IF34348 - PEMROGRAMAN LANJUT

EXCEPTION

09

Oleh : Andri Heryandi, M.T.

MATERI HARI INI

- Exception
- Penanganan Exception
  - Penanganan Exception satu catch
  - Penanganan Exception dengan banyak catch
  - Pengaganan Exception dengan finally
- Membuat exception sendiri

EXCEPTION

- Exception adalah suatu kejadian/event yang terjadi ketika eksekusi program yang akan mengganggu alur program.
- Dengan exception, alur program dapat diatur agar melakukan suatu proses ketika sebuah error terjadi.
- Contoh kejadian :
  - Pembagian dengan angka 0
  - File tidak ditemukan
  - Koneksi jaringan terputus ketika membaca data

EXCEPTION

- Contoh exception:

```
import java.util.Scanner;
public class ContohException1 {
  public static void main(String[] args) {
        Scanner kb=new Scanner(System.in);
        int b1,b2,hasil;
        System.out.print("Bilangan 1 : ");b1=kb.nextInt();
        System.out.print("Bilangan 2 : ");b2=kb.nextInt();
        hasil=b1/b2;
        kb.close();// tutup scanner
        System.out.println("Hasil : "+hasil);
        System.out.println("Program ditutup normal.");
  }
}
```

EXCEPTION

- Contoh exception:

```
Hasil RUN (Normal):
-----------------------
Bilangan 1 : 25
Bilangan 2 : 4
Hasil : 6
Program ditutup normal.
```

```
Hasil RUN (tidak normal):
-----------------------
Bilangan 1 : 25
Bilangan 2 : 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ContohException1.main(ContohException1.java:8)
```

Nama Class Exception yang terjadi

Baris perintah setelah lokasi terjadinya kesalahan (menuliskan hasil dan "program ditutup normal") tidak tereksekusi karena program di-*terminate*.

PENANGANAN EXCEPTION

- Penanganan exception dapat dilakukan dengan menggunakan
  - Block try catch
  - Block try catch finally

Sintak

try catch

```
try{
    baris penyebab exception;
}catch(ClassException1 e1){
        penanganan exception 1;
}catch(ClassException2 e2){
    penanganan exception 2;
}
```

Try catch finally

```
try{
    baris penyebab exception;
}catch(ClassException1 e1){
        penanganan exception 1;
}catch(ClassException2 e2){
    penanganan exception 2;
}finally{
        statement-statement;
}
```

PENANGANAN EXCEPTION

- Contoh penanganan exception:

```
import java.util.Scanner;
public class ContohException2 {
  public static void main(String[] args) {
        Scanner kb=new Scanner(System.in);
        int b1,b2,hasil;
        try{
           System.out.print("Bilangan 1 : ");b1=kb.nextInt();
           System.out.print("Bilangan 2 : ");b2=kb.nextInt();
           hasil=b1/b2;
           kb.close();// tutup scanner
           System.out.println("Hasil : "+hasil);
        }
        catch(ArithmeticException e){
           System.out.println("Terjadi Error Pembagian dengan 0.");
        }
        System.out.println("Program ditutup normal.");
```

PENANGANAN EXCEPTION

- Contoh penanganan exception:

```
import java.util.Scanner;
public class ContohException2 {
  public static void main(String[] args) {
        Scanner kb=new Scanner(System.in);
        int b1,b2,hasil;
        try{
           System.out.print("Bilangan 1 : ");b1=kb.nextInt();
           System.out.print("Bilangan 2 : ");b2=kb.nextInt();
           hasil=b1/b2;
           kb.close();// tutup scanner
           System.out.println("Hasil : "+hasil);
        }
        catch(ArithmeticException e){
           System.out.println("Terjadi Error Pembagian dengan 0.");
        }
        System.out.println("Program ditutup normal.");
```

PENANGANAN EXCEPTION

- Contoh penanganan exception:

```
Hasil RUN (Normal):
-----------------------
Bilangan 1 : 25
Bilangan 2 : 4
Hasil : 6
Program ditutup normal.
```

```
Hasil RUN (tidak normal):
-----------------------
Bilangan 1 : 25
Bilangan 2 : 0
Terjadi Error Pembagian dengan 0.
Program ditutup normal.
```

Baris perintah setelah lokasi terjadinya kesalahan (menuliskan hasil dan "program ditutup normal") TETAP tereksekusi walau pun terjadi exception.

PENANGANAN EXCEPTION BANYAK CATCH

- Jika sebuah/sekumpulan statement memiliki banyak kemungkinan exception, maka untuk penanganan tiap exception bisa menggunakan block try yang memiliki banyak catch.
- Contoh Kasus :
  - Pada contoh program sebelumnya (class ContohException2) ada baris kb.nextInt() yang berguna untuk membaca sebuah integer dari keyboard. Bagaimana kalau yang diinputkan bukan integer, tapi string?

```
Hasil RUN (tidak normal):
-------------------------
Bilangan 1 : 25
Bilangan 2 : angka
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Unknown Source)
        at java.util.Scanner.next(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at ContohException3.main(ContohException3.java:9)
```

Nama Class Exception yang terjadi

PENANGANAN EXCEPTION BANYAK CATCH

- Solusi :
  - Berarti ada 2 exception yang harus dihandle dalam program tersebut yaitu ArithmeticException dan InputMismatchException

Bagaimana saya tahu exception yang dilempar oleh sebuah Method?

