COMP 6481

Tutorial 2:
Polymorphism and Abstract Classes

Coding Exercise 1

- Given an abstract class Shape and followed by main class ShapeDriver:

```
package shapes;


//an abstract class Shape
abstract class Shape {
// abstract method getArea
     abstract double getArea();
}
```

Coding Exercise 1 (class ShapeDriver)

```
package shapes;
public class ShapeDriver {
 public static void main(String[] args)
 {
   Circle c1= new Circle();
   c1.setRadius(2.0);
   System.out.println("Area of c1  " +c1);
   Circle c2= new Circle();
   c2.setRadius(4.0);
   System.out.println("Area of c2  " +c2);
```

Coding Exercise 1 (class ShapeDriver ..Continue)

```
    Rectangle r1= new Rectangle();
    r1.setHeight(2.0);
    r1.setWidth(4.0);
    System.out.println("Area of r1 " +r1);
    Rectangle r2= new Rectangle();
    r2.setHeight(3.0);
    r2.setWidth(6.0);
    System.out.println("Area of r2 " +r2);
    Shape shapes[]={c1,c2,r1,r2};
   // We are using the "totalArea" method here
    System.out.println("Total Area is: " + totalArea(shapes));
  } //TODO: Define method "totalArea" here
}
```

Coding Exercise 1(Continue)

- Define classes Rectangle and Circle which extend Shape and provide the expected result below:

Expected result:

```
Area of c1 Circle: 12.56
Area of c2 Circle: 50.24
Area of r1 Rectangle: 8.0
Area of r2 Rectangle: 18.0
```

Note:

Area of Rectangle = height*width;

Area of Circle = 3.14*radius*radius;

Coding Exercise 2 (Continue)

- The capability to reference instances of Rectangle and Circle as Shape types brings the advantage of treating a set of different types of shapes as one common type. Define a method "totalArea" in the class ShapeDriver in order to get the result below:

Expected result:

```
Area of c1 Circle: 12.56
Area of c2 Circle: 50.24
Area of r1 Rectangle: 8.0
Area of r2 Rectangle: 18.0
Total Area is: 88.8
```

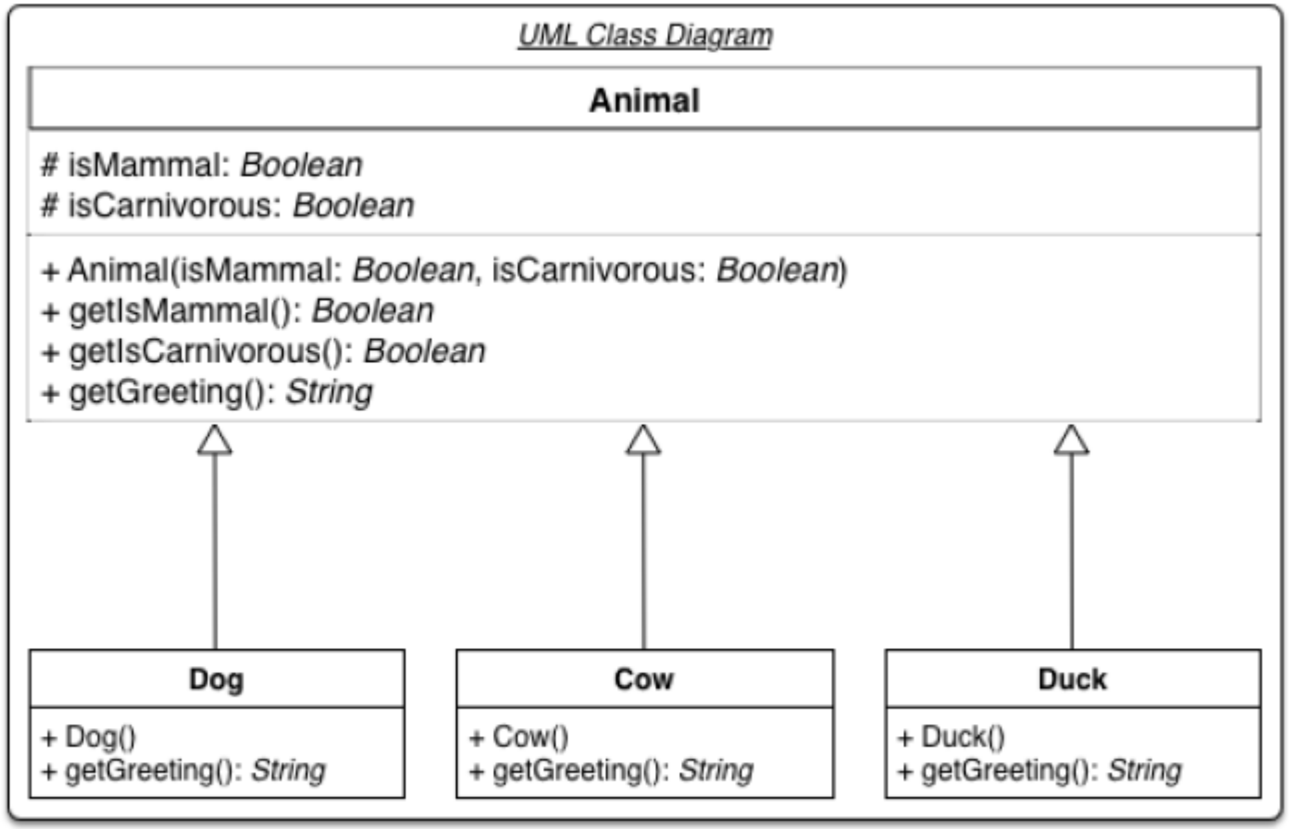
Coding Exercise 3

- Rewrite the class Shape without using an abstract class. This new class Shape should be still match classes ShapeDriver, Rectangle and Circle (programmed in previous question) and the same expected result below.

Expected result:

```
Area of c1 Circle: 12.56
Area of c2 Circle: 50.24
Area of r1 Rectangle: 8.0
Area of r2 Rectangle: 18.0
Total Area is: 88.8
```

Coding Exercise 4



A UML diagram of Animal, Dog, Cow, and Duck classes. Recall that `-` denotes *private*, `+` denotes *public*, and `#` denotes *protected*.

Things to do:

1. Declare an abstract class named Animal with the implementations for getIsMammal() and getIsCarnivorous() methods, as well as an abstract method named getGreeting() .

2. Create Dog, Cow , and Duck objects.

3. Call the getIsMammal() , getIsCarnivorous() , and getGreeting() methods on each of these respective objects.

4. Three classes named Dog , Cow , and Duck that inherit from the Animal class.

5. No-argument constructors for each class that initialize the instance variables inherited from the superclass.

6. Each class must implement the getGreeting() method:
- For a Dog object, this must return the string ruff .
- For a Cow object, this must return the string moo .
- For a Duck object, this must return the string quack .

Input Format
There is no input for this challenge.

Output Format
The getGreeting() method must always return a string denoting the appropriate greeting for the implementing class.

Sample Output

```
A dog says 'ruff', is carnivorous, and is a mammal.
A cow says 'moo', is not carnivorous, and is a mammal.
A duck says 'quack', is not carnivorous, and is not a mammal.
```