2110101 : การทำโปรแกรมคอมพิวเตอร์

เรื่องต้องรู้ (บ้าง)

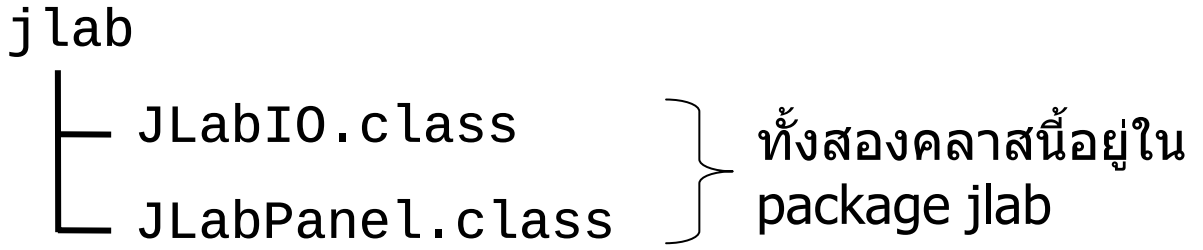
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

หัวข้อ

- packages, import
- private, public, final, static
- Error & Exception Handling (Optional)

Package

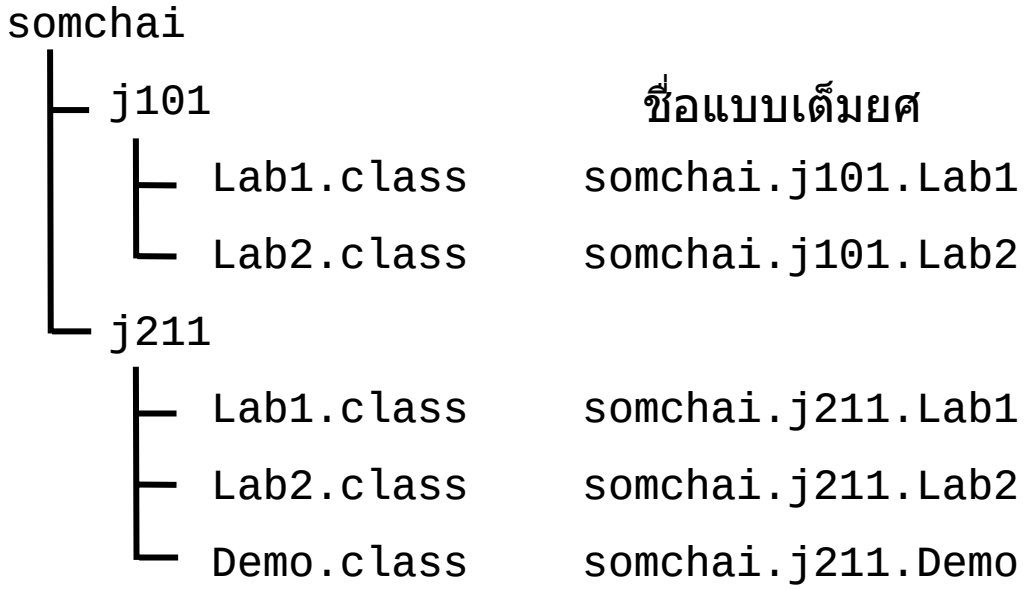
- โปรแกรมจาวาประกอบด้วยคลาสมากมาย ทั้งที่เขียนเอง ของระบบจาวา หรือของคนอื่น
- จะจัดเก็บคลาสต่างๆ อย่างไร ที่ไหน และมีวิธีป้องกันชื่อซ้ำได้อย่างไร
- จัดเก็บคลาสต่างๆ ในรูปของ package
- คลาสที่อยู่ใน package เดียวกัน คืออยู่ใน directory เดียวกัน



ชื่อ package มักขึ้นด้วยตัวเล็ก

Package

- Package มีลำดับชั้น : มี subdirectory ได้
- ต้องระบุชื่อ package ในโปรแกรมด้วย



	ชื่อแบบเต็มยศ
Lab1.class	somchai.j101.Lab1
Lab2.class	somchai.j101.Lab2
Lab1.class	somchai.j211.Lab1
Lab2.class	somchai.j211.Lab2
Demo.class	somchai.j211.Demo

```
package somchai.j101;

public class Lab1 {
  ...
```

