Implementing Classes

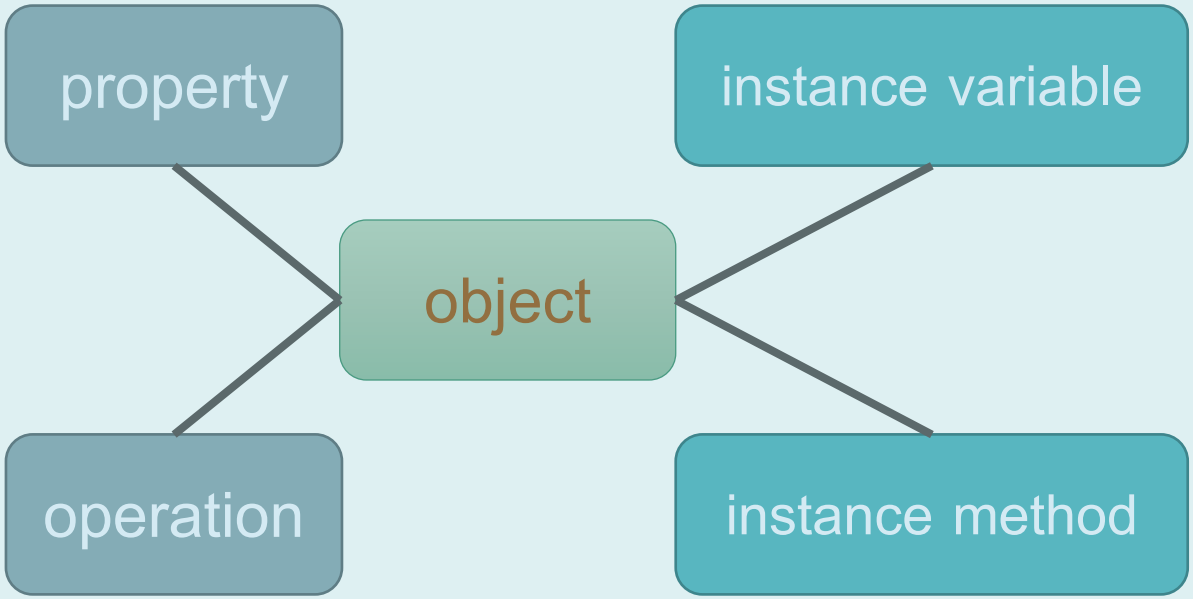
การสร้างคลาส

วัตถุ (object)

พิกัด (x,y) และ ขนาดของรูปสี่เหลี่ยม

ชื่อ รหัสนิสิต เกรด ... ของนิสิต

เลขบัญชี ชนิดบัญชี ชื่อเจ้าของ ยอดเงิน ... ของบัญชีเงินฝาก



การเลื่อนตำแหน่งของรูปสี่เหลี่ยม การหาพื้นที่ ...

การแก้ชื่อ การแก้เกรด ... ของนิสิต

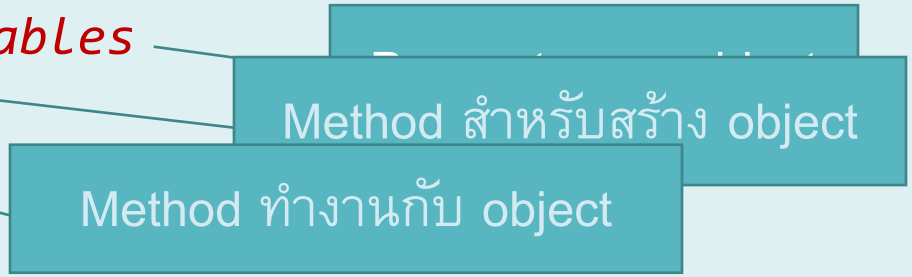
การแก้ชื่อเจ้าของ การฝากเงิน การถอนเงิน ... ของบัญชีเงินฝาก

This chapter

- ประกาศคลาส
- ตัวแปร instance (instance variable)
  - ตัวแปรสาธารณะ (public variable)
  - ตัวแปรส่วนตัว (private variable)
  - ตัวแปรเฉพาะที่ (local variable)
- เมทอด (method)
  - เมทอดสร้างวัตถุ (constructor method)

