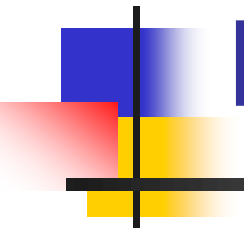
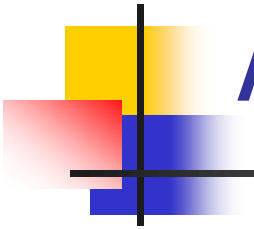


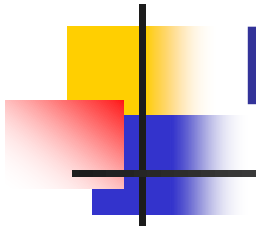
Exception Handling



Agenda

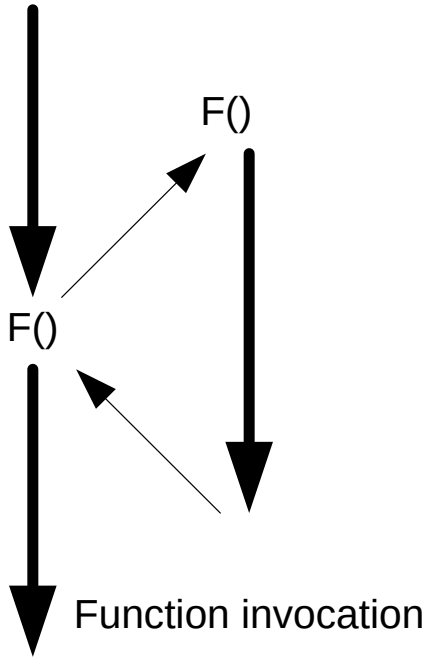
- Exception Handling
- Throwable Class
- Error Class
- Exception Class
- Exception Handling in Java
- Try-Catch Blocks
- Finally Block
- Method That Throws Exception
- Throw Statement
- User-defined Exception
- Exception Propagation

