INTPROG TB2: OBJECT ORIENTED PROGRAMMING

Practical Worksheet 8: Inheritance in Java

1. Introduction

The practical session this week will be focusing on implementing inheritance using Java. The principles of inheritance remain the same as those experienced in Python. However, there are some subtle differences between Java and Python – particularly that Java does not have access any instances variables which are set to private. In order to overcome this, we use set the variables which are required by the subclass to a protected access modifier. If there are variables within the superclass which are not needed by the subclass, they must remain private.

As usual, work through this practical session at your own speed. Make sure you understand everything included within the example before you move onto the questions at the end of the worksheet. If you have questions, either ask your practical tutor, or ask a question using the anonymous question form on Moodle.

2. Example – Ball Inheritance

In this week's practical we will be using an example of balls which are used within sports – namely footballs, and rugby balls. These will both inherit features from the Ball class, but will need to have their own features. On Moodle, you will find the start of this project – called "BallInheritance". This class has already implemented the complete Ball object, and a class called BallStorage, which will be used to store any balls we create in an ArrayList, so that we can operate on every item in the collection.

Firstly, we are going to implement the Football. Create a class called "Football". As Football inherits from the Ball class, we need to use the "`extends`" keyword. In addition to the instance variables in the superclass, the football will also have a second Color variable which will set the alternate colour for the ball. This variable is passed as a parameter, and set within the constructor.

```
import java.awt.*;

public class Football extends Ball
{
    private Color alternateColour;

    public Football(double startX, double startY, double xVel, double yVel, Canvas canvas,
Color firstColour, Color secondColour)
    {
        super(100, firstColour, xVel, yVel, canvas, startX, startY);
        alternateColour = secondColour;
    }

    // rest of class goes here...
}
```

The Football class will override a number of methods within the superclass. The first one which we will consider is the drawBall() method. It will firstly call the superclass method, to draw the basic ball background. Once this has been completed, it will draw concentric circles using a for loop.

```
public void drawBall()
{
    super.drawBall();

    for(int i = 0; i < 4; i++)
    {
        if(i % 2 == 0)
        {
            win.setForegroundColor(alternateColour);
        }
        else
        {
            win.setForegroundColor(colour);
        }

        double divisor = i + 1.25;

        win.fillCircle(currentX, currentY, diameter/divisor);

    }

}
```

The next method we need to override is moveBall() method. In the superclass, the method updates the currentX and currentY variables depending on the direction the user has entered. However, when using the Canvas class, we need to erase the shape, then redraw it in the new position. Therefore, we need to create a method which erases all the elements of the Football before updating the position coordinates, and redrawing the ball.

```
public void eraseBall()
{
    super.eraseBall();

    for(int i = 0; i < 4; i++)
    {
        double divisor = i + 1.25;
        win.eraseCircle(currentX, currentY, diameter/divisor);

    }

}

public void moveBall(char direction)
{
    eraseBall();
    super.moveBall(direction);
    drawBall();

}
```

The final method we need in the create in the Football class is the printInformation() method. This will call the superclass version of the printInformation() method, then add the additional information which is provided by the subclass.

```
public void printInformation()
{
    super.printInformation();
    System.out.println("Secondary Colour: " + alternateColour);
    System.out.println("Volume: " + calculateVolume());
}
```

This completes the Football class. Next we will write the RugbyBall class. Below is the code for this class – before entering it into your project, work your way through it and try to understand what each method is doing, and why.

```
import java.awt.*;

public class RugbyBall extends Ball
{
    private double height;

    public RugbyBall(double startX, double startY, double xVel, double yVel, Canvas canvas,
Color colour)
    {
        super(200, colour, xVel, yVel, canvas, startX, startY);
        height = 100;
    }

    public void drawBall()
    {
        win.setForegroundColor(colour);
        win.fillSemiCircle(currentX, currentY, diameter, height, false, true);
        win.fillSemiCircle(currentX, currentY, diameter, height, false, false);
    }

    public void moveBall(char direction)
    {
        eraseBall();
        super.moveBall(direction);
        drawBall();
    }

    public void eraseBall()
    {
        win.eraseSemiCircle(currentX, currentY, diameter, height, false, true);
        win.eraseSemiCircle(currentX, currentY, diameter, height, false, false);
    }

    public double calculateVolume()
    {
        double volume = (4/3) * Math.PI * diameter * Math.pow(height, 2);

        return volume;
    }

    public void printInformation()
    {
        super.printInformation();
        System.out.println("Volume: " + calculateVolume());
    }

}
```

In the file which contains the main method, uncomment the two lines which create two objects – one of each type of Ball. Run the method. It should show two balls moving across the screen, and then print the information about those two objects in the console window. Make sure you add a couple more objects – all of which should be either Footballs or Rugby Balls. Make sure you understand how these two classes are working before you move onto the practical questions.

3. Practice Questions

The following questions are designed to give you some practice at implementing inheritance in Java. For all the questions, make sure that you include appropriate JavaDoc comments. If you have any questions, please make sure you ask the members of staff within the practical sessions.