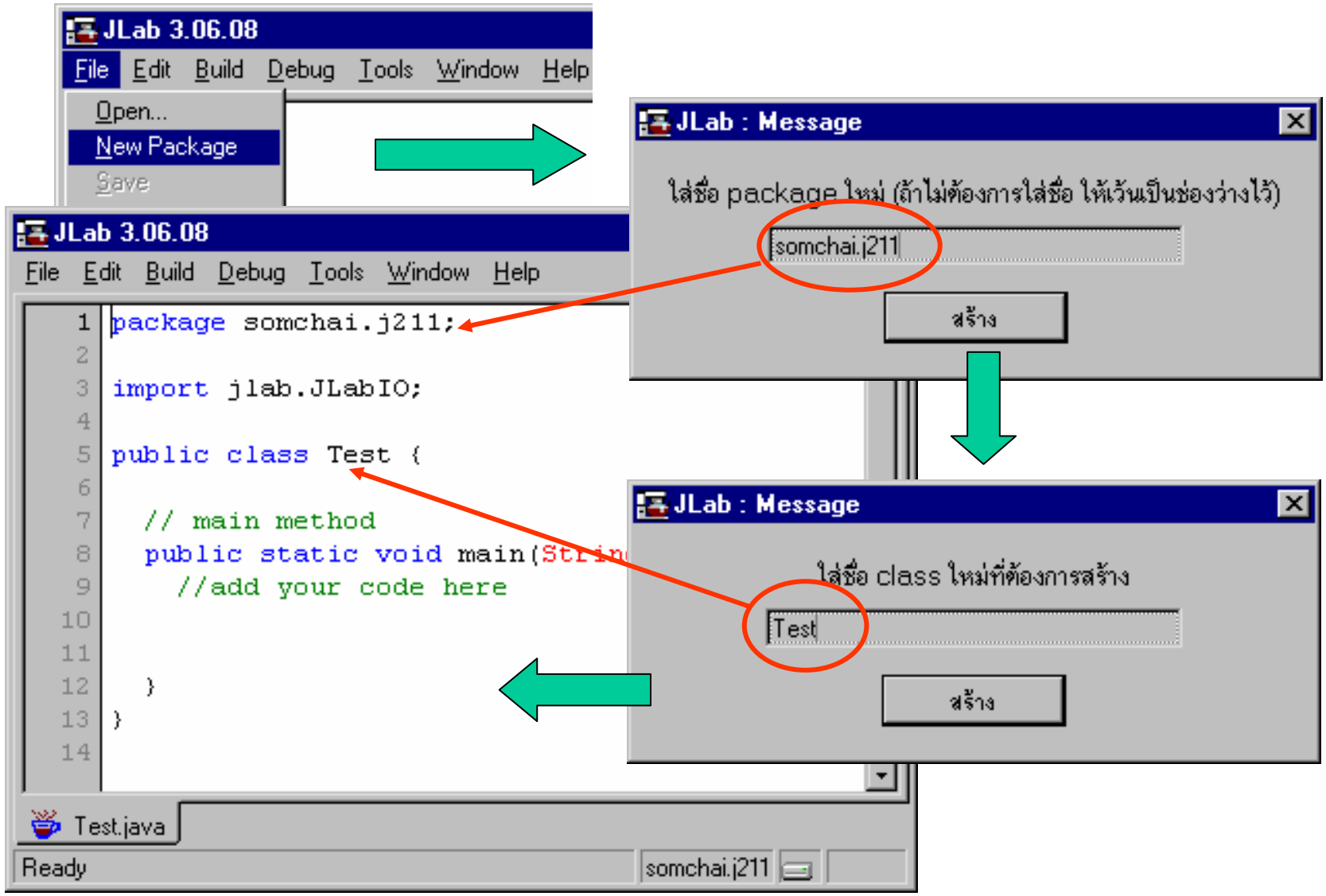
```
package somchai.j211;

public class Lab1 {
  ...
```

อ้างอิงชื่อคลาสแบบเต็มยศ

```
public class Other {
  ...
  somchai.j101.Lab1  lab101;
  somchai.j211.Lab1  lab211;
  ...
```