การประกาศคลาส (class declaration)

```
accessSpecifier  class  className {
    instanceVariables
    constructors
    methods
}
```



accessSpecifier อาจเป็น `public` หรือ `private`

- ตอนนี้ใช้ `public` ไปก่อน
- ถ้าสร้างคลาสในคลาส อาจให้คลาสที่ข้างในเป็น private เพื่อใช้ในคลาสข้างนอกเท่านั้น

Example

```
package rectangle;

public class Rectangle {
    private int x, y, width, height;  //instance variables

    public Rectangle(int w, int h) {
      x=0; y=0; width=w; height=h;
    }

    public Rectangle(int px, int py, int w, int h) {
      x=px; y=py; width=w; height=h;
    }

    public void translate(int dx, int dy) {
      x+=dx; y+=dy;     // x and y are instance variables
    }

    public int calArea() {
      int area = width*height;    // area is a local variable
      return area;
    }

}
```

Variables

instance variable

instance variable เป็นตัวแปรของ object ใน class นั้น

ถ้ามี object `b1` และ `b2` ในคลาส `Rectangle`

instance variable `width` ของ `b1` กับ instance variable `width` ของ `b2` ไม่เกี่ยวข้องกัน

การกำหนด instance variable ของคลาส

```
accessSpecifier  class  className {
   accessSpecifier type instanceVariable;
   …
}
```

accessSpecifier อาจเป็น `public` หรือ `private`

instance variable ควรเป็น `private` แล้วมีเมทอดที่เป็น public ที่ทำงานกับ instance variable

ตัวอย่าง

```
public class Rectangle {
   private int x, y, width, height;
   …
}
```

`x,y,width,height` เป็น instance variables (ควรเป็น private)

private and public variables

```
public class Rectangle {
 private int x,y,width,height;
   …
}
```

```
public class Rectangle {
 public int x,y,width,height;
   …
}
```

```
public class testRect {
  Rectangle b=new Rectangle(3,4);
  b.x=0;
}
```

`x,y,width,height` เป็น private variables

เกิด error ที่ `b.x=0;` เพราะไม่สามารถอ้างถึงตัวแปร `x` ของ `b` ได้

`x,y,width,height` เป็น public variables

ไม่เกิด error ที่ `b.x=0;` เพราะตัวแปร `x` ของ `b` เป็น public variable

ไม่แนะนำ

ตัวแปร local

- ตัวแปร local เป็นตัวแปรที่ประกาศในเมทอด และ ใช้ได้เมทอดนั้นเท่านั้น
- คล้ายพารามิเตอร์
- ตัวแปร local และ พารามิเตอร์ มีชีวิตเมื่อเมทอดนั้นทำงานอยู่
- เพื่อให้โปรแกรมเมอร์ไม่ต้องห่วงว่าตั้งชื่อตัวแปรซ้ำกับเมทอดอื่นๆ
- ต่างจาก instance variable ที่มีชีวิตอยู่เมื่อวัตถุยังมีชีวิตอยู่

```
public class Rectangle {
   private int x, y, width, height;  //instance variables
   …

   public void translate(int dx, int dy) { //dx and dy are parameters
      x+=dx; y+=dy;     // x and y are instance variables
   }

   public int calArea() {
      int area = width*height;    // area is a local variable
      return area;
   }
}
```

Instance variable / Local variable / Parameter

```
public class Rectangle {
   private int x, y, width, height;  // x is an instance variable
   …

   public void translate(int x, int y) { // x is also parameter
      this.x += x;   // this.x refers to instance variable
      this.y += y;   // plain x refers to parameter x
   }
   …
}
```

เทียบกับ

```
public class Rectangle {
   private int x, y, width, height;  // x is an instance variable
   …

   public void translate(int dx, int dy) { // use different name
      x += dx;
      y += dy;   // plain x refers to parameter x
   }
   …
}
```

Method

การกำหนด constructor ของคลาส

```
accessSpecifier  class  className {
    …
    public className ( parameterList ) { // constructor
        …
    }
    …
}
```

- Constructor เป็นเมทอดที่ใช้ชื่อเดียวกับคลาส
- เป็นเมทอดที่ไม่มี return type เหมือนเมทอดอื่นๆ (return type ไม่ใช่ void)
- ถูกเรียกใช้ด้วยฟังก์ชัน new ที่คืนค่าเป็นวัตถุในคลาสนั้น
- อาจมี constructor หลายตัวที่มี *parameterList* ต่างกัน เรียกว่า overloaded method

