Dasar-Dasar Pemrograman 2

Lab 08

Exception Handling

I. Exception Handling

Exception Handling adalah event yang terjadi ketika program menemui kesalahan pada saat instruksi program dijalankan. Banyak hal yang dapat menimbulkan event ini; misalnya crash, hard disk rusak dengan tiba-tiba, sehingga program-program tidak bisa mengakses file-file tertentu. Programmer pun dapat menimbulkan event ini, misalnya dengan melakukan pembagian dengan bilangan nol, atau pengisian elemen array melebihi jumlah elemen array yang dialokasikan dan sebagainya.

Try, Catch, Finally

Catching merupakan salah satu cara yang dapat dilakukan untuk menangani exception yang muncul. Untuk melakukan catching exceptions, kita akan mengenal dengan yang namanya **try block, catch block, dan finally block**:

- **Try block** berisi potongan program yang dapat memunculkan exception.
- **Catch block** berisi perintah-perintah yang dilakukan ketika menangani exception yang muncul pada try block. Catch dapat ditulis berkali-kali dengan menangkap berbagai jenis exception yang berbeda-beda.
- **Finally block** akan selalu tereksekusi ketika keluar dari try block. Hal ini memastikan bahwa perintah-perintah yang berada di dalam finally block akan selalu tereksekusi meskipun nanti ternyata muncul exception yang tidak terduga. Kegunaan finally lebih dari hanya handling exception. Finally bisa memungkinkan programmer menghindari kode cleanup terlewati karena return, continue, atau break. Meletakkan kode cleanup di dalam finally block merupakan praktik yang bagus meskipun tidak akan muncul exception

Contoh program try dan catch:

```java
class Example1 {
    Run | Debug
    public static void main(String[] args) {
        int num1, num2;
        // Try block
        try {
            num1 = 0;
            num2 = 17 / num1;
            System.out.println(num2);
            System.out.println("=== AKHIR DARI TRY BLOCK ===");
        }
        // Catch block
        catch (ArithmeticException e) {
            // Block ini hanya tereksekusi jika ArithmetixException muncul
            System.out.println("Tidak bisa membagi angka dengan 0.");
        }
        catch (Exception e) {
            /**
            * Blok ini menangkap exception umum.
            * Dieksekusi ketika ada exception yang belum ter-handle
            * di catch sebelum-sebelumnya.
            */
            System.out.println("Terjadi exception.");
        }
        System.out.println("=== AKHIR DARI TRY-CATCH BLOCKS ===");
    }
}
```

Contoh program try, catch, dan finally:

```java
class Example2 {
    Run | Debug
    public static void main(String[] args) {

        int[] arr = new int[4];

        // Try block
        try {
            int i = arr[4];

            // Baris kode ini tidak akan pernah dijalankan
            // karena terjadi exception di baris kode atas.
            System.out.println("=== AKHIR DARI TRY BLOCK ===");
        }
        // Catch block
        catch (ArrayIndexOutOfBoundsException ex) {
            System.out.println("Index berada di luar batas array.");
        }
        // Finally block
        finally {
            System.out.println("=== FINALLY BLOCK ===");
        }
    }
}
```

Throwing Exceptions

Throw

Throw dalam Java digunakan untuk melemparkan exception secara eksplisit pada method atau suatu block kode. Keyword throw biasanya digunakan untuk melemparkan exception buatan sendiri (custom exception).

Contoh program throw:

```
class Example3 {
    static void throwEx() {
        try {
            throw new ArithmeticException("Pembagian oleh angka nol.");
        }
        catch (ArithmeticException e) {
            System.out.println("Caught in throwEx().");

            // rethrowing the exception
            throw e;
        }
    }

    Run | Debug
    public static void main(String args[]) {
        try {
            throwEx();
        }
        catch (NullPointerException e) {
            System.out.println("Caught in main.");
        }
    }
}
```

Pada contoh di atas, blok kode yang mengandung throw tersebut akan menghasilkan sebuah exception bernama ArithmeticException. Exception tersebut kemudian dapat di-handle oleh yang menggunakan method yang mengandung potongan program tersebut dengan melakukan try-catch yang telah dijelaskan pada sub-materi Try, Catch, Finally.

