Java Lab Pertemuan ke-10

Exception Handling

Merupakan salah satu mekanise untuk mencegah berhentinya sebuah program yang dikarenakan terjadi error pada saat runtime. Pada praktikum kali ini, akan diberikan contoh program untuk menangani exception menggunakan `try, catch, finally, throw, throws`.

1. `try-catch`

   Lihat contoh program pada https://github.com/harkespan/pbo/blob/master/eksepsi/Eksepsi1.java Program tersebut akan menangani error yang diakibatkan karena index array melebihi yang ditentukan.

   Ingat: jika *n* adalah panjang array, maka index maksimumnya adalah *n-1*. Hal ini dikarenakan pada Java, index array dimulai dari 0.

2. *Multi Catch*

   Lihat contoh program pada https://github.com/harkespan/pbo/blob/master/eksepsi/MultiCatch.java

   Program ini menunjukkan bahwa Java mengijinkan penggunaan *multi catch* sebagai mekanisme menangani *multi exception*.

   Latihan:

   Sisipkan potongan kode berikut ke baris 14 dan 15 pada file MultiCatch.java

   ```
   int d = s.nextInt();
   System.out.println(c/d);
   ```

   Tambahkan kode yang dapat dipakai untuk menangkap exception yang akan muncul pada saat MultiCatch.java dieksekusi.

3. Finally

   Kata kunci finally berada setelah try dan atau multi catch. Finally adalah blok yang akan selalu dieksekusi tidak peduli terjadi *exception* atau tidak.

   Lihat contoh program pada https://github.com/harkespan/pbo/blob/master/eksepsi/ExceptionTest.java

   Pada potongan program tersebut terjadi *exception*, di mana index array yang dipanggil (10) melebihi index maksimumnya (9), sehingga blok `try` tidak dijalankan, blok `catch` dijalankan, namun perhatikan pada blok `finally`. Blok tersebut akan dijalankan pada saat *exception* terjadi.

   Latihan:

   Isilah arr[0] - arr[10] dengan sembarang angka. Kemudian jalankan kembali programnya.

4. Membuat *Exception* sendiri

   Developer dapat membuat *exception* sendiri untuk menangani error yang terjadi pada program yang dibuatnya. Untuk dapat membuat *exception*, semua class exception yang dibuat harus subclass dari class `Throwable`. Jika ingin membuat *checked exception* (exception yang dapat diperkirakan terjadi), class harus extends ke class `Exception`. Sedangkan jika ingin membuat *runtime exception*, class yang dibuat harus extends ke class `RuntimeException`. Pada praktikum kali ini, *exception* yang dibuat adalah *checked exception*.

   Perhatikan kode program pada
   https://github.com/harkespan/pbo/blob/master/eksepsi/ExceptionTest.java

```
package eksepsi;
import java.io.*;
public class InsufficientFundsException extends Exception {
      private double amount;
      public InsufficientFundsException(double amount){
            this.amount = amount;
      }
      public double getAmount()
      {
            return amount;
      }
}
```

Class exception yang dibuat adalah `InsufficientFundsException`. Class ini merupakan subclass dari class `Exception`. Pada class ini terdapat 2 method, method paling atas bertindak sebagai constructor yang akan memberi nilai pada variabel amount. Sedangkan method getAmount() akan mengembalikan nilai amount yang ada pada class `InsufficientFundsException`.

Lihat kode program pada
https://github.com/harkespan/pbo/blob/master/eksepsi/CheckingAccount.java secara khusus pada blok berikut.

```
public void tarikUang(double jumlahtarik) throws
InsufficientFundsException
      {
            if(jumlahtarik <= saldo)
            {
                  saldo -= jumlahtarik;
            }
            else
            {
                  double kebutuhan = jumlahtarik - saldo;
                  throw new InsufficientFundsException(kebutuhan);
            }
      }
```

Pada blok tersebut terdapat kata kunci *throws* yang berada di akhir nama method dan parameternya, diikuti dengan memanggil class `InsufficientFundsException`. Kemudian pada blok `else` terdapat pemanggilan *exception* menggunakan kata kunci *throw*. Pemanggilan exception ini akan memberi nilai (hasil pengurangan jumlahtarik – saldo yang disimpan pada variabel kebutuhan) pada variabel amount pada class `InsufficientFundsException`.

Perhatikan kode program pada https://github.com/harkespan/pbo/blob/master/eksepsi/Bank.java kita akan gunakan blok try dan catch untuk proses penarikan uang. Pada baris 11 – 13, program masih akan tetap dijalankan karena tidak terjadi exception. Baris ke-14 terjadi exception karena jumlah uang yang ditarik (600.000) melebihi saldo yang saat ini dimiliki (400.000, berasal dari saldo awal 500.000 dikurangi 100.000, yang dieksekusi pada baris ke-12). Exception kemudian ditangkap dengan memanggil class `InsufficientFundsException`. Sehingga, jika Bank.java dijalankan, akan menghasilkan output sebagai berikut.

```
Menyimpan Rp 500000

Tarik Uang Rp 100000

Tarik Uang Rp 600000
Maaf saldo Anda tidak mencukupi Rp 200000.0
eksepsi.InsufficientFundsException
        at pbogab/eksepsi.CheckingAccount.tarikUang(CheckingAccount.java:25)
        at pbogab/eksepsi.Bank.main(Bank.java:14)
```

Catatan: baris berwarna merah itu bukan error, tetapi hasil pemanggilan method printStackTrace() pada baris ke-19.