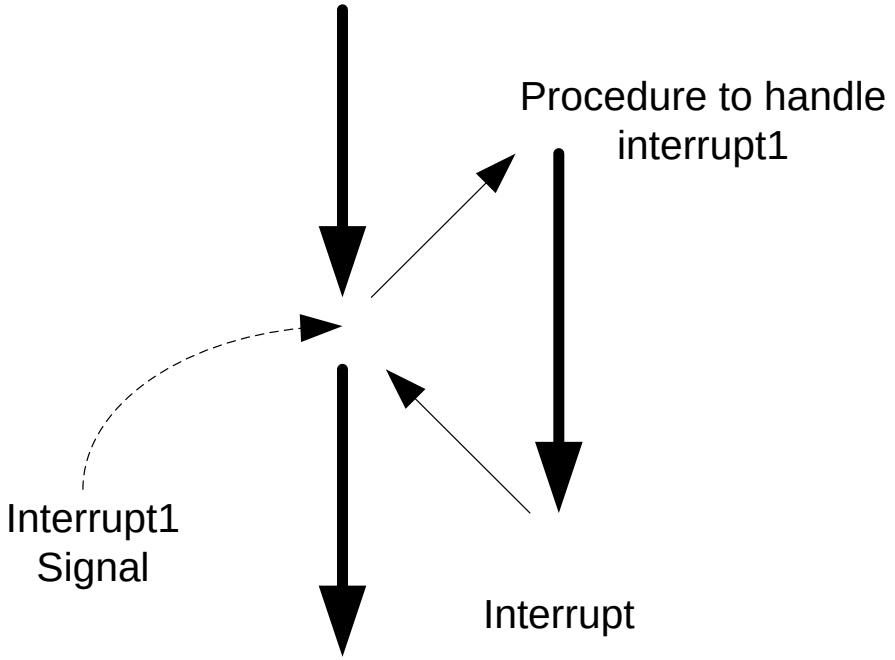
Exception Handling



- เป็นกลไกที่ช่วยจัดการเกี่ยวกับข้อผิดพลาดที่อาจจะเกิดขึ้นได้ในระหว่างที่โปรแกรมทำงาน
- เป็นการนำ Interrupt มาพัฒนาบนภาษาชั้นสูง (เพียงแต่จะจัดการเฉพาะข้อผิดพลาดที่เกิดขึ้นเท่านั้น)
- ข้อแตกต่างระหว่าง Interrupt กับ Exception คือ Interrupt สามารถที่จะเกิดขึ้นเมื่อไหร่ก็ได้ แต่ Exception จะเกิดขึ้นเมื่อทำงานกับบาง statement เท่านั้น (เป็นความผิดพลาดของ statement นั้น ๆ)
- ประโยชน์ของ Exception Handling คือทำให้โปรแกรมสามารถแยก statement ที่ทำงานปกติออกจาก statement ที่ใช้สำหรับตรวจสอบความผิดพลาด (เพราะถ้ารวมกัน โปรแกรมจะตรวจสอบความผิดพลาดทุกครั้ง ถึงแม้ว่ามันจะไม่เกิดขึ้นก็ตาม)

Exception Handling





Throwable Class

- ภาษา Java มี Throwable class ใช้สำหรับสร้าง Instance สำหรับความผิดพลาดที่เกิดขึ้นของโปรแกรม ซึ่งมี constructor ดังนี้

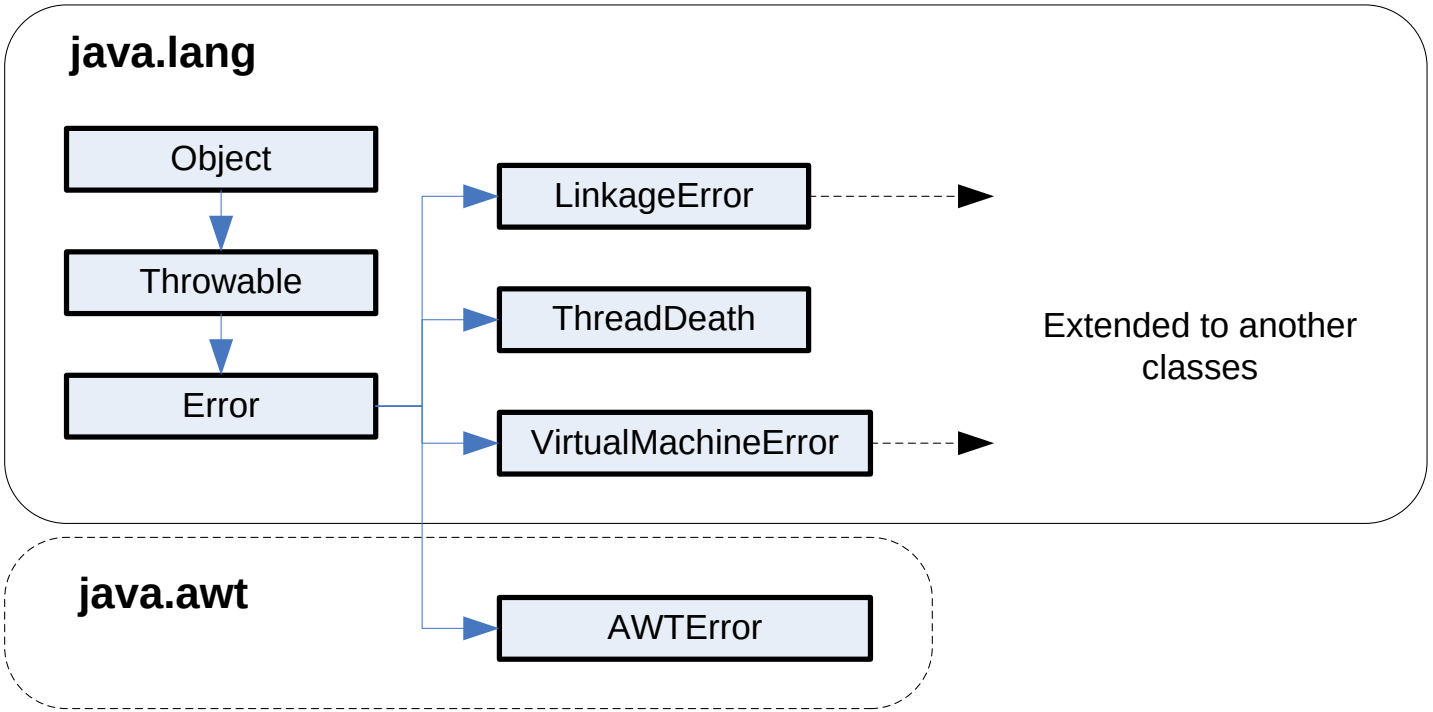
```
public Throwable();
public Throwable(String message);
```

