EKSEPSI

Tim Pengampu Pemrograman Berorientasi Objek
Departemen Ilmu Komputer Informatika Universitas Diponegoro

Agenda

- Eksepsi
- Menggunakan Blok `try-catch`
- Hierarki Eksepsi
- Kategori Checked dan UnChecked
- Penggunaan Klausa `throws`
- Kesimpulan

1 Eksepsi

- Adalah kondisi eksekusi program di luar jalur normal.
- Bisa terjadi karena kesalahan-kesalahan tertentu.
- Perlu mekanisme khusus untuk **menangani** kesalahan dan **mendelegasikannya** ke dalam objek-objek tertentu dalam **hirarki** tertentu pula.
- Eksepsi merupakan mekanisme program, **bagian dari implementasi** yang sengaja dideklarasikan untuk menangani kondisi abnormal program.

2 Menggunakan Blok try-catch (1)

- Deklarasi eksepsi menggunakan blok `try-catch`, `try-catch*` atau `try-catch*-finally`.
  - Blok `try{}` berisi kode yang di dalamnya ada kemungkinan terjadi kesalahan sehingga keluar dari jalur eksekusi normal program.
  - Blok `catch{}` digunakan untuk menangani kesalahan seperti melakukan recovery menampilkan pesan kesalahan, dsb.
  - Blok `finally{}` digunakan untuk mengeksekusi kode apapun kondisi eksekusi program.

2 Menggunakan Blok try-catch (2)

try-catch

```
try{
  //kode yang kemungkinan akan terjadi kesalahan
}catch(ExceptionClass  reference){
  //kode jika kesalahan
  //yang ditangani oleh ExceptionClass terjadi
}
```

2 Menggunakan Blok try-catch (3)

try-catch*

```
try{
  //kode yang kemungkinan akan terjadi kesalahan

}catch(ExceptionCls3  reference){
  //kode jika kesalahan yang ditangani
  //oleh ExceptionCls3 terjadi
}catch(ExceptionCls2  reference){
  //kode jika kesalahan yang ditangani
  //oleh ExceptionCls2 terjadi
}catch(ExceptionCls1  reference){
  //kode jika kesalahan yang ditangani
  //oleh ExceptionCls1 terjadi
}
```

