ชื่อ ..... รหัสนักศึกษา.....

**05506003 Programming Fundamentals**

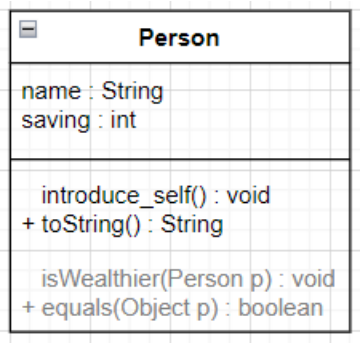
*Lab Week 12*

## วัตถุประสงค์

A. ฝึกทักษะการสร้าง class

## In class

กิจกรรมที่ 1 เติม q1.1 – q1.3 ให้สอดคล้องกับ class diagram (กำหนด access modifier เป็น default ให้หมดก่อน ยกเว้น toString() มี access modifier เป็น public)




```
class Person {
  String name;
  int saving;
  /* q1.1 */ (String n, int amount) {
    name = n;
    saving = amount;
  }
  void /* q1.2 */ () {
    System.out.println("Hi,
                My name is " + name);
    }
    @Override
    /* q1.3 */ {
        return "I am " + name + ". My
savings amount is " + saving;
}
```

## Homework

กิจกรรมที่ 2 ต่อยอด Person.java ด้วยการ Save as เป็น Person_xxyyyy.java แก้ code ให้เรียบร้อย

เพิ่ม

void isWealthier(Person p) { ... }

public boolean equals(Object theOther) { ... }

กิจกรรมที่ 3 เพิ่ม

int reward(Person p, int amount) { ... } โดยหมายถึงให้(เงิน)รางวัล (จาก saving ตนเอง ดังนั้น saving ของ p ย่อมเพิ่มขึ้น)

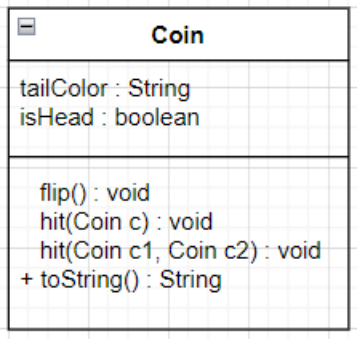
```java
public class ProFun12_Person_Main {
    public static void main(String[] args) {
        demo_1_initialize_object();
        demo_2_interaction_equality();
        demo_3_more_interaction();
    }
    static void demo_3_more_interaction() {
        System.out.println("- person's saving change----");
        Person yd1 = new Person("yindee", 2000 );
        Person pd = new Person("Preeda", 1000);
        yd1.reward(pd, 400);
        System.out.println(yd1);
        System.out.println(pd);
    }
    static void demo_2_interaction_equality() {
        System.out.println("-object interaction----");
        Person yd1 = new Person("yindee", 2000 );
        Person pd = new Person("Preeda", 1000);
        System.out.println(yd1.isWealthier(pd));

        Person yd2 = new Person("yindee",2000);
        System.out.println("== tests on references' address " + (yd1 == yd2));
        System.out.println(".equals() " + yd1.equals(yd2));
        System.out.println(".equals() " + yd2.equals(yd1));
        System.out.println(".equals() " + yd1.equals(pd));
    }
    static void demo_1_initialize_object() {
        System.out.println("-new and toString()----");
        Person yd = new Person("yindee", 2000 );
        Person pd = new Person("Preeda", 1000);
        yd.introduce_self();
        pd.introduce_self();
        System.out.println( yd ); // invoke toString()
        System.out.println( pd );
    }
}
```

```
-new and toString()----
Hi, My name is yindee
Hi, My name is Preeda
I am yindee. My savings amount is 2000
I am Preeda. My savings amount is 1000
-object interaction----
true
== tests on references' address false
.equals() true
.equals() true
.equals() false
- person's saving change----
I am yindee. My savings amount is 1600
I am Preeda. My savings amount is 1400
```

ชื่อ ..... รหัสนักศึกษา.....

