MODUL PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

EXCEPTION HANDLING

Deskripsi Singkat

Praktikum pemrograman berorientasi objek adalah praktikum yang menggunakan bahasa Java sebagai bantuan dalam memahami konsep pemrograman berorientasi objek. Materi praktikum berisi teori, latihan dan soal pemrograman.

Tujuan

1. Menggunakan blok try-catch.
2. Menggunakan finally.
3. Menggunakan throw dan throws.

Prasyarat

Siswa telah melakukan praktikum 1-10.

Materi 1 : Exception dan Exception Handling

Exception merupakan suatu kondisi abnormal yang terjadi saat menjalankan program yang dapat mengganggu jalannya program secara normal. Contohnya pada operasi pembagian, jika penyebut bernilai 0 akan memunculkan ArithmeticException.

```
int a=50/0;      //ArithmeticException
```

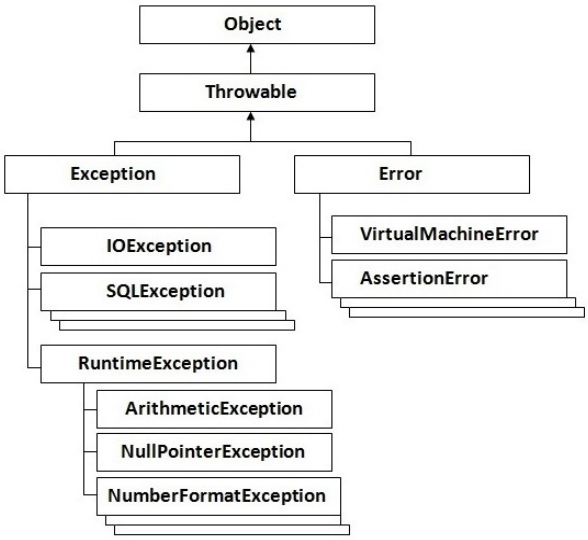
Karena dalam Java segala sesuatu merupakan objek, maka exception juga direpresentasikan dalam sebuah objek yang menjelaskan tentang exception tersebut. Mekanisme untuk menangani exception tersebut dinamakan exception handling. Sehingga exception handling akan menjaga aliran normal dari program/aplikasi.

Secara umum terdapat 3 tipe exception yaitu:

1. Checked exception. Exception tipe ini akan dicek saat **kompilasi**. Semua class yang extends (turunan) class Throwable (kecuali class RuntimeException dan Error) termasuk ke dalam tipe checked exception.
2. Unchecked exception. Exception tipe ini baru akan dicek saat runtime. Semua class yang extends (turunan) class RuntimeException termasuk ke dalam tipe unchecked exception. Contoh seperti ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException.

3. Error. Error merupakan kesalahan fatal yang tidak dapat diperbaiki. Contoh seperti OutOfMemory, AssertionError, VirtualMachineError.

Hirarki tipe exception tersebut dapat dilihat pada gambar berikut ini.



Terdapat 2 penanganan exception:

1. Menangani sendiri exception. Dengan menggunakan blok try-catch-finally.
2. Meneruskannya ke luar dengan cara membuat objek tentang exception tersebut dan melemparnya (throw) keluar agar ditangani oleh kode memanggil method tersebut.

Terdapat 5 keyword dalam Java untuk menangani exception yaitu try, catch, finally, throw dan throws.

Materi 2 : Blok try-catch-finally

Exception yang ditangani sendiri menggunakan blok try-catch-finally. Walaupun namanya try-catch-finally, catch dan finally adalah *optional*/pilihan. Boleh ada atau tidak ada.

- Blok try akan berisi bagian yang akan menghasilkan exception.
- Blok catch akan menangkap exception lalu menanganinya.
- Blok finally merupakan blok yang akan selalu dilakukan, saat ada atau tiada exception.

Sintaks:

```
try{
      //code yang akan menghasilkan exception
}
```

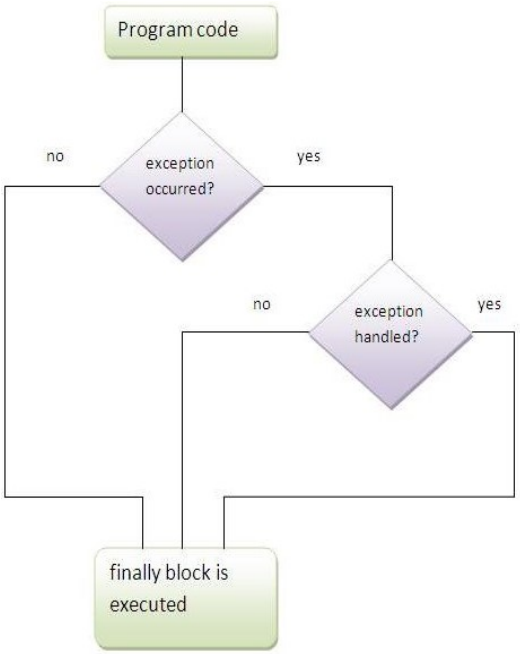
```
catch(Nama_class_exception ref){}

finally{}
```

Catch dapat dibuat lebih dari satu jika kode program yang dibuat akan menghasilkan banyak exception. Blok catch yang banyak (lebih dari satu) haruslah disusun dari yang paling spesifik hingga yang paling umum. Jika muncul exception tertentu, maka catch yang paling sesuai yang akan digunakan.

