Praktikum 9

Exception Handling

Tujuan

Memahami dan menerapkan konsep penanganan error pada pemrograman berorientasi objek.

Dasar Teori

Exception merupakan subkelas dari kelas *java.lang.Throwable*, bukalah dokumentasi java (*java doc*) untuk lebih menyakinkan anda. Karena *Exception* adalah sebuah kelas maka hakikatnya ketika program berjalan dan muncul sebuah bug atau kesalahan maka bug tersebut dapat dianggap sebuah object. Sehingga ketika object ini di tampilkan di layar maka java akan secara otomatis memanggil method *toString* yang terdapat dalam object bertipe *Exception* ini. Java memberikan akses kepada developer untuk mengambil object bug yang terjadi ini dengan mekanisme yang dikenal *Exception Handling*. *Exception handling* merupakan fasilitas di java yang memberikan flexibilitas kepada developer untuk menangkap bug atau kesalahan yang terjadi ketika program berjalan. Contoh *Exception Handling* akan dibahas pada bagian berikutnya.

Perbedaan antara *Class Error* dan *Class Exception* di java, seperti yang telah dijelaskan diatas bahwa kelas *Exception* merupakan kelas turunan dari kelas *Throwable* di *package Java.Lang*. Selain Exception, java.lang.Throwable juga memiliki subclass yaitu class Error. Tentu, kita bertanya-tanya, sebetulnya apa sih perbedaan antara class Error dengan class Exception. *Error* dan *exception* pada dasarnya berbeda, *error* merupakan masalah yang muncul tapi tidak ada alasan yang kuat untuk menangkapnya. Sedangkan *exception* merupakan kesalahan kecil yang muncul dan ingin diperlakukan sesuai keinginan developer. Ada 5 keyword penting dalam java dalam hal *exception handling* :

1. try

Keyword *try* biasanya digunakan dalam suatu block program. Keyword ini digunakan untuk mencoba menjalankan block program kemudian mengenai sasaran dimana munculnya kesalahan yang ingin diproses. Keyword ini juga harus dipasangkan dengan keyword *catch* atau keyword *finally* yang akan dibahas pada point kedua dan ketiga.

2. catch

Dalam java, keyword *catch* harus dipasangkan dengan *try*. Kegunaan keyword *catch* adalah untuk menangkap kesalahan atau bug yang terjadi dalam block *try*. Setelah

menangkap kesalahan yang terjadi maka programmer dapat melakukan hal apapun pada block *catch* sesuai keinginan programmer.

Contoh penggunaan :

```
public class A
{
    public static void main(String[] args) {
        try
        {
            int a = 1 / 0;
            // berpotensi untuk menimbulkan kesalahan yaitu
            // pembagian dengan bilangan 0
            System.out.println("perintah selanjutnya");
        }
        catch (Exception kesalahan)
        {
            System.err.println(kesalahan);
        }
    }
}
```

Perhatikan contoh diatas, ada beberapa hal penting yang perlu dilihat. Pertama, block program yang diyakini akan menimbulkan kesalahan maka ada di dalam block *try* and *catch*. Kedua, kesalahan yang muncul akan dianggap sebagai object dan ditangkap *catch* kemudian di assign ke variable kesalahan dengan tipe *Exception*. Ketiga, perintah setelah munculnya kesalahan pada block *try* tidak akan dieksekusi. Keyword *catch* juga dapat diletakan berulang-ulang sesuai dengan kebutuhan. Contoh penggunaan:

```
public class A
{
    public static void main(String[] args) {
        try
        {
            int a = 1/0;
            // berpotensi untuk menimbulkan kesalahan
            // yaitu pembagian dengan bilangan 0

            System.out.println("perintah selanjutnya");
        }
        catch(NullPointerException e)
        {
        }
        catch(ArrayIndexOutOfBoundsException e)
        {
        }
        catch(Exception e)
        {
        }
    }
}
```

