Bab 9

Exception Handling dan Keandalan Program

9.1 Konsep Exception dan Try-Catch

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 3.1: Mengimplementasikan exception handling dalam program Java

Konsep Penting

Exception handling adalah mekanisme untuk menangani kondisi tidak terduga selama runtime, memastikan program tetap berjalan stabil [1]. Exception handling membantu program tetap stabil dan memberikan pesan kesalahan yang jelas.

9.1.1 Jenis Exception

- **Checked Exception**: wajib ditangani (contoh: `IOException`)
- **Unchecked Exception**: turunan `RuntimeException`
- **Error**: kesalahan serius yang biasanya tidak ditangani

9.1.2 Struktur Try-Catch

```
public class DemoException {
    public static void main(String[] args) {
        try {
            int hasil = 10 / 0;
            System.out.println(hasil);
```

```
        } catch (ArithmeticException e) {
            System.out.println("Terjadi kesalahan: " + e.
               getMessage());
        }
    }
}
```

Kode Program 9.1: Try-Catch Sederhana

9.2 Custom Exception dan Best Practice

9.2.1 Membuat Custom Exception

Custom exception digunakan untuk mewakili kesalahan yang spesifik pada domain aplikasi.

```
class SaldoTidakCukupException extends Exception {
    public SaldoTidakCukupException(String pesan) {
        super(pesan);
    }
}

class Rekening {
    private double saldo;

    public void tarik(double jumlah) throws
       SaldoTidakCukupException {
        if (jumlah > saldo) {
            throw new SaldoTidakCukupException("Saldo tidak
               mencukupi");
        }
        saldo -= jumlah;
    }
}
```

Kode Program 9.2: Custom Exception

9.2.2 Finally dan Try-with-Resources

Gunakan `finally` untuk memastikan resource ditutup. Di Java modern, gunakan try-with-resources.

Aktivitas Pembelajaran

1. **Eksperimen**: Buat program yang memicu `NumberFormatException` dan tangani dengan try-catch.

2. **Custom Exception**: Buat exception `StokHabisException` pada sistem inventori.

3. **Diskusi**: Kapan sebaiknya melempar exception dan kapan mengembalikan nilai default?

4. **Refactoring**: Ganti penanganan error manual dengan try-with-resources pada contoh file I/O.

Latihan dan Refleksi

1. Jelaskan perbedaan checked dan unchecked exception.

2. Buat program input angka yang aman dari kesalahan format input.

3. Buat custom exception untuk validasi umur minimal 17 tahun.

4. Jelaskan kapan `finally` tetap dieksekusi.

5. **Refleksi**: Bagaimana exception handling membantu kualitas aplikasi Anda?

Asesmen (Evaluasi Kinerja)

Instrumen Penilaian untuk Sub-CPMK 3.1

A. Pilihan Ganda

1. Exception yang wajib ditangani adalah:
   (a) Checked exception
   (b) Runtime exception
   (c) Error
   (d) Logical error

2. Try-with-resources digunakan untuk:
   (a) Menghindari overloading
   (b) Menutup resource secara otomatis
   (c) Menghapus object
   (d) Mempercepat program

B. Tugas Praktik

- Buat class `Login` yang melempar exception jika password salah tiga kali.

Rubrik Penilaian: Lihat Lampiran A

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya memahami konsep exception dan error

☐ Saya dapat menggunakan try-catch dengan benar

☐ Saya mampu membuat custom exception

☐ Saya memahami penggunaan finally dan try-with-resources

☐ Saya dapat menentukan kapan melempar exception

Rangkuman

Exception handling menjaga program tetap stabil saat terjadi kesalahan sesuai spesifikasi Java [2]. Dengan custom exception dan try-with-resources, aplikasi menjadi lebih robust dan mudah dipelihara.

Kata Kunci: ***Exception***, ***try-catch***, ***checked***, ***unchecked***, ***try-with-resources***

Daftar Pustaka

[1] Cay S. Horstmann. *Core Java Volume I–Fundamentals*. Prentice Hall, 11th edition, 2019.

[2] Oracle. The java tutorials. https://docs.oracle.com/javase/tutorial/, 2024.