Praktikum Pemrograman Lanjut TI - C

EXCEPTION HANDLING

Try - Catch

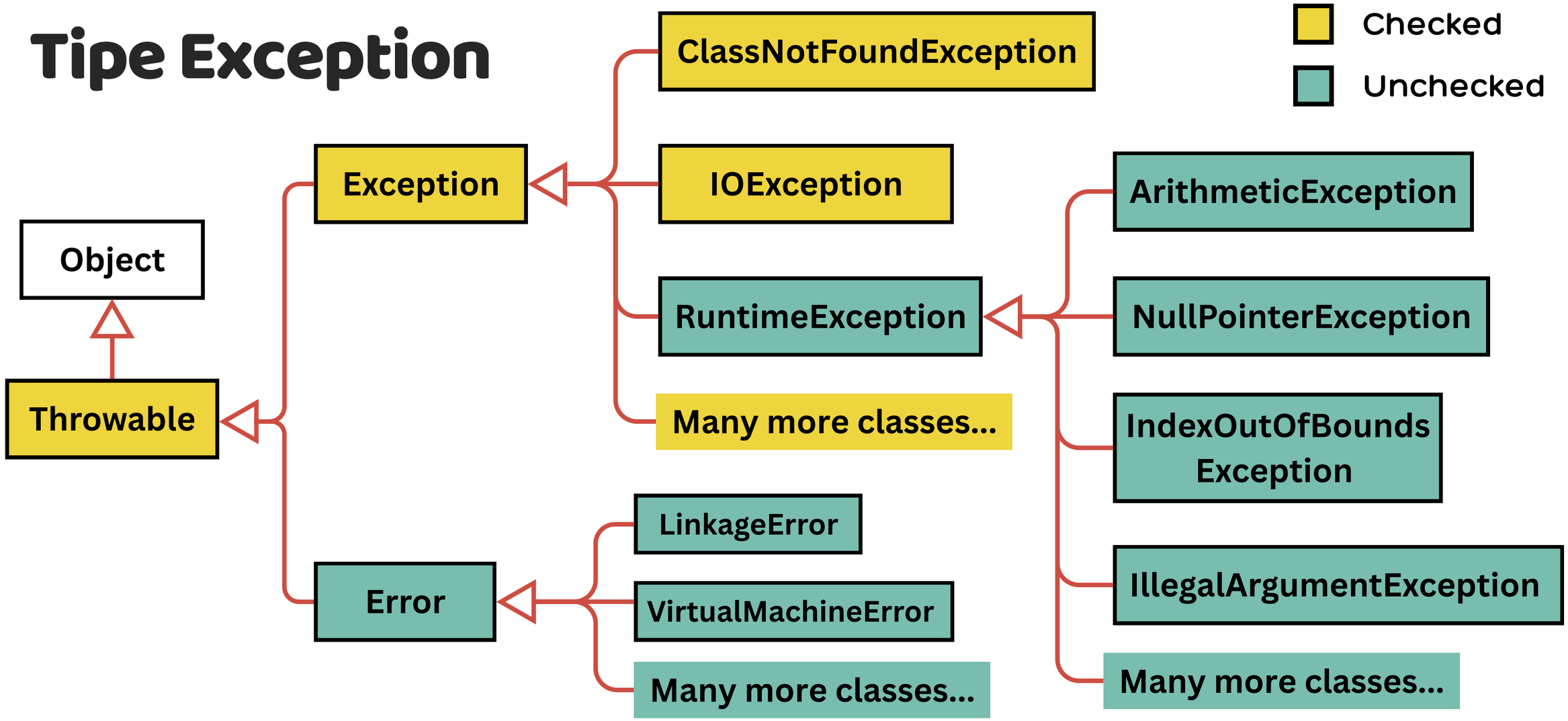
Lidwina Eleonora Dora

Definisi

Exception adalah suatu kondisi / kejadian / event yang tidak diharapkan / diinginkan yang terjadi saat eksekusi program yang mengganggu alur instruksi program.

Ini terjadi karena sesuatu yang tidak di-expect terjadi misalnya seperti mengakses index yang invalid, membagi suatu angka dengan nol, atau membuka file yang tidak ada, dan lain sebagainya.

Tipe Exception



Perbedaan Error dan Exception

ERROR

Masalah serius yang biasanya tidak bisa ditangani, eksternal, di luar kontrol program. (bukan urusan programmer)
Contohnya: OutOfMemoryError

EXCEPTION

Masalah yang bisa ditangani dengan coding oleh programmer. Contohnya: menghindari angka dibagi nol.