Instance class yang bisa dilemparkan oleh throw haruslah merupakan **subclass dari class Throwable**. Biasanya, class exception buatan sendiri dibuat dengan **meng-extend class Exception atau RuntimeException** karena kedua class tersebut sudah merupakan subclass dari Throwable.

Throws

Throws merupakan salah satu cara untuk meng-handle kemunculan exception selain menggunakan try-catch. Dengan menggunakan throws, exception yang muncul pada suatu blok kode dalam suatu method akan dilemparkan lagi menuju kode yang memanggil method tersebut. Dengan demikian, biarkan yang menggunakan method tersebut yang melakukan handle lebih lanjut dari exception yang muncul.

Contoh program throws:

```
class Example4 {
    static int divide(int x, int y) throws ArithmeticException {

        // Exception jika y bernilai 0
        return x / y;
    }

    Run | Debug
    public static void main(String[] args) {
        // akan muncul exception seakan-akan dihasilkan oleh method divide
        int x = divide(8, 0);
    }
}
```

Pada contoh di atas, exception akan dilemparkan dari hasil pembagian ketika pembagi y bernilai 0. Kemudian kita melakukan handle dengan cara melemparkan lagi exception tersebut keluar method kepada kode yang melakukan pemanggilan method divide (saat ini kita tidak perlu memikirkan dulu bagaimana nanti exception tersebut di-handle oleh kode yang memanggil method divide tersebut). Dengan demikian, seakan-akan method divide-lah yang menghasilkan exception tersebut.

Custom Exception

Custom Exception merupakan exception yang kita buat sendiri. Misalnya kita ingin memastikan bahwa input umur oleh user tidak boleh kurang dari 0. Kita ingin kode kita mengeluarkan sebuah exception yang bernama InvalidAgeException. Dengan prinsip inheritance, kita dapat membuat sebuah class yang meng-extend class Exception (class Exception secara default sudah ada pada Java tanpa harus dibuat lagi).

Contoh custom exception:

```
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}
```

Pada kode di atas dapat dilihat bahwa constructor yang dibuat harus menerima parameter string yang merupakan error message yang akan ditampilkan saat exception terjadi. Sama dengan exception pada umumnya, kita juga dapat melakukan throw pada custom exception.

Contoh penggunaan custom exception:

```
if (age < 0) {
    throw new InvalidAgeException("Umur tidak boleh negatif.");
}
```

Soal Lab 08

Dogegochi

Source: https://www.wartaekonomi.co.id/read340405/lagi-elon-musk-pompom-dogecoin-harganya-to-the-moon-lagi

Setelah kamu keluar dari Pacilgeon, kamu melanjutkan hari-hari normal kamu berkuliah di Fasilkom UI. Namun, kamu mulai kelelahan dan jenuh sehingga kamu butuh suatu hiburan. Akhirnya, Angewomon yang mendengar isi hati kamu datang dan memberikan kamu sebuah mainan seperti tamagotchi yang bernama Dogegochi.

Dogegochi merupakan mainan virtual pet di mana kamu dapat memelihara doge. Kamu bisa memberi makan dan bermain dengan doge sehingga kamu dapat terhibur saat memainkannya. Doge yang kamu pelihara merupakan anjing yang sangat malas karena hanya bisa makan dan main saja. Doge juga merupakan anjing yang cepat bosan dalam melakukan suatu hal sehingga dia tidak mau melakukan aktivitas yang sama 3 kali berturut-turut. Karena kamu sangat menyukai dogegochi, kamu ingin membuat salinan dogegochi versi kamu sendiri untuk diberikan kepada teman-teman kamu agar mereka juga bisa menghibur diri mereka sendiri. Di waktu yang bersamaan, kamu sudah belajar mengenai **Exception Handling** sehingga kamu akan mengaplikasikan hal tersebut dalam pembuatan salinan dogegochi milik kamu.

Spesifikasi Program:

- **Simulator:**
    - **Aktivitas:**
        - Main: kamu bermain dengan doge.

          Command untuk melakukan aktivitas ini: 1

          Output jika berhasil: “[SUCCESS : Dogenya bermain dengan senang hati]”

        - Makan: kamu memberi makan doge. Saat melakukan aktivitas makan, program akan meminta input lagi berupa: <makanan> <jumlah makanan>. Tidak perlu meng-handle jika jumlah makanan yang dimasukkan kurang dari 1. Input makanan dipastikan hanya berupa 1 kata.

