

## LAPORAN PRAKTIKUM MINGGU 9

### Exception Handling, Custom Exception, dan Penerapan Design Pattern

**NAMA** : ALVIRA LIBRA RAMADHANI

**NIM** : 240202851

**KELAS** : 3IKRA

#### 1. TUJUAN

1. Mahasiswa mampu menjelaskan perbedaan antara *error* dan *exception*.
2. Mahasiswa mampu mengimplementasikan struktur *try-catch-finally* untuk menangani kondisi tidak normal pada program.
3. Mahasiswa dapat membuat dan menggunakan *custom exception* untuk validasi spesifik pada sistem keranjang belanja Agri-POS.
4. Mahasiswa memahami dasar penerapan *design pattern* seperti Singleton dan MVC dalam pengembangan aplikasi.

#### 2. DASAR TEORI

1. **Error vs Exception:** *Error* adalah kondisi fatal yang tidak dapat ditangani program (seperti *OutOfMemoryError*), sedangkan *Exception* adalah kondisi tidak normal yang masih dapat diantisipasi dan ditangani.
2. **Try-Catch-Finally:** Blok *try* berisi kode berisiko, *catch* menangani kesalahan yang muncul, dan *finally* berisi kode yang akan selalu dijalankan apa pun hasilnya.
3. **Custom Exception:** Kelas pengecualian buatan sendiri yang mewarisi class *Exception* untuk memberikan informasi kesalahan yang lebih spesifik bagi logika bisnis aplikasi.
4. **Singleton Pattern:** Pola desain yang memastikan sebuah *class* hanya memiliki satu *instance* global di seluruh aplikasi.

#### 3. LANGKAH PRAKTIKUM

1. Membuat tiga jenis *custom exception*: *InvalidQuantityException*, *ProductNotFoundException*, dan *InsufficientStockException*.
2. Memperbarui model *Product* agar memiliki atribut stok dan metode *reduceStock*.
3. Mengimplementasikan logika validasi pada class *ShoppingCart* menggunakan kata kunci *throws* dan *throw*.
4. Menyusun skenario pengujian pada *MainExceptionDemo* untuk memicu kesalahan input jumlah negatif, penghapusan produk yang tidak ada, dan stok tidak mencukupi.
5. Melakukan pengujian pola desain Singleton pada *ProductService*.

#### 4. CODE PROGRAM

##### a) EmptyStockException.java

```
praktikum > week9-exception-handling > src > main > java > com > upb > agripos  
1 package com.upb.agripos;  
2  
3 public class EmptyStockException extends Exception {  
4  
5     public EmptyStockException(String message) {  
6         super(message);  
7     }  
8 }
```

##### b) InsufficientStockException.java

```
praktikum > week9-exception-handling > src > main > java > com > upb > agripos > J Insuffic  
1 package com.upb.agripos;  
2  
3 public class InsufficientStockException extends Exception {  
4     public InsufficientStockException(String msg) { super(msg); }  
5 }
```

##### c) InvalidPriceException

```
praktikum > week9-exception-handling > src > main > java > com > upb > agripos > J Inv  
1 package com.upb.agripos;  
2  
3 public class InvalidPriceException extends Exception {  
4  
5     public InvalidPriceException(String message) {  
6         super(message);  
7     }  
8 }
```

##### d) InvalidQuantityException.java

```
praktikum > week9-exception-handling > src > main > java > com > upb > agripos > J InvalidQ  
1 package com.upb.agripos;  
2  
3 public class InvalidQuantityException extends Exception {  
4     public InvalidQuantityException(String msg) { super(msg); }  
5 }
```

##### e) MainExceptionDemo.java

```
praktikum > week9-exception-handling > src > main > java > com > upb > agripos > J MainExceptionDemo.java > Language Sup  
1 package com.upb.agripos;  
2  
3 public class MainExceptionDemo {  
4     Run main | Debug main | Run | Debug  
5     public static void main(String[] args) {  
6         System.out.println("Hello, Alvira-240202851 (Week9)");  
7  
8         ShoppingCart cart = new ShoppingCart();  
9         Product p1 = new Product(code: "P01", name: "Pupuk Organik", price: 25000, stock: 3);  
10        // contoh penambahan stok  
11        try {  
12            // menambahkan produk dengan stok negatif  
13            cart.addProduct(p1, -1);  
14        } catch (InvalidQuantityException e) {  
15            // menampilkan informasi quantity tidak valid  
16            System.out.println("Kesalahan: " + e.getMessage());  
17        }  
18        // contoh pengurangan stok  
19        try {  
20            cart.removeProduct(p1);  
21        } catch (ProductNotFoundException e) {  
22            // menampilkan informasi produk tidak ditemukan  
23            System.out.println("Kesalahan: " + e.getMessage());  
24        }  
25    }  
26 }
```

```

24         // jika stok kurang dari 5 maka tidak bisa checkout
25         try {
26             cart.addProduct(p1, qty: 5);
27             cart.checkout();
28         } catch (Exception e) {
29             System.out.println("Kesalahan: " + e.getMessage());
30         }
31         // insufficient stock
32         try {
33             cart.checkout();
34         } catch (InsufficientStockException e) {
35             System.out.println("Kesalahan: " + e.getMessage());
36         }
37     }
38 }