2 Menggunakan Blok try-catch (4)

try-catch*-finally

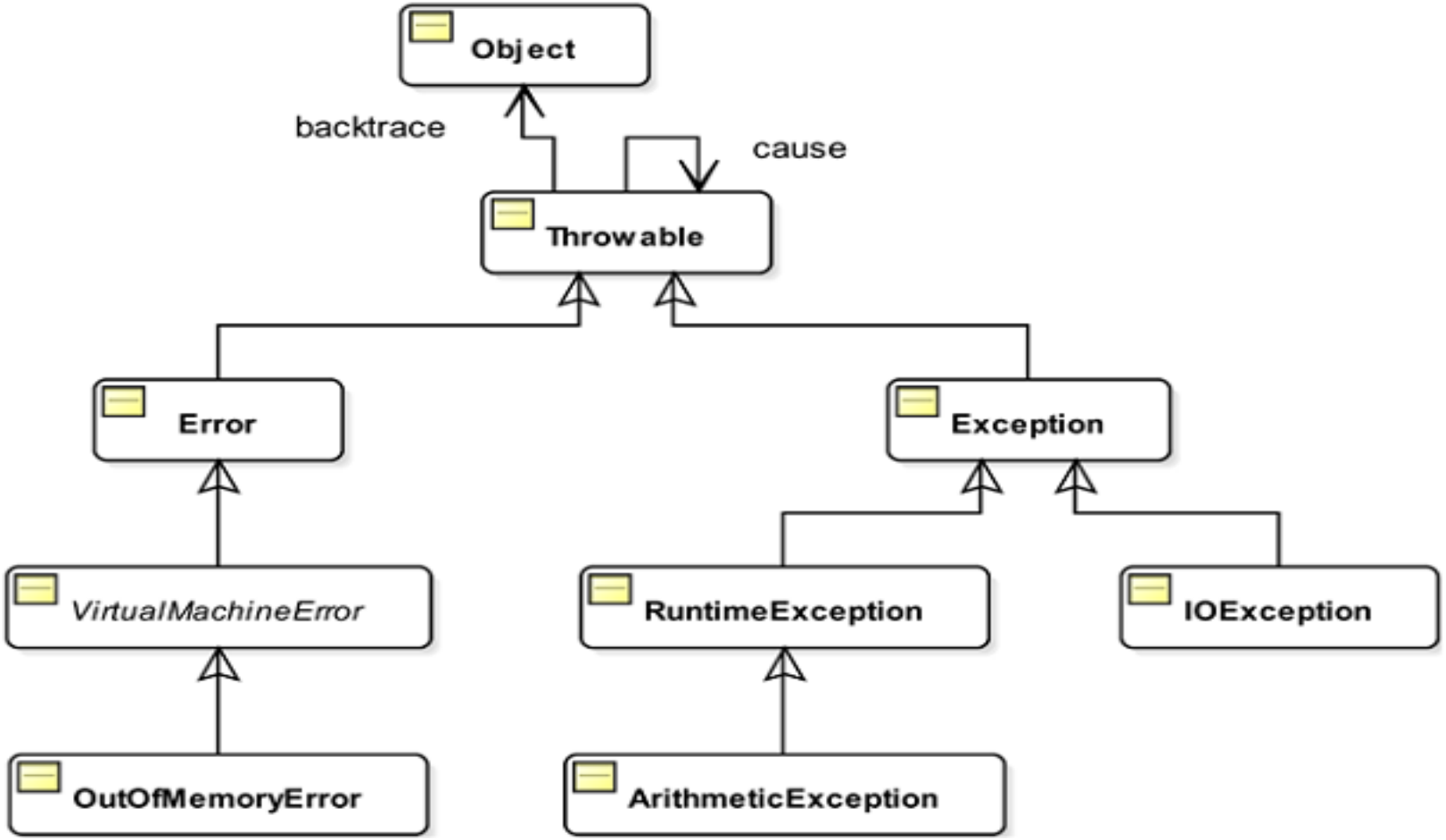
```
try{
  //kode yang kemungkinan akan terjadi kesalahan

}catch(ExceptionCls3  reference){
  //kode jika kesalahan yang ditangani
  //oleh ExceptionCls3 terjadi
}catch(ExceptionCls2  reference){
  //kode jika kesalahan yang ditangani
  //oleh ExceptionCls2 terjadi
}catch(ExceptionCls1  reference){
  //kode jika kesalahan yang ditangani
  //oleh ExceptionCls1 terjadi
} finally{
  //kode yang harus dijalankan, apapun kondisinya
}
```

2 Menggunakan Blok try-catch (5)

- Pada `try-multi catch`, apabila terjadi kesalahan pada blok try, hanya ada satu eksekusi blok `catch`.
- Apabila terdapat blok `finally`, maka setelah eksekusi `try` atau salah satu `catch`, maka blok `finally` dieksekusi.

3 Hierarki Kelas Eksepsi (1)

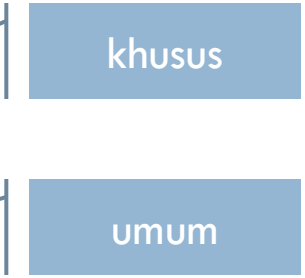


Gambar 1. Beberapa kelas pada hierarki kelas Exception

3 Hierarki Kelas Eksepsi (2)

- Pada eksepsi menggunakan `try-catch*`, hirarki objek eksepsi dapat dimanfaatkan, dengan meletakkan kelas eksepsi yang lebih khusus di **awal**.

```
try{
   ....
} catch(IOException e){
   ....
} catch(Exception ioe){
   ....
}
```

