

# Penjebakan Eksepsi

Nur hasanah, M.Cs

# Pendahuluan

- Dalam Java, *runtime error* (kesalahan-kesalahan yang terjadi pada saat program sedang berjalan) disebut eksepsi.
- Terdapat 5 buah kata kunci : `try`, `catch`, `throw`, `throws` dan `finally`.
- Kata kunci `try` digunakan untuk membuat blok berisi statemen-statemen yang mungkin menimbulkan eksepsi.
- Apabila dalam proses eksekusi runtunan statemen tsb terjadi sebuah eksepsi, maka eksepsi akan dilempar ke bagian blok penangkap yang dibuat dengan kata kunci `catch`.

- Pada kasus-kasus tertentu, terkadang kita juga ingin melempar eksepsi secara manual. Untuk melakukan hal tersebut, maka gunakan kata kunci `throw`.
- Apabila kita ingin membangkitkan sebuah eksepsi tanpa menuliskan blok `try`, maka kita perlu menambahkan kata kunci `throws` pada saat pendeklarasian `method`.
- Dalam mendefinisikan blok `try`, kita juga diizinkan untuk menulis statemen tambahan, yaitu kata kunci `finally`.
- Statemen `finally` pasti akan dieksekusi baik terjadi eksepsi atau tidak.

- Bentuk umum penanganan eksepsi:

```
try {  
    //kumpulan statemen yang mungkin menimbulkan eksepsi  
} catch(TipeEksepsi1 objekEksepsi1) {  
    //penanganan untuk tipe eksepsi1  
} catch (TipeEksepsi2 objekEksepsi2) {  
    //penanganan untuk tipe eksepsi2  
}  
...  
finally {  
    //statemen tambahan yang pasti akan dieksekusi  
}
```



# Contoh

```
class ContohEksepsi1 {
    public static void main(String[] args) {
        int[] A = new int[5];
        A[5] = 100; // SALAH, karena tidak terdapat indeks ke-5
    }
}
```

```
class ContohEksepsi2 {
    public static void main(String[] args) {
        int pembilang = 2;
        int penyebut = 0;
        int hasil = pembilang/penyebut; // SALAH
        System.out.println("Hasil = " + hasil);
    }
}
```

- Kedua contoh di atas menunjukkan terjadinya eksepsi dan maka program dihentikan secara tidak normal

# Menggunakan kata kunci `try` dan `catch`

```
class ContohEksepsi3 {
    public static void main(String[] args) {
        int pembilang = 2;
        int penyebut = 0;
        try {
            int hasil = pembilang/penyebut; // menimbulkan eksepsi
            System.out.println("Hasil = " + hasil); //tdk dieksekusi
        } catch (ArithmeticException ae) {
            System.out.println("KESALAHAN: " +
                "Terdapat pembagian dengan nol");
        }
        System.out.println("Statemen setelah blok try-catch");
    }
}
```

- Meskipun ada kesalahan, tetapi program tidak dihentikan secara tiba2, karena diatasi blok *try-catch*.

# Menggunakan kata kunci `try` dan `catch`

- Apabila kita tidak mengetahui kemungkinan kesalahan yang akan timbul, maka dapat menggunakan tipe `Exception` sebagai parameternya.

```
class ContohEksepsi4 {
    public static void main(String[] args) {
        int pembilang = 2;
        int penyebut = 0;
        try {
            int hasil = pembilang/penyebut;           // SALAH
            System.out.println("Hasil = " + hasil); //tdk dieksekusi
        } catch (Exception e) {
            System.out.println("KESALAHAN: " +
                "Terdapat pembagian dengan nol");
        }
        System.out.println("Statemen setelah blok try-catch");
    }
}
```



# Menggunakan kata kunci `try` dan `catch`

- Pada kasus-kasus tertentu, mungkin kita ingin menampilkan pesan sebenarnya yang terkandung dalam eksepsi yang ditimbulkan.
- Kita dapat menggunakan method `getMessage()`

```
class ContohEksepsi5 {
    public static void main(String[] args) {
        int pembilang = 2;
        int penyebut = 0;
        try {
            int hasil = pembilang/penyebut;           // SALAH
            System.out.println("Hasil = " + hasil); // tdk dieksekusi
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        System.out.println("Statemen setelah blok try-catch");
    }
}
```

