CSC 2220 – Programming in Java
Fall 2020 Semester, Block A
Programming Assignment 3
Due Date: Wednesday, September 16, 2020, by 11:55pm, to Moodle

General Rules of Engagement

Unless otherwise stated, you are not to collaborate on this, or any, assignment. Refer to the Cheating Policy in the syllabus for more information.

When you turn in on Moodle, you'll probably have several Java classes to submit. Submit everything via .zip file. All of the .java files that are necessary.

Although you won't necessarily see a deduction, do take care to make sure spelling and spacing have been addressed and are correct.

Make sure to comment your code. You should have a header for each of your .java files, which includes name and description of what the class does. You should also have comments throughout your code. Failure to do so will result in a loss of points.

No late work will be accepted. If you are up against a deadline and Moodle isn't responsive for some reason, remember you can always email your zip file to me, at rjh7g@mcs.uvawise.edu for no penalty. Just remember to use common sense.

Make sure that you name things meaningfully. This is just good programming practice and a way to self-document your code beyond writing comments.

Programming Assignment Description

In lectures, we've been working on digitizing the wooden baseball game I showed you at the beginning of the semester. Now, between the first assignment, and what you have done in class, we should be able to start working with line-ups for home and away and track stats.

As opposed to the first assignment, let's not keep score yet. Let's practice recording the results of plays as appropriate stats for a batter in the line-up.

We're going to kind of take a step back from what you did for the previous programming assignment and firm up not only the Batter class, but also new classes called BaseballPlayer and Pitcher.

For this assignment, you're given BaseballPlayer.java and BaseballGameDriver.java. You will need to utilize the BaseballGame.java file I had posted as a potential correct solution.

BaseballPlayer.java
We're playing around with inheritance and polymorphism with this assignment. In order to give you some direction, BaseballPlayer.java has been provided for you.

You'll notice some things. In the previous assignment, hits, walks, and strikeouts were all concerns just for the Batter class. It turns out that these are also vital statistics for pitchers, too. Therefore, to allow for both classes to inherit these items, they have been moved up to the top of the hierarchy. Also, baseball player has been declared with a constructor that takes three arguments, representing strings of first name, last name, and position. Hits, walks, and strikeouts are all initialized to zero. You'll also see three methods declared with the protected keyword. This is because I want each subclass (Batter and Pitcher) to be able to utilize these methods.

To be an abstract class, you need abstract methods, and there are three: toString, scorePlay, and printBoxScoreHeader. By definition, you will need to make sure that both subclasses have these methods and they are defined. The method toString returns the object (in whichever class) as a String. And there's a certain order to follow. See output below for guidance. The same story is true with printBoxScoreHeader, as this prints out each column header, if you will, in the order that it needs to be printed. Don't forget to format your output.

The last method, scorePlay, behaves differently for each class. The two arguments are the same: you're given a PlayResult value and an integer value representing the number of runs. For Batter, the runs are added to the object's RBIs (runs batted in). For the pitcher, these runs are added to the earned runs attribute. You'll need to update certain stats based on the play for a Batter; the rules are the same as before. For the Pitcher class, you're interested in outs. Sac fly, flyout, etc., are each one out. Double plays are two (just go with it for now…we'll worry about game situations later). You'll also need to record for hits, walks, and strikeouts.

Batter.java

Other than the changes that have been outlined above, you should largely retain the functionality that you used in Programming Assignment 2.

Pitcher.java

Additional data that you'll need to include for this class includes earned runs and innings pitched. These are necessary for calculations.

You'll need methods to determine earned run average (ERA), walks and hits per inning pitched (WHIP), and strikeouts per nine innings (k/9).

To calculate ERA – earned runs divided by (innings pitched divided by 9). It should be a decimal value. Print out to two decimal places.

To calculate WHIP – add up total walks and hits and divided by the number of innings pitched. Print to two decimal places.

To calculate k/9 – strikeouts divided by (innings pitched divided by 9). Should be a decimal value. Print to two decimal places.

BaseballGame.java

You'll need to update this file with one additional method, called print. It has an argument of BaseballPlayer. Call both the printBoxScoreHeader method, and print out the BaseballPlayer's information. You get to see Polymorphism in action here, combined with…

BaseballGameDriver.java

This file will be provided to you, as well, as it's a driver demonstrating the capabilities of Batter and Pitcher. When I test your submissions, I will utilize this driver, so I highly recommend not altering the file when you do your final testing.

Two objects are set up, as well as an array. Things are filled in and updated. BaseballGame's print gets called. You also get to see polymorphic behavior kind of take place with the scorePlay calls. You definitely see it at the end, with the printing.

Last notes:

Other than making sure you print out the correct data in the correct order, I don't care how you implement your classes.

You'll notice that I pass in 54.33 for innings pitched, but it prints out as 54.1. Baseball is weird with its notation. .1 means a third of an inning; .2 means two-thirds of an inning, and .0 means a complete/whole inning.

Remember a couple of things here: one, practicing polymorphism. Two, back at the beginning of the semester, I said we would do refinement with the game, mainly because we talk about it and never implement it with our programs. Hopefully you're starting to see us doing that here.

Program output

Your output should match mine. Here's the output from the provided driver file:

```
 ----jGRASP exec: java BaseballGameDriver
First Name          Last Name           Pos  ab   r    h    2b   3b   hr   rbi  sf   bb   so   obp       slg       avg
Fernando            Tatis, Jr.          ss   175  45   53   10   2    15   39   0    23   47   0.384     0.640     0.303


First Name          Last Name           ip        er   h    bb   so   era       whip      k/9
Zach                Davies              54.1      15   37   15   50   2.48      0.96      8.28


First Name          Last Name           Pos  ab   r    h    2b   3b   hr   rbi  sf   bb   so   obp       slg       avg
Fernando            Tatis, Jr.          ss   179  46   55   10   3    16   44   0    23   48   0.386     0.665     0.307


First Name          Last Name           ip        er   h    bb   so   era       whip      k/9
Zach                Davies              54.2      17   38   16   51   2.80      0.99      8.40
```

Submission

Submit your .java files via zip file by 11:55pm on Wednesday, September 16. If you feel it necessary, you may utilize the files we've worked on in class.