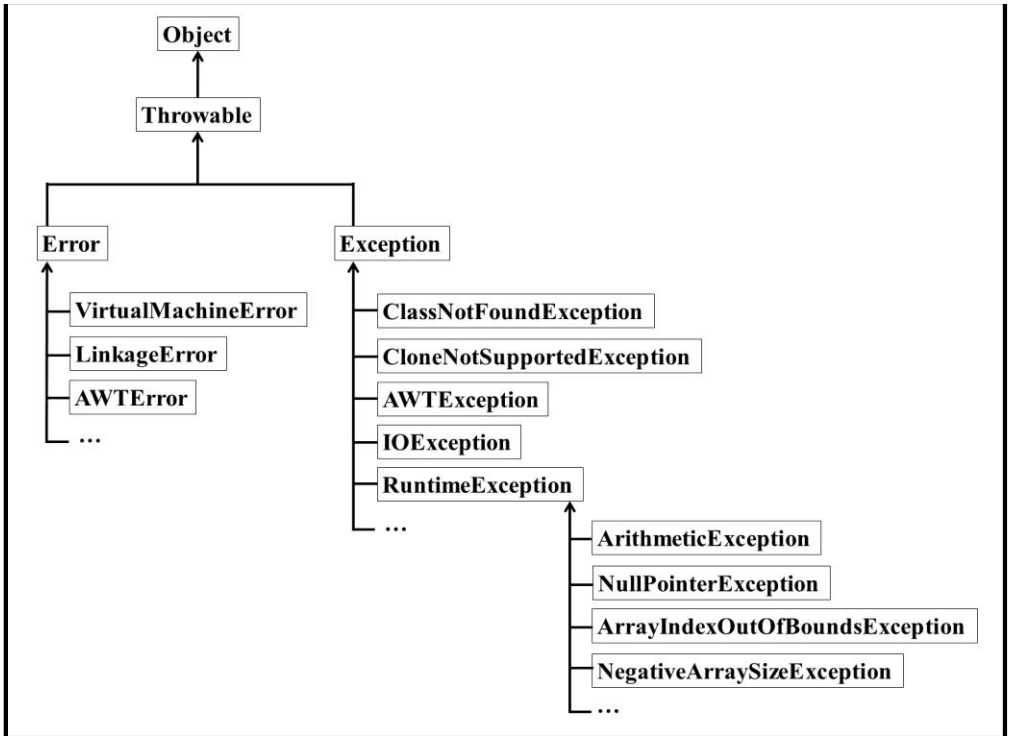
1. Konsep *exception handling*.

Kategori *error* dalam pemrograman diantaranya:

a. *Compilation error*, yaitu *error* yang terjadi saat program dikompilasi.
b. *Runtime error*, yaitu *error* yang terjadi saat program dieksekusi/dijalankan.
c. *Logic error*, yaitu *error* yang terjadi saat program tidak berjalan sesuai dengan harapan.

Exception (eksepsi atau pengecualian) adalah kondisi yang menyebabkan program menjadi *hang* (tergantung) atau *quit* (kaluar) dari alur normal yang telah ditentukan pada saat program dijalankan. *Exception* dipicu oleh *runtime error*, yaitu *error* atau kesalahan yang terjadi saat program dieksekusi oleh *interpreter*. Sedangkan *exception handling* (penanganan pengecualian) merupakan mekanisme untuk menangkap *exception* tersebut.

2. Hierarki *class* Exception.



Gambar 7.1 Hierarki *Exception*

3. *Keyword* pada *exception handling*.

Keyword pada *exception handling* diantaranya:

a. *Try*. Blok *try* digunakan untuk menempatkan kode program yang mungkin menyebabkan terjadinya *exception*.
b. *Catch*. Blok *catch* digunakan untuk menangkap kesalahan yang terjadi pada blok *try*.
c. *Finally*. Blok *finally* akan selalu dijalankan, tidak peduli apakah terjadi *exception* atau tidak.
d. *Throw*. Klausa *throw* digunakan untuk melemparkan *exception* yang terjadi.
e. *Throws*. Klausa *throws* digunakan untuk mendeklarasikan *method* yang mungkin akan mengalami *exception*.

4. Pengenalan *class* JList dan JComboBox.

Class JList merupakan *class* dari *package* javax.swing yang digunakan untuk membuat komponen berupa *list* atau daftar pilihan. Dengan menggunakan *list*, seluruh alternatif pilihan yang tersedia dapat ditampilkan secara bersamaan.

Class JComboBox merupakan *class* dari *package* javax.swing yang digunakan untuk membuat komponen berupa *combo box* atau kotak kombo yang berisi pilihan seperti pada *list*. Perbedaannya adalah dengan menggunakan *combo box*, seluruh alternatif pilihan yang tersedia ditampilkan seperti menu *drop down*. Oleh karena itu, *combo box* juga dikenal sebagai *drop down list*.

