## Soal Tugas & Evaluasi

Jelaskan apa yang dimaksud dengan **Exception Handling** dan sebutkan tujuannya!

## Jawaban

Exception handling adalah mekanisme untuk menangani salah satu permasalahan yang terjadi Ketika mengeksekusi Program. Dimana pada saat terjadi sebuah kesalahan pada kode program kita, alur jalanya program akan terganggu dan bisa jadi berakhir

Tujuan Utama

* Mengidentifikasi situasi tidak normal
* Memberikan informasi yang berguna
* Menghindari penghetian mendadak
* Menangani kodisi tidak terduga
* Menigkatkan keamanan system

## Source Code

Tulis kode program dikotak ini...

1 kotak dan 1 Penjelasan untuk 1 Class

**Penjelasan**

Tulis Penjelasan disini …

**Output**

|  |
| --- |
| Masukan screenshot output disini |

## Soal Tugas & Evaluasi

Sebutkan kategori **Exception Handling** beserta penjelasannya!

## Jawaban

- Checked Exception adalah masalah yang terjadi di saat proses compile program sedang berlangsung. Kategori exception ini tidak dapat dibiarkan begitu saja, kita harus menangani kesalahan ini jika tidak ingin program kita berhenti di Tengah jalan

- unChecked Exception adalah masalah yang tidak harus di tangani seperti halnya pada Checked Exception. Sederhanannya Exception ini terjadi pada saat program menemukan sebuah Bug seperti kesalahan Logika pemrograman

- Error Bukan termasuk kedalam Exception karena masalah ini bukan di sebabkan oleh user maupun programmer. Melaikan biasanya terjadi karena kondisi yang serius dan tidak dapat di pulihkan

-

## Source Code

Tulis kode program dikotak ini...

1 kotak dan 1 Penjelasan untuk 1 Class

**Penjelasan**

Tulis Penjelasan disini …

**Output**

|  |
| --- |
| Masukan screenshot output disini |

## Soal Tugas & Evaluasi

Buatlah kode program yang menunjukkan implementasi kategori Exception Handling berikut:

-*Checked Exception*: Buat program yang menangani *FileNotFoundException* dan *IOException*.

-*Unchecked Exception*: Buat program yang menunjukkan penanganan *ArithmeticException* dan *NullPointerException*.

## Jawaban

Ketik jawaban disini …

## Source Code

import java.io.\*;

public class CheckedException {

    public static void main(String[] *args*) {

        try (FileReader fileReader = new FileReader("nonexistent\_file.txt");

             BufferedReader bufferedReader = new BufferedReader(fileReader)) {

            String line;

            while ((line = bufferedReader.readLine()) != null) {

                System.out.println(line);

            }

        } catch (FileNotFoundException *e*) {

            System.out.println("Error: File tidak ditemukan.");

        } catch (IOException *e*) {

            System.out.println("Error: Terjadi kesalahan saat membaca file.");

        }

    }

}

**Penjelasan**

Kode ini membaca file menggunakan FileReader dan BufferedReader dalam blok try-with-resources, memastikan sumber daya ditutup otomatis. Jika file tidak ditemukan, FileNotFoundException ditangani dengan pesan "File tidak ditemukan." Jika terjadi kesalahan lain saat membaca file, IOException ditangani dengan pesan "Terjadi kesalahan saat membaca file."

**Output**

|  |
| --- |
|  |

## Source Code

public class UncheckedExceptionExample {

    public static void main(String[] *args*) {

        try {

            int a = 10;

            int b = 0;

            int result = a / b;

        } catch (Exception *e*) {

            System.out.println(e.getMessage());

        }

*// Contoh NullPointerException*

        try {

            String str = null;

            System.out.println(str.length());

        } catch (NullPointerException *e*) {

            System.out.println("Error: Objek bernilai null.");

        }

    }

}

**Penjelasan**

Kode ini menunjukkan contoh penanganan **Unchecked Exception** di Java, seperti ArithmeticException dan NullPointerException. Pada blok pertama, pembagian angka dengan nol menyebabkan ArithmeticException, yang ditangani dengan mencetak pesan error. Pada blok kedua, mencoba mengakses metode pada objek null menghasilkan NullPointerException, dan ditangani dengan pesan "Error: Objek bernilai null."

**Output**

|  |
| --- |
|  |

## Soal Tugas & Evaluasi

Buat sebuah program yang melakukan operasi pembagian angka dengan penanganan menggunakan blok *try-catchfinally*. Program ini harus dapat menangani input yang tidak valid, termasuk pembagian dengan nol.

## Jawaban

Ketik jawaban disini …

## Source Code

import java.util.Scanner;

public class DivisionWithExceptionHandling {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Masukkan angka pertama: ");

int angka1 = Integer.parseInt(scanner.nextLine());

System.out.print("Masukkan angka kedua: ");

int angka2 = Integer.parseInt(scanner.nextLine());

int hasil = angka1 / angka2;

System.out.println("Hasil pembagian: " + hasil);

} catch (NumberFormatException e) {

System.out.println("Error: Input harus berupa angka.");

} catch (ArithmeticException e) {

System.out.println("Error: Tidak dapat membagi dengan nol.");

} finally {

System.out.println("Operasi pembagian selesai.");

scanner.close();

}

}

}

**Penjelasan**

Kode ini adalah contoh penanganan exception saat membagi dua angka yang dimasukkan oleh pengguna. Input dibaca menggunakan Scanner dan dikonversi menjadi integer. Jika input bukan angka, NumberFormatException ditangani dengan pesan "Input harus berupa angka." Jika terjadi pembagian dengan nol, ArithmeticException ditangani dengan pesan "Tidak dapat membagi dengan nol." Blok finally memastikan pesan "Operasi pembagian selesai." dicetak dan Scanner ditutup untuk mencegah kebocoran sumber daya.

**Output**

|  |
| --- |
|  |