CSC 1302 Principles of Computer Science II

Assignment 4: Java Class, Objects

(Due on 11:59 pm, 7/6/2021)

Purpose:

A class is a blueprint or prototype from which objects/instances are created. It includes fields or methods that are common to all objects of the same blueprint/class. In general, a class declaration has components: class header, constructors, methods such as getter/setter, and so on. The constructors are used for initializing new instances. Fields are variables representing the state of the class and its objects. The methods are used to implement the behavior of the class and its objects.

In this assignment, we will practice how to define classes and create objects. Once a class is available, it's like we have a new data type. Variables can be defined with a type of this class and can refer to an object of this class. Methods can use this class as parameter type or return type.

Program #1:

Add *divide* method to take another Fraction as the parameter and return a new Fraction that is the division of current Fraction and the Fraction in the parameter. (public Fraction divide(Fraction f)).

- Add *scaleup* and *scaledown* methods to the Fraction class. The *scaleup* method will take a factor as the parameter and multiply the numerator by the factor. The *scaledown* method will take a factor as the parameter and multiply the denominator by the factor.
- Add a *scale* method which will have two parameters: *factor* and *flag*. The *flag* is *boolean*. If *flag* is true, then scale up the fraction; otherwise scale down the fraction.
- Both *scaledown* and *scale* methods must check if the factor is 0. If it is 0, a warning message is printed out and no scaling is operated.
- Add two more constructors. One of the constructors will have no parameters; it initializes the fraction to 0/1. The other constructor will have one parameter, representing the numerator of the fraction; the denominator of the fraction will be 1.

- Write a program named *FractionScale* that prompts the user of a fraction and a scale factor. Here is what the user will see on the screen:

This program performs the scaling operations on a fraction.

Enter a fraction: **3/7**

Scale up or down (1: up, 0: down): **1**

Enter a scale factor: **2**

Scaled fraction is: 6/7

You can assume that the user always enters two integers separated by a slash (/) for the faction. However, the user may enter any number of spaces before and after each integer.

```
public class Fraction {
  // Instance variables
  private int numerator;    // Numerator of fraction
  private int denominator;  // Denominator of fraction

  // Constructors
  public Fraction(int num, int denom) {
    numerator = num;
    denominator = denom;
  }

  // Instance methods
  public int getNumerator() {
    return numerator;
  }

  public int getDenominator() {
    return denominator;
  }

  public Fraction add(Fraction f) {
    int num = numerator * f.denominator +
              f.numerator * denominator;
    int denom = denominator * f.denominator;
    return new Fraction(num, denom);
  }

}
```

Program #2:

Following is a part of the source codes for NBATeam and NBA classes. Finish the programs for both classes to produce an output screen similar to the following figure.

```
Creat the NBA team of Heats ......
Add a play to Heats? (yes/no): yes
What is the name to be added? James
Add one more play to Heats? (yes/no): yes
What is the name to be added? Wade
Add one more play to Heats? (yes/no): no
Creat the NBA team of Spurs ......
Add a play to Spurs? (yes/no): yes
What is the name to be added? Duncan
Add one more play to Spurs? (yes/no): yes
What is the name to be added? Parker
Add one more play to Spurs? (yes/no): no
Game on Now ......
Spurs ***WIN the series***
Heats[ James Wade ] win #: 2 los : 4
Spurs[ Duncan Parker ] win #: 4 los : 2
```

//File NBA.java

```
import java.util.Scanner;

public class NBA {
  public static void main(String[] args) {

            Scanner input = new Scanner (System.in);
            String ifAddPlayer;
            String playerName;

            //construct Team Heat
            System.out.println("Creat the NBA team of Heats ...... ");
            NBATeam heat= new NBATeam("Heats");
            System.out.print("Add a play to Heats? (yes/no): ");
            ifAddPlayer= input.next();

            while (ifAddPlayer.equalsIgnoreCase("yes")){
                  System.out.print("What is the name to be added? ");
                  playerName=input.next();
                  heat.addAPlayer(playerName);

                  System.out.print("Add one more play to Heats?
(yes/no): ");
                  ifAddPlayer= input.next();
             }
```

```
            //construct Team spurs

.............................//Your code here


        /*simulate a series of atmost 7 games by generating a random
        number; if the random number is bigger than 0.5, Heat wins;
        otherwise Heat losses a game. As soon as team wins 4 games, the
        series is over. */
.............................//Your code here

            System.out.println(heat);
            System.out.println(spurs);

    }
}
```

//File NBATeam.java

```
public class NBATeam {
    private String sTeamName;
    private int nWin;
    private int nLoss;

    private String [] playerArray;

.............................//Your code here


}//end of class definition
```

Criteria:

1. Upload all of the .java and the .class files to the CSc1302 dropbox on http://icollege.gsu.edu.
2. Your assignment will be graded based on the following criteria: (a) Are your programs runnable without errors? (b) Do your programs complete the tasks with specified outputs? (c) Do you follow the specified rules to define your methods and programs? (d) Do you provide necessary comments include the programmer information, date, title of the program and brief description of the program.

3. Make sure that both the .java and .class files are named and uploaded to icollege correctly. If any special package is used in the program, be sure to upload the package too. Should you use any other subdirectory (whatsoever) your program would not be graded, and you will receive a **0 (zero)**.
4. No copying allowed. If it is found that students copy from each other, all of these programs will get **0**.