3. finally

Keyword *finally* merupakan keyword yang menunjukan bahwa block program tersebut akan selalu dieksekusi meskipun adanya kesalahan yang muncul atau pun tidak ada. Contoh implementasinya pada program :

```
public class A
{
    public static void main(String[] args) {
        try
        {
            int a = 1/0;
        }
        catch (Exception e)
        {
            System.out.println("ada kesalahan yang muncul");
        }
        finally
        {
            System.out.println("terima kasih telah menjalankan program");
        }
    }
}
```

Perhatikan kedua contoh diatas, block *finally* akan selalu dieksekusi meskipun adanya kesalahan atau tidak pada block *try*. Berbeda dengan keyword *catch* keyword *finally* hanya dapat diletakan 1 kali setelah keyword *try*.

4. throw

Keyword ini digunakan untuk melemparkan suatu *bug* yang dibuat secara manual. Contoh program:

```
public class A
{
    public static void main(String[] args) {
        try
        {
            throw new Exception("kesalahan terjadi");
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

Seperti yang anda lihat pada program diatas, pada keyword `throw new Exception("kesalahan terjadi");` akan melempar object bertipe *exception* yang merupakan subclass dari class *Exception* sehingga akan dianggap sebagai suatu kesalahan yang harus ditangkap oleh keyword *catch*. Perhatikan contoh berikut ini :

```
public class A
{
    public static void main(String[] args) {
        try
        {
            throw new B();
            //cobalah hilangkan "throw" ganti dengan "new B();" saja
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
class B extends Exception
{
```

```
    B()
    {
    }
    public String toString()
    {
        return "object dengan tipe kelas B";
    }
}
```

Program diatas telah mendefinisikan suatu kelas B *extends* dari kelas *Exception*. Ketika kita melakukan throw new B(); maka object dari kelas bertipe B ini akan dianggap kesalahan dan ditangkap oleh block catch. Sekarang jika anda menghilangkan keyword throw apa yang terjadi?

5. throws

Keyword throws digunakan dalam suatu method atau kelas yang mungkin menghasilkan suatu kesalahan sehingga perlu ditangkap error-nya. Cara mendefinisikannya dalam method adalah sebagai berikut : <method modifier> type method-name throws exception-list1, exceptio-list2, … {}. Contoh Program :

```
public class A
{
    public static void main(String[] args) {
        try
        {
            f();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
    public static void f() throws
           NullPointerException,ArrayIndexOutOfBoundsException
    {
        //implementasi method
        throw new NullPointerException();
        //throw new ArrayIndexOutOfBoundsException();
    }
}
```

Perhatikan contoh pengguaan keyword *throws* pada method. Ketika method tersebut dipanggil dalam block *try*. Maka method tersebut akan membuat object yang merupakan *subclass* dari *class* Throwable dan method tersebut akan melemparkan kesalahan yang ada dalam block method kedalam block *try*. Di dalam block *try*, kesalahan tersebut kemudian ditangkap kedalam block *catch*.

Percobaan 1

Jalankan program dibawah ini, amati error yang terjadi. Kemudian perbaiki seperti pada program dibawahnya.

```
public class Exception {

    public static void main(String[] args) {
       int a[]=new int[5];
       a[5]=100;
    }
}
```

Pembetulan program:

```
public class Exception {

    public static void main(String[] args) {
        int a[]=new int[5];
        try
        {
            a[5]=100;
        }
        catch(Exception e)
        {
            System.out.println("Terjadi pelanggaran memory");
        }
    }

}
```

Percobaan 2

Jalankan program dibawah ini, amati error yang terjadi. Kemudian perbaiki seperti pada program dibawahnya.