กิจกรรมที่ 4 จะให้เรียกเป็นอะไรเพราะมันเป็นเหรียญ

กำหนด Coin.java




แต่ละเหรียญจะมีหน้า tail เป็นสี ไม่เหมือนกัน (จริงๆไม่จำเป็น) และรู้สถานะว่าตอนนี้ head หรือ tail ที่หงายอยู่ (เวลา toString() จะแสดงหน้าปัจจุบันของเหรียญ)

กำหนดให้หน้าเหรียญตอนเริ่มเป็น head

flip() หมายถึงเหรียญนั้นสลับหน้า

hit(Coin c) หมายถึง การชนกันของเหรียญ ทำให้ทั้งคู่เกิดการ flip()

hit(Coin c1, Coin c2) หมายถึงเหรียญ this ชนกับ c1 และไปชน c2 ต่อ (การสร้าง method ชื่อเดียวกัน แต่ signature ต่างกันเรียกว่าการทำ overloading)

```
class Coin {
  String tailColor;
  boolean isHead;
  Coin(String color) {
        tailColor = color;
        isHead = true;
  }
  @Override
  public String toString() {
    return "I am a coin object with tailColor = " + tailColor
                                    + ". My isHead is " + isHead;
  }
  void flip() {
    if (isHead)
        System.out.println("From flip() -> I am head
                  my color is SILVER (my tailColor is " + tailColor + ")");
    else
        System.out.println("From flip() -> My face color is " + tailColor);
    isHead = !isHead;
  }
  void hit(Coin c) {
        // ? if (!this.equals(c)) for coin can't hit itself
        /* q2.1 */
  }
  void hit(Coin c1, Coin c2) {
        /* q2.2 */  }
}
```

หมายเหตุ 1. ตามหลัก OOP ควรมี debit(int amount) ให้ p จาก reward(Person p) เรียก ไม่ใช่แก้ค่าตรงๆ

2. user-defined-type ที่ไม่เป็นไปตาม OOP จะเขียนแบบ structured programming ดังแสดงใน demo4_non_oop_static_way() (ไม่มี method ก็เป็นเพียง type ที่ซับซ้อนกว่า primitive)

```
    static void demo3_overloaded_hit() {
        System.out.println("-demo overloaded method (from demo2)");
        Coin c1 = new Coin("Burgandy");
        Coin c2 = new Coin("Cabala");
        Coin c3 = new Coin("Danube");
        c1.hit(c2, c3);
    }
    static void demo2_hit() {
        System.out.println("-demo object interaction--");
        Coin c1 = new Coin("Burgandy");
        Coin c2 = new Coin("Cabala");
        Coin c3 = new Coin("Danube");
        c1.hit(c2);
        c1.hit(c3);
        System.out.println(c1);
        System.out.println(c2);
        System.out.println(c3);
    }
    static void demo_1_instantiation() {
        System.out.println("-demo instantiation and call object method--");
        Coin c1 = new Coin("Burgandy");
        Coin c2 = new Coin("Cabala");
        Coin c3 = new Coin("Danube");
        System.out.println(c1);
        System.out.println(c2);
        System.out.println(c3);
        c2.flip();
        System.out.println(c2);
    }
    static void demo4_non_oop_static_way() {
      Coin c1 = new Coin("Burgandy");
      Coin c2 = new Coin("Cabala");
      static_hit(c1, c2);
    }
    static void static_hit(Coin c1, Coin c2) {
      if (c1.isHead)
        System.out.println("From flip() -> I am head my color is
                            SILVER (my tailColor is " + c1.tailColor + ")");
      else
        System.out.println("From flip() -> My face color is " + c1.tailColor);
      c1.isHead = !c1.isHead;

      if (c2.isHead)
        System.out.println("From flip() -> I am head my color is
                            SILVER (my tailColor is " + c2.tailColor + ")");
      else
        System.out.println("From flip() -> My face color is " + c2.tailColor);
      c2.isHead = !c2.isHead;
    }
```

```
-demo instantiation and call object method--
I am a coin object with tailColor = Burgandy. My isHead is true
I am a coin object with tailColor = Cabala. My isHead is true
I am a coin object with tailColor = Danube. My isHead is true
From flip() -> I am head my color is SILVER (my tailColor is Cabala)
I am a coin object with tailColor = Cabala. My isHead is false
-demo object interaction--
From flip() -> I am head my color is SILVER (my tailColor is Burgandy)
From flip() -> I am head my color is SILVER (my tailColor is Cabala)
From flip() -> My face color is Burgandy
From flip() -> I am head my color is SILVER (my tailColor is Danube)
I am a coin object with tailColor = Burgandy. My isHead is true
I am a coin object with tailColor = Cabala. My isHead is false
I am a coin object with tailColor = Danube. My isHead is false
-demo overloaded method (from demo2)
Coin with tailColor = Burgandy hits me.
From flip() -> I am head my color is SILVER (my tailColor is Cabala)
Coin with tailColor = Burgandy hits me.
From flip() -> I am head my color is SILVER (my tailColor is Danube)
From flip() -> I am head my color is SILVER (my tailColor is Burgandy)
```

กำหนดส่ง TBA