Lab 6: Objects and Classes

Due on Friday Oct 18, 2019 by 11:59PM

Make sure you follow the style guidelines (https://google.github.io/styleguide/javaguide.html#s4-formatting) and refer to the submission guidelines.

Goals

The main goals of this project are:

- Get practice solving problems end-to-end
- Practice the following topics:
  - Design Classes and draw UML class diagrams
  - Java Classes
- Get some easy lab points.

Instructions

Programming and UML. Complete the following problems. They involve writing Java programs and creating UML class diagrams in some cases:

1. Design a class named Square to represent a square. The class contains:

   - A double data field named width that specifies the width of the square. The default values are 1 for the width.
   - no-arg constructor that creates a default square.
   - constructor that creates a square with the specified width.
   - method named getArea() that returns the area of this square.
   - method named getPerimeter() that returns the perimeter.

   Draw the UML diagram for the class and then implement the class. Write a test program that creates two Square objects—one with width 40 and the other with width 35.9. Display the width, area, and perimeter of each square in this order.

2. Design a class named Bond to represent a financial bond. The class contains:

   - A double data field named **coupon** that specifies the amount of a coupon payment.
   - An integer data field named **payments** that specifies the number of coupon payments.
   - A double data field named **interest** that specifies the interest rate of the bond.
   - A double data field name **valueMaturity** that specifies the value at maturity.
   - no-arg constructor Bond() that creates a default bond.

- constructor that creates a bond with the specified coupon, payments, interest and value at maturity.
- method named getPrice() that returns the price of the bond.

The price of a bond is calculated with the following formula:

C = coupon payment
N = number of payments
i = interest rate
M = value at maturity

$$price = C * \left[1 - \left[\frac{1}{(1+i)^n}\right]\right] + M * \frac{1}{(1+i)^n}$$

Draw the UML diagram for the class and then implement the class. Write a test program that creates a bond with a coupon payment of $500, with 100 payments, and interest rate of 5% and a value at maturity of $10,000. Display the price of the bond.

3. You are building a system to keep track of matches played by a soccer team. You need to create a class to represent your matches. Call this class SoccerMatch. The class will have the following data fields and methods:
   - A Date data field called startTime to register the date and time for the start of the match.
   - A Date data field called endTime to register the date and time for the end of the match.
   - A String data field called location for the venue of the match
   - A String data field called home for the home team name.
   - A String data field called visitor for the visitor's team name.
   - An array of a Player class (to be designed also) called homePlayers of size 11 for the home team players (assume no substitutions).
   - An array of Player class called visitorPlayers of size 11 for the visiting team players.
   - An array of Goal class (to be designed also) data field called homeGoals for the home team goals (size 10).
   - An array of Goal class data field called visitorGoals for the visitor team goals (size 10).
   - A no-arg constructor to create a new match.
   - A constructor that creates a new match on a specific date and time between two teams.
   - An addHomePlayer(Player p) method that adds a player to the home team.
   - An addVisitorPlayer(Player p) method that adds a player to the visitor team.
   - A getWinner() method that returns the name of the winner. In case of a tie it returns the word "tie"
   - An addHomeGoal(Goal g) that adds a goal for the home team.
   - An addVisitorGoal(Goal g) that adds a goal for the visitor team.

- A getHomeGoals() method to get a list of the goals for the home team.
- A getVisitorGoals() method to get the goals of the visitor team.

Create a Player class for the players with the following data fields and methods:

- A String field called **name** for the name of the player.
- An int called **goals** to track the lifetime goals of the player.
- A String field called **team** for the name of team of the player.
- A no-arg constructor to create a new player.
- Getters and setters for all the data fields.

Create a Goal class to track goals with the following data fields and methods:

- An int field call minute for the minute at which the goal is scored.
- A Player fields called player for the player scoring the goal.
- Contructors, getters and setters for this class.

Draw the UML diagram for the three classes and then implement the classes. Write a test program for a simulated soccer match.

For the UML class diagrams, create a PDF document containing all your diagrams.

Submission

Place your .java and a .pdf file under the corresponding folder in your local copy of the GitHub repository, commit and push it to the remote repository. Make sure that the professor has access to the repository (jfac65-marist).

```
cmpt220lastname\
   hw\
   prj\
   labs\
      1\
      2\
      3\
      4\
      5\
      6\
         Place your files in here.
```