```
public class Exception2 {
    public static void main(String[] args) {
       int i=0;
       String greeting[]={
            "Hello World!",
            "No, I mean it!",
            "Hello World"
       };
       while(i<4)
       {
            System.out.println(greeting[i]);
            i++;
       }
    }
}
```

Pembetulan program:

```
public class Exception2 {
    public static void main(String[] args) {
        int i=0;
        String greetings[]={
            "Hello World!",
            "No,I mean it!",
            "HELLO WORLD!"
        };
        while(i<4)
        {
            try
            {
                System.out.println(greetings[i]);
                i++;
            }
            catch(ArrayIndexOutOfBoundsException e)
            {
                System.out.println("Resetting index value");
                i=0;
            }
        }
    }
}
```

Percobaan 3

Jalankan program dibawah ini, amati error yang terjadi. Kemudian perbaiki seperti pada program dibawahnya.

```
public class Exception3 {

    public static void main(String[] args) {
        int bil=10;
        System.out.println(bil/0);
    }
}
```

Pembetulan Program:

```
public class Exception3 {

    public static void main(String[] args) {
       int bil=10;
       try
       {
           System.out.println(bil/0);
       }
       catch(Exception e)
       {
           System.out.println("Ini menghandle error yang terjadi");
       }
    }
}
```

Pembetulan Program:

```
public class CobaException3 {

    public static void main(String[] args) {
        int bil=10;
        try
        {
            System.out.println(bil/0);
        }
        catch(ArithmeticException e)
        {
            System.out.println("Terjadi Aritmatika error");
        }
        catch(Exception e)
        {
            System.out.println("Ini menghandle error yang terjadi");
        }
    }
}
```

Percobaan 4

Jalankan program dibawah ini, amati error yang terjadi. Kemudian perbaiki seperti pada program dibawahnya.

```
public class CobaException4 {
    public static void main(String[] args) {
       int bil=10;
       String b[]={"a","b","c"};
       try
       {
           System.out.println(b[3]);
           System.out.println(bil/0);
       }
       catch(ArithmeticException e)
       {
           System.out.println("Terjadi Aritmatika error");
       }
       catch(ArrayIndexOutOfBoundsException e)
       {
           System.out.println("Melebihi jumlah array");
       }
       catch(Exception e)
       {
           System.out.println("Ini menghandle error yang terjadi");
       }
       }
}
```

Pembetulan Program:

```
public class CobaException4 {
    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 e)
        {
            System.out.println("Terjadi Aritmatika error");
        }
        catch(ArrayIndexOutOfBoundsException e)
        {
            System.out.println("Melebihi jumlah array");
        }
        catch(Exception e)
        {
            System.out.println("Ini menghandle error yang terjadi");
        }
        }
}
```

Percobaan 5

Jalankan program dibawah ini, berikan analisa penggunaan try dan catch pada program dibawah ini.

```
public class Exception5 {

    public static void main(String[] args) {
        int bil=10;
        try
        {
            System.out.println(bil/0);
        }
        catch(ArithmeticException e)
        {
            System.out.println("Pesan error:  ");
            System.out.println(e.getMessage());
            System.out.println("Info stack erase");
            e.printStackTrace();
            e.printStackTrace(System.out);
        }
        catch(Exception e)
        {
            System.out.println("Ini menghandle error yang terjadi");
        }
    }

}
```

Percobaan 6

Jalankan program dibawah ini, berikan analisa penggunaan try dan catch pada program dibawah ini.

```
public class ThrowExample {

    static void demo()
    {
        NullPointerException t;
        t=new NullPointerException("Coba Throw");
        throw t;
```

```
        // Baris ini tidak lagi dikerjakan;
        System.out.println("Ini tidak lagi dicetak");
    }

    public static void main(String[] args) {
        try
        {
            demo();
            System.out.println("Selesai");
        }
        catch(NullPointerException e)
        {
            System.out.println("Ada pesan error: "+e);
        }
    }

}
```

Percobaan 7

Jalankan program dibawah ini, berikan analisa penggunaan try dan catch pada program dibawah ini.