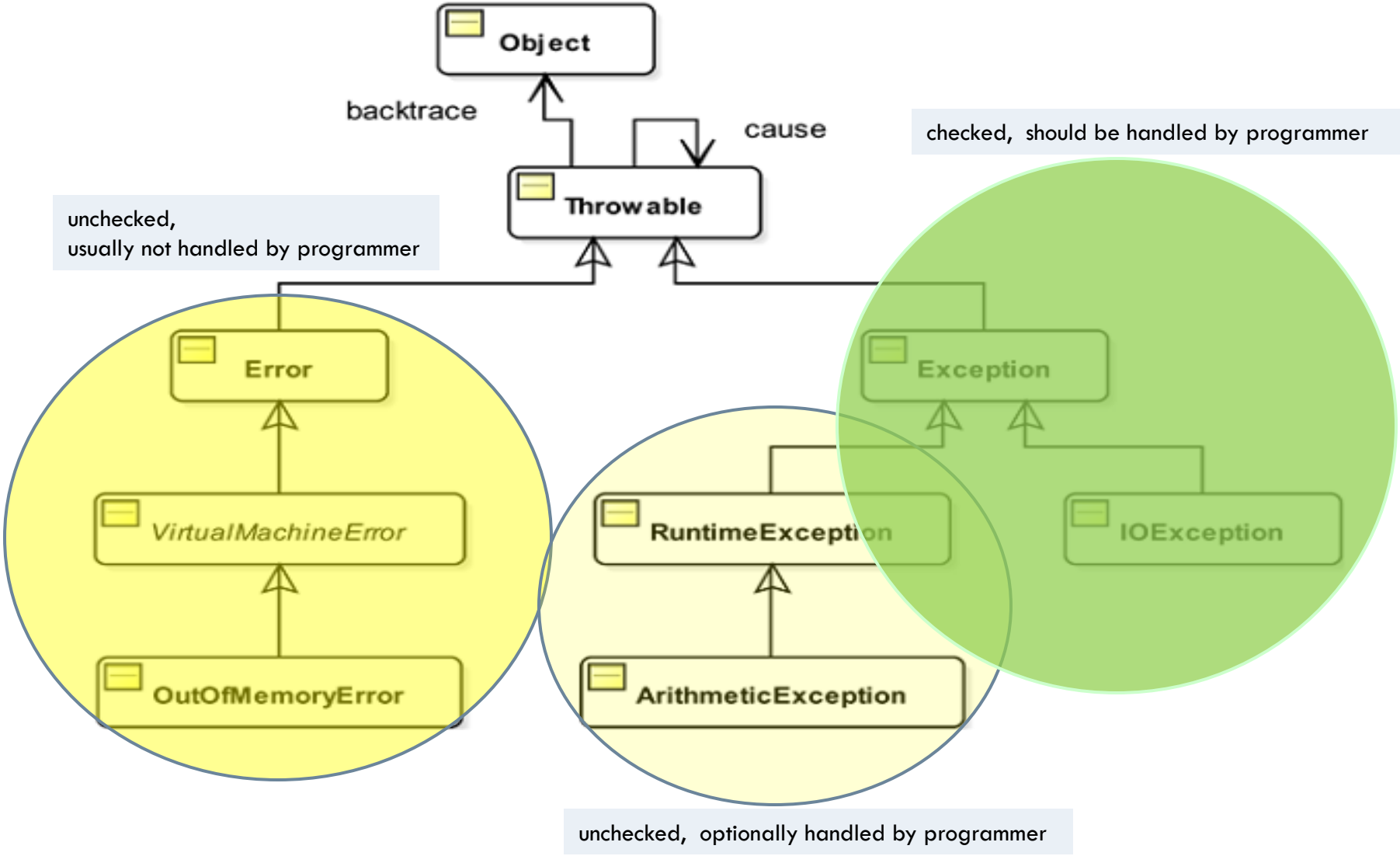


- Tidak disarankan meletakkan objek eksepsi yang umum diikuti yang khusus

4 Kategori Checked dan Unchecked (1)

- Kategori eksepsi :
  - **Checked**, compiler memeriksa eksepsi dan meminta untuk ditangani (pada kode program)
  - **Unchecked**, compiler tidak memeriksa apakah telah ditangani pada kode program
- Kelas `Throwable`, `Exception` serta turunannya, (kecuali `RuntimeException` dan turunannya) digunakan untuk menangani *checked exception*
- Selebihnya, digunakan untuk menangani *unchecked exception*.