การสร้าง Package ใน JLab



import

- การใช้คลาสที่อยู่นอก package ของตัวเอง
  - เขียนชื่อ class รวมทั้ง package ให้ครบถ้วน
  - หรือใช้ import ก่อน แล้วอ้างเฉพาะชื่อคลาสได้เลย
    - import แบบระบุชื่อคลาส
    - import แบบเหมาหมดทุกคลาส

```
import jlab.JLabIO;

public class Happy {
  public void A() {
    int x = JLabIO.readI
    ...
```

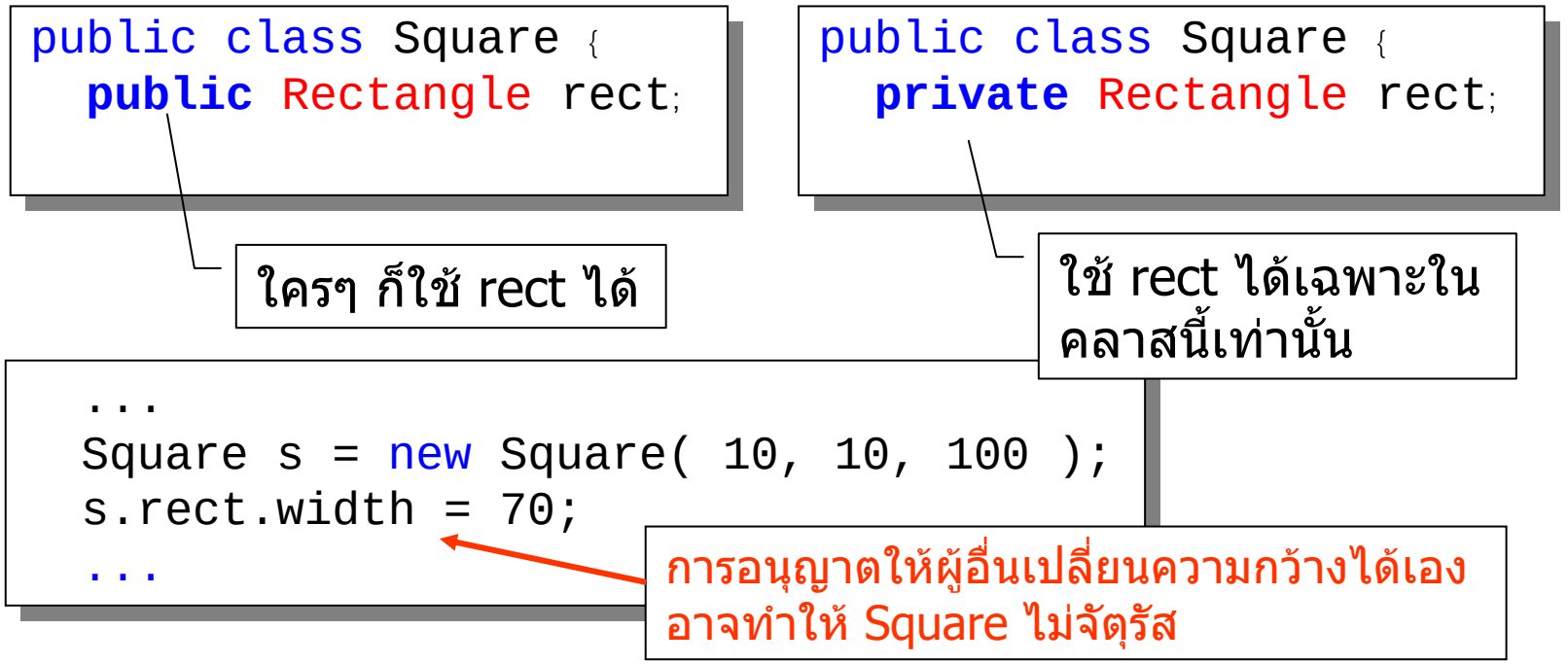
```
import jlab.*;

public class Happy {
  public void A() {
    int x = JLabIO.readInt();
```

```
public class Happy {
  public void A() {
    int x = jlab.JLabIO.readInt();
    ...
  }
  ...
```

