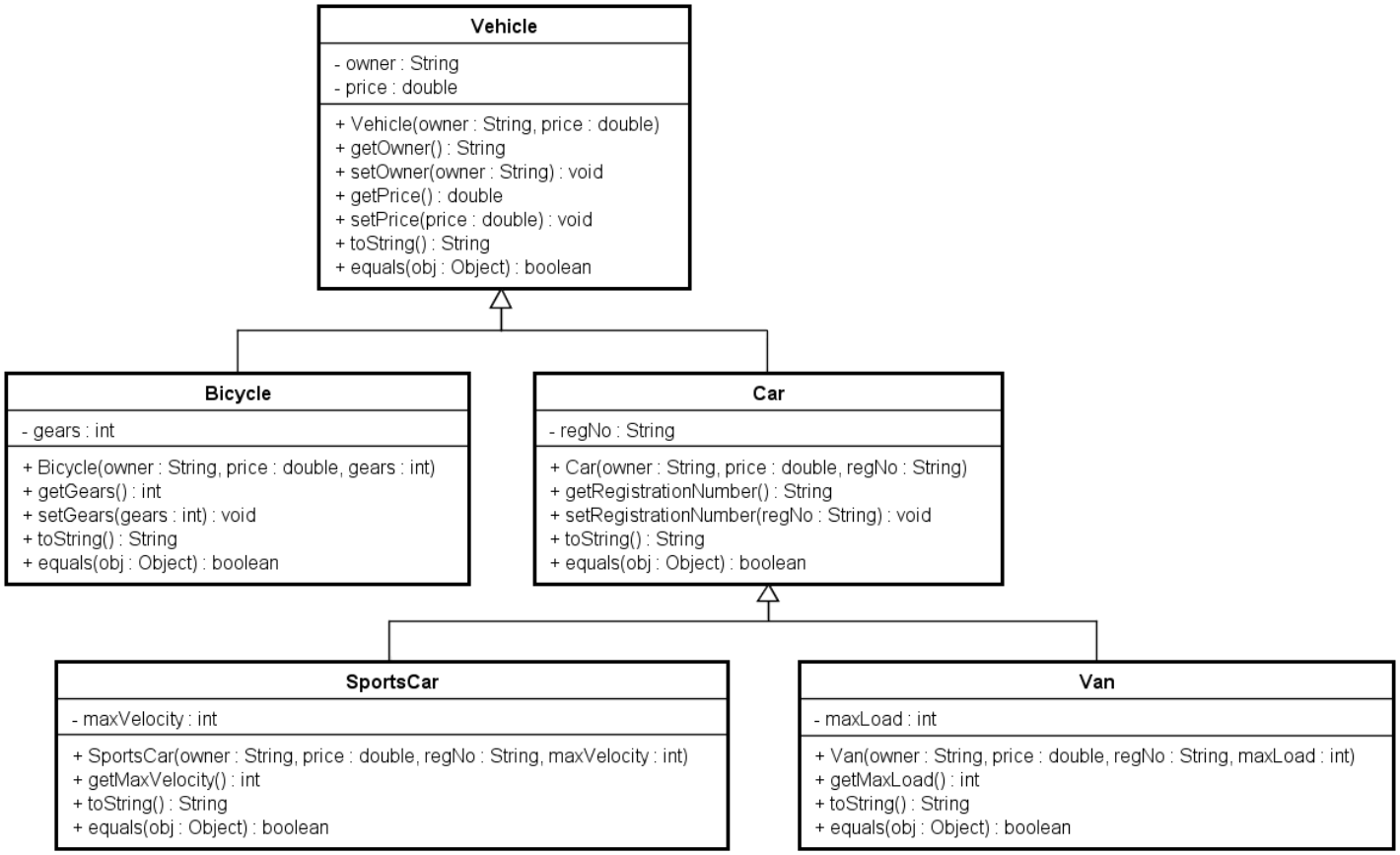
Programming Exercises - PRO1 - Session 20