Lihat deskripsi method tersebut di javadoc

```
public int nextInt()

Scans the next token of the input as an int.
An invocation of this method of the form nextInt() behaves in e
Returns:
        the int scanned from the input
Throws:
        InputMismatchException - if the next token does not m
        NoSuchElementException - if input is exhausted
        IllegalStateException - if this scanner is closed
```

PENANGANAN EXCEPTION BANYAK CATCH

- Contoh penanganan exception:

```
import java.util.InputMismatchException;
import java.util.Scanner;
public class ContohException3 {
        public static void main(String[] args) {
                Scanner kb=new Scanner(System.in);
                int b1,b2,hasil;
                try{
                        System.out.print("Bilangan 1 : ");b1=kb.nextInt();
                        System.out.print("Bilangan 2 : ");b2=kb.nextInt();
                        hasil=b1/b2;
                        kb.close();// tutup scanner
                        System.out.println("Hasil : "+hasil);
                }
                catch(ArithmeticException e){
                        System.out.println("Terjadi Error Pembagian dengan 0.");
                }
                catch(InputMismatchException e){
                        System.out.println("Data yang anda masukan bukan angka");
                }
                System.out.println("Program ditutup normal.");
        }
```

PENANGANAN EXCEPTION BANYAK CATCH


```
Hasil RUN (tidak normal):
-------------------------
Bilangan 1 : 25
Bilangan 2 : angka
Data yang anda masukan bukan angka
Program ditutup normal.
```

PENGANGANAN EXCEPTION DENGAN TRY, CATCH, FINALLY

- Jika sebuah exception terjadi, maka baris-baris di blok try setelah baris terjadinya exception pasti tidak akan tereksekusi karena alur program akan pindah ke blok catch. Jika baris-baris dibawah lokasi exception ingin tetap dijalankan ketika terjadi exception atau pun tidak terjadi exception, maka tulislah perintah-perintah tersebut di bagian finally.
- Finally ditulis di bawah catch.
- Blok finally PASTI dieksekusi walau pun tidak terjadi exception.
- Blok finally biasanya digunakan untuk melepas resource (tutup file, tutup koneksi ke jaringan, free memory dll).

PENGANGANAN EXCEPTION DENGAN TRY, CATCH, FINALLY


```
import java.util.InputMismatchException;
import java.util.Scanner;
public class ContohException4 {
        public static void main(String[] args) {
                Scanner kb=new Scanner(System.in);
                int b1,b2,hasil;
                try{
                        System.out.print("Bilangan 1 : ");b1=kb.nextInt();
                        System.out.print("Bilangan 2 : ");b2=kb.nextInt();
                        hasil=b1/b2;
                        System.out.println("Hasil : "+hasil);
                }
                catch(ArithmeticException e){
                        System.out.println("Terjadi Error Pembagian dengan 0.");
                }
                catch(InputMismatchException e){
                        System.out.println("Data yang anda masukan bukan angka");
                }
                finally{
                        System.out.println("Tutup Scanner");
                        kb.close();// tutup scanner
                }
                System.out.println("Program ditutup normal.");
        }
```

PENGANGANAN EXCEPTION DENGAN TRY, CATCH, FINALLY


```
Hasil RUN (normal):
-----------------------
Bilangan 1 : 28
Bilangan 2 : 7
Hasil : 4
Tutup Scanner
Program ditutup normal.
```

```
Hasil RUN (tidak normal):
-----------------------
Bilangan 1 : 25
Bilangan 2 : angka
Data yang anda masukan bukan angka
Tutup Scanner
Program ditutup normal.
```

"Tutup Scanner" akan dieksekusi baik ketika terjadi exception atau tidak

MEMBUAT EXCEPTION SENDIRI

- Tidak semua exception yang telah didefinisikan oleh Java mendukung semua kebutuhan kita.
- Contoh Kasus :
  - Untuk nilai suatu mata kuliah, nilai hanya boleh 0 sampai 100.
  - Java tidak memiliki exception untuk menangani exception tersebut.
- Solusi : Buatlah exception sendiri
- Cara membuat exception sendiri :
  - Buat sebuah class turunan dari Exception atau RuntimeException
    - Exception yang menggunakan super class Exception disebut sebagai checked exception. Jika suatu method melemparkan exception jenis ini, maka pemanggilan method ini harus dalam blok try-catch.
    - Exception yang menggunakan super class RuntimeException disebut sebagai unchecked exception. Jika suatu method melemparkan exception jenis ini, maka pemanggilan method ini tidak harus dalam blok try-catch.

MEMBUAT EXCEPTION SENDIRI

- Class NilaiErrorException

```
public class NilaiErrorException extends Exception {
        public String getMessage(){
                return "Nilai hanya boleh 0 sampai 100";
        }
}
```

Atau

```
public class NilaiErrorException extends RuntimeException {
        public String getMessage(){
                return "Nilai hanya boleh 0 sampai 100";
        }
}
```

THROWING EXCEPTION

- Throwing Exception adalah suatu operasi melemparkan sebuah exception ketika sebuah exception terjadi.
- Perintah untuk melempar sebuah exception adalah throw.
- Untuk mendefinisikan sebuah method bisa melemparkan exception tertentu, maka gunakan keyword throws di pendefinisian method, diikuti dengan class exceptionnya.

THROWING EXCEPTION


```
public class Nilai {
        private double nilai;
        public void setNilai(double n) throws NilaiErrorException{
                if((n<0)||(n>100))
                        throw new NilaiErrorException();
                else
                        nilai=n;
        }
        public double getNilai(){
                return nilai;
        }
        public static void main(String[] args){
                Nilai n1;
                n1=new Nilai();
                try{
                        n1.setNilai(50);
                        System.out.println("Nilai 1 : "+n1.getNilai());
                        n1.setNilai(150);
                        System.out.println("Nilai 2 : "+n1.getNilai());

                }catch(NilaiErrorException e){
                        System.out.println(e.getMessage());
                }
```