          Command untuk melakukan aktivitas ini: 2

          Output jika berhasil: “[SUCCESS : Dogenya makan <makanan> sebanyak <jumlah makanan> dengan gembira]”

        - Selesai bermain: kamu selesai bermain dengan doge, program akan berhenti.

          Command untuk melakukan aktivitas ini: 3

          Output: “[Bye-bye doge!!]”

    - **Keterangan:**
        - Input untuk menjalankan aktivitas dipastikan antara 1, 2, atau 3

- **Dogegochi**
    - **Constructor:**
        - Constructor tidak memiliki parameter. Saat objek dogegochi diinstansiasi, atribut energy dari objek dogegochi akan bernilai 50.

- **Method:**
  Berikut adalah macam-macam aktivitas yang dapat dilakukan doge:
  - eat(String food, int quantity): kamu memberikan makanan berupa <food> sebanyak <quantity> untuk dimakan oleh doge. Setelah doge makan, energi doge akan bertambah sebanyak 2 kali lipat dari <quantity>.

    Output: “[SUCCESS : Dogenya makan <food> sebanyak <quantity> dengan gembira]”

  - play( ): kamu bermain dengan doge. Jika doge bermain, energinya akan berkurang sebanyak 20.

    Output: “[SUCCESS: Dogenya bermain dengan senang hati]”

- **Keterangan:**
  - Energi doge tidak bisa lebih kecil dari 0 atau lebih besar dari 100.
  - Saat melakukan penambahan energi yang belum bernilai 100, jika energi menjadi lebih besar dari 100, maka energi tersebut akan tetap 100.
  - Saat melakukan pengurangan energi yang belum bernilai 0, jika nilainya menjadi lebih rendah dari 0, maka energi tersebut akan tetap 0.
  - Jika tidak ada kasus exception yang ditemukan, maka setelah melakukan sebuah aktivitas, print output sesuai dengan penjelasan di atas.

- **Kasus-Kasus Exception Handling:**
  - **Built-in Exception:**
    - **ArrayIndexOutOfBoundsException:** jika input untuk aktivitas makan tidak lengkap, yaitu hanya makanan saja tanpa jumlahnya. Jika kasus ini terjadi, doge tidak melakukan aktivitas makan.

      Error message: “[FAILED: Kamu tidak memberikan jumlah makanan yang ingin diberikan]”

        - **NumberFormatException:** jika input jumlah makanan bukan berupa angka. Jika kasus ini terjadi, doge tidak melakukan aktivitas makan.

          Error message: “[FAILED: Jumlah makanan harus berupa angka]”

    - **Custom Exception:**
        - **FullException:** jika energi doge sudah penuh (bernilai 100) namun masih tetap diberi makan. Jika kasus ini terjadi, doge tidak melakukan aktivitas makan.

          Error message: “[FAILED: Dogenya sudah buncit tidak kuat makan lagi :(]“

        - **HungerException:** terjadi ketika energi doge sudah habis (bernilai 0) namun masih tetap melakukan aktivitas lain selain makan. Jika kasus ini terjadi, aktivitas yang ingin dilakukan tidak jadi dilakukan.

          Error message: “[FAILED: Dogenya lemes butuh makan :(]“

        - **BoredException:** terjadi jika doge melakukan 3 aktivitas yang sama berturut-turut (Terjadi ketika doge sudah melakukan 2 aktivitas yang sama namun akan melakukan aktivitas yang sama lagi untuk ke-3 kalinya). Jika kasus ini terjadi, aktivitas yang ingin dilakukan tidak jadi dilakukan.

          Error message: “[FAILED: Dogenya bosan nih, gak mau melakukan hal lain saja?]”

Anda wajib membuat class Exception yang sesuai dan pastikan jika menemukan kasus-kasus di atas, kamu print error message yang sesuai dengan penjelasan di atas. Ubahlah code Simulator.java dan Dogegochi.java agar dapat memanfaatkan class Exception yang kamu buat.

