Object and classes

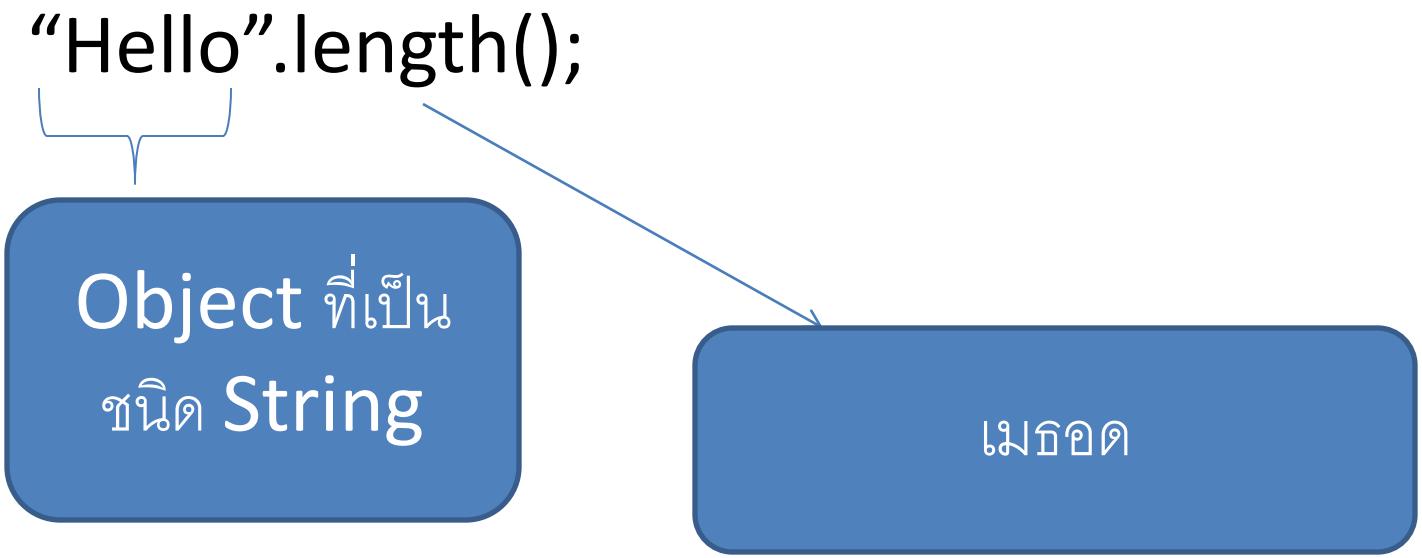
บทนี้เรียนอะไรบ้าง

- แนะนำ Object-oriented programming
- บอกวิธีการสร้าง object จากคลาสที่จาวาให้มา
- สอนวิธีสร้างคลาสของเราเอง

แนะนำ Object-Oriented Programming

- โปรแกรมประกอบขึ้นจากออบเจ็กต์
- แต่ละออบเจ็กต์จะมีฟังก์ชั่นให้เรียกใช้
  - คนเรียกใช้ไม่จำเป็นต้องรู้ว่าข้างในออบเจ็กต์ทำงานยังไง
  - ขอให้ใช้ได้ตามที่ต้องการก็พอ
- วิธีคิดในการโปรแกรม
  - เมื่อต้องการแก้ปัญหา ให้ออกแบบตัวข้อมูลที่ต้องใช้ก่อน
  - แล้ววิธีแก้ปัญหาค่อยตามมา

ตัวอย่าง



เมื่อรันแล้วจะได้ 5 เป็นคำตอบออกมา สามารถ print ออกมาได้ เช่น

System.out.println(“Hello”.length());

System.out.println(“Hello”.length());

นี่เป็น implicit พารามิเตอร์ของเมธอด println ครั้งนี้

จริงๆมันก็คือ ออบเจ็กต์ที่เรียกเมธอดนั่นเอง

นี่เป็น explicit พารามิเตอร์ของเมธอด println ครั้งนี้

การสร้างออบเจ็กต์

- ตัวอย่าง คลาส Rectangle

```
Rectangle box = new Rectangle(5,10,20,30);
```

พารามิเตอร์ ของคอนสตรัคเตอร์

ควรเก็บไว้ในตัวแปร ไม่งั้นเอาไปใช้ต่อยาก

สร้างออบเจ็กต์

คอนสตัรคเตอร์ ไม่มีพารามิเตอร์ก็ได้

```
System.out.println(new Rectangle());
```

เอาออบเจ็กต์ ที่สร้างไปเป็นพารามิเตอร์เมธอดอื่นก็ได้

API documentation

- คือ help file ที่บอกวิธีใช้งานทุกคลาสและเมธอดที่คนเขียนจาวาทำมาแล้ว
- เราเขียนของคลาสเราเองได้ด้วยนะ
- ลองทำความรู้จัก จากหน้า

  java.sun.com/javase/7/docs/api/index.html
  - ลองหาของ Rectangle ดู

เราสามารถเรียกเมธอดของ Rectangle กับออบเจ็กต์ชนิดนี้ที่เราพึ่งสร้างขึ้นมา เช่น

```
box.translate(15,25);
```