E. Latihan

1. Program tanpa *exception handling*.

Tulislah dan simpan kode program di bawah ini:

```
public class WithoutExceptionHandling
{
    public static void main(String[] args)
    {
        System.out.println("\n********* PROGRAM TANPA EXCEPTION
HANDLING *********\n");
        int hasil = 9/0; //penyebab exception
        System.out.println("Hasil pembagian = "+hasil);
        System.out.println("Pernyataan setelah bebas dari
exception.");
    }
}
```

a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program ini!

2. Menangkap *exception* dengan blok *try-catch*.

Tulislah dan simpan kode program di bawah ini:

```
public class WithArithmeticException
{
    public static void main(String[] args)
    {
        System.out.println("\n********* PROGRAM DENGAN EXCEPTION
HANDLING *********\n");
        try
        {
            int hasil = 9/0; //penyebab exception
            System.out.println("Hasil pembagian = "+hasil);
            System.out.println("Pernyataan dalam blok try setelah
bebas dari exception.");
        }
        catch(ArithmeticException exc)
        {
            System.err.println("ArithmeticException menangkap
exception hasil pembagian oleh nol.");
            System.err.println("\nException yang ditangkap adalah
: "+exc);
        }
```

```
        System.out.println("\nPernyataan di luar blok try-
catch.");
    }
}
```

a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program ini!

c. Buatlah pernyataan yang tidak menyebabkan *exception* sehingga pernyataan terakhir dalam blok *try* dapat ditampilkan! Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya! Berikan penjelasan singkat!

3. Membuat blok *try-catch-finally*.

Tulislah dan simpan kode program di bawah ini:

```
public class TryCatchFinally
{
    public static void main(String[] args)
    {
        System.out.println("\n********* MENGGUNAKAN BLOK TRY-
CATCH-FINALLY *********\n");
        try
        {
            int hasil = 9/0; //penyebab exception
            System.out.println("Hasil pembagian = "+hasil);
            System.out.println("Pernyataan dalam blok try setelah
bebas dari exception.");
        }
        catch(ArithmeticException exc)
        {
            System.err.println("ArithmeticException menangkap
exception hasil pembagian oleh nol.");
            System.err.println("\nException yang ditangkap adalah
: "+exc);
        }
        finally
        {
            System.out.println("\nPernyataan dalam blok
finally.");
        }
        System.out.println("\nPernyataan di luar blok try-catch-
finally.");
    }
}
```

a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program ini!

4. Membuat *catch* secara bertingkat.

Tulislah dan simpan kode program di bawah ini:

```
public class MultipleCatch
{
    public static void main(String[] args)
    {
        System.out.println("\n********* MENGGUNAKAN MULTIPLE CATCH
*********\n");
        try
        {
            int[] array = new int[9]; //deklarasi array berukuran
9 buah elemen
            array[9] = 13; //penyebab exception
            System.out.println("Elemen array indeks ke 9 adalah
"+array[9]);
            System.out.println("Pernyataan dalam blok try setelah
bebas dari exception.");
        }
        catch(ArrayIndexOutOfBoundsException exc)
        {
            System.err.println("Anda mengakses array di luar
indeks yang dideklarasikan.");
        }
        catch(NegativeArraySizeException exc)
        {
            System.err.println("Anda mendeklarasikan array dengan
ukuran negatif.");
        }
        catch(Exception exc)
        {
            System.err.println("Anda melakukan pembagian bilangan
oleh nol.");
        }
    }
}
```

a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program ini!

c. Buatlah pernyataan yang tidak menyebabkan *exception* sehingga pernyataan terakhir dalam blok *try* dapat ditampilkan! Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya! Berikan penjelasan singkat!

d. Buatlah pernyataan yang mengandung *exception* sehingga blok `catch (NegativeArraySizeException exc)` dijalankan! Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya! Berikan penjelasan singkat!

e. Buatlah pernyataan yang mengandung *exception* sehingga blok `catch (Exception exc)` dijalankan! Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya! Berikan penjelasan singkat!

f. Pindahkan letak blok `catch(Exception exc)` di bagian sebelum blok `catch(ArrayIndexOutOfBoundsException exc)`! Lakukan kompilasi program kemudian tunjukkan hasilnya! Apakah blok `catch (Exception exc)` harus diletakkan di bagian terakhir dalam *multiple exception*? Jelaskan!

