CS 1331 Homework 5

Due Thursday, February 14th , 2013 8:00pm

Introduction

This week the homework covers more on classes, OOP design, and javadocing.

You will be creating two separate programs. The first is very similar in format to the last homework; We provided you a driver file, and you must make the java file for the class that it uses.

The second portion is much different; here, you are actually creating an entire program on your own. You must have the functional requirements we list, but we will not be giving you any of the specific method headers. It is up to you to decide how exactly to create your class, but make sure that whatever you do follows good Object Oriented design.

If you get stuck, feel free to post on Piazza or come to office hours!

5.0 Javadoc

Now that we are making some classes that relate to each other in more complex ways, we want to start documenting our code so that someone can look at the documentation and know what is going on with our programs.

Throughout the semester you should have been using the Java API quite a bit; all of what you have been reading on there is the product of javadocs! The authors of the code for each class wrote up a lot of helpful documentation so that you can know what exactly each method does without having to actually look at the code.

From here on out, you will be adding javadocs to all of your code.

The style guide in the resources section describes how javadocs work and what they should look like. I have also uploaded it with this assignment. Please refer to that document for detailed instructions.

Remember that you need to javadoc all class and method headers!

5.1 Counter.java

For this part of the assignment, your job is to create a class that keeps track of the number of items of a certain type that you currently have, as well as the overall number of items that are being counted. Each Counter object will have an item that it is counting, the individual count of that item. Counter will also have a static variable to keep track of the overall counts.

Here is an outline of the variables and methods your class should have:

- Instance Variables(make sure these are the correct visibility!)
    - String itemName
    - int count
- Static Variables (again, make sure the visibility is correct)
    - int totalCount
- Methods
    - Constructor
        - `public Counter(String itemName, int count){…}`
    - reset method
        - `public void reset(){…}`
        - This method should set the count for the item to 0. Make sure you correctly change the total count as well
    - methods to change the count
        - `public int changeCount(boolean increment) {…}`
        - `public int changeCount(boolean increment, int amount) {…}`
        - The boolean variable indicates whether the method should increase or decrease the count by the given amount.
        - The method that does not take in an amount should change the count by one.
        - For good OOP, think about how you can call one of these methods from the other to avoid copying code
    - getters
        - `public int getCount(){…}`
        - `public static int getTotalCount(){…}`

5.2 BakeSale.java

We provide this file for you (**DO NOT MODIFY IT – although if you want to change the initial item values for testing purposes you may**) – all you need to do is make sure that you write your Counter class to the specifications listed above. When you compile and run your program, you should get output that looks something like this:

```
Welcome to our bake sale!
Would you like to buy something? Or do you have a delivery? buy
We have 24 cupcakes, 30 brownies, and 12 cookies.
What item would you like to buy? cupcake
Here you go!

Welcome to our bake sale!
Would you like to buy something? Or do you have a delivery? buy
We have 23 cupcakes, 30 brownies, and 12 cookies.
What item would you like to buy? all the brownies
Wow! Here are all of our brownies :]

Welcome to our bake sale!
Would you like to buy something? Or do you have a delivery? buy
We have 23 cupcakes, 0 brownies, and 12 cookies.
What item would you like to buy? brownie
I'm sorry, we don't have any more brownies :[

Welcome to our bake sale!
Would you like to buy something? Or do you have a delivery? delivery
Great! What item do you have for us? cupcakes
How many do you have? 6
Thanks!

Welcome to our bake sale!
Would you like to buy something? Or do you have a delivery? buy
We have 29 cupcakes, 0 brownies, and 12 cookies.
What item would you like to buy? all the cupcakes
Wow! Here are all of our cupcakes :]

Welcome to our bake sale!
Would you like to buy something? Or do you have a delivery? buy
We have 0 cupcakes, 0 brownies, and 12 cookies.
What item would you like to buy? all the cookies
Wow! Here are all of our cookies :]

We sold out! Goodbye
```

5.3 RPSPlayer.java

For the second major part of this homework, you will be making a program that will allow the user to play Rock Paper Scissors against the computer. In order to do this, we first want to create a class to define our players. The way we are writing this, we could later implement a system that has two human players instead of one human and one computer; we are writing our code in such a way that it is not tied to one specific implementation.

Your player should have a name, a score, and a choice (rock, paper, or scissors – this should be a String). A player should be able to take in a new choice, give its choice to the system, increase its score, give its score to the system, and give its name to the system.

5.4 RPSGame.java

This file will contain the logic for a single Rock Paper Scissors game. It will have the two players, a way to set their choices, and a way to determine who won based on those choices. When it determines who won, it should also increment that player's score. We will then create instances of this class in our RPSTournament class.

It should also have String constants for rock, paper, and scissors. This way, if we change how we represent Rock Paper and Scissors, we only need to change it in one place.

This class should NOT prompt for any user input. It should simply have a method that takes in two choices and assigns those to the correct user. We want to design our class like this so that it will work with a text based implementation like we are writing here, or with something GUI based. We are abstracting the logic away from the actual implementation (it's good program design :]).

5.5 RPSTournament.java

This is the class that will do all of the user facing work. It will consist of the main method, and a helper method for generating the computer's choices.

You will need to get the player's name, and then create an RPSPlayer object for the user and one for the computer. You should also prompt the user for the number of games they would like to play. Your program should allow the user to play as many or as few games as they would like; for every game, you must create a new RPSGame object, prompt the user for their choice, generate a computer choice, and use the RPSGame object to determine who won. After every game, the current score should be printed to the user.

In addition to the main method, you should write a static method to assist with generating user choices. This method must be static so that we can reference it from the main method. We are making this a separate method because it is something that we want to do again and again, and the code can easily be extracted into a separate method that makes sense. You will need to use a Random generator here, and translate the random numbers you get into the choices of Rock, Paper, or Scissors.

When you compile and run your program, you should get output that looks something like this:

```
Please enter your name: Elizabeth
Enter the number of games you wish to play: 3

Elizabeth, enter your choice: rock
Your Rival chose scissors
Elizabeth won this game.
Current Score:      Elizabeth - 1       Your Rival - 0

Elizabeth, enter your choice: scissors
Your Rival chose scissors
You tied!
Current Score:      Elizabeth – 1       Your Rival - 0

Elizabeth, enter your choice: paper
Your Rival chose rock
Elizabeth won this game.
Current Score:      Elizabeth - 2       Your Rival - 0

Elizabeth won, with 2/3 total games.
```

Turn-in Procedure

Turn in the following files to T-Square. When you are ready, make sure that you have actually **submitted** your files, and not just saved them as a draft.

- Counter.java
- BakeSale.java
- RPSGame.java
- RPSPlayer.java
- RPSTournament.java

Note** Always submit .java files - never submit your .class files. And be 100% certain that the files you turn in compile and that the program runs - submissions that do not will receive an automatic 0. Also, make sure that your files are in on time; the real deadline is 8 pm. While you have until 2 am to get it turned in, we will not accept homework past 2 am for any reason. Don’t wait until the last minute!

Verify the Success of Your HW Turn-in

Practice "safe submission"! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
    a. It helps insure that you turn in the correct files.
    b. It helps you realize if you omit a file or files. (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
    c. Helps find last minute causes of files not compiling and/or running.