ตัวอย่างการกำหนด constructor ของคลาส

```
public class Rectangle {
   private int x, y, width, height;

   public Rectangle(int w, int h) {
      x=0; y=0; width=w; height=h;
   }

   public Rectangle(int px, int py, int w, int h) {
      x=px; y=py; width=w; height=h;
   }
   ...
}
```

ในตัวอย่างนี้ Constructor ของคลาส `Rectangle` เป็น overloaded method

การใช้ constructor ของคลาส

- Constructor ถูกเรียกใช้ด้วยฟังก์ชัน new และคืนค่าเป็นวัตถุในคลาสนั้น

```
new className( parameterList )
```

- สร้างวัตถุในคลาส *className*

```
className referenceVariable = new className( parameterList );
```

- สร้างวัตถุในคลาส *className* และเก็บ object reference ในตัวแปร *referenceVariable*

ตัวอย่างการใช้ constructor ของคลาส

```
public class Rectangle {
   private int x, y, width, height;

   public Rectangle(int w, int h) { x=0; y=0; width=w; height=h;  }

   public Rectangle(int px, int py, int w, int h) {
      x=px; y=py; width=w; height=h;
   }
}
```

ถ้าใช้ `Rectangle box = new Rectangle(3,5)` จะได้

- object ในคลาส `Rectangle` ที่มี instance variable `x=0, y=0, width=3, height=5`
- Reference variable ของ object นี้ คือ box

ถ้าใช้ `Rectangle box = new Rectangle(6,7,3,5)` จะได้

- object ในคลาส `Rectangle` ที่มี instance variable `x=6, y=7, width=3, height=5`

ตัวอย่างการใช้ constructor ของคลาส

```
public class Rectangle {
   private int x, y, width, height;

   public Rectangle(int w, int h) { x=0; y=0; width=w; height=h;  }

   public Rectangle(int px, int py, int w, int h) {
      x=px; y=py; width=w; height=h;
   }
}
```

เมื่อใช้ `(new Rectangle(2,4)).calArea()` เพื่อสร้าง object ในคลาส `Rectangle` แล้วใช้ . เรียกใช้เมทอดเลย

วัตถุที่สร้างมานี้ไม่มี reference variable นั่นคือเป็น **anonymous object**

ตัวอย่างการใช้ constructor ของคลาส

```
public class Rectangle {
   private int x, y, width, height;

   public Rectangle(int w, int h) { x=0; y=0; width=w; height=h;  }

   public Rectangle(int px, int py, int w, int h) {
      x=px; y=py; width=w; height=h;
   }
}
```

ถ้าให้ reference variable `Rectangle b;`

เมื่อใช้ `b = new Rectangle(2,4);` เพื่อสร้าง object ในคลาส `Rectangle` แล้วเก็บใน reference variable `b` แล้วเรียกใช้ method ผ่าน reference variable `b` เช่น `b.translate(2,3); int a=b.calArea();` ก็ได้

การกำหนดเมทอดของคลาส

```
accessSpecifier  class  className {
   …
   accessSpecifier returnType methodName ( parameterList ) {
      …
   }
   …
}
```

- ถ้า *accessSpecifier* เป็น public เมทอดถูกเรียกใช้จากนอกคลาสได้
- ถ้า *accessSpecifier* เป็น private เมทอดถูกเรียกใช้ได้จากในคลาสเท่านั้น
- ถ้าไม่ระบุ *accessSpecifier* เมทอดถูกเรียกใช้ได้จากใน package เดียวกัน
- เมทอดของคลาสเดียวกันที่มีชื่อเดียวกันต้องมี *parameterList* ต่างกัน คือ เป็น overloaded method

ตัวอย่างการกำหนดเมทอดของคลาส

```
public class Rectangle {
   private int x, y, width, height; // changed via public methods
   …
   public void translate(int dx, int dy) {  // move the rect
      x+=dx; y+=dy;  return;    // can omit return statement
   }
   public int calArea() {  // find the area
      return width*height; // return int
   }
   …
}
```

translate และ calArea เป็น instance method ของคลาส Rectangle

translate แก้ค่าของ instance variable x และ y แล้วไม่ส่งค่าคืน (void)

calArea คำนวณพื้นที่ของสี่เหลี่ยม แล้วส่งค่าคืนมาเป็นจำนวนเต็ม (int)