เคลื่อนสี่เหลี่ยมไป 15 จุดตามแนวแกน x และ 25 จุดตามแนวแกน y

อย่าลืม ว่าถ้าจะเอา Rectangle มาใช้ได้ เราต้อง import package ที่มันอยู่ เข้ามาในโปรแกรมที่เราเขียนก่อน

```
import java.awt.Rectangle;
```

Package คือกลุ่มของคลาสที่จัดไว้ชุดเดียวกัน ไว้ใช้งานร่วมกัน

exercise

- จงใช้คลาส Day ใน API เขียนโปรแกรมที่รับวันเดือนปีเกิด แล้วพิมพ์คำตอบว่า คนที่เกิดวันเดือนปีนั้น มีชีวิตมาแล้วทั้งหมดกี่วัน
- จงใช้คลาส Picture มาเขียนโปรแกรมในการเปลี่ยนรูปภาพ ทดลองเมธอดต่างๆตามที่ต้องการ

คลาส

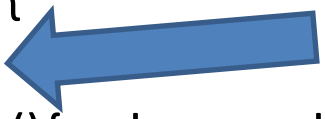
- คลาสคือ template ที่เราจะใช้สร้างออบเจ็กต์
  - พูดง่ายๆ มันเป็นตัวกำหนดว่า ออบเจ็กต์ที่สร้างจากมัน จะมีข้อมูลอะไรอยู่ภายในบ้าง
  - คลาสเปรียบเสมือนแม่พิมพ์ ส่วนออบเจ็กต์ก็เป็นสิ่งที่แม่พิมพ์พิมพ์ออกมา
- เมื่อเราสร้างออบเจ็กต์ขึ้นจากคลาส
  - เราเรียกเหตุการณ์นี้ว่า การสร้าง instance ของคลาส

- ภายในแต่ละออบเจ็กต์
  - ข้อมูล หรือดาต้า เราเรียกว่า instance fields หรือ instance variables
    - แต่ละออบเจ็กต์ จะมีค่าของแต่ละ instance field เป็นของตนเอง ไม่ใช้ค่าร่วมกับออบเจ็กต์อื่น
    - สถานะของค่าตัวแปรเหล่านี้ทั้งหมด ถือเป็น state ของออบเจ็กต์
    - ซึ่งเมื่อเรียกเมธอด state ของออบเจ็กต์อาจเปลี่ยนได้
  - ส่วนฟังก์ชั่นที่ออบเจ็กต์นั้นเรียกเพื่อจัดการข้อมูลข้างต้น เรียกว่า methods

- ห้ามเข้าถึงค่าของ instance field โดยตรง
- ต้องอ่าน และเขียนค่าตัวแปร จากเมธอดเท่านั้น
- ทั้งนี้เพื่อ
  - ป้องกันการยัดค่าผิดๆ ใส่ลงไปใน instance field
    - เช่น a.x = 100; จริงๆ x อาจจะมีค่าเกิน 99 ไม่ได้ ดังนั้นเขียนเมธอดคุมดีกว่า
  - เวลาจะเปลี่ยนพฤติกรรมการทำอะไร จะได้แก้ในเมธอด ทีเดียว ไม่ต้องมาแก้หลายๆที่

ตัวอย่าง

อ่านได้เฉพาะเมธอดของคลาส Counter เท่านั้น เพราะเป็น private คือ ต้องเรียกใช้ตัวแปรนี้ผ่านเมธอดที่คลาส Counter มี เท่านั้น



```
public class Counter{
    private int value;
    public void count(){ value = value+1;}
    public int getValue(){return value;}
}

public class Auser{
    …
    Counter tally = new Counter();
    Counter tally02 = new Counter();
    tally.count();
    tally.count();
    int result = tally.getValue(); //result is set to 2
    int result02 = tally02.getValue(); //result 02is 0, a default value
}
```

เรียกว่า

encapsulation

Instance var มีค่า default แต่ local var ต้อง initiailize ทุกครั้งนะ ไม่เหมือนกัน

อีกตัวอย่าง

Constructor ห้าม return นะ

```
public class BankAccount{
    private double  balance;
    public BankAccount(){…}
    public BankAccount(double initialBalance){…}
```

อย่าลืม initialize ในคอนสตรัคเตอร์ เพราะ null เรียกเมธอดไม่ได้นะ

mutator

```
    public void deposit(double amount){…}
    public void withdraw(double amount){…}
    public double getBalance(){…}
```

accessor

จากหน้าที่แล้ว ถ้าเราต้องการทำให้คลาสของเรามี help file แบบที่ API ตามปกติมี

เติมคอมเม้นไว้ด้านบนของคลาสหรือเมธอดที่ต้องการสร้าง API ดังตัวอย่างนี้

```
/**
    Withdraws money from the bank account.
    @param amount the amount to withdraw
*/
public void withdraw(double amount){...}


/**
    Gets the current balance of the bank account.
    @return the current balance
*/
public double getBalance(){...}
```