5. Melemparkan *exception* dengan klausa *throw*.

Tulislah dan simpan kode program di bawah ini:

```
public class KlausaThrow
{
    public static void main(String[] args)
    {
        String input = "Throw RuntimeException";

        System.out.println("\n********* MENGGUNAKAN KLAUSA THROW
*********\n");
        try
        {
            if(input.equals("Throw RuntimeException"))
            {
                throw new RuntimeException("Melempar Exception");
            }
            else if(input==null)
            {
                throw new NullPointerException();
            }
            else
            {
                System.out.println("Input adalah : "+input);
            }
            System.out.println("\nPernyataan dalam blok try
setelah bebas dari pelemparan exception.");
        }
        catch(Exception exc)
        {
            System.err.println("Exception ditangkap di sini.");
            System.err.println("\nException yang ditangkap adalah
: "+exc);
        }
    }
}
```

a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program ini!

c. Buatlah pernyataan yang dapat menjalankan blok **`else if(input==null)`**! Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya! Berikan penjelasan singkat!

d. Buatlah pernyataan yang dapat menjalankan blok **`else`**! Lakukan kompilasi dan eksekusi program lalu tunjukkan hasilnya! Berikan penjelasan singkat!

6. Menggunakan klausa *throws*.

Tulislah dan simpan kode program di bawah ini:

```
public class KlausaThrows
{
    public static void uji(int angka) throws NullPointerException,
ArithmeticException
    {
        if(angka<0)
        {
            throw new NullPointerException("KESALAHAN : Null
Pointer Exception");
        }
        else
        {
            throw new ArithmeticException("KESALAHAN : Arithmetic
Exception");
        }
    }
    public static void main(String[] args)
    {
        System.out.println("\n********* MENGGUNAKAN KLAUSA THROWS
*********\n");
        try
        {
            //uji(-12); //penyebab NullPointerException
            //uji(0); //penyebab ArithmeticException
        }
        catch(Exception exc)
        {
            System.err.println("Exception ditangkap di sini");
            System.err.println("\n\tPemberitahuan!!!
\n"+exc.getMessage());
        }
        System.out.println("\nPernyataan di luar blok try-
catch.");
    }
}
```

a. Lakukan kompilasi dan eksekusi program terhadap penyebab terjadinya NullPointerException dan ArithmeticException lalu tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program ini!

7. Membuat *class exception* sendiri.

Tulislah dan simpan kode program di bawah ini:

Kode program *class* CreateExceptionHandling:

```
class CreateExceptionHandling extends Exception
{
    private int bilangan;

    CreateExceptionHandling()
    {

    }

    CreateExceptionHandling(String pesan)
    {
        super(pesan);
    }

    CreateExceptionHandling(String pesan, int nilai)
    {
        super(pesan);
        bilangan = nilai;
    }

    public int getBilangan()
    {
        return bilangan;
    }
}
```

Kode program *class* TryExceptionHandling:

```
public class TryExceptionHandling
{
    public static int hitungFaktorial(int n) throws
CreateExceptionHandling
    {
        if(n<0)
        {
            throw new CreateExceptionHandling("Bilangan tidak
boleh negatif", n);
        }
        int hasil = 1;
        for(int i=n; i>=1; i--)
        {
            hasil *= i;
        }
        return hasil;
    }

    public static void main(String[] args)
    {
        System.out.println("\n********* MEMBUAT CLASS EXCEPTION
SENDIRI *********\n");
        try
        {
            System.out.println("Pada saat menghitung 5!");
```

```
            System.out.println("Hasil = "+hitungFaktorial(5));
            System.out.println("\nPada saat menghitung -5!");
            System.out.println("Hasil = "+hitungFaktorial(-5));
        }
        catch(CreateExceptionHandling exc)
        {
            System.err.println("Bilangan = "+exc.getBilangan());
            exc.printStackTrace();
        }
    }
}
```

a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program ini!

c. Jelaskan fungsi perintah `exc.printStackTrace();`!

8. Menggunakan *class* JList.

Tulislah dan simpan kode program di bawah ini:

```
import javax.swing.*;
import javax.swing.event.*;

public class TryList extends JFrame implements
ListSelectionListener
{
    private JList daftar;
    private JTextArea tArea;

    public TryList()
    {
        super("MENGGUNAKAN LIST");
        setSize(350,125);

        String mataKuliah[] = {"Fisika Teknik", "Komputasi
Numerik", "Matematika Diskrit"}; //mengatur item pada list
        daftar = new JList(mataKuliah); //menugaskan objek daftar
dari class JList
        daftar.setSelectedIndex(0);
        daftar.addListSelectionListener(this); //mendeteksi event
dari list
        tArea = new JTextArea(4,15);
        tArea.setEditable(false);

        JPanel p1 = new JPanel();
        p1.add(daftar);
        add("West", p1);
        JPanel p2 = new JPanel();
        p2.add(tArea);
        add("East", p2);

        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
```

```
    public void valueChanged(ListSelectionEvent e) //menangani
event
    {
        tArea.setText("\nItem yang terpilih :
\n"+daftar.getSelectedValue()); //menampilkan pilihan
    }

    public static void main(String[] args)
    {
        TryList frame = new TryList();
    }
}
```

a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program ini!

c. Jelaskan fungsi perintah `daftar.setSelectedIndex(0);`!