Try dapat dibuat nested/bersarang. Nested try bermakna try berada di dalam try yang lain. Nested try dibuat bila diperlukan.

Blok finally akan selalu dieksekusi saat ada exception maupun tidak ada. Blok finally ditulis sesudah blok try-catch. Blok finally biasanya digunakan untuk eksekusi kode penting seperti tutup connection, tutup stream. Ilustrasi blok finally dapat dilihat pada gambar berikut ini.



Materi 3 : Keyword throw dan throws

Kata kunci throw digunakan untuk melempar exception secara eksplisit. Kata kunci throw bisa digunakan untuk tipe exception checked maupun unchecked. Kata kunci throw sering dipakai untuk custom exception (exception buatan sendiri).

Kata kunci throws menginformasikan kepada programmer telah terjadi exception dan diharapkan untuk segera membuat exception handling.

Kata kunci throw dan throws dapat muncul bersamaan. Pada penggunaan throw, jika exception yang dilempar merupakan checked exception , maka pada method haruslah ditambah dengan throws.

LATIHAN 1

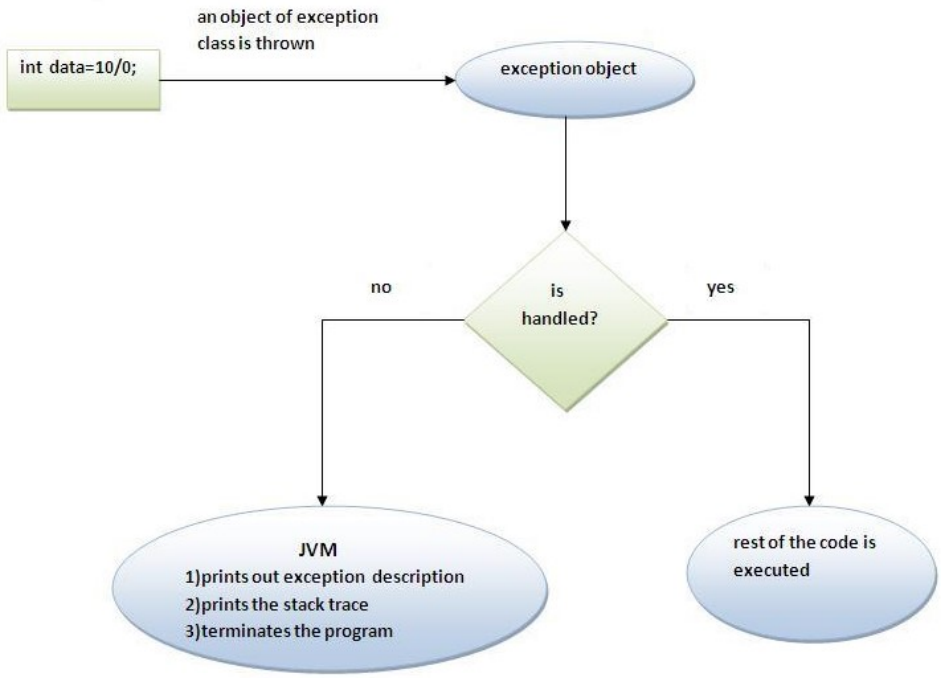
Ketik dan jalankan coding di bawah:

```
public class Testtrycatch2{
      public static void main(String args[]){
            try{
                  int data=10/0;
            }catch(ArithmeticException e){System.out.println(e);}

            System.out.println("rest of the code...");
      }
}
```

Apakah hasil outputnya ketika coding di atas dijalankan?

Ilustrasinya:



LATIHAN 2

Ketik dan jalankan coding di bawah:

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

```
                int a[]=new int[5];
                a[5]=30/0;
            }
            catch(ArithmeticException e){
                   System.out.println("task1 is completed");
            }
            catch(ArrayIndexOutOfBoundsException e){
                   System.out.println("task 2 completed");
            }
            catch(Exception e){
                   System.out.println("common task completed");
            }

            System.out.println("rest of the code...");
      }
}
```

Apakah hasil outputnya ketika coding di atas dijalankan? Coba anda tukar posisi catch. Apakah output yang dihasilkan berbeda?

LATIHAN 3

Ketik dan jalankan coding di bawah:

```
public class TestFinallyBlock2{
     public static void main(String args[]){
          try{
               int data=25/0;
               System.out.println(data);
          }
          catch(ArithmeticException e){
               System.out.println(e);
          }
          finally{
               System.out.println("finally    block    is    always
executed");
          }
          System.out.println("rest of the code...");
  }
}
```

Apakah hasil outputnya ketika coding di atas dijalankan? Pahami output yang terhasil.

Coba ubah 25/0 menjadi 25/5. Kompil dan jalankan program. Apa perbedaannya.

SOAL-SOAL

1. .