การกำหนดเมทอดของคลาส

ในการเรียกใช้เมทอดโดยอ้างถึง object (เช่น `box.translate(4, 5)`) object ที่อ้างถึง (เช่น box) เป็น implicit parameter

ในนิยามเมทอดของคลาส

- อ้างถึง instance variable `x` ของ implicit parameter ได้โดยใช้ชื่อของ instance variable `x` หรือ `this.x`
- อ้างถึงเมทอด `f(…)` ของ implicit parameter ได้โดยใช้ชื่อของเมทอด `f(…)` หรือ `this.f(…)`
- อ้างถึง constructor ของ implicit parameter ได้โดยใช้ `this(…)`

```
public Rectangle(int w, int h) {
   x=0; y=0; width=w; height=h;
}
public Rectangle(int px, int py, int w, int h) {
   this(w,h); this.x=px; y=py;
}
```

การเรียกใช้เมทอดของคลาส

```
referenceVariable.methodName ( parameterList )
```

- เมทอดถูกเรียกใช้ด้วย object member access operator คือ `.` ผ่าน reference variable
- reference variable เป็น implicit parameter ที่อ้างถึงได้ในเมทอด

```
(new constructor(parameterList)).methodName(parameterList)
```

- วัตถุที่สร้างมานี้ไม่มี reference variable นั่นคือเป็น **anonymous object**

ตัวอย่างการเรียกใช้เมทอดของคลาส

```
public class Test {
   public static void main(String[] args) {
      Rectangle box = new Rectangle(3,4);
      box.translate(4, 5);
      System.out.println(new Rectangle(2,4).calArea());
   }
}
```

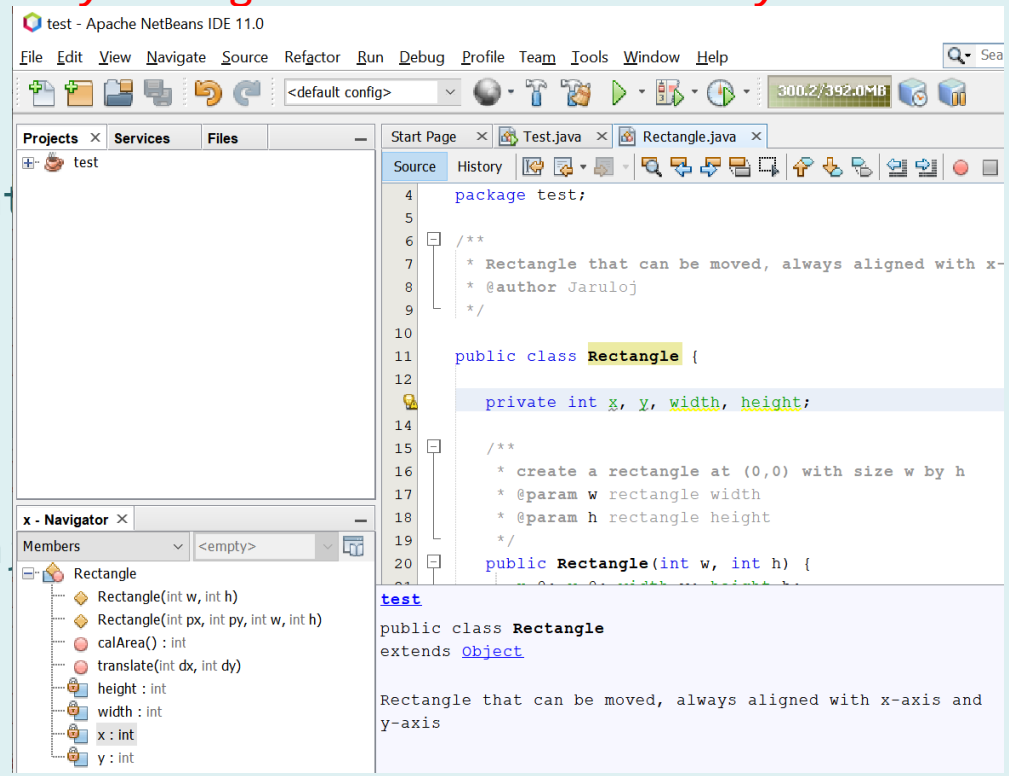
```
public class Rectangle {
   private int x, y, width, height;
   public Rectangle(int w, int h) { … }
   public Rectangle(int px, int py, int w, int h) { … }
   public void translate(int dx, int dy) { … }
   public int calArea() { … }
}
```

