Pertemuan 2: Exception Handling

Exception Handling di Java merupakan mekanisme untuk menangani kesalahan (error) yang terjadi saat program berjalan (runtime). Penggunaan exception handling bertujuan agar program dapat menangani kesalahan dengan baik, sehingga tidak langsung berhenti ketika terjadi error.

1. Konsep Exception

Exception adalah kondisi tidak normal yang terjadi saat eksekusi program. Exception dapat disebabkan oleh kesalahan seperti pembagian dengan nol, file tidak ditemukan, dan lain-lain.

2. Jenis-Jenis Exception

1. **Checked Exception**: Kesalahan yang terdeteksi pada waktu kompilasi dan harus ditangani oleh programmer. Contoh: `IOException`, `SQLException`.
2. **Unchecked Exception**: Kesalahan yang terjadi pada waktu runtime dan tidak perlu ditangani oleh programmer. Contoh: `ArithmeticException`, `NullPointerException`.
3. **Error**: Masalah yang terjadi di luar kendali program, biasanya terkait dengan resource sistem, misalnya `OutOfMemoryError`.

3. Blok Try-Catch

Untuk menangani exception di Java, kita menggunakan blok **try-catch**. Kode yang mungkin menimbulkan exception ditempatkan dalam blok `try`, dan penanganannya dituliskan dalam blok `catch`.

Sintaks:

```
try {
    // Kode yang dapat menimbulkan exception
} catch (Exception e) {
    // Penanganan exception
}
```

```java
public class ContohException {
    public static void main(String[] args) {
        try {
            int hasil = 10 / 0;  // Akan menimbulkan ArithmeticException
            System.out.println("Hasil: " + hasil);
        } catch (ArithmeticException e) {
            System.out.println("Kesalahan: Pembagian dengan nol tidak diperbolehkan.");
        }
    }
}
```

Output:

Kesalahan: Pembagian dengan nol tidak diperbolehkan.

4. Blok Finally

Blok **finally** selalu dijalankan setelah eksekusi blok `try` dan `catch`, baik exception terjadi atau tidak. Biasanya digunakan untuk membersihkan resource seperti menutup file atau koneksi database.

Sintaks:

```java
try {
    // Kode yang dapat menimbulkan exception
} catch (Exception e) {
    // Penanganan exception
} finally {
    // Kode yang akan selalu dijalankan
}
```

Contoh:

```java
public class ContohFinally {
    public static void main(String[] args) {
        try {
            int[] angka = {1, 2, 3};
            System.out.println(angka[5]);  // Akan menimbulkan ArrayIndexOutOfBoundsEx
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Kesalahan: Indeks array di luar batas.");
        } finally {
            System.out.println("Blok finally selalu dijalankan.");
        }
    }
}
```

Output:

```
Kesalahan: Indeks array di luar batas.
Blok finally selalu dijalankan.
```

5. Throws dan Throw

- **`throws`**: Digunakan pada deklarasi metode untuk menunjukkan bahwa metode tersebut dapat menimbulkan exception.
- **`throw`**: Digunakan untuk secara eksplisit melempar exception.

Contoh `throws`:

```java
import java.io.*;

public class ContohThrows {
    public static void bacaFile(String namaFile) throws IOException {
        FileReader file = new FileReader(namaFile);
        BufferedReader reader = new BufferedReader(file);
        System.out.println(reader.readLine());
        reader.close();
    }

    public static void main(String[] args) {
        try {
            bacaFile("file_tidak_ada.txt");  // Akan menimbulkan FileNotFoundException
        } catch (IOException e) {
            System.out.println("Kesalahan: File tidak ditemukan.");
        }
    }
}
```

Contoh `throw`:

```java
public class ContohThrow {
    public static void periksaUsia(int usia) {
        if (usia < 18) {
            throw new ArithmeticException("Tidak memenuhi syarat untuk mendaftar.");
        } else {
            System.out.println("Pendaftaran berhasil.");
        }
    }

    public static void main(String[] args) {
        try {
            periksaUsia(16);  // Akan menimbulkan ArithmeticException
        } catch (ArithmeticException e) {
            System.out.println("Kesalahan: " + e.getMessage());
        }
    }
}
```

Output:

Kesalahan: Tidak memenuhi syarat untuk mendaftar.

Ringkasan Exception Handling:

1. **Try-Catch** digunakan untuk menangani kesalahan yang mungkin terjadi saat runtime.
2. **Finally** selalu dijalankan, baik exception terjadi atau tidak, biasanya untuk membersihkan resource.
3. **Throws** menunjukkan bahwa sebuah metode dapat menimbulkan exception.
4. **Throw** digunakan untuk melempar exception secara eksplisit.