การควบคุมการใช้ fields และ methods

- ใช้ public กำกับ
  - เปิดเผย : คลาสอื่นๆ ใช้ได้หมด
- ใช้ private กำกับ
  - ปกปิด : ใช้ได้เฉพาะภายในคลาสเจ้าของเท่านั้น



ข้อแนะนำ

- ให้ทุกๆ fields เป็น private
- มี public method ให้หยิบหรือเปลี่ยนค่าของ field ได้

```
public class Square {
  private Rectangle rect;

  public Square(int x, int y, int len) {
    this.rect = new Rectangle(x, y, len, len);
  }
  public int getLength( ) {
    return this.rect.width;
  }
  public void setLength(int len) {
    if (len > 0)
      this.rect.width = this.rect.height = len;
  }
  ...
```

ปลอดภัย และตรวจหาจุดบกพร่องของโปรแกรมได้ง่ายขึ้น

ยังมีแบบอื่น

- นอกจาก private และ public แล้วยังมี
  - protected อนุญาตให้เฉพาะคลาสใน package เดียวกัน และ subclass เท่านั้นที่ใช้ protected ของคลาสอื่น
  - ไม่ต้องใส่อะไรกำกับ อนุญาตให้เฉพาะคลาสใน package เดียวกันเท่านั้น

```
package mars;
public class A {
  protected int count;
  ...
```

```
package venus;
public class C {
  protected int number;
  ...
```

```
package earth;
public class B extends mars.A {

  public void retire( ) {
    super.count = 5;     // OK
    C c = new C();
    c.number = 7;        // fail
    ...
```

final

- เติม final หน้า field
  - ไม่ต้องการให้เปลี่ยนแปลงค่าได้ (constant)
- เติม final หน้า method
  - ไม่ต้องการให้ subclass override ได้
- เติม final หน้า class
  - ไม่ต้องการให้ใคร extends ได้

มักตั้งชื่อ final field ด้วยตัวอักษรตัวใหญ่ทุกตัว

```
final String COMPUTER = "cp";
final String ELECTRICAL = "ee";
final String CIVIL = "ce";

...
```

static

- เติม static หน้า method ทำให้เป็น class method
- เติม static หน้า field ทำให้เป็น class data
  - ที่เก็บไม่ได้อยู่ประจำแต่ละออบเจกต์ที่สร้าง
  - แต่อยู่ประจำคลาสเจ้าของ ดังนั้นจึงมีเพียงที่เก็บเดียว

```
public class Undergrad {
  public static final int FRESHY = 1;
  public static final int SOPHOMORE = 2;
  public static final int JUNIOR = 3;
  public static final int SENIOR = 4;
  private int status;
  public void setStatus( int s ) {
    switch( s ) {
      ...
      this.status = s;
      ...
    }
  }
```

```
Undergrad s = new Undergrad();
s.setStatus( Undergrad.FRESHY );
...
```

Error และ Exception

- การทำงานของโปรแกรมมีโอกาสเกิดความผิดพลาดหรือพบสิ่งผิดปกติ
- ความผิดปกติอาจเกิดที่
  - ภายในตัวระบบจาวา รวมทั้งภายในเมท็อดของคลาสมาตรฐานของจาวา
    - หน่วยความจำหมด, อ้างอิง index ของอาเรย์นอกช่วงที่จองไว้, ...
  - ภายในเมท็อดของผู้เขียนเอง
    - ข้อมูลในตัวแปร ไม่เป็นไปตามที่หวัง (เช่นต้องไม่ติดลบ, ...)