# Menggunakan kata kunci `try` dan `catch`

- Apabila kita ingin mengeluarkan informasi *stack trace*, gunakan method `printStackTrace()`.

```
class ContohEksepsi6 {
    public static void main(String[] args) {
        int pembilang = 2;
        int penyebut = 0;
        try {
            int hasil = pembilang/penyebut;           // SALAH
            System.out.println("Hasil = " + hasil); //tdk dieksekusi
        } catch (Exception e) {
            e.printStackTrace(); // mencetak stack trace
        }
        System.out.println("Statemen setelah blok try-catch");
    }
}
```

# Menggunakan kata kunci `try` dan `catch`

- Contoh program yang mengatasi kesalahan dalam pengaksesan indeks array

```
class EksepsiIndeksArray {
    public static void main(String[] args) {
        int[] A = new int[5];

        try {
            // mengisi elemen array
            for (int i=0; i<5; i++) {
                A[i] = i * 10;
            }
            // akan menimbulkan eksepsi (mengakses indeks array ke-9)
            System.out.println("Nilai pada indeks ke-9 : " + A[9]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Tidak terdapat indeks ke-" + e.getMessage());
        }

        // menampilkan elemen array
        for (int i=0; i<5; i++) {
            System.out.println("Elemen ke-" + i + " : " + A[i]);
        }
    }
}
```

# Penjebakan Beberapa Tipe Eksepsi

- Pada saat terjadi eksepsi dg tipe tertentu dalam blok `try`, maka program akan mencari statemen `catch` yang sesuai dengan tipe eksepsi yang dibangkitkan.

```
class BanyakEksepsi {
    public static void test(int a, int b) {
        try {
            int c = a / b;
            System.out.println("Hasil bagi: " + c);

            int[] Arr = {1,2,3,4,5}; // array dengan 5 elemen
            Arr[10] = 11; // mengakses indeks ke-10
        } catch (ArithmeticException ae) {
            System.out.println("Terdapat pembagian dengan 0");
            System.out.println(ae);
        } catch (ArrayIndexOutOfBoundsException oobe) {
            System.out.println("Indeks di luar rentang");
            System.out.println(oobe);
        }
    }

    public static void main(String[] args) {
        test(4, 0); // menimbulkan ArithmeticException
        System.out.println();
        test(12, 4); // menimbulkan ArrayIndexOutOfBoundsException
    }
}
```

# Penjebakan Beberapa Tipe Eksepsi

- Urutan tipe eksepsi harus diperhatikan
- Kelas eksepsi turunan harus ditempatkan lebih awal dibandingkan dengan kelas induknya.

```
class UrutanCatch3 {
    public static void main(String[] args) {

        int bilangan = 12;
        int pembagi = 4;

        try {
            int hasil = bilangan / pembagi;
            System.out.println("Hasil : " + hasil);
            int[] Arr = {1,2,3,4,5};
            Arr[10] = 11;
        } catch (ArithmeticException ae) {
            System.out.println("Terjadi pembagian dengan 0");
        } catch (Exception e) {
            System.out.println("Eksepsi generik dieksekusi");
        }
    }
}
```

# Menggunakan kata kunci `throw`

- Jika kita ingin membangkitkan eksepsi secara manual, kita gunakan kata kunci `throw`.
- Bentuk umum:  
*throw* *eksepsi*;

```

class Barang {
    private String kode;
    private String nama;
    private double harga;

    public void setKode(String vKode) {
        try {
            kode = vKode;
            if (kode == null) {
                throw new NullPointerException();
            }
        } catch (NullPointerException npe) {
            System.out.println("KESALAHAN: " +
                "Kode barang tidak boleh null");
        }
    }

    public String getKode() {
        return kode;
    }

    public void setNama(String vNama) {
        try {
            nama = vNama;
            if (nama == null) {
                throw new NullPointerException();
            }
        } catch (NullPointerException npe) {
            System.out.println("KESALAHAN: " +
                "Nama barang tidak boleh null");
        }
    }
}

```

```

    public String getNama() {
        return nama;
    }

    public void setHarga(int vHarga) {
        harga = vHarga;
    }

    public double getHarga() {
        return harga;
    }
}

class DemoThrow {
    public static void main(String[] args)
    {

        Barang obj = new Barang();

        obj.setKode(null);
        obj.setNama("Buku tulis");
        obj.setHarga(2500);

        System.out.println("\nKode : " +
            obj.getKode());
        System.out.println("Nama : " +
            obj.getNama());
        System.out.println("Harga : " +
            obj.getHarga());
    }
}