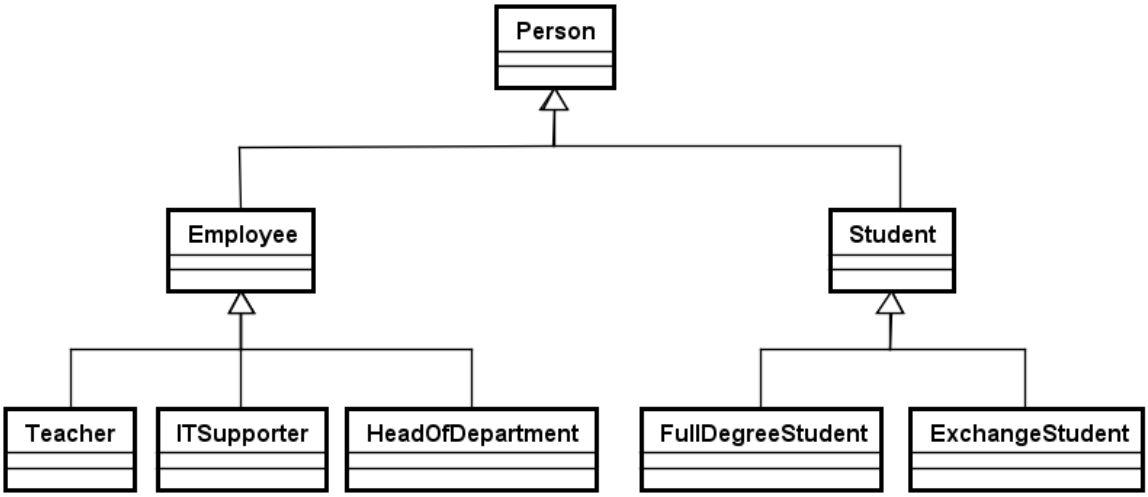
Exercise 20.01

Implement the classes shown in the class diagram below. Then create a test class to test the functionality.



Exercise 20.02

Based on the example I showed during the lessons, then create the class hierarchy shown in the class diagram below:



All classes should have relevant instance variables, plus the necessary get and set methods. Furthermore, the classes should all have a `toString` and an `equals` method.

Some of the classes you can take directly from my example, but for the others you should try to come up with some relevant instance variables yourself, and then create get and set methods for them. A few suggestions for some of the new classes could be:

In class Teacher: `String[] courses` (or `ArrayList<String> courses`)
In class ITSupporter: `String workArea` (e.g. “software” or “hardware”)
In class HeadOfDepartment: `String department`

First draw a UML class diagram of your solution (with all instance variables, constructors, and methods), and then implement the classes in Java. Then (as always) create a test class to test the functionality.

Exercise 20.03

[Gaddis] Programming Challenges 7+8, p. 699

Exercise 20.04 - An old SDJ1 test

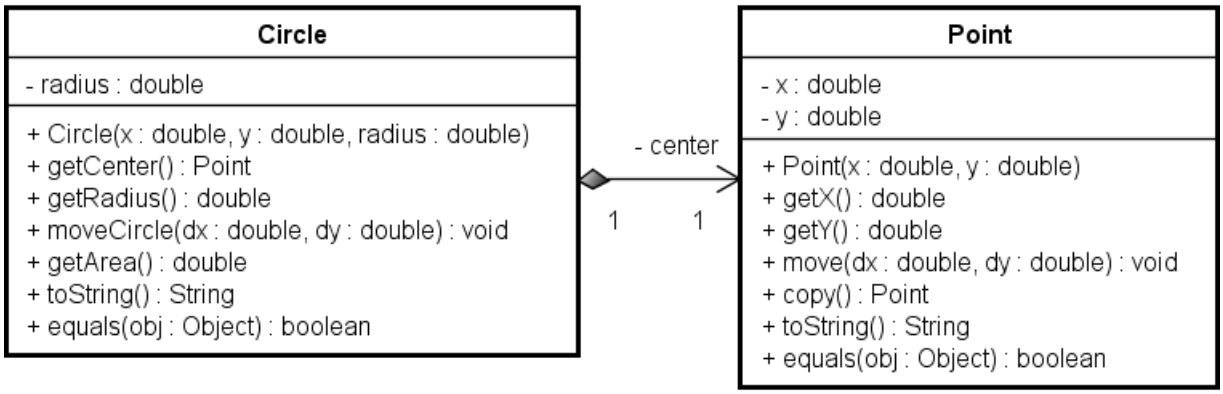
Part 1

Implement the Point class shown in the diagram on the next page. The class holds information of a point in the plane (the x-coordinate and y-coordinate). The class is partially implemented below:

```
public class Point
{
    private double x;
    private double y;

    public Point(double x, double y)
    {
        this.x = x;
        this.y = y;
    }

    public double getX()
    {
        return x;
    }

    public double getY()
    {
        return y;
    }

    public void move(double dx, double dy)
    {
        this.x += dx;
        this.y += dy;
    }

    public String toString()
    {
        return "(" + x + ", " + y + ")";
    }

    public Point copy()
    {
        // To be implemented
    }

    public boolean equals(Object obj)
    {
        // To be implemented
    }
}
```

