University of Dhaka
Dept. of Computer Science and Engineering

CSE-2112: Object Oriented Programming Lab (Spring, 2023)
Lab Teachers: Dr. Muhammad Ibrahim and Mr. Md. Ashraful Islam

Lab Exam 1: Up to method overriding.
Date : February 28, 2023

1) Consider a simplified system for shop management. Suppose you have only two types of food items in the shop: grain (rice and wheat) and egg.

   While these items have some common properties (like product code and name), they have some differences from the shop owner's perspective. (For example, grains are sold in 100 grams but eggs are sold per piece.) Properties of eggs include type (hen or duck) and size (small and large etc.). Properties of rice include coarseness (thick or thin). Properties of wheat include level of fiber (i.e. white or red wheat).

   From the shop owner's perspective, each food item has three actions associated with it, which are: (1) Being brought into the stock (i.e., stocking in). (2) Being sold to the customers. (Both of these actions need to be correctly updated with the stock amount.) (3) Calculating the total price of a particular food item.

   You must use the concept of "using superclass variables for referring to subclass objects and invoking overridden methods". So you must use inheritance and method overriding as appropriate. In your main function, demonstrate usage of all these concepts.

   You should write standard code as much as possible. For example, you should use constructors in all the classes. You should use meaningful variable names.

2) Consider the following program. Here method overriding is used apparently successfully. But it turns out that there is a real problem when using a superclass variable for referring to a subclass object -- as shown in the later part of the main function. In this example, you've learnt how an apparently benign code of inheritance and method overriding may turn into a catastrophe. So you must use inheritance and method overriding wisely. This is the lesson you've learnt from this code.

   Now, your task is to modify the code in such a way that minimal change is needed to fend off the demonstrated problem while maintaining (as much as possible) the intended benefit of inheritance and method overriding. There are multiple solutions to this problem, so think about the best possible solution and write the code accordingly. You may also write multiple solutions if time permits.

```
class Rectangle {
    protected double width;
    protected double length;
    public Rectangle (double width, double length){
        this.width = width;
        this.length = length;
    }
    public void set_width (double width){
        this.width = width;
    }
    public void set_length (double length){
        this.length = length;
    }
    public double calculate_area (){
        return width*length;
    }

}

class Square extends Rectangle {
    public Square (double width){
        super(width, width);
    }
    public void set_width (double width){
        this.width = width;
        this.length = width;
    }
    public void set_length (double length){
        this.width = length;
        this.length = length;
    }
}

class another_class {
    public static void to_another_programmer(Rectangle r){
       System.out.println("I am another programmer...I am always very
cautious, so I'm checking the type of variable r whether it is of
Type Rectangle or not ...");
       System.out.println("Type of r: " + (r instanceof Rectangle));
      System.out.println("All good, so I'm changing the dimensions of
the rectangle to 4 and 3 ...");
        r.set_width(4);
```

```
        r.set_length(3);
       System.out.println("Now I'm expecting to get the area of the
rectangle to be 12 ...");
                    System.out.println("Area   of   rectangle:   "   +
r.calculate_area());
       System.out.println("Oh, what happened???");
    }
}

class Demo {
    public static void main(String[] args) {
        Rectangle rect = new Rectangle (5, 6);
                     System.out.println("Area   of   square:   "   +
rect.calculate_area());
        rect = new Square (5);
                     System.out.println("Area   of   square:   "   +
rect.calculate_area());
        another_class.to_another_programmer(rect);
    }
}
```