- และมี accessor ดังนี้

```
public String getMessage();
```

Error Class

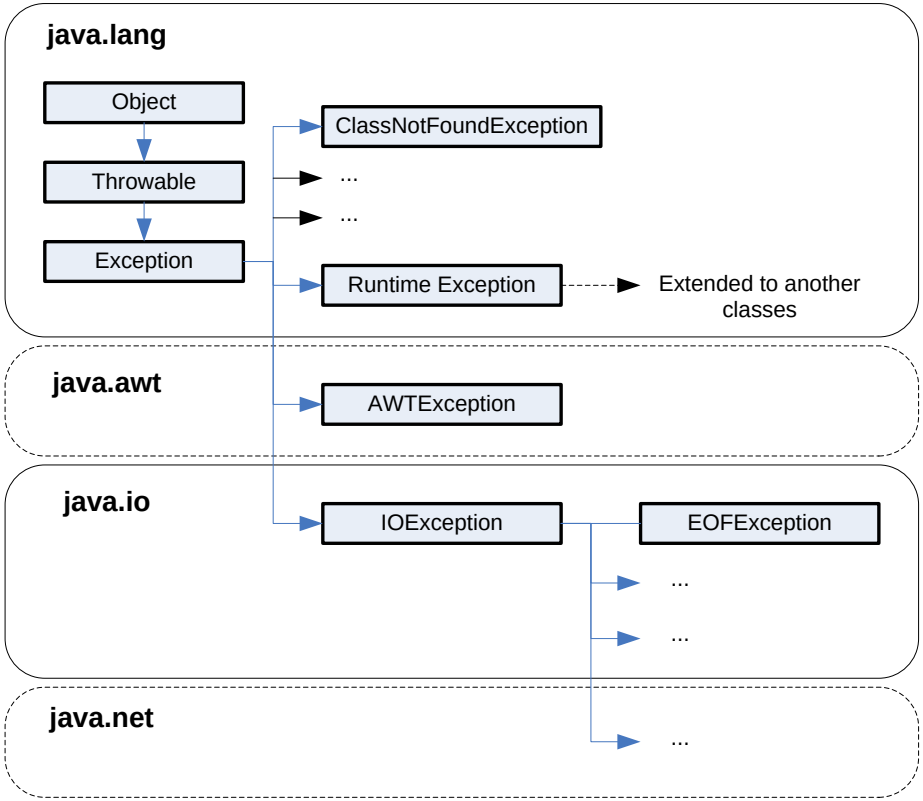
Error class จะขยายมาจาก **Throwable class** ทำหน้าที่สร้าง **instance** ที่เกิดจากความผิดพลาดที่ร้ายแรง แต่โดยปกติจะไม่จับ **instance** ที่สร้างจาก **class** นี้ เนื่องจากว่าจับไปก็ทำอะไรไม่ได้ จึงปล่อยให้ **Interpreter** เป็นตัวจัดการ



Exception Class

Exception class ขยายมาจาก **Throwable class** ทำหน้าที่สร้าง **instance** ที่ไม่ร้ายแรง โดยมีโครงสร้างดังนี้