4 Kategori Checked dan Unchecked (2)



Contoh1 Checked Exception (1)

```
import java.io.*;
class Example {
   public static void main(String args[])
   {
         FileInputStream fis = null;
         // throws checked exception: FileNotFoundException
         fis = new FileInputStream("B:/myfile.txt");
         int k;
         // throws checked exception: IOException
         while(( k = fis.read() ) != -1)
         {
           System.out.print((char)k);
         }
         // throws checked exception: IOException
         fis.close();
   }
}
```

Compile kode program ini, apakah berhasil?

Contoh1 Checked Exception (2)

```
import java.io.*;
class Example {
   public static void main(String args[])
   {
         FileInputStream fis = null;
         // throws checked exception: FileNotFoundException
         fis = new FileInputStream("B:/myfile.txt");
         int k;
         // throws checked exception: IOException
         while(( k = fis.read() ) != -1)
         {
           System.out.print((char)k);
         }
         // throws checked exception: IOException
         fis.close();
   }
}
```

```
C:\testexception>javac Example.java
Example.java:10: error: unreported exception FileNotFoundException; must be caught or declared to be thrown
       fis = new FileInputStream("B:/myfile.txt");
             ^
Example.java:16: error: unreported exception IOException; must be caught or declared to be thrown
       while(( k = fis.read() ) != -1)
                           ^
Example.java:23: error: unreported exception IOException; must be caught or declared to be thrown
       fis.close();
               ^
3 errors
```

Contoh1 Checked Exception (3)

```
import java.io.*;
class Example {

    public static void main(String args[]) {
        FileInputStream fis = null;
        try {
            fis = new FileInputStream("B:/myfile.txt");
        } catch (FileNotFoundException ex) {
            System.out.println("Terjadi FileNotFoundException: " + ex);
        }
        int k;
        try {
            while ((k = fis.read()) != -1) {
                System.out.print((char) k);
            }
            fis.close();
        } catch (IOException ex) {
            System.out.println("Terjadi IOException Berikut: " + ex);
        }
    }
}
```

Compile kode program ini, apakah berhasil? Bagaimana output yang dihasilkan?

Contoh2 UnChecked Exception (1)

```
public class UncheckedException {
    public static void main(String[] args) {
        int num1 = 10;
        int num2 = 0;
        // throws unchecked Exception: ArithmeticException
        int res = num1 / num2;
        System.out.println(res);
    }
}
```

```
public class UncheckedException2 {
    public static void main(String[] args) {
        int arr[] = {1, 2, 3, 4, 5};
        // throws unchecked exception: ArrayIndexOutOfBoundsException
        System.out.println(arr[7]);
    }
}
```

Compile kode program ini, apakah berhasil?

Bagaimana ketika program ini running?

Contoh2 UnChecked Exception (1)

```
public class Uncheckedexception {
    public static void main(String[] args) {
        int num1 = 10;
        int num2 = 0;
        // throws unchecked Exception: ArithmeticException
        int res = num1 / num2;
        System.out.println(res);
    }
}
```

```
C:\testexception>java Uncheckedexception
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Uncheckedexception.main(Uncheckedexception.java:6)
```

```
public class Uncheckedexception2 {
    public static void main(String[] args) {
        int arr[] = {1, 2, 3, 4, 5};
        // throws unchecked exception: ArrayIndexOutOfBoundsException
        System.out.println(arr[7]);
    }
}
```

```
C:\testexception>java Uncheckedexception2
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 7
        at Uncheckedexception2.main(Uncheckedexception2.java:5)
```

Apakah bisa ditangani?