Checkhed vs Unchecked

Checked Exception	Unchecked
Checked at compile time.	Checked at run time.
Diturunkan dari Exception	Diturunkan dari Runtime Exception
Harus di-handle dengan try-catch block atau dideklarasikan dengan “throws”	Tidak perlu, tidak wajib handling, tapi bisa dilakukan untuk mencegah crash

Contoh Unchecked

Without try statement

```
public class ExceptionExample_v0 {
    public static void main(String[] args) {
        int pembilang, penyebut;
        Scanner input = new Scanner(System.in);

        System.out.print("Masukkan pembilang: ");
        pembilang = input.nextInt();
        System.out.print("Masukkan penyebut: ");
        penyebut = input.nextInt();
        System.out.println("Hasil pembagiannya: " + pembilang / penyebut);

        System.out.println("Akhir dari program");
    }
}
```

Contoh Unchecked

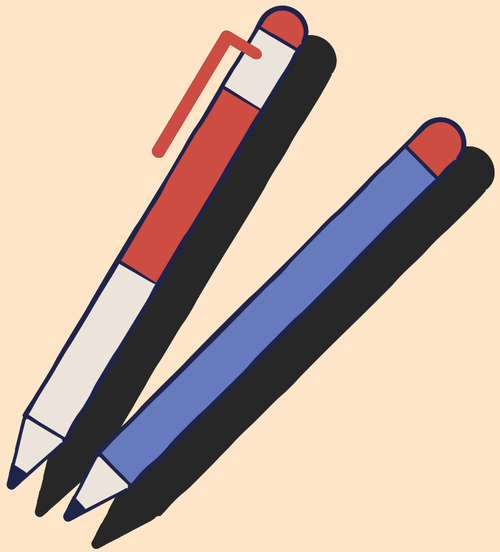
With try statement

```
public class ExceptionExample_v1 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        boolean inputValid = false; // Loop sampai berhasil

        while (!inputValid) {
            try {
                System.out.print("Masukkan pembilang: ");
                int pembilang = input.nextInt();

                System.out.print("Masukkan penyebut: ");
                int penyebut = input.nextInt();

                System.out.println("Hasil pembagian: " + (pembilang / penyebut));
                inputValid = true; // Keluar dari loop jika sukses

            } catch (InputMismatchException ex) {
                System.out.println("Error: Input harus berupa angka. Coba lagi.\n");
                input.nextLine();
            } catch (ArithmeticException ex) {
                System.out.println("Error: Pembagian oleh nol tidak diizinkan. Coba lagi.\n");
            } catch (Exception ex) {
                System.out.println("Terjadi kesalahan tidak terduga: " + ex.getMessage());
                input.nextLine();
            }
        }

        System.out.println("Akhir dari program");
        input.close();
    }
}
```

```
public class FileReading {
    public static void main(String[] args) {
        FileReader reader = null;
        try {
            // Gunakan path yang sama dengan parameter
            reader = new FileReader("src/exception/checked/HelloFriend.txt");
            readFile(reader);
        } catch (IOException ex) {
            System.out.println("Message: " + ex.getMessage());
        } finally {
            try {
                System.out.println("\nIt's a finally block");
                if (reader != null) {
                    reader.close();
                }
            } catch (IOException ex) {
                System.out.println("There is a file closing problem.");
            }
        }
    }

    static void readFile(FileReader r) throws IOException {
        int c;
        while ((c = r.read()) != -1) {
            System.out.print((char) c);
        }
    }
}
```

Contoh Checked

With try statement + throws clause

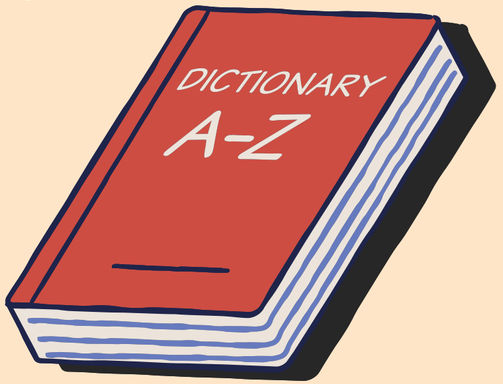


Contoh Checked

Custom Exception

With try statement + throw

https://gitlab.com/lidwinae/praktikumpemlan/-/blob/main/src/exception/checked/ProgramLogin.java

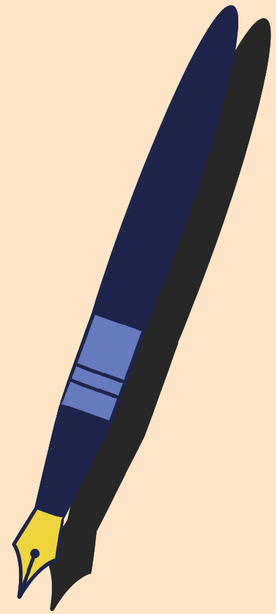


Try-Catch-Finally

Finally (opsional)

Pada penggunaan blok / statement try-catch, wajib menggunakan 1 try dan minimal 1 catch (bisa lebih dari 1 catch), finally tidak wajib digunakan.

Finally ini berguna untuk cleanup kode seperti: menutup file, koneksi database, membersihkan memori, etc. Blok finally ini berisikan kode penanganan setelah penggunaan try and catch, yang akan selalu tereksekusi meskipun exception terjadi.



Throw vs Throws

Throw	Throws
Digunakan untuk secara aktif melemparkan exception, melempar instance exception.	Digunakan untuk mendeklarasikan bahwa method mungkin melemparkan exception, mendeklarasikan tipe exception.
Ditulis dalam body method	Ditulis di deklarasi method
Hanya satu exception setiap kali, contoh : throw new NullPointerException();	Bisa beberapa exception, contoh : void method() throws IOException, SQLException

Kegunaan Exception Handling

- Menangani error secara terkontrol.
- Mencegah program berhenti secara tiba-tiba (crash).
- Memisahkan logika normal dan logika penanganan kesalahan.
- Memberikan informasi yang jelas ke pengguna atau developer.
- etc.

Sekian dan Terima Kasih