# บทที่ 9

# Exception

สอนโดย อ.เรวดี พิพัฒน์สูงเนิน สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ มหาวิทยาลัยราชภัฏอุดรธานี

# Exception

- ในการทำงานของโปรแกรม จะมีโอกาสเกิดข้อผิดพลาดขณะที่ทำการรันโปรแกรมอยู่ ข้อผิดพลาด พวกนี้อาจจะทำให้โปรแกรมหยุดการทำงาน เช่น การหารด้วย ศูนย์ การใช้ข้อมูลผิดประเภท หรือแม้แต่การเปิดไฟล์ที่ไม่มีอยู่จริงในเครื่องคอมพิวเตอร์ เป็นต้น
- ในการแก้ปัญหาโดยทั่วไปอาจจะทำการตรวจสอบข้อผิดพลาดเหล่านี้ ก่อนที่จะทำงานบางอย่างเพื่อไม่ให้โปรแกรมหยุดการทำงาน

# Exception

- วิธีดั้งเดิมในการดักจับข้อผิดพลาด

```
public boolean withdraw(double amount)
{   if (amount > balance)
    {   return false;
    }
    balance = balance - amount;
    return true;
}
```

ต้องการถอนเงิน แต่เงินที่ต้องการถอนมากกว่าเงินที่มี

- ปัญหาที่อาจพบคือ ผู้เขียนโปรแกรมมักจะลืมเขียนโปรแกรมดัก error
- ปัญหาอีกอย่างคือ การเขียนแบบนี้ก็ไม่สามารถทำอะไรเกี่ยวกับความล้มเหลวที่พบ ดังนั้นอาจมีเขียนโค๊ตเพิ่มเติม

```
System.out.println("Not enough balance");
```

# Exception

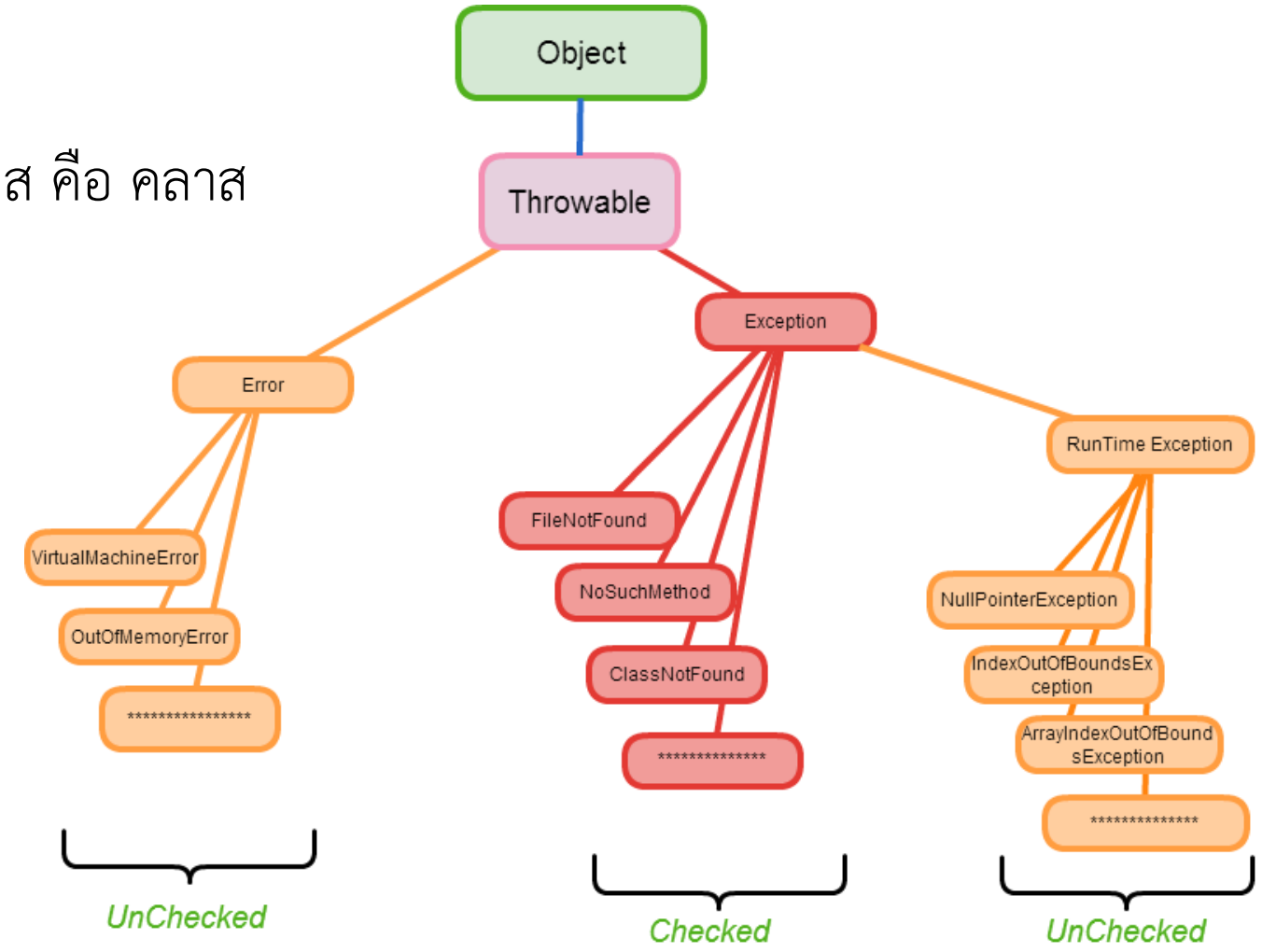
- แต่สำหรับในภาษา Java มีการใช้งานการจับข้อผิดพลาด นั่นคือ การนำ Exception มาช่วยตรวจสอบและดักจับข้อผิดพลาด โดยการดักจับข้อผิดพลาดจะไม่เข้าไปเป็นส่วนหนึ่งของการทำงานหลักของโปรแกรม จะเป็นการครอบการดักจับดังนั้น สามารถแก้ไขและตรวจสอบได้ง่าย จึงเป็นที่นิยมมาก