9. Menggunakan *class* JComboBox.

Tulislah dan simpan kode program di bawah ini:

```
import javax.swing.*;
import java.awt.event.*;

public class TryComboBox extends JFrame implements ActionListener
{
    private JComboBox daftar;
    private JTextArea tArea;

    public TryComboBox()
    {
        super("MENGGUNAKAN COMBO BOX");
        setSize(350,125);

        String mataKuliah[] = {"Fisika Teknik", "Komputasi
Numerik", "Matematika Diskrit"}; //mengatur item pada combo box
        daftar = new JComboBox(mataKuliah); //menugaskan objek
daftar dari class JComboBox
        daftar.addActionListener(this); //mendeteksi event dari
combo box
        tArea = new JTextArea(4,15);
        tArea.setEditable(false);

        JPanel p1 = new JPanel();
        p1.add(daftar);
        add("West", p1);
        JPanel p2 = new JPanel();
        p2.add(tArea);
        add("East", p2);

        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
```

```
    public void actionPerformed(ActionEvent e) //menangani event
    {
        JComboBox cbb = (JComboBox)e.getSource();
        String keterangan = (String)cbb.getSelectedItem();
        tArea.setText("\nItem yang terpilih : \n"+keterangan);
//menampilkan pilihan
    }
    public static void main(String[] args)
    {
        TryComboBox frame = new TryComboBox();
    }
}
```

a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program ini!

10. Menghubungkan *frame*.

Tulislah dan simpan kode program di bawah ini:

Kode program *class* Button1:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class Button1 extends JFrame implements ActionListener
{
    private JButton btnEvent;
    private JLabel label;

    public Button1()
    {
        super("SHOW BUTTON 1");
        setSize(300,200);
        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout(FlowLayout.CENTER,0,50));
        label = new JLabel("Menampilkan");
        btnEvent = new JButton("BUTTON 1");
        panel.add(label);
        panel.add(btnEvent);
        add("North", panel);
        btnEvent.addActionListener(this);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource()==btnEvent)
        {
            Button2 frame = new Button2(); //menampilkan frame
dari class Button2
            setVisible(false); //menyembunyikan frame dari class
Button1
        }
    }
}
```

Kode program *class* Button2:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class Button2 extends JFrame implements ActionListener
{
    private JButton btnEvent;
    private JLabel label;

    public Button2()
    {
        super("SHOW BUTTON 2");
        setSize(300,200);
        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout(FlowLayout.CENTER,0,50));
        label = new JLabel("Menampilkan");
        btnEvent = new JButton("BUTTON 2");
        panel.add(label);
        panel.add(btnEvent);
        add("South", panel);
        btnEvent.addActionListener(this);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource()==btnEvent)
        {
            Button1 frame = new Button1(); //menampilkan frame
dari class Button1
            setVisible(false); //menyembunyikan frame dari class
Button2
        }
    }
}
```

Kode program *class* TryFrame:

```
public class TryFrame
{
    public static void main(String[] args)
    {
        Button1 frame = new Button1(); //menampilkan frame dari
class Button1
    }
}
```

a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program ini!

F. Tugas Praktikum

1. Membuat *multiple catch* dengan *class exception handling* sendiri.

a. Buatlah sebuah program berbasis *console* di bidang perfilman dengan membuat *class exception handling* sendiri! Terapkan penggunaan *class* BufferedReader untuk membaca *input* data dari *user*! Gunakan blok *try*, minimal 3 *catch* dan *finally*! Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program yang Anda buat!

2. Menggunakan *list* dan *combo box*.

a. Buatlah sebuah program berbasis GUI dengan memanfaatkan penggunaan *list* dan *combo box*! Buatlah lebih dari satu *frame* sehingga antara satu *frame* dengan *frame* yang lain dapat saling berhubungan! Tampilkan gambar yang berbeda sebagai akibat dari pemilihan item yang berbeda pada *list* atau *combo box*! Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program yang Anda buat!

G. Tugas Rumah

1. Buatlah sebuah program login system yang meminta inputan username dan password dengan dilengkapi exeption handling. Rincian program sebagai berikut:
   - Username boleh nama(string) atau angka(int) atau terdiri dari kombinasi nama dan angka
   - Password boleh nama(String) atau angka (int) atau terdiri dari kombinasi nama dan angka
   - Program akan menangkap dan menampilkan pesan apabila terjadi kesalahan saat proses login dilakukan. Contoh (Login Failed, Username / Password Salah)
   - Program dapat menampilkan exeption ketika kita menambahkan tanda baca saat proses login.
2. Jelaskan megapa diperlukan exception handling
3. Jelaskan kapan kondisi yang tepat penggunaan exception handling pada program