1. Add the following balls to the above Ball Inheritance example:
   (a) Golf ball

```
Ball Information
Primary colour: java.awt.Color[r=255,g=255,b=0]
Volume: 25132.741228718343
```

   (b) Tennis ball


```
Ball Information
Primary colour: java.awt.Color[r=255,g=255,b=0]
Volume: 201061.92982974675
Brand: S
```

2. Download the Network project from Moodle. This project contains 2 classes – a NewsFeed class, and a Post class. Look at both classes, and make sure you understand what purpose each has.
   (a) Create MessagePost class, which will inherit from the Post class. It requires an additional instance variable which holds the message. It will also need the following method: display(). The below image shows the output of the display() method. You will also need to write an appropriate accessor method for the class.

```
Alice
0 seconds ago
   No comments.
Hello World
```

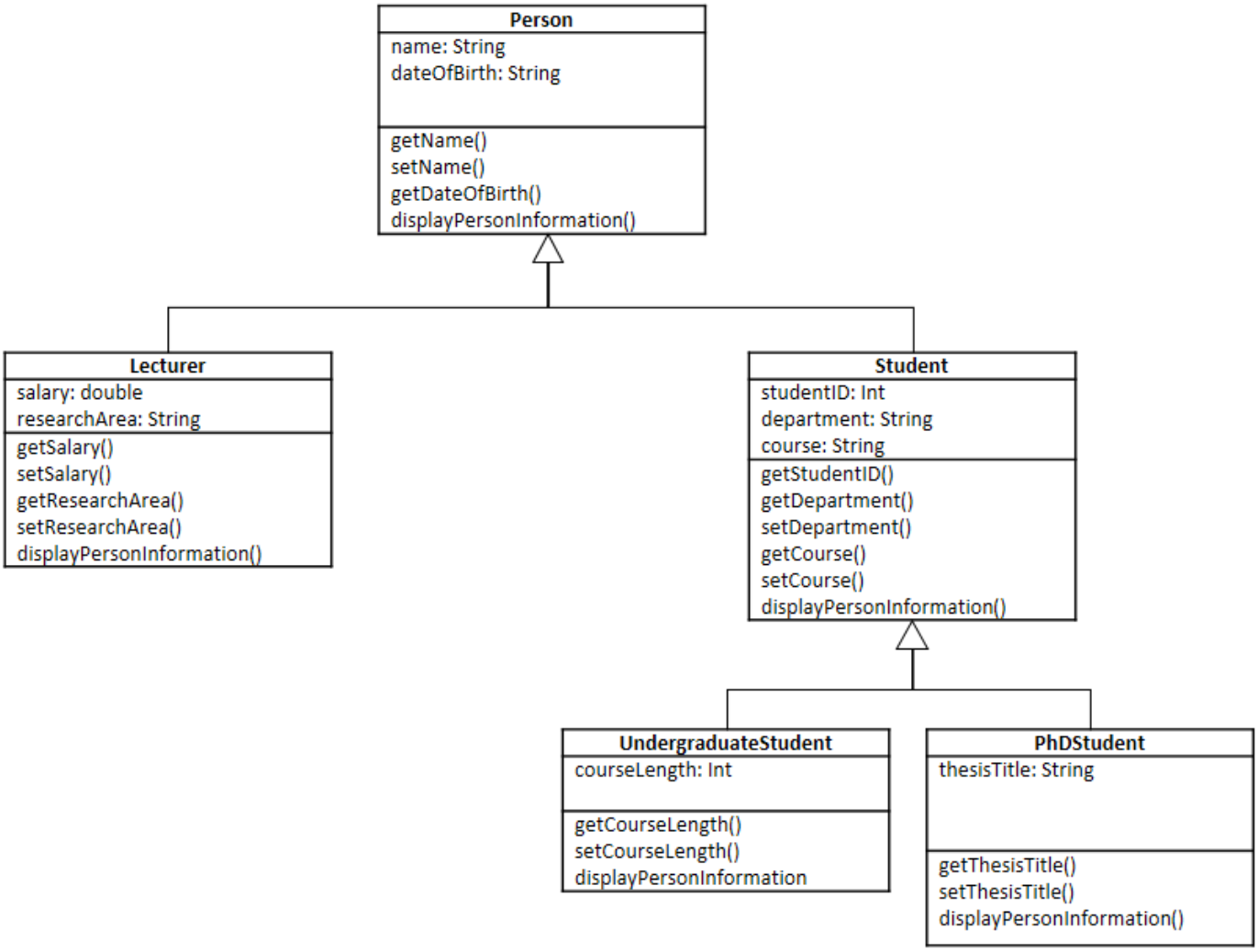
(b) Create another class called PhotoPost, which will inherit from the Post class. It will require the following additional variables: filename, and caption. It will also need a display() method, which shows the filename (in square brackets) and the caption in the Terminal window, as shown below. Also, write appropriate accessor methods for the class.

```
Alice
0 seconds ago
   No comments.
  [/img/helloworld.jpg]
  Hello World Picture
```

(c) Create a main method within another class, which instantiates a NewsFeed object. Use the NewsFeed's Post method to add the some MessagePosts and some PhotoPosts. Finally, use the show() method to display the NewsFeed.

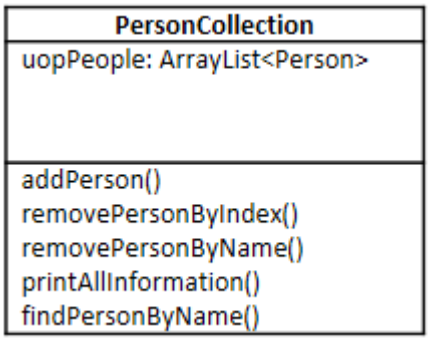
3. Start a new project called People.

(a) Implement the below inheritance hierarchy. Use method overriding to implement the displayPersonInformation() method throughout the project.



Hint: Make sure you think about your mutator methods – don't let them be assigned to invalid data. You can assume that the maximum length of a course at the university is 4 years.

(b) Write the below class, which then can be used within a main method to test whether your inheritance hierarchy has been correctly implemented.



The findPersonByName() method should take the name of a person as a parameter, and print their information.