Prioritas exception jika terjadi kondisi yang melanggar lebih dari satu exception :

1. ArrayIndexOutOfBoundsException
2. NumberFormatException

3. BoredException
4. FullException
5. HungerException

Test Case

Test case 1:

```
Selamat Datang di Dogegochi!
Perintah yang dapat dijalankan Dogegochi :
1. Main
2. Makan
3. Selesai Bermain

Status Doge:
Energy = 50
Masukkan Perintah: 1
[SUCCESS : Dogenya bermain dengan senang hati]

Status Doge:
Energy = 30
Masukkan Perintah: 1
[SUCCESS : Dogenya bermain dengan senang hati]

Status Doge:
Energy = 10
Masukkan Perintah: 2
Masukkan Makanan dan jumlahnya: wagyu a5
[FAILED: Jumlah makanan harus berupa angka]

Status Doge:
Energy = 10
Masukkan Perintah: 2
Masukkan Makanan dan jumlahnya: wagyu sapi
[FAILED: Jumlah makanan harus berupa angka]

Status Doge:
Energy = 10
Masukkan Perintah: 2
Masukkan Makanan dan jumlahnya: wagyu 1
[SUCCESS : Dogenya makan wagyu sebanyak 1 dengan gembira]

Status Doge:
Energy = 12
Masukkan Perintah: 1
[SUCCESS : Dogenya bermain dengan senang hati]
```

```
Status Doge:
Energy = 0
Masukkan Perintah: 1
[FAILED : Dogenya lemes butuh makan :(]

Status Doge:
Energy = 0
Masukkan Perintah: 3
[Bye-bye doge!!]
```

Test case 2:

```
Selamat Datang di Dogegochi!
Perintah yang dapat dijalankan Dogegochi :
1. Main
2. Makan
3. Selesai Bermain

Status Doge:
Energy = 50
Masukkan Perintah: 2
Masukkan Makanan dan jumlahnya: mieghetti 25
[SUCCESS : Dogenya makan mieghetti sebanyak 25 dengan gembira]

Status Doge:
Energy = 100
Masukkan Perintah: 2
Masukkan Makanan dan jumlahnya: buncis 1
[FAILED : Dogenya sudah buncit tidak kuat makan lagi :(]

Status Doge:
Energy = 100
Masukkan Perintah: 2
Masukkan Makanan dan jumlahnya: okonomiyaki 1
[FAILED : Dogenya sudah buncit tidak kuat makan lagi :(]

Status Doge:
Energy = 100
Masukkan Perintah: 1
[SUCCESS : Dogenya bermain dengan senang hati]

Status Doge:
Energy = 80
Masukkan Perintah: 1
[SUCCESS : Dogenya bermain dengan senang hati]

Status Doge:
Energy = 60
Masukkan Perintah: 1
[FAILED : Dogenya bosan nih, gak mau melakukan hal lain saja?]
```

```
Status Doge:
Energy = 60
Masukkan Perintah: 2
Masukkan Makanan dan jumlahnya: samyang
[FAILED: Kamu tidak memberikan jumlah makanan yang ingin diberikan]

Status Doge:
Energy = 60
Masukkan Perintah: 3
[Bye-bye doge!!]
```

Komponen Penilaian

- 45% Implementasi custom class Exception untuk kasus-kasus yang ada
- 25% Throw Exception pada Dogegochi.java
- 20% Exception handling pada Simulator.java
- 10% Dokumentasi dan kerapian kode (may be changed if desired)

Komponen Penilaian

- Revisi 1 (highlight berwarna kuning): 10 Mei 2021 18.30 WIB
    - Memperjelas bahwa jika Built-in Exception Java terjadi, tidak dihitung sebagai aktivitas
- Revisi 2 (highlight berwarna hijau): 10 Mei 2021 19.45 WIB
    - Mengganti error message pada NumberFormatException
    - Memperjelas urutan error handling
- Revisi 3 (highlight berwarna merah): 10 Mei 2021 21.20 WIB
    - Memperjelas penjelasan pada HungerException

Kumpulkan berkas .java yang telah di-zip dengan format penamaan seperti berikut.
`Lab08_[Kelas]_[KodeAsdos]_[NPM]_[NamaLengkap].zip`
Contoh:
`Lab08_A_LN_1234567890_DekDepe.zip`