Exceptions ที่เกิดขึ้นจากระบบจาวา

- การทำงานมีโอกาสผิดปกติ (Exception)

```
// ArithmeticException
int a = 4, b = 0;
int c = a / b;
```

```
// ArrayIndexOutOfBoundsException
int [] a = int[40];
a[40] = 20;
```

```
// StackOverflowError
void Jeng() {
 Jeng();
}
```

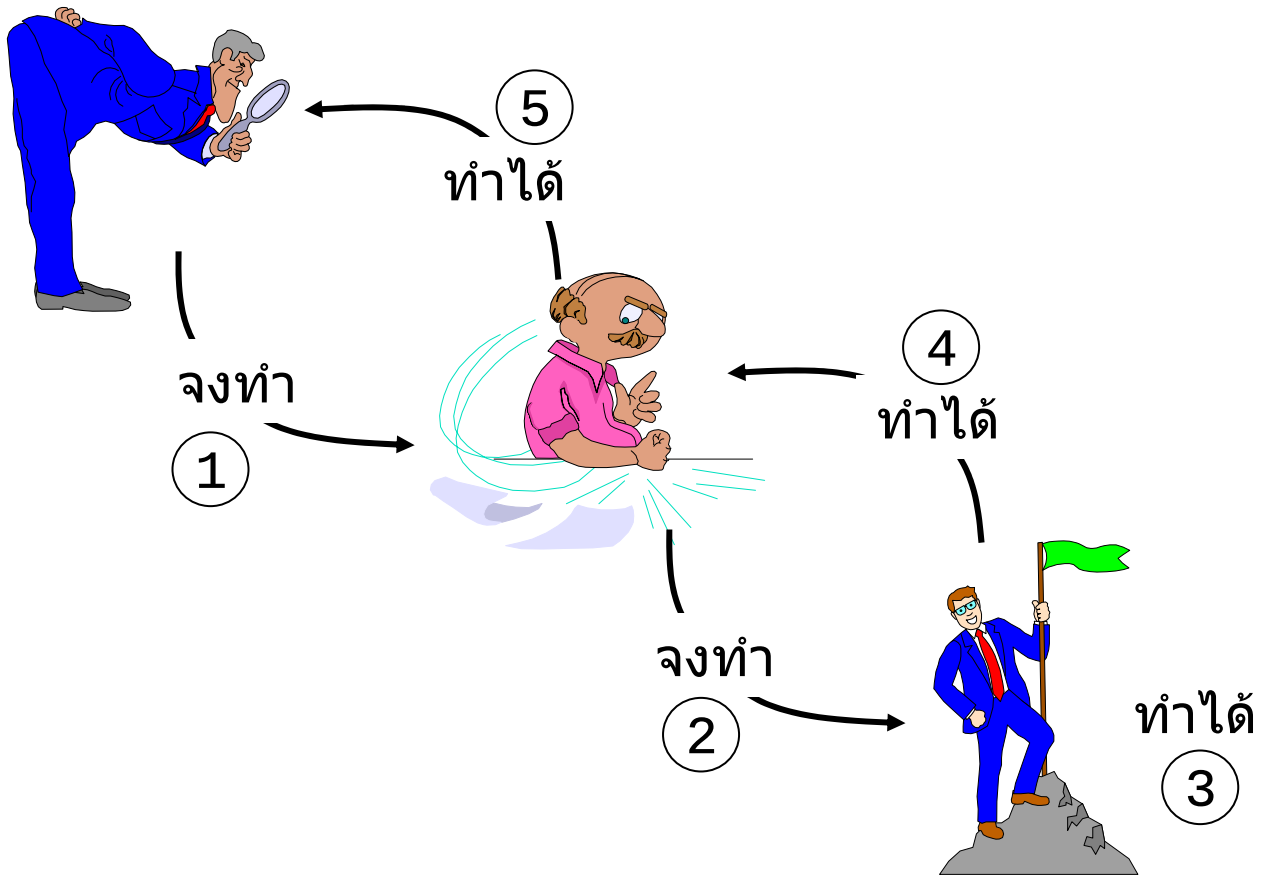
```
// ClassCastException
Object a = new Rectangle();
System.out.println((String) a);
```

```
// NegativeArraySizeException
int n = -100;
Vector [] a = new Vector[n];
```