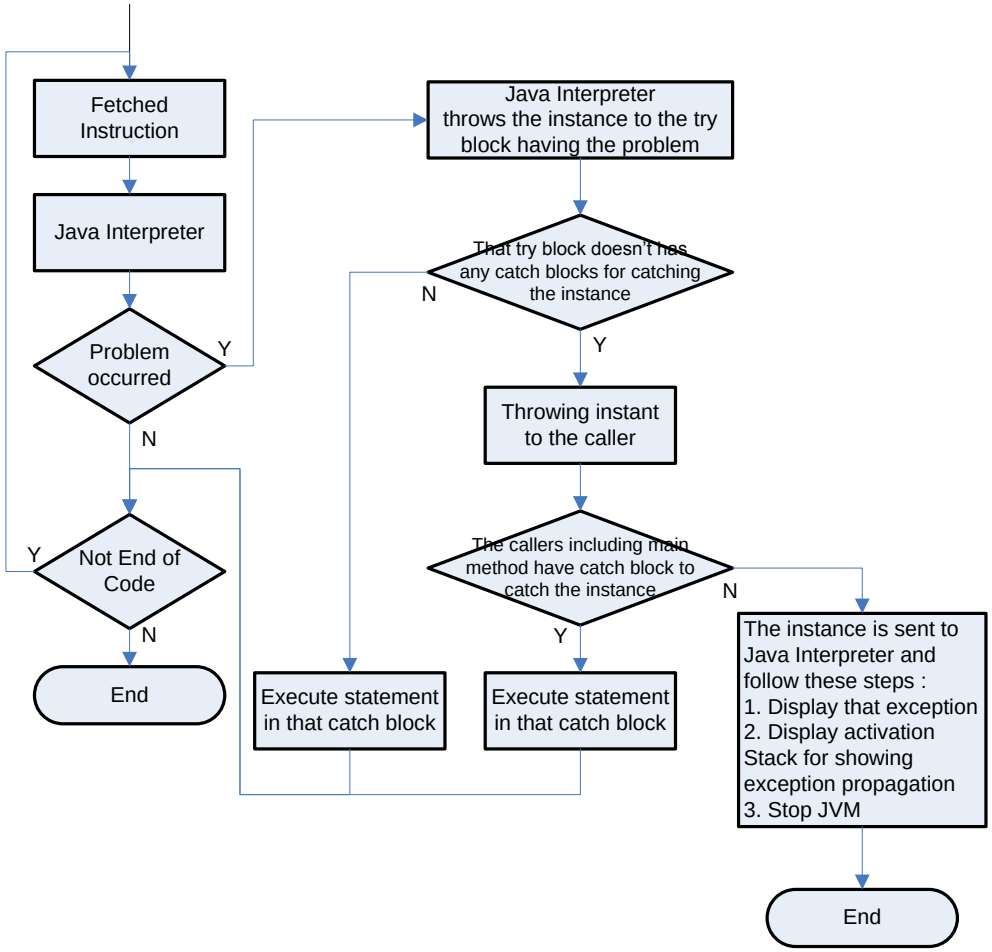
```
public class Exception extends Throwable {
    public Exception();
    public Exception(String s);
}
```



Exception Handling in Java

- **Java มี class ที่ขยายมาจาก exception class ไว้ให้ Java Interpreter สร้าง Instance เวลาเกิดข้อผิดพลาดของคำสั่ง เราเรียก class เหล่านี้ว่า pre-defined exception**

- **User-defined exception คือ class ที่ผู้ใช้สร้างขึ้นเองโดยการขยายมาจาก exception class (exception แบบนี้ผู้ใช้จะต้องโยนออกมาเองโดยใช้คำสั่ง throw)**



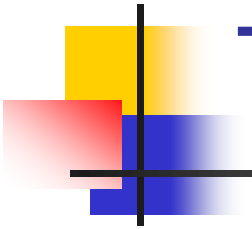
Exception Handling in Java

แสดงการอ้างถึงสมาชิกนอกขอบเขต Array

```
// ArrayOut1.java
class ArrayOut1 {
    public static void main(String args[]) {
        System.out.println("Hello! "+ args[0]);
    }
}
```

เมื่อเราเรียก **Java Interpreter** โดยที่ไม่มีการใส่ **argument** ใด ๆ เข้าไปในโปรแกรมนี้ จะได้ผลลัพธ์ดังนี้

```
java.lang.ArrayIndexOutOfBoundException : 0
        at ArrayOut1.main(ArrayOut1.java.java:5)
```



Try-Catch Blocks

- Try-Catch Block แบ่งเป็น 3 ส่วน (ตามลำดับ) ดังนี้
  - Try block
  - Catch block
  - Finally block
- Try-Catch Block มีโครงสร้างดังนี้

```
try { /* statements in try block */ }
[catch (Throwable t) { /* statement in catch block */ }]
[finally {  /* statement in finally block */ }]
```