```

# Menggunakan kata kunci `throw`

- Pada Java, eksepsi hanya memiliki 2 tipe constructor yaitu:
  - Constructor tanpa parameter
  - Constructor yang memiliki 1 parameter bertipe `String`.
- Pada contoh sebelumnya, kita hanya menggunakan constructor bentuk yg pertama.
- Sedangkan constructor bentuk yg kedua, terlihat pada contoh berikutnya



```
class Barang {
    private String kode;
    private String nama;
    private double harga;

    public void setKode(String vKode) {
        try {
            kode = vKode;
            if (kode == null) {
                throw new
NullPointerException("KESALAHAN: " +
                "Kode barang tidak boleh null");
            }
        } catch (NullPointerException npe) {
            System.out.println(e.getMessage);
        }
    }

    public String getKode() {
        return kode; }

    public void setNama(String vNama) {
        try {
            nama = vNama;
            if (nama == null) {
                throw new
NullPointerException("KESALAHAN: " +
                "Nama barang tidak boleh null");
            }
        } catch (NullPointerException npe) {
            System.out.println(npe.getMessage);
        }
    }
}
```

```
public String getName() {
    return nama;
}

public void setHarga(int vHarga) {
    harga = vHarga;
}

public double getHarga() {
    return harga; }
}
```

# Menggunakan kata kunci `throws`

- Apabila kita tidak menyertakan blok `try-catch` di dlm `method`, maka kita harus menyertakan klausa `throws` pada saat pendeklarasian `method` bersangkutan.
- Jika tidak, maka program tidak dapat dikompilasi.
- Cara ini juga dapat digunakan untuk beberapa tipe eksepsi.

- Bentuk umum:

```
tipe nama-method(daftar-parameter) throws tipe-  
eksepsi1, tipe-eksepsi2,...{
```

```
//badan method
```

```
}
```

# Menggunakan kata kunci `throws`

```
class DemoThrows {
    public static void test() throws IllegalAccessException {
        throw new IllegalAccessException(
            "KESALAHAN: illegal access");
    }

    public static void main(String[] args) {
        try {
            test();
        } catch (Exception e) {
            System.out.println("Eksepsi ditangkap di sini...");
            System.out.println(e.getMessage());
        }
        System.out.println("Statemen setelah blok try-catch");
    }
}
```

# Menggunakan kata kunci `throws`

- Kata kunci `throws` juga dapat digunakan untuk menangkap beberapa tipe eksepsi. Pemisahan antar-tipe eksepsi dilakukan dengan menggunakan tanda koma.
- Perhatikan contoh selanjutnya.

# Menggunakan kata kunci `throws`

```
class DemoThrows2 {
    public static void test(int n)
        throws NullPointerException, ArithmeticException {

        if (n < 0) {
            throw new NullPointerException(
                "KESALAHAN: null pointer");
        } else {
            throw new ArithmeticException(
                "KESALAHAN: arithmetic exception");
        }
    }

    public static void main(String[] args) {
        try {
            //test(-12); // menimbulkan eksepsi NullPointerException
            test(0); // menimbulkan eksepsi ArithmeticException
        } catch (Exception e) {
            System.out.println("Eksepsi ditangkap di sini...");
            System.out.println(e.getMessage());
        }
        System.out.println("Statemen setelah blok try-catch");
    }
}
```

# Menggunakan kata kunci `finally`

- Blok finalisasi digunakan pada saat kita ingin menempatkan kode yang pasti akan dieksekusi, baik terjadi eksepsi atau tidak.
- Blok finalisasi dibuat dengan menggunakan kata kunci `finally`
- Bentuk umum:

```
try {  
    // statemen yang mungkin menimbulkan eksepsi A, B dan C  
} catch (A ea) {  
    // blok penangkap untuk eksepsi A  
} catch (B eb) {  
    // blok penangkap untuk eksepsi B  
} catch (C ec) {  
    // blok penangkap untuk eksepsi C  
} finally {  
    // statemen yang pasti akan dieksekusi, baik terjadi eksepsi maupun  
    tidak  
}
```

# Menggunakan kata kunci `finally`

```
class DemoFinally {
    private static int i = 0;

    public static void main(String[] args) {
        while (true) {
            try {
                System.out.print("Pada saat i = " + i + ": ");
                if (i++ == 0) {
                    throw new Exception(); // melempar eksepsi
                }
                System.out.println("Tidak terjadi eksepsi");
            } catch (Exception e) {
                System.out.println("Terdapat eksepsi");
            } finally {
                System.out.println("Statemen dalam blok finally\n");
                if (i == 2) {
                    break; // pada saat i==2, pengulangan akan berhenti
                }
            }
        }
    }
}
```

# Reference

- Budi Rahardjo dkk. (2012). “*Mudah Belajar Java*”. Penerbit Informatika Bandung.