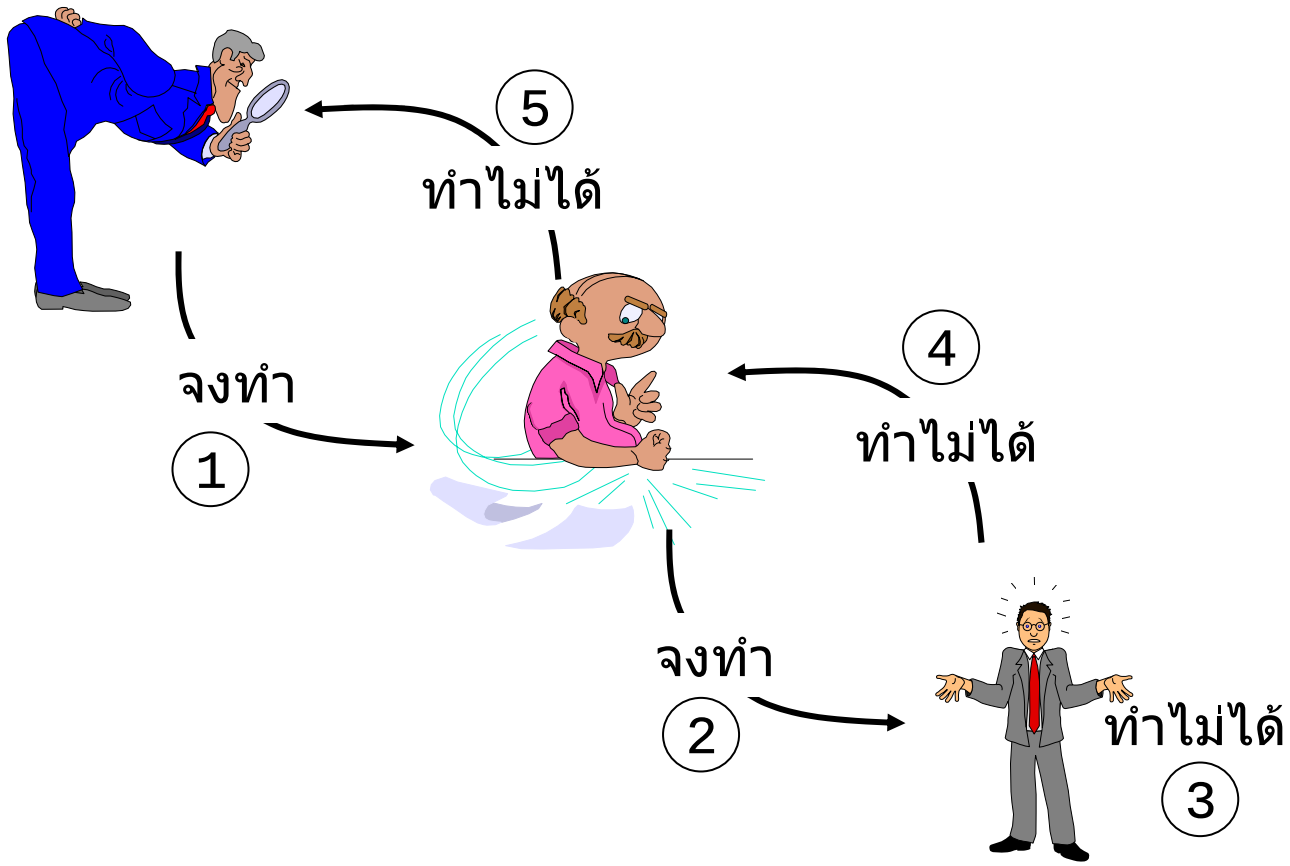
```
// ArrayStoreException
Object [] a = new String[5];
a[0] = new Rectangle();
```

```
 // FileNotFoundException
BufferedReader in = JLabIO.openFile( "C:\\DATA.IN" );
String txt = in.readLine();
```