# Exception

- Exception คือการที่โปรแกรมพยายามจะทำงานบางอย่าง แต่เกิดข้อผิดพลาดขึ้น แล้วโปรแกรมไม่สามารถจัดการข้อผิดพลาดนั้นได้ ซึ่งทำให้เกิด exception ขึ้น และส่งผลทำให้โปรแกรมหยุดทำงาน
- ในขณะที่รันโปรแกรมตรวจจับความผิดพลาดขณะรันโปรแกรม ภาษา Java มีการดักจักด้วยการโยน (throws) ความผิดพลาดที่เกิดขึ้นให้กับชุดการดักจับข้อผิดพลาด

# Exception

- คลาส Throwable
- คลาส Throwable มีคลาสลูก 2 คลาส คือ คลาส Error และคลาส Exception



# ความแตกต่างระหว่าง Error และ Exception

- **Error**
  - คือ ความผิดพลาดที่เกิดจาก Syntax error คือ การเขียนชุดคำสั่งผิด ตรวจสอบได้ด้วย Compiler
  - หรืออาจจะเกิดจาก Logic Error คือ การที่ใช้ตรรกะผิดวัตถุประสงค์ ซึ่ง Error นี้เกิดจากความเข้าใจผิดของผู้เขียนโปรแกรมเอง ตรวจพบได้ยาก
  - compiler ไม่สามารถตรวจสอบได้

# ความแตกต่างระหว่าง Error และ Exception

- **Exception**
  - คือความผิดพลาดที่ไม่ร้ายแรง ทำให้การทำงานของโปรแกรมไม่เป็นไปตามขั้นตอนที่กำหนดไว้
  - ข้อผิดพลาดที่เกิดขณะรันโปรแกรม อาจจะเกิดจากการหารด้วย 0 หรือ การเปิดไฟล์ที่ไม่มีอยู่จริง หรือการกรอกข้อมูลผิดประเภท
  - เป็นข้อผิดพลาดที่คนเขียนโปรแกรมไม่สามารถทราบได้ขณะที่กำลังพัฒนาโปรแกรม
  - ตรวจสอบได้ยากเนื่องจากต้องทดลองการใช้งานโปรแกรมจึงจะตรวจสอบได้ แต่สามารถป้องกันได้