```

f) Product.java

```

praktikum > week9-exception-handling > src > main > java > com > upb > agripis > J Product.java >
1  package com.upb.agripis;
2
3  public class Product {
4      private final String code;
5      private final String name;
6      private final double price;
7      private int stock;
8
9      public Product(String code, String name, double price, int stock) {
10         this.code = code;
11         this.name = name;
12         this.price = price;
13         this.stock = stock;
14     }
15
16     public String getCode() { return code; }
17     public String getName() { return name; }
18     public double getPrice() { return price; }
19     public int getStock() { return stock; }
20     public void reduceStock(int qty) { this.stock -= qty; }
21 }

```

g) ProdukNotFoundException.java

```

praktikum > week9-exception-handling > src > main > java > com > upb > agripis > J Prod
1  package com.upb.agripis;
2
3  public class ProductNotFoundException extends Exception {
4      public ProductNotFoundException(String msg) { super(msg); }
5  }

```

# h) ShoppingCart.java

```
> weeks-exception-handling > src > main > java > com > upb > agripos > ShoppingCart.java > Language Support for Java
1  package com.upb.agripos;
2
3  import java.util.HashMap;
4  import java.util.Map;
5
6  public class ShoppingCart {
7      private final Map<Product, Integer> items = new HashMap<>();
8
9      public void addProduct(Product p, int qty) throws InvalidQuantityException {
10         if (qty <= 0) {
11             throw new InvalidQuantityException(msg: "Quantity harus lebih dari 0.");
12         }
13         items.put(p, items.getOrDefault(p, defaultValue: 0) + qty);
14     }
15
16     public void removeProduct(Product p) throws ProductNotFoundException {
17         if (!items.containsKey(p)) {
18             throw new ProductNotFoundException(msg: "Produk tidak ada dalam keranjang.");
19         }
20         items.remove(p);
21     }
22
23     public void checkout() throws InsufficientStockException {
24         for (Map.Entry<Product, Integer> entry : items.entrySet()) {
25             Product product = entry.getKey();
26             int qty = entry.getValue();
27             if (product.getStock() < qty) {
28                 throw new InsufficientStockException(
29                     "Stok tidak cukup untuk: " + product.getName()
30                 );
31             }
32         }
33         // contoh pengurangan stok bila semua cukup
34         for (Map.Entry<Product, Integer> entry : items.entrySet()) {
35             entry.getKey().reduceStock(entry.getValue());
36         }
37     }
38 }
```

## 5. HASIL EKSEKUSI

```
PS C:\Users\LENOVO\oop-202501-240202851> & 'C:\Program Files\IBM\SPSS\Statistics\25\JRE\bin\java.exe' '-cp' 'C:\Users\LENOVO\AppData\Roaming\Code\User\workspaceStorage\43d5f3510c604989e781f371f498d2b7\redhat.java\jdt_ws\oop-202501-240202851_edc4669d\bin' 'com.upb.agripos.MainExceptionDemo'
Hello, Alvira-240202851 (Week9)
Kesalahan: Quantity harus lebih dari 0.
Kesalahan: Produk tidak ada dalam keranjang.
Kesalahan: Stok tidak cukup untuk: Pupuk Organik
Kesalahan: Stok tidak cukup untuk: Pupuk Organik
PS C:\Users\LENOVO\oop-202501-240202851>
```

- Analisis

Analisis hasil eksekusi menunjukkan bahwa program **Agri-POS** telah berhasil menerapkan sistem keamanan data melalui **Exception Handling**. Dengan adanya *custom exception* seperti `InvalidQuantityException` dan `InsufficientStockException`, program tidak langsung berhenti (*crash*) saat pengguna melakukan kesalahan input, melainkan memberikan pesan edukatif yang informatif. Penggunaan blok `try-catch` memisahkan logika utama program dengan logika penanganan kesalahan, sehingga alur aplikasi menjadi lebih kuat (*robust*) dan terorganisir sesuai standar pengembangan aplikasi profesional.

## 6. KESIMPULAN

Penerapan *Exception Handling* dan *Custom Exception* sangat krusial dalam aplikasi Agri-POS untuk memastikan integritas data, terutama pada manajemen stok dan transaksi. Melalui praktikum ini, dapat disimpulkan bahwa penanganan kesalahan yang baik tidak hanya mencegah kegagalan sistem, tetapi juga meningkatkan pengalaman pengguna melalui pesan kesalahan yang jelas dan terkendali.

## 7. QUIZ

### 1. Jelaskan perbedaan error dan exception.

**JAWABAN:**

Error bersifat fatal dan berhubungan dengan kegagalan sistem/hardware (tidak bisa ditangani), sedangkan exception adalah kesalahan logika atau input yang dapat ditangkap dan diperbaiki oleh program.

### 2. Apa fungsi finally dalam blok try-catch-finally?

**JAWABAN:**

Fungsi finally adalah untuk menjalankan kode penting (seperti menutup koneksi atau membersihkan memori) yang harus tetap dieksekusi baik saat terjadi kesalahan maupun saat program berjalan normal.

### 3. Mengapa custom exception diperlukan?

**JAWABAN:**

Agar program dapat memberikan identifikasi kesalahan yang spesifik sesuai kebutuhan bisnis (seperti stok habis), sehingga lebih mudah dipahami oleh pengembang maupun pengguna dibandingkan pengecekan error umum.

### 4. Berikan contoh kasus bisnis dalam POS yang membutuhkan custom exception.

**JAWABAN:**

Contohnya adalah saat kasir memasukkan jumlah barang negatif, saat memproses pembayaran dengan saldo yang tidak mencukupi, atau saat memindai kode barang yang belum terdaftar di database.