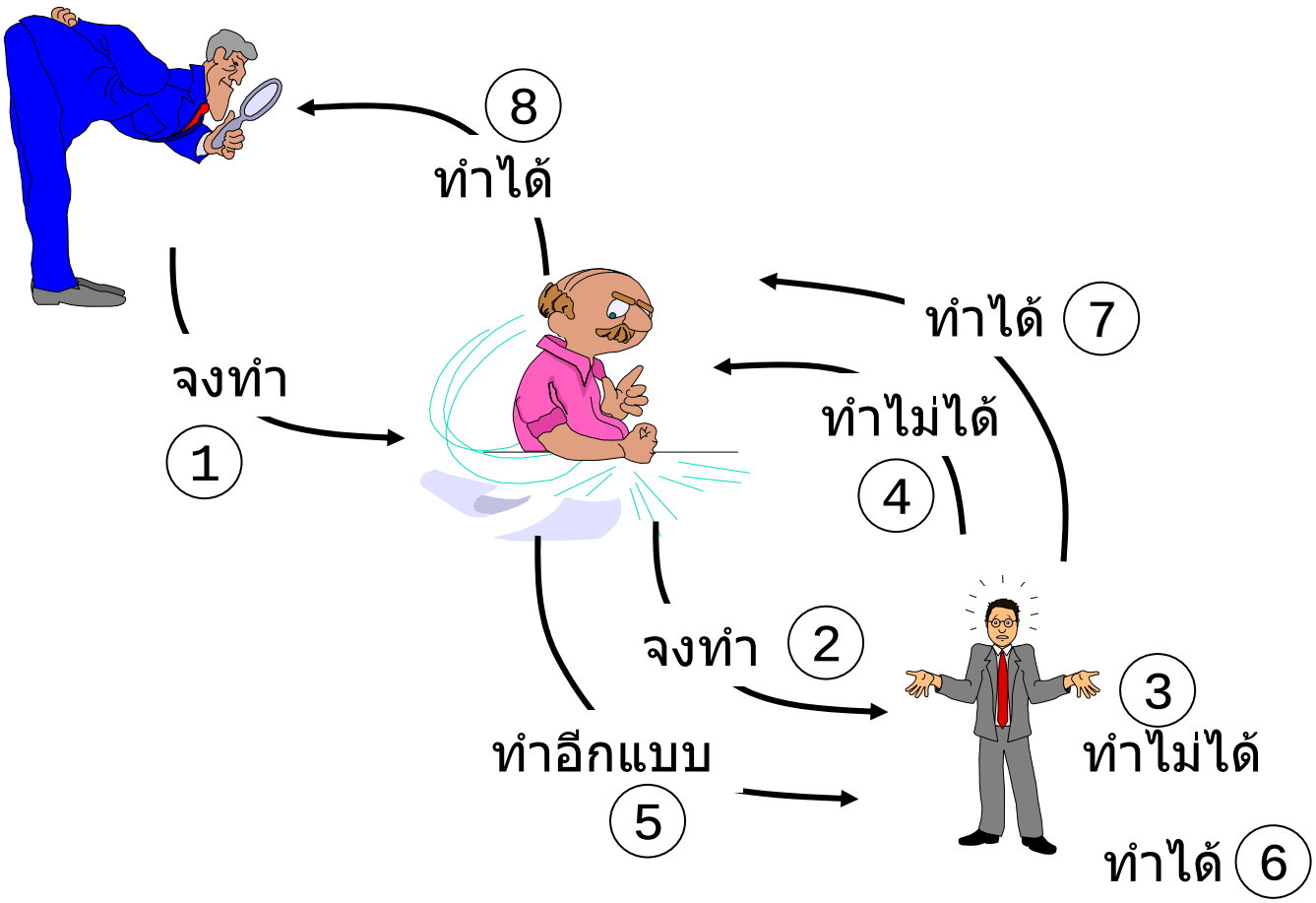
การเรียกเมท็อด



ผิดปกติแล้วไม่จัดการ



ผิดปกติต้องจัดการ



Robustness

- ถ้าเกิดความผิดปกติ แล้วไม่มีส่วนใดของโปรแกรมที่รองรับสถานะการณ์ที่เกิดขึ้น
  - ระบบจะแสดงข้อความอธิบายความผิดปกติออกจอภาพ
  - แล้วเลิกการทำงาน
- ระบบที่แข็งแกร่งควรจัดการกับความผิดปกติที่อาจเกิดขึ้นให้มากที่สุด โดยทำให้โปรแกรมยังทำงานต่อไปได้
- ตัวอย่าง : ต้องการเปิดแฟ้มข้อมูล แต่หาไม่พบ
  - แทนที่จะบอกผู้ใช้ว่า "file not found" แล้วเลิกการทำงาน
  - ก็บอกผู้ใช้ให้ใส่ชื่อแฟ้มใหม่ แล้วลองเปิดแฟ้มใหม่

