

1. ให้ศึกษาและทดลองพิมพ์ Code ตัวอย่างการสืบทอดคลาสจากตัวอย่างต่อไปนี้

```
public abstract class GeometricObject {  
    private String color = "white";  
    private boolean filled;  
    /** Default constructor */  
    protected GeometricObject() {  
    }  
    /** Convenience constructor */  
    protected GeometricObject(String color, boolean filled) {  
        this.color = color;  
        this.filled = filled;  
    }  
    public String getColor() {  
        return color;  
    }  
    public void setColor(String color) {  
        this.color = color;  
    }  
    public boolean isFilled() {  
        return filled;  
    }  
    public void setFilled(boolean filled) {  
        this.filled = filled;  
    }  
    public abstract double findArea();  
    public abstract double findPerimeter();  
}  
public class Circle extends GeometricObject {  
    private double radius;  
  
    /** Default constructor */  
    public Circle() {  
        this(1.0);  
    }  
  
    /** Radius convenience constructor */  
    public Circle(double radius) {  
        this(radius, "white", false);  
    }  
  
    /** Convenience constructor for all properties */  
    public Circle(double radius, String color, boolean filled) {  
        super(color, filled);  
        this.radius = radius;  
    }  
  
    /**  
     * Return the radius  
     * @return radius Current radius of Circle  
     */  
    public double getRadius() {  
        return radius;  
    }  
  
    /**  
     * Set the radius of the circle  
     */
```

```

public void setRadius(double radius) {
    this.radius = radius;
}

/**
 * Returns the area of the current circle
 * Implementation of abstract method in GeometricObject
 * @return area of the circle
 */
public double findArea() {
    return radius * radius * Math.PI;
}

/**
 * Returns the perimeter of the current circle
 * Implementation of abstract method in GeometricObject
 * @return perimeter of the circle
 */
public double findPerimeter() {
    return 2 * radius * Math.PI;
}

/**
 * Provide a string representation of the object
 */
public String toString() {
    return "Circle: radius = " + radius;
}
}

```

1.1 ให้อธิบายลำดับการเรียกใช้ constructor เมื่อสร้างวัตถุจากคลาส Circle

- Circle() → Circle (double radius)
→ Circle (double radius, string color, boolean filled)

1.2 ให้สร้างคลาส Rectangle ตาม UML Class diagram ด้านบน

2. จงหาข้อผิดพลาดของส่วนของโปรแกรมต่อไปนี้

```
public class Circle{
    private double radius;

    public Circle(double radius) {
        radius = radius; ②
    }
    public double getRadius() {
        return radius;
    }
    public double findArea() {
        return radius * radius * Math.PI;
    }
}
class Cylinder extends Circle{
    private double length;

    Cylinder(double radius, double length) {
        Circle(radius); ①
        ③ length = length;
    }
}
```

① ในการจะใช้ attributes ของคลาสจะมีอยู่ทั้งนี้ super

②, ③ ในการเขียน code attributes ของ constructor ต้องใช้ keyword this

3. จงหาผลลัพธ์การรันโปรแกรมต่อไปนี้

3.1

```
class A{
    public A(){
        System.out.println("The no-arg constructor of A is
invoked");
    }
}
class B extends A{
    public B(){
    }
}
public class C{
    public static void main(String[] args){
        B b = new B();
    }
}
```

ผลลัพธ์การรันโปรแกรม

The no-arg constructor
of A is invoke

3.2

```
Animal.java:
01  public class Animal {
02      public Animal() {
03          System.out.println("A new
04          animal has been created!");
05      }
06      public void sleep() {
07          System.out.println("An animal
08          sleeps...");
09      }
10      public void eat() {
11          System.out.println("An animal
12          eats...");
13  }
```

```
Bird.java:
01  public class Bird extends Animal {
02      public Bird() {
03          super();
04          System.out.println("A new
05          bird has been created!");
06      }
07      @Override
08      public void sleep() {
09          System.out.println("A bird
10          sleeps...");
11      }
12      @Override
13      public void eat() {
14          System.out.println("A bird
15          eats...");
16  }
```

```
Dog.java:
01  public class Dog extends Animal {
02      public Dog() {
03          super();
04          System.out.println("A new dog
05          has been created!");
06      }
07      @Override
08      public void sleep() {
09          System.out.println("A dog
10          sleeps...");
11      }
12      @Override
13      public void eat() {
14          System.out.println("A dog
15          eats...");
16  }
```

```
MainClass.java:
01  public class MainClass {
02      public static void main(String[]
03      args) {
04          Animal animal = new Animal();
05          Bird bird = new Bird();
06          Dog dog = new Dog();
07          System.out.println();
08          animal.sleep();
09          animal.eat();
10          bird.sleep();
11          bird.eat();
12          dog.sleep();
13          dog.eat();
14      }
15  }
```

ผลลัพธ์การรันโปรแกรม
 A new animal has been created
 A new bird has been created
 A new dog has been created

An animal sleeps...

An animal eats...

A bird sleeps...

A bird eats...

A dog sleeps...

A dog eats...

ชื่อ-นามสกุล นิตย์ พานิช ชุมวิจิตร

รหัสประจำตัวนักศึกษา 660466263004

ปีการศึกษา 2567

4. จากโปรแกรมด้านล่างต่อไปนี้ ข้อใดคือ Overriding ที่สามารถนำมาเติมลงในบรรทัดที่ 9 ของโปรแกรมได้โดยไม่ทำให้เกิดข้อผิดพลาดขึ้นกับการทำงานของโปรแกรม อธิบายเหตุผลประกอบ

```

1 class SuperClass{
2     private int num=1;
3     protected int getNumber(){
4         return num;
5     }
6 }
7 class Subclass extends SuperClass{
8     private int num=10;
9     //overriding method
10    public static void main(String[] args){
11        Subclass s= new Subclass();
12        System.out.println(s.getNumber());
13    }
14 }
```

โปรแกรม	สามารถนำมาเติมลงในบรรทัดที่ 9 ของโปรแกรมได้โดยไม่ทำให้เกิด ข้อผิดพลาดหรือไม่	อธิบายเหตุผลประกอบ
protected int getNumbers(){ return num+5; }	ไม่ได้	ใช้การระบุไว้แล้วว่าเป็น protected แต่ จะไม่สามารถ override ได้
protected long getNumber(){ return num+5; }	ไม่ได้	return type เปลี่ยนแปลงไป
protected int getNumber(){ return num+5; }	ไม่ได้	แล้วจะผิดตามปกติ
public int getNumber(){ return num+5; }	ไม่ได้	ไม่สามารถoverride (public สามารถ override protected)
protected int getNumber(int num){ return num+5; }	ได้	ใช้ในการ Overload method
int getNumber(){ return num+5; }	ไม่ได้	แล้วจะผิดตามปกติ ไม่ได้ access modifier
private int getNumber(){ return num+5; }	ไม่ได้	Overriding ไม่สามารถทำได้ร ถ้า access type ของ subclass เป็น private