```
public class ThrowExample2 {

    public static void main(String[] args) {
        try
        {
            throw new Exception("Here's my Exception");
        }
        catch(Exception e)
        {
            System.out.println("Caught Exception");
            System.out.println("e.getMessage():"+e.getMessage());
            System.out.println("e.toString():"+e.toString());
            System.out.println("e.printStackTrace():");
            e.printStackTrace();
        }
    }
}
```

Percobaan 8

Jalankan program dibawah ini, berikan analisa penggunaan throws pada program dibawah ini.

```
import java.io.*;

public class Test3 {
    public void methodA(){
        System.out.println("Method A");
    }
    public void methodB() throws IOException
    {
        System.out.println(20/0);
        System.out.println("Method B");
    }
}
```

```
class Utama
{
     public static void main(String[] args) throws IOException
     {
        Test3 c=new Test3();
        c.methodA();
        c.methodB();
     }
}
```

Kemudian coba ubah class utama diatas dengan yang program baru di bawah ini:

```
class Utama
{
     public static void main(String[] args)
     {
           Test3 o=new Test3();
           o.methodA();
           try
           {
                o.methodB();
           }
           catch(Exception e)
           {
               System.out.println("Error di Method B");
           }
           finally
           {
               System.out.println("Ini selalu dicetak");
           };
     }
}
```

Percobaan 9

Jalankan program membalik string (reverse) berikut ini, coba hapus isi dari method reverse (`"This is a string")` kemudian jalankan kembali.

```
class Propagate {

    public static void main(String[] args)
    {
        try
        {
            System.out.println(reverse("This is a string"));
        }
        catch(Exception e)
        {
            System.out.println("The String was blank");
        }
        finally
        {
            System.out.println("All done");
        }
    }


    public static String reverse(String s) throws Exception
    {
         if(s.length()==0)
```

```
          {
               throw new Exception();
          }
          String reverseStr = "";
          for(int i=s.length()-1 ; i>=0 ; --i){
              reverseStr+=s.charAt(i);
          }
          return reverseStr;
     }

}
```

Percobaan 10

Program berikut ini menerapkan IOException ketika membuat objek dari class RandomAccessFile (java.io) yang menghasilkan file txt.

```
class RandomAccessRevisi
{
    public static void main(String[] args) {

        String bookList[]={"Satu","Dua","Tiga"};
        int yearList[]={1920,1230,1940};
        try
        {
            RandomAccessFile books = new RandomAccessFile
("books.txt","rw");

            for(int i=0;i<3;i++)
            {
                 books.writeUTF(bookList[i]);
                 books.writeInt(yearList[i]);
            }
            books.seek(0);
            System.out.println(books.readUTF()+" "+books.readInt());
            System.out.println(books.readUTF()+ " "+books.readInt());
            books.close();

        }
        catch(IOException e)
        {
            System.out.println("Indeks melebihi batas");
        }
        System.out.println("test");
    }

}
```

Percobaan 11

Program berikut ini menunjukkan proses throw and catch pada class yang extends Throwable.

```
class RangeErrorException extends Throwable
{
    public RangeErrorException(String s)
    {
         super(s);
    }
```

```
    public static void main(String[] args)
    {
         int position=1;
         try
         {
             if(position>0)
             {
                 throw new RangeErrorException("Position " +position);
             }
         }
         catch(RangeErrorException e)
         {
             System.out.println("Range error: " +e.getMessage());
         }
         System.out.println("This is the last program.");
    }
}
```

Percobaan 12

Program berikut ini menunjukkan proses throw and catch pada class yang extends Exception.

```
class MyException extends Exception{

    private String Teks;
    MyException(String s)
    {
        Teks="Exception generated by: "+s;
        System.out.println(Teks);
    }
}

class Eksepsi
{
    static void tampil(String s)throws Exception
    {
         System.out.println("Tampil");
         if(s.equals("amir"))
         {
               throw new MyException(s);
         }
         System.out.println("OK!");
    }

    public static void main(String[] args)throws Exception
    {
        try
        {
            tampil("ali");
            tampil("amir");
        }
        catch(MyException ex)
        {
            System.out.println("Tangkap:"+ex);
        }
    }
}
```