# Exception

exception ก็มีคลาสลูกอีกมากมาย ซึ่งแบ่งได้ 2 กลุ่ม คือ

- Checked Exception
  - เป็น exception ที่คอมไพเลอร์สามารถตรวจสอบพบได้ในช่วงคอมไพล์โปรแกรม จึงต้องมีการจัดการ ไม่เช่นนั้นจะคอมไพล์ไม่ผ่าน
- Unchecked Exception
  - เป็น exception ที่คอมไพเลอร์ไม่สามารถตรวจสอบพบได้ตอนคอมไพล์โปรแกรม แต่จะตรวจสอบพบได้ในช่วงรันโปรแกรม (run time)
  - เป็น subclass ของคลาส RuntimeException ซึ่งเราอาจจะไม่จัดการกับมันก็ได้

# ตัวอย่าง โปรแกรมที่ไม่มีการจัดการข้อผิดพลาด

- ตัวอย่าง โปรแกรม DivideByZero มีเมธอดชื่อ quotient() รับค่าตัวเลขจำนวนเต็มและส่งค่ากลับเป็นผลลัพธ์จากการหาร ข้อผิดพลาดที่อาจจะเกิดขึ้นระหว่างการทำงานคือ หากเลขตัวหารเป็น 0 จะทำให้เกิดข้อผิดพลาด แต่ในตัวอย่างไม่มีการจัดการกับข้อผิดพลาดที่เกิดขึ้น หรือหากป้อนข้อมูลที่ไม่ตัวเลขก็จะเกิดข้อผิดพลาด

```java
import java.util.Scanner;

public class DivideByZero
{
    public static int quotient(int num , int denominator)
    {
        return num / denominator ; //possible division by zero
    }

    public static void main(String args[])
    {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter integer number => ");
        int num = input.nextInt();
        System.out.print("Enter integer denominator => ");
        int deno = input.nextInt();

        int result = quotient(num , deno);
        System.out.printf("Result of : %d / %d = %d\n" ,num,deno,result);
    }
}
```

# ตัวอย่างผลลัพธ์

```
Enter integer number => 100
Enter integer denominator => 7
Result of : 100 / 7 = 14
```

```
Enter integer number => 100
Enter integer denominator => 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at DivideByZero.quotient(DivideByZero.java:7)
    at DivideByZero.main(DivideByZero.java:19)
```

```
Enter integer number => 100
Enter integer denominator => zero
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:864)
    at java.util.Scanner.next(Scanner.java:1485)
    at java.util.Scanner.nextInt(Scanner.java:2117)
    at java.util.Scanner.nextInt(Scanner.java:2076)
    at DivideByZero.main(DivideByZero.java:17)
```

# โปรแกรม IndexOutOfBound เป็นโปรแกรมแสดงข้อมูลในอาร์เรย์ โดยรับหมายเลขสมาชิก

```
import java.util.Scanner ;

public class IndexOutOfBound
{
    public static void main(String[] args)
    {
        String member[] = {"Rewadee" ,"Kunawut" , "Khwanchai",
                           "Phanupan" , "Vilaiporn" };

        Scanner input = new Scanner(System.in);

        System.out.print("Enter member no.(1-5) => ");
        int no = input.nextInt();

        while(no > 0) {
            System.out.printf("Member no.%d is %s\n",no,member[no-1]);
            System.out.println("----------------------------------------");
            System.out.print("Enter member no.(1-5) => ");
            no = input.nextInt();
        }
    }
}
```

# ตัวอย่างผลลัพธ์

```
Enter member no.(1-5) => 1
Member no.1 is Rewadee
---------------------------------------
Enter member no.(1-5) => 3
Member no.3 is Khwanchai
---------------------------------------
Enter member no.(1-5) => 5
Member no.5 is Vilaiporn
---------------------------------------
Enter member no.(1-5) => 7
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 6
    at IndexOutOfBound.main(IndexOutOfBound.java:16)
```

```java
import java.io.File;
import java.io.FileReader;

public class FileNotFound {

    public static void main(String args[]) {
        File file = new File("E://file.txt");
        FileReader fr = new FileReader(file);
    }
}
```

```
D:\Java_code\exception\FileNotFound.java:8: error: unreported exception FileNotFoundException; must be caught or declared to be thrown
      FileReader fr = new FileReader(file);
                      ^
1 error
```

# การจัดการข้อผิดพลาด(Exception handling)

- การจัดการกับข้อผิดพลาดใน ภาษา Java เราจำเป็นต้องเขียนโปรแกรมเพื่อจัดการกับข้อผิดพลาดเหล่านั้นที่เกิดขึ้น ว่าจะให้โปรแกรมทำงานอย่างไรต่อหากมีข้อผิดพลาดเกิดขึ้น
- การใช้งานการดักจับข้อผิดพลาดมีใช้งาน 2 คำสั่งนั่นคือ
  - try catch
  - throws