Try-Catch Blocks

- Try block คือ block ที่บรรจุคำสั่งทั่ว ๆ ไป ซึ่งประโยคเหล่านี้อาจส่ง instance ของ exception class ออกมาในกรณีที่เกิดความผิดปกติ
- Catch Block คือ block ที่บรรจุคำสั่งที่ต้องการจะให้กระทำในกรณีที่คำสั่งใน try block เกิดความผิดปกติ
- Catch Block จะมี parameter เป็น Reference เอาไว้สำหรับรับ Instance ของ Exception ที่ถูกส่งออกมาจาก Try Block
- Finally Block คือ block ที่บรรจุคำสั่งที่ต้องการจะให้กระทำทุกครั้ง(ไม่ว่าโปรแกรมจะผ่านหรือไม่ผ่าน try, catch block ก็ตาม

Try-Catch Blocks

แสดงการใช้ **Try-Catch Block** ในการจับ **ArrayIndexOutOfBoundException**

```
// ArrayOut2.java
class ArrayOut2 {
    public static void main(String args[]) {
        try {
            System.out.println("Hello!  "+ args[0]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Hello! whoever you are. ");
        }
        System.out.println("How are you today? ");
    }
}
```

ถ้าป้อน **argument** เป็น **john** จะได้ผลลัพธ์

Hello! John
How are you today?

ถ้าไม่ป้อน **argument** จะได้ผลลัพธ์

Hello! Whoever you are
How are you today?

Try-Catch Blocks

แสดงการใช้ **Try-Catch Block** ในการจับ **Exception** หลายประเภท

```
class DivByZero {
    public static void main(String args[]) {
        try {
            int i1 = Integer.parseInt(args[0]);
            int i2 = Integer.parseInt(args[1]);
            System.out.println(i1 / i2);
        } catch (NumberFormatException e) {
            System.out.println(e.getMessage());
        } catch (ArithmeticException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- **Argument** ไม่สมบูรณ์จะเกิดอะไรขึ้น
- **parseInt** ทำงานไม่สมบูรณ์แบบจะเกิดอะไรขึ้น
- **i2** เป็น **0** จะเป็นอย่างไร

Finally Blocks

```
class FinallyTest {
    public static void main(String args[]) {
        try {
            if (args[0].equals("John"))
                return;
            System.out.println("Hello!  "+ args[0]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Hello! whoever you are.");
        }
        finally {
            System.out.println("How are you today?");
        }
        System.out.println("It's nice to see you.");
    }
}
```

ถ้าไม่ป้อน **argument** จะได้ผลลัพธ์

Hello! Whoever you are
How are you today?
It's nice to see you.

ถ้าป้อน **argument** เป็น **Joe** จะได้ผลลัพธ์

Hello! Joe
How are you today?
It's nice to see you.

ถ้าป้อน **argument** เป็น **john** จะได้ผลลัพธ์

How are you today?

Method Throwing Exception

```
class Throws {                          // เป็นเพียง Keyword ที่บอกให้รู้เท่านั้น
    static int div(int x, int y) throws ArithmeticException {
        return x / y;
    }
    public static void main(String args[]) {
        try {
            int i1 = Integer.parseInt(args[0]);
            int i2 = Integer.parseInt(args[1]);
            System.out.println(div(i1, i2));
        } catch (ArithmeticException e) {
            System.out.println(e);
        }
    }
}
```

Throw Statements

```
class ThrowTest1 {
    static int div(int x, int y) {
        try {
            if ( y == 0)
                throw new ArithmeticException();
            return x / y;
        } catch (ArithmeticException e) {
            return Integer.MAX_VALUE;
        }
    }
    public static void main(String args[]) {
        System.out.println(div(1,0));
    }
}
```

Throw Statements

ปกติเรานิยมที่จะโยน **Exception** ที่เกิดขึ้นออกจาก **method** และไปดักจับ **Exception** ที่ **caller method** ซึ่งจะทำให้ **method** ที่ทำงานจริงไม่ต้องมาสนใจกับการตรวจจับ **Exception**

```
class ThrowTest2 {
    static int div(int x, int y)throws ArithmeticException {
        if (y == 0)
            throw new ArithmeticException();
        return x / y;
    }
    public static void main(String args[]) {
        try {
            System.out.println(div(1,0));
        } catch (ArithmeticException e) {
            System.out.println(Integer.MAX_VALUE);
        }
    }
}
```