Throwing Exceptions

- เมื่อเกิดความผิดปกติภายในเมท็อด และต้องการแจ้งให้เมท็อดที่เรียกมาให้ทราบ
  - สร้างออบเจกต์ของคลาสที่แทนความผิดปกติที่เกิดขึ้น
  - แล้ว "โยน" ออบเจกต์นั้นให้ผู้เรียกรับทราบ
- ที่หัวเมท็อดต้องระบุด้วยว่า เมท็อดนี้อาจมีการ "โยน" สิ่งผิดปกติประเภทใดบ้าง

```
public double [] qRoot( double a, double b, double c )
                  throws IllegalArgumentException {
  double t = b * b - 4 * a * c;
  if ( t < 0 ) {
    IllegalArgumentException e =
       new IllegalArgumentException("imaginary root");
    throw e;
  }
  ...
 }
```

$$\frac{-b+\sqrt{b^2-4ac}}{2a}$$

Exception Classes

- จาวาใช้คลาสเพื่อเขียนบรรยายความผิดปกติ
- ชื่อคลาสจะสื่อความหมายถึงประเภทของความผิด
  - IOException, OutOfMemoryError, ...

```
package java.lang;

public class IllegalArgumentException extends RuntimeException {
  public IllegalArgumentException() {
    super();
  }
  public IllegalArgumentException(String s) {
    super(s);   // s is detail exception message
  }
}
```

คลาส IllegalArgumentException มีแค่นี้เอง คลาสนี้มักใช้แทนความผิดปกติกรณีที่เมท็อดได้รับข้อมูลมาไม่ตรงตามที่ได้ตกลงกันไว้

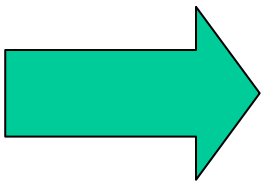
Catching Exception

- เมื่อใช้อะไรที่อาจ "โยน" สิ่งผิดปกติ
- ก็ต้องคอย "รับ" สิ่งผิดปกติด้วย
- ใช้คำสั่ง try-catch

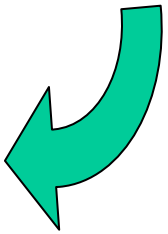
```
try {

  statement
  statement
    ...
  statement

} catch( Exception1 varName ) {
  statement
  ...
  statement
}
```



ถ้าข้อความสั่งใดโยนออบเจกต์ของคลาส Exception1 การทำงานจะกระโดดไปทำงานต่อที่ catch block



ตัวแปรนี้ก็คือตัวอ้างอิงออบเจกต์ที่ถูกโยนออกมาเนื่องจากเกิดสิ่งผิดปกติ

มีได้หลาย catch blocks

```
try {

  statement
    ...
  statement

} catch( Exception1 varName ) {
  statement
    ...
  statement
} catch( Exception2 varName ) {
  statement
    ...
  statement
} catch( Exception3 varName ) {
  statement
    ...
  statement
}
```

การทำงานจะกระโดดไปทำงานต่อที่ catch block ใด ก็ขึ้นกับว่าออบเจกต์ที่ถูกโยนมาตรงกับคลาสของ catch block ใด

ตัวอย่าง

- เมท็อด dumpFile อาจโยน IOException object

```
public void dumpFile(String fn) throws IOException {
  BufferedReader in = JLabIO.openFile(fn);
  String txt;
  while ( (txt = in.readLine()) != null ) {
    System.out.println(txt);
  }
  in.close();
}
```

```
do {
  try {
    String fn = JLabIO.readString("file = ");
    dumpFile(fn);
    success = true;
  } catch( IOException e ) {
    System.err.println(e.getMessage() + " : try again");
    success = false;
  }
} while ( !success );
```

ยังมีเรื่องจุกจิกอีกมาก

- ถ้าไม่ "รับ" ก็ต้อง "โยน" ต่อ
- คลาสของความผิดปกติทั้งหลาย extends มาจากคลาสมาตรฐาน 3 ประเภทคือ
  - Error, Exception และ RuntimeException
- ลำดับของ catch block มีความหมาย
- ไม่ต้อง catch ความผิดปกติที่ extends จาก RuntimeException ก็ได้
- ...