# การใช้ throws

- เป็นคำสั่งส่งข้อผิดพลาดออกไปจัดการด้วย objectที่ต้องการ ซึ่งมีรูปแบบการใช้งานดังนี้

```
[modifier] returntype MethodName([parameter]) throws Exception1,[Exception2] {
        ชุดคาสั่งใน Method
}
```

- สังเกตว่า การใช้ throws จะเป็นการโยนข้อผิดพลาดไปให้ผู้ที่เรียกใช้ Method นั้นๆ ซึ่งการใช้ throws เป็นการโยนความรับผิดชอบไปให้ผู้ใช้งาน ดังนั้นผู้ใช้งานต้องทำการตรวจจับข้อผิดพลาดนั้น ๆ เองโดยใช้ try catch

# การใช้ throws

- throws (มี s) : ใช้ระบุว่าเมธอดนี้ อาจจะส่ง exception
- throw (ไม่มี s) : ใช้ส่ง exception

```
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class FileNotFound {

   public static void main(String args[]) throws IOException {
      File file = new File("E://file.txt");
      if (!file.exists()) {
            throw new IllegalArgumentException("File not found!");

      }
      FileReader fr = new FileReader(file);
   }
}
```

**ตัวอย่างผลลัพธ์**

```
Exception in thread "main" java.lang.IllegalArgumentException: File not found!
    at FileNotFound.main(FileNotFound.java:10)
```

# การใช้งาน try...catch

- การใช้งาน try catch เป็นการใช้งานคล้ายกับการใช้ if statement คือ ถ้าเกิดข้อผิดพลาดขณะที่โปรแกรมทำงานในส่วนที่ try ดูแลอยู่ โปรแกรมจะตรวจสอบที่คำสั่ง catch ว่าเป็นข้อผิดพลาดอะไร และทำงานให้ตรงการคำสั่ง catch ที่ตรงกับข้อผิดพลาดนั้น

```
try{
        ชุดคำสั่งที่อาจจะเกิดข้อผิดพลาดได้;
}catch (TheException){
        ชุดคำสั่งที่ทำงานในกรณีที่เกิดข้อผิดพลาด;
}finally{
        ชุดคำสั่งที่ทำงานปรกติไม่ว่าจะมี Exception เกิดหรือไม่ก็ตาม
}
```

# การใช้งาน try...catch

- ในบล็อคคำสั่งของ try เป็นส่วนของโปรแกรมที่อาจจะทำให้เกิด exception ขึ้น และในแต่ละ catch บล็อคเป็นการจัดการกับ exception แต่ละแบบ และบล็อค finally โปรแกรมจะเข้ามาทำงานเสมอไม่ว่าจะเกิด exception ในบล็อคของคำสั่ง try หรือไม่ก็ตาม
- ในภาษา Java นั้นมีไลบรารี่มาตรฐานของ exception ที่สามารถให้เราจัดการกับข้อผิดพลาดประเภทต่างๆ ได้ ซึ่ง extends มาจากคลาส Exception สำหรับในบทเรียนนี้ จะยกตัวอย่างการใช้ในบางส่วนเท่านั้น

# ตัวอย่างการใช้ try catch ดักจับและจัดการ Ex

```java
import java.util.Scanner;

public class DivideByZero {
    public static int quotient(int num , int denominator) {
        return num / denominator ; //possible division by zero
    }

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter integer number => ");
        int num = input.nextInt();
        System.out.print("Enter integer denominator => ");
        try {
            int deno = input.nextInt();
            int result = quotient(num , deno);
            System.out.printf("Result of : %d / %d = %d\n" ,num,deno,result);
        }catch (ArithmeticException ex) {
            System.out.println("Zero is an invalid denominator!!!");
            System.out.println("Can not divide by zero...");
        }
    }
}
```