Implement the following methods:

a) A copy method that returns a new copy of the Point object.
b) An equals method that returns true if the other Point object has the same x-value and y-value, and false otherwise.



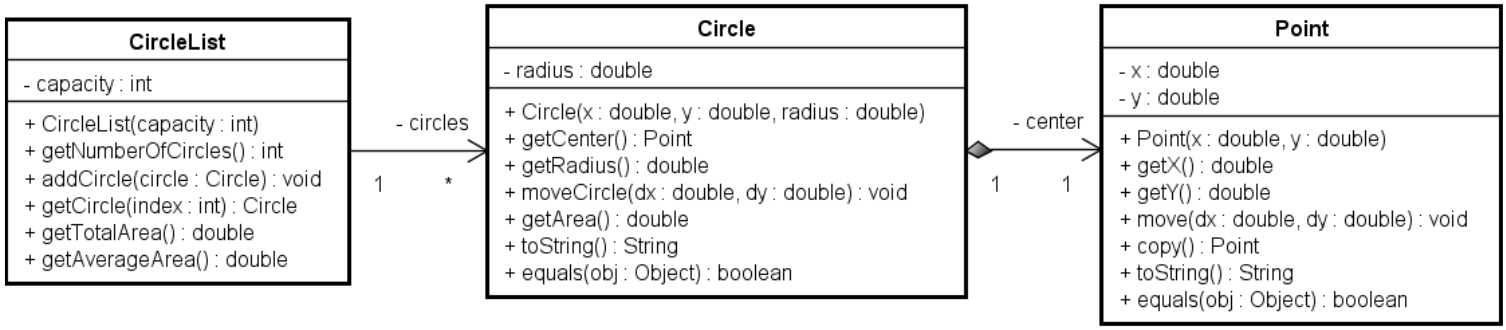
Part 2

Implement a class Circle that holds information of a circle in the plane (a center and a radius). You have to implement the Circle class from the class diagram above. That is, the field, method, and parameter names and types have to be identical to the ones used in the diagram. The relation between Circle and Point has to be implemented as a composition relationship. The class contains the following:

a) Two instance variables: center and radius.
b) A 3-argument constructor setting both instance variables. Use x and y for the center of the circle.
c) Get-methods for radius and center.
d) A moveCircle method that moves the circle, so that its new center is in (x + dx, y + dy).
e) A getArea method, that returns the area ($r^2*\pi$) of the circle. Use Math.PI to get the value of π.
f) A toString method that returns all the information of the object.
g) An equals method that returns true if the other Circle object has the same center and radius, and false otherwise.

Part 3

Now implement the class CircleList exactly as it appears in the diagram below.



The class contains the following:

a) Two instance variables: circles and capacity. The circles variable should be able to store a number of Circle objects, and capacity should indicate the maximum number of Circle objects that can be stored in the class.

b) A 1-argument constructor.
c) A getNumberOfCircles method returning the number of Circle objects currently stored.
d) An addCircle method adding the Circle given as argument to the stored circles.
e) A getCircle method returning the Circle object stored at the index given as argument.
f) A getTotalArea method returning the sum of the areas of the stored Circle objects.
g) A getAverageArea method returning the average of the areas of the stored Circle objects.

Part 4

Implement a test class (CirclesTest) with a main method and test the classes in the following way:

a) Create a Circle object.
b) Get the center of the Circle object and store it as a Point object.
c) Move the Circle object 150 to the right.
d) Make a second Circle object with a center located at the coordinates of the Point object.
e) Now create a CircleList object with a capacity for storing a maximum of 3 Circle objects.
f) Add the two Circle objects to the CircleList.
g) Print out the total area and the average area of the Circle objects in the CircleList.
h) Run through the CircleList and print out all information of each of the stored Circle objects