2110211 : โครงสร้างข้อมูล

แนะนำจาวาตอน ๓

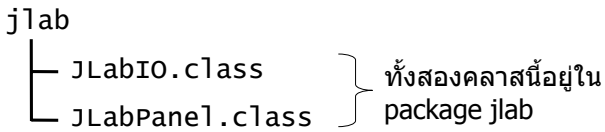
สมชาย ประสิทธิ์จูตระกูล
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

หัวข้อ

- Package, Import
- public, private
- final, static
- Inheritance
  - subclass และ superclass
  - Overriding vs. Overloading
  - class Object
  - Inheritance Hierarchy
  - upcasting และ downcasting

Package

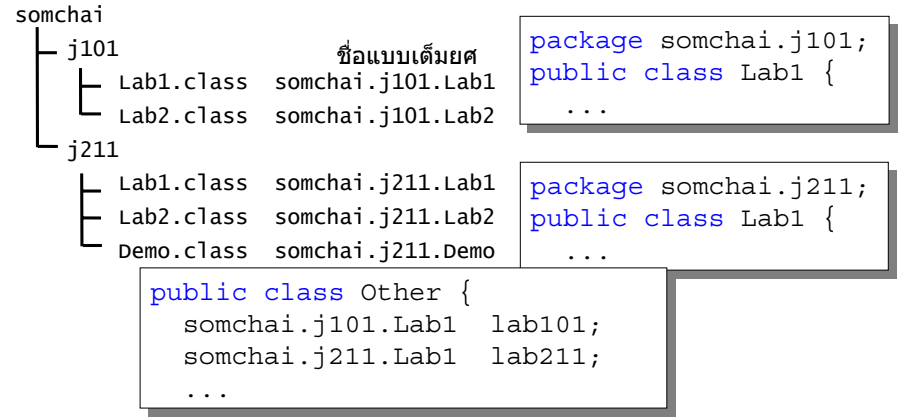
- โปรแกรมจาวาประกอบด้วยคลาสมากมาย ทั้งที่เขียนเอง ของระบบจาวา หรือของคนอื่น
- เราสามารถจัดเก็บคลาสต่างๆ ที่เกี่ยวข้องกันเป็นกลุ่มๆ เรียกว่า package
- คลาสที่อยู่ใน package เดียวกัน คืออยู่ใต้ directory เดียวกัน



ชื่อ package มักขึ้นด้วยตัวเล็ก

Package

- Package มีลำดับชั้นเป็น subdirectory ได้
- ต้องระบุชื่อ package ในโปรแกรมด้วย



import

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

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

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


การควบคุมการใช้ fields และ methods

- public : เปิดเผย คลาสอื่นๆ ใช้ได้หมด

```
public class Square {
  public Rectangle rect;
  public Square(int x, int y, int len) {
    this.rect = new Rectangle(x, y);
    this.rect.width = len;
    this.rect.length = len;
  }
  ...
```

```
...
Square  s = new Square(0,0,100);
s.rect.width = 10;
...
```


การควบคุมการใช้ fields และ methods

- private : ปกปิด ใช้ได้เฉพาะภายในคลาสเท่านั้น

```
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;
  }
  ...
```


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

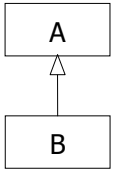
- เติม static หน้า method ทำให้เป็น class method
- เติม static หน้า field ทำให้เป็น class data
  - ที่เก็บไม่ได้อยู่ประจำแต่ละออบเจกต์ที่สร้าง
  - แต่อยู่ประจำคลาสเจ้าของ ดังนั้นจึงมีเพียงที่เก็บเดียว

```
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;
  ...
```

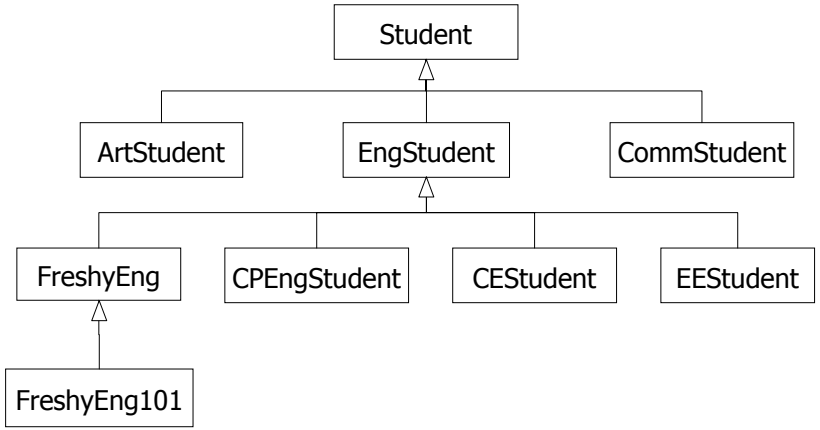
```
Undergrad s = new Undergrad();
s.setStatus( Undergrad.FRESHY );
...
```


Inheritance

- เขียนคลาสใหม่ (B) ซึ่ง "ขยาย" คุณสมบัติจากคลาสเก่า (A) ที่มีอยู่ (class B extends A)
- เรียกคลาส B ว่า "รับมรดก" จากคลาส A
  - B มีทุกๆ fields ของ A (โดยอัตโนมัติ)
  - B มีทุกๆ methods ของ A (โดยอัตโนมัติ) (ยกเว้น constructors)
  - เรียก B ว่าเป็น subclass ของ A
  - เรียก A ว่าเป็น superclass ของ B
  - ตีความได้ว่า B เป็นคลาสที่เป็นกรณีพิเศษของ A
    - ออบเจกต์ของ B เป็น A
    - แต่ออบเจกต์ของ A ไม่จำเป็นต้องเป็น B
    - ตัวอย่างเช่น class Cat extends Mammal