# ตัวอย่างผลลัพธ์

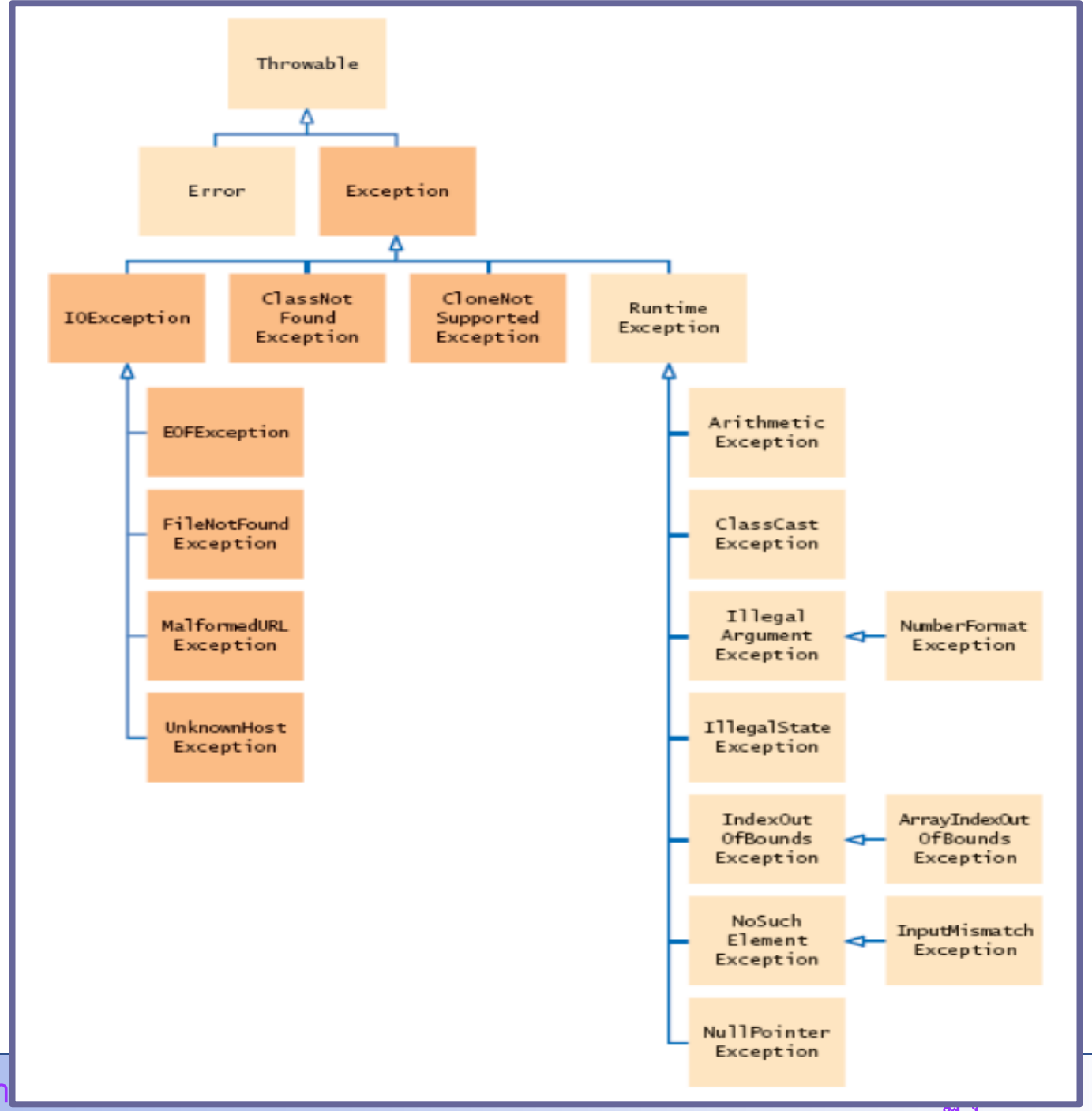
```
Enter integer number => 100
Enter integer denominator => 20
Result of : 100 / 20 = 5
```

```
Enter integer number => 100
Enter integer denominator => 0
Zero is an invalid denominator!!!
Can not divide by zero...
```

# การใช้งาน try...catch

- การจัดการกับข้อผิดพลาด หลายๆ ประเภท เราสามารถมีชุดของ catch ได้มากกว่า 1 ชุด
- ในกรณีที่มีการจัดการกับข้อผิดพลาดมากกว่า 1 ชุด จะทำการตรวจสอบชนิดของข้อผิดพลาดจากคลาสที่มีการเรียงตามลำดับการสืบทอด
- ดังนั้นการดักจับข้อผิดพลาดควรเรียงจากลำดับ Subclass ไปหา Superclass ของ Exception