Contoh2 UnChecked Exception (1)

```
public class Uncheckedhandled {
    public static void main(String[] args) {
        int num1 = 10;
        int num2 = 0;
        // throw unchecked Exception: ArithmeticException
        try {
            int res = num1 / num2;
            System.out.println(res);
        } catch (ArithmeticException e) {
            System.out.println("Terjadi ArithmeticException: " + e.getMessage());
        }
    }
}
```

```
C:\testexception>java Uncheckedhandled
Terjadi ArithmeticException: / by zero
```

```
public class Uncheckedhandled2 {

    public static void main(String[] args) {
        int arr[] = {1, 2, 3, 4, 5};
        try {
            System.out.println(arr[7]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Terjadi ArrayIndexOutOfBoundsException: "
               + e.getMessage());
        }
    }
}
```

```
C:\testexception>java Uncheckedhandled2
Terjadi ArrayIndexOutOfBoundsException: 7
```

5 Penggunaan klausa throws

- Method dapat mengabaikan penanganan Eksepsi menggunakan `throws`. Ketika terjadi Eksepsi, method tersebut melemparkan (dan mengirim) ke method yang memanggilnya. Ini artinya method pemanggil harus menangkap (catch) atau melempar (throw) eksepsi yang didapat

eksepsi ditangani sendiri dengan blok try-catch

```
public static void openFile(String name)
{
    try
    {
        FileInputStream f = new FileInputStream(name);
    }
    catch (FileNotFoundException e)
    {
        System.out.println("File not found.");
    }
}
```

eksepsi dilemparkan ke method pemanggil menggunakan klausa throws

```
public static void openFile(String name) throws FileNotFoundException
{
    FileInputStream f = new FileInputStream(name);
}
```

Contoh3 Throwing an Exception (1)

```
public class KonversiString {
    public static void keInteger1(String string) {
        try {
            int hasil = Integer.parseInt(string);
            System.out.println(hasil);
        } catch (NumberFormatException e) {
            System.out.println("Kesalahan keInteger1 :" + e.getMessage());
        }
    }
    public static void keInteger2(String string) throws NumberFormatException{
        int hasil = Integer.parseInt(string);
        System.out.println(hasil);
    }
    public static void main(String[] args) {
        keInteger1("hai");
        try {
            keInteger2("hai");
        } catch (NumberFormatException e) {
            System.out.println("Kesalahan keInteger2 :" + e.getMessage());
        }
    }
}
```

Bagaimana output hasil eksekusi ?

Contoh3 Throwing an Exception (2)

```
public class KonversiString {
    public static void keInteger1(String string) {
        try {
            int hasil = Integer.parseInt(string);
            System.out.println(hasil);
        } catch (NumberFormatException e) {
            System.out.println("Kesalahan keInteger1 :" + e.getMessage());
        }
    }
    public static void keInteger2(String string) throws NumberFormatException{
        int hasil = Integer.parseInt(string);
        System.out.println(hasil);
    }
    public static void main(String[] args) {
        keInteger1("hai");
        try {
            keInteger2("hai");
        } catch (NumberFormatException e) {
            System.out.println("Kesalahan keInteger2 :" + e.getMessage());
        }
    }
}
```

```
run:
Kesalahan keInteger1 :For input string: "hai"
Kesalahan keInteger2 :For input string: "hai"
```

Review

- Eksepsi adalah ....
- Eksepsi dapat didelegasikan kepada ....
- Deklarasi eksepsi menggunakan ....
- Apakah blok finally selalu dieksekusi ?
- Apa perbedaan eksespsi Checked dan Unchecked
- Apa tujuan penggunaan klausa throws?

Kesimpulan

- Eksepsi adalah kondisi di luar kondisi normal eksekusi program yang terjadi karena kesalahan tertentu.
- Penanganan eksepsi (*checked* maupun *unchecked*) dapat dilakukan oleh objek-objek khusus dengan **hirarki** tertentu dan **deklarasi** tertentu.