Inheritance Hierarchy



การแสดง Inheritance ใน Help File

```
java.lang.Object
  |
  +-JSci.maths.Matrix
      |
      +-JSci.maths.ComplexMatrix
          |
          +-JSci.maths.ComplexSquareMatrix
              |
              +-JSci.maths.ComplexTridiagonalMatrix
                  |
                  +-JSci.maths.ComplexDiagonalMatrix
```

Inheritance : แบบง่ายสุดๆ

- เขียนคลาสใหม่ เพียงแค่บอกว่าขยายจากอีกคลาส

```
import java.awt.*;
public class R1 extends Rectangle { }
```

- เหมือน Rectangle ทุกประการ (ยกเว้น constructors)
- R1 มี fields และ methods ต่างๆ ตามที่ Rectangle มี
- ไม่มี constructor อะไรเลย ระบบจะเติม constructor แบบไม่รับพารามิเตอร์ให้

```
import java.awt.*;
public class R1 extends Rectangle {
  public R1() {
    super();
  }
}
```

เรียก constructor ของ superclass

Inheritance : แบบที่ควรทำ

- กำหนดให้คลาสใหม่ขยายจากคลาสเก่าที่มี
- เพิ่ม fields ที่จำเป็น
- เขียน constructors ที่อยากให้บริการ
- เพิ่ม methods ใหม่ๆ ที่ superclass ไม่มี
- เปลี่ยน methods เดิมของ superclass ให้มีพฤติกรรมใหม่ (เรียกว่า overriding method)
- จาวาไม่อนุญาตให้ลบ methods ของ superclass ที่ subclass ไม่อยากได้

Inheritance

```
import java.awt.*;
public class DrawableRect extends Rectangle {
  public Color color;

  public DrawableRect(int x, int y, int w, int h, Color c) {
    super(x, y, w, h);
    this.color = c;
  }

  public boolean equals(Object r) {
    if ( !super.equals(r) ) return false;
    if ( this.getClass() != r.getClass() ) return false;
    return this.color.equals(((DrawableRect) r).color);
  }
  public void draw(Graphics g) {
    g.setColor(this.color);
    g.fillRect(super.x, super.y, super.width, super.height);
  }
}
```

เพิ่ม field ใหม่

เขียน constructor

เปลี่ยน method เดิม

เพิ่ม method ใหม่

Overriding ไม่เหมือน Overloading

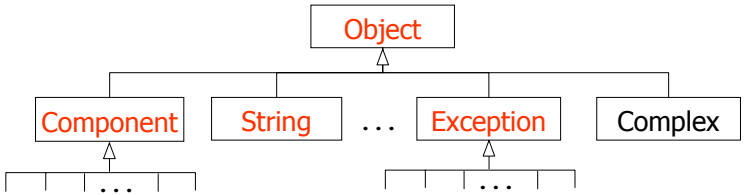
- Overriding เป็นการเขียนเมท็อดใน subclass ซึ่งมีชื่อและรายการพารามิเตอร์ปรากฏอยู่แล้วใน superclass (เป็นการ "ลบล้าง" ของเก่าที่มีอยู่)
- Overloading เป็นการเขียนเมท็อดใหม่ ซึ่งมีชื่อซ้ำ แต่รายการพารามิเตอร์ไม่เหมือนกับของเดิม (เป็นการ "เพิ่มภาระ" ให้กับชื่อเมท็อดว่าทำงานหลายแบบ ขึ้นกับรายการพารามิเตอร์ที่ได้รับ)

```
public class A {
  ...
  public int f() {
    ...
  }
}
```

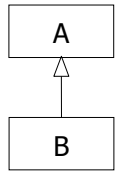
```
public class B extends A {
  public int f() {
    ... // overriding
  }
  public int f(int x) {
    ...  // overloading
  }
```

คลาส Object

- จาวามีคลาสมาตรฐานชื่อว่า Object (ขอย้ำ ชื่อ Object นี้เป็นชื่อคลาส ขึ้นต้นด้วยโอใหญ่)
- คลาส Object เป็น "บรรพบุรุษ" ของทุกคลาสในจาวา
- การเขียนคลาสใหม่ที่ไม่ได้ extends จากคลาสใด ถือได้ว่า extends จากคลาส Object



Casting



- กำหนดให้ B extends A
  - B รับพฤติกรรมทุกอย่างจาก A
  - B "เป็น" A (คือเป็นกรณีพิเศษของ A)
- Upcasting
  - เราสามารถ cast ออบเจกต์ของ B ให้เป็น A ได้
  - upcast ได้เสมอ
- Downcasting
  - ออบเจกต์ของ B ที่เคยถูก upcast ให้เป็น A สามารถถูก cast กลับลงมาเป็น B ได้
  - ถ้า downcast ไม่ได้จะเกิด ClassCastException

Up & Down

```
String s;
Object data;

data = "String"; // upcast

s = data.toUpperCase();  // error

s = ((String) data).toUpperCase(); // downcast

int i = ((Rectangle) data).width; // error
```