# Hierarchy of Exception class



```java
import java.util.Scanner;
import java.util.InputMismatchException;

public class DivideByZero {
    public static int quotient(int num , int denominator) {
        return num / denominator ; //possible division by zero
    }

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        try {
            System.out.print("Enter integer number => ");
            int num = input.nextInt();
            System.out.print("Enter integer denominator => ");

            int deno = input.nextInt();
            int result = quotient(num , deno);
            System.out.printf("Result of : %d / %d = %d\n" ,num,deno,result);
        }
        catch(InputMismatchException ex1) {
            input.nextLine();
            System.out.println("You must enter integer....");
        }
        catch(ArithmeticException ex2) {
            System.out.println("Zero is an invalid denominator!!!");
            System.out.println("Can not divide by zero...");
        }
    }
}
```

# ตัวอย่างผลลัพธ์

```
Enter integer number => 100
Enter integer denominator => 7
Result of : 100 / 7 = 14
```

```
Enter integer number => 100
Enter integer denominator => 0
Zero is an invalid denominator!!!
Can not divide by zero...
```

```
Enter integer number => hello
You must enter integer....
```

```
Enter integer number => 100
Enter integer denominator => nine
You must enter integer....
```

# การใช้ try...catch ร่วมกับ throws

- เมธอด go() ของคลาส A จะโยน exception เมื่อพารามิเตอร์ n < 10

```
class A {
    void go(int n) throws Exception {
        if (n < 10) {
            throw new Exception();
        }
        System.out.println(n);
    }
}
public class Test {
    public static void main(String[] args) {
        try {
            A a = new A();
            a.go(5);
        } catch (Exception ex) {
            System.out.println("catch");
        }
    }
}
```

**ตัวอย่างผลลัพธ์**

```
catch
```

# การใช้ try...catch ร่วมกับ throws

- เมธอดสามารถโยน exception ได้มากกว่าหนึ่งชนิด

```
import java.io.*;
import java.sql.*;
import java.util.*;

class A {
    void go(int n) throws IOException, SQLException {
        switch (n) {
            case 0:
                System.out.println();
                break;
            case 1:
                throw new IOException();
            case 2:
                throw new SQLException();
            default:
                throw new IllegalArgumentException();
        }
    }
}
```

# การใช้ try...catch ร่วมกับ throws

- เมธอดสามารถโยน exception ได้มากกว่าหนึ่งชนิด

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

        Scanner input = new Scanner(System.in);
        System.out.print("Enter menu ==> ");
        int n = input.nextInt();
        while(n>0){
            try {
                System.out.print(n);
                A a = new A();
                a.go(n);
            } catch (IOException ex) {
                System.out.println(" catch io exception");
            } catch (SQLException ex) {
                System.out.println(" catch sql exception");
            } catch (Exception ex) {
                System.out.println(" catch exception");
            }

            System.out.print("Enter menu ==> ");
            n = input.nextInt();
        }
    }
}
```

# ตัวอย่างผลลัพธ์

```
Enter menu ==> 1
1 catch io exception
Enter menu ==> 2
2 catch sql exception
Enter menu ==> 3
3 catch exception
Enter menu ==> 4
4 catch exception
Enter menu ==> 0
```

# การใช้งาน finally{}

- ในบล็อคของคำสั่ง finally นั้นจะประมวลผลเสมอ ไม่ว่าจะเกิด exception หรือไม่ มันใช้สำหรับการทำงานที่สำคัญ เช่น การปิดไฟล์ หรือปิดการเชื่อมต่อของ stream

```
do{
    try {
        System.out.print("Enter integer number => ");
        int num = input.nextInt();
        System.out.print("Enter integer denominator => ");

        int deno = input.nextInt();
        int result = quotient(num , deno);
        System.out.printf("Result of : %d / %d = %d\n" ,num,deno,result);
        loop = false;
    }
    catch(InputMismatchException ex1) {
        input.nextLine();
        System.out.println("You must enter integer....");
    }
    catch(ArithmeticException ex2) {
        System.out.println("Zero is an invalid denominator!!!");
        System.out.println("Can not divide by zero...");
    }
    finally {
        System.out.println("-----------------------------------");
    }
}while(loop);
```

# จบการนำเสนอ

# END.

# Q & A