`box.translate(4, 5)` เรียกใช้เมทอดผ่าน object box

`new Rectangle(2,4).calArea()` เรียกใช้เมทอดผ่าน anonymous object

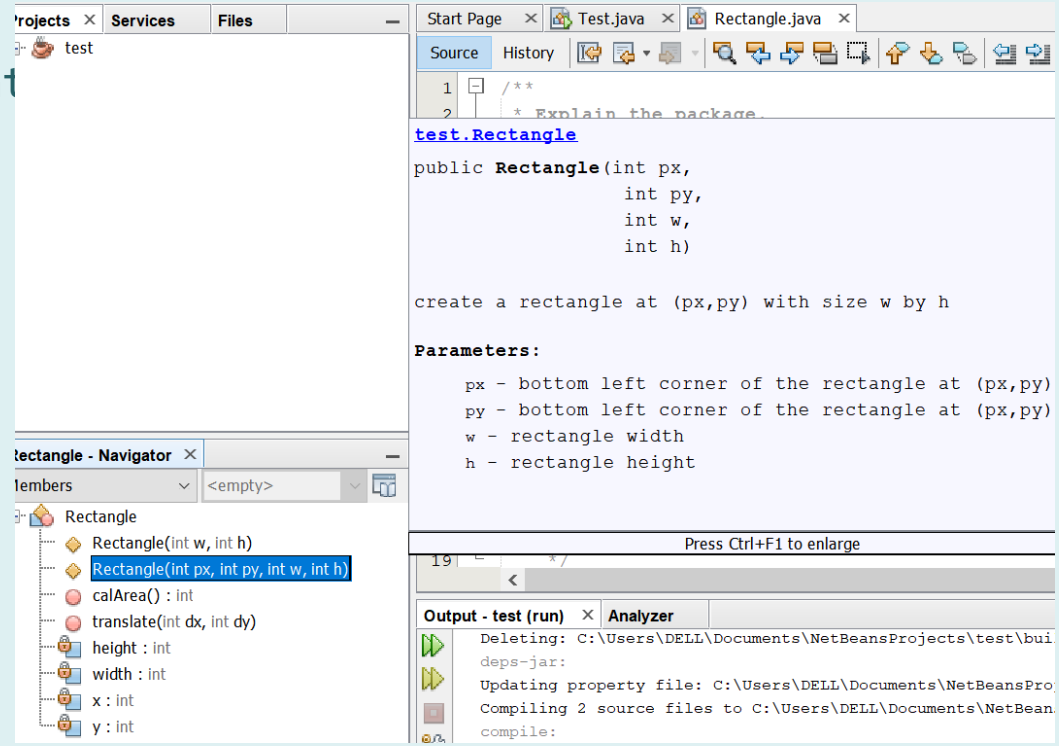
javadoc

```
/**
 * Explain the package.
 */
package test;

/**
 * Rectangle that can be moved, always aligned with x-axis and y-axis
 * @author Jaruloj
 */

public class Rectangle {
   private int x, y, width, height

   /**
    * create a rectangle at (0,0)
    * @param w rectangle width
    * @param h rectangle height
    */
   public Rectangle(int w, int h)
      x=0; y=0; width=w; height=h
   }
```



javadoc

```
/**
 * create a rectangle at (px,py) with size w by h
 * @param px bottom left corner of the rectangle at (px,py)
 * @param py bottom left corner of the rectangle at (px,py)
 * @param w rectangle width
 * @param h rectangle height
 */
public Rectangle(int px, int
    x=px; y=py;
    width=w; height=h;
}
```



javadoc

```
    /**
     * move the rectangle
     * @param dx the translation in x-axis
     * @param dy the translation in y-axis
     */
    public void translate(int dx, int dy) {
        x+=dx; y+=dy;
    }

    /**
    * calculate the area of rectangle
    * @return area as integer
    */
    public int calArea() {
        return width*height;
    }
}
```