ใช้ javadoc สร้างไฟล์ API จากคอมเม้นได้อัตโนมัติ ทุก tool มีให้ทดลอง

สรุป วิธีเขียนคลาสของเราขึ้นมาใช้งานเอง

- คิดก่อน ว่าคลาสของเราต้องมีเมธอดอะไรที่เรียกใช้ได้บ้าง
- เขียนหัวเมธอดเหล่านั้นออกมาก่อน
- เขียนคอมเม้นของเมธอดนั้นให้เรียบร้อย
- หาว่า ต้องใช้ instance variable อะไรบ้าง
- เขียนโค้ดของคอนสตรัคเตอร์และเมธอด
- Test ตัวโค้ดทีละเมธอด ด้วยเครื่องมือ อย่างเช่น Junit

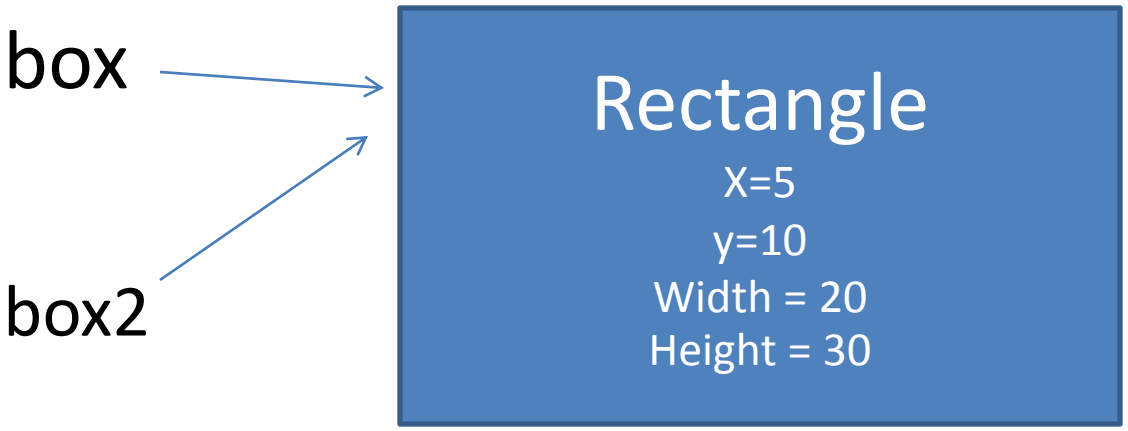
(คลาสไม่จำเป็นต้องมี main นะ ขอให้มีเมธอดให้เรียกใช้ได้ก็โอเคแล้ว)

คราวนี้มาดูเรื่อง object ref

- ตัวแปรที่มีไทป์เป็นคลาส ไม่ได้เก็บออบเจ็กต์
  - แต่เก็บตำแหน่งในเมมโมรี่ ของออบเจ็กต์นั้น
- Object reference หมายถึงตำแหน่งในเมมโมรี่ของอ็อบเจ็กต์
- ตัวแปรที่เก็บตำแหน่งในเมมโมรี่ของอ็อบเจ็กต์ เราเรียกว่าตัวแปรนั้น refer to อ็อบเจ็กต์นั้น

```
Rectangle box = new Rectangle(5,10,20,30);

Rectangle box2 = box;
```



เปรียบเทียบ int กับ อ็อบเจ็กต์

```
int n =13;
int n2 = n;
int n2 = 12;
```

เมื่อรันเสร็จ n จะเท่ากับ 13 ส่น n2 จะเท่ากับ 12

```
Rectangle box = new
Rectangle(5,10,20,30);
Rectangle box2 = box;
box2.translate(15,25);
```

เมื่อรันเสร็จจะได้box กับ box2 ที่ข้างในเหมือนกันทั้งคู่

this

ถ้าใช้ในเมธอด หรือคอนสตรัคเตอร์ จะหมายถึงออบเจ็กต์ที่เมธอดนั้นๆถูกเรียก เช่น

```
public void deposit(double amount){
    balance = balance + amount;
}
```

ซึ่งมีความหมายเหมือนกับ

```
public void deposit(double amount){
    this.balance = this.balance + amount;
}
```

- หรือจะใช้เพื่อแยก instance var กับ local var ก็ได้ เช่น

```
public BankAccount(double balance){
   this.balance = balance;
}
```

ใช้คอนสตรัคเตอร์ตัวหนึ่งเป็นฐาน เรียกตัวอื่น

```
public BankAccount(double initialBalance){
   balance = initialBalance;
}

public BankAccount(){
   this(0); // การเรียกคอนสตรัคเตอร์อื่น ทำได้แค่บรรทัดแรกของ
            //คอนสตรัคเตอร์เท่านั้น
}
```