SPECIFICATIONS

Project #1

As discussed in class, in this project you will write an application that will design a class called aDie which will simulate an actual die. That class should have a method called roll() which returns a random integer between 1 and 6 (inclusive), just like a real die. In order to control the sequence of numbers generated by your aDie class you will provide a constructor that takes a seed. You will also have a default constructor that uses the default seed mentioned in class.

You will use 2 picture boxes in your GUI to display the faces of two dice.

You will use a button to control when you roll the dice. The button should have the text "ROLL!" on it.

You will also have another button with the text "STATISTICS!" and a list box (or ComboBox) which will hold the number of rolls you want to make in order to see if your die is fair. The list box will have the following choices: 60,600,6000 and 60000.

You will use a TextBox to specify the seed. You will use a label in front of the TextBox to indicate that the TextBox is to hold the seed. If the TextBox is empty this means that you want to use the default seed. When your application starts, the seed TextBox should contain the number 999. When you (or the TA) are testing the application the seed will always start with 999. Then after hitting RESET you can change the seed to see what happens.

When you click on the "STATISTICS!" button you will roll each one of the two dice the number of times specified by the selected number in the listbox.

You will be using a Chart control to display a histogram for each one of the two dice. That histogram will display 2 series which will display how many times each face appeared for each die. A fair die should have each face show up roughly the same number of times. This should be reflected in your output.

The updating of the chart should be in real-time. This means that as you roll the dice, the histograms should be updated after a certain number N of rolls. An appropriate approach is to pick N to be some fraction of the total number of rolls. For example, you might decide N to be 1/100th of the total number of rolls. If you are doing 6000 rolls, you would update the chart every 6000/100 = 60 rolls.

You will display the mean value of the rolls for each die as well as the minimum and maximum counts and the associated face. This would look something like this:

Stats: Mean 4.4 ,Min Count: 988 Face 3, Max Count: 1010 Face 4 (Your mileage may vary). This means that the average of all your rolls is 4.4, that the number 3 showed up the least amount of times with a count of 988 and that the number 4 appeared the most often with a count of 1010.

Finally you should have a button with text “SUM OF 2 DICE”. When that button is pressed you should roll the dice by using the selected number of rolls from the list box. For each roll, you will calculate the SUM of the faces shown by the 2 dice. And, as in the case of the generation of single die histogram, you will display the histogram of the SUM of the 2 dice.

You should also have a “RESET” button to clear your display before asking for more rolls, recompute your stats (perhaps with more rolls) or compute the sum of 2 dice.

Every class, methods, properties, and member variables your solution must have an XML comment. Here is an article on why and how you should comment your code: C# Comments: A Complete Guide, Including Examples - SubMain Blog

You should also look at C# Documentation: A Start to Finish Guide - SubMain Blog

Each missing comment will cost 5 points! Don’t skip them! Commenting is a habit you absolutely want to develop! Those comments will then help you with your code by documenting your classes, methods, properties and member variables. Visual studio will use those comments to help you in coding as well as others who might be using your classes!

The names of your controls should follow the convention we discussed in class.

CAVEATS!

One major problem we discussed in class is what happens when multiple RANDOM objects are created in a very short amount of time. Think about how you will solve that problem.

Also, why is there only one SEED input in the GUI when we have multiple dice?

You will need to think hard about how to implement your solution with a SINGLE RANDOM object!!!

It is HIGHLY RECOMMENDED that you take a look at the Microsoft Documentation about STATIC at Static Classes and Static Class Members - C# Programming Guide | Microsoft Docs , in particular the 2nd part on STATIC MEMBER VARIABLES!!!

You are not allowed to use If-Then or Switch statements to update the counts of the faces in your solution! In other words, you cannot have something like this:

If[face == 1] count1++;

Else If[face == 2] count2++;

up to Else If[face == 6] count6++;

Imagine if you wanted to use this approach with a die that has 100 faces!! If this doesn’t make sense to you, it might be time to review the use of Arrays (or Lists) or to ask questions in class.

GOAL:

This project will help you better understand INHERITANCE, ENCAPSULATION, STATIC Members and SINGLETONS. It will also help you learn to use Lists in C# as well as the use of CONTROLS and handling EVENTS in GUIs. You will also learn how to use RANDOM objects, especially when you have multiple classes using such objects.

It will also teach you the importance of comments in making your code understandable.

BEWARE!

Someone has already shown me a partial solution found on the internet. What was suspicious is that that student didn't understand anything about the solution!

Today, one can find almost anything on the Internet. The problem is that you don't know what might be in that solution. If later, in your professional life, you simply search for solutions to problems (even partial problems) on the Internet, you might add code you don't understand in your product. Then, unbeknownst to you, there might be a vulnerability in the code you inserted in your product. Perhaps that code creates a backdoor to your system or purposefully corrupts your data or somehow modifies your data or spies on your company's product. DO NOT DO THIS. If your solution is flagged by Turnitin as having been copied from the Internet, you will receive an F for the course!

Write YOUR OWN SOLUTION! It is the best way to learn. Furthermore, you have to write comments about your solution. That will be quite difficult if you copy code from somewhere else or someone and don't know what the code is doing. The comments will be scrutinized to see if they actually explain your code (class, method, etc…)

The report should be an explanation of why you designed your project the way you did. This can be a separate report (in a separate text file put in your project and called report.txt) or by writing a number of explanatory comments interspersed in your code.

This project is not an open source project where multiple programmers work together on a project's code scrutinize each other's code for errors, bugs, viruses, etc …

Grading: If your project works, **<u>you start with the maximum score of 100 points</u>** however, you lose different amount of points each time a feature fails or is not implemented.

1) If you don't have comments or a report, you lose 20 points. So make sure you provide a separate report or that your comments are numerous enough to provide a good explanation of what you are doing.

2) If you don't have a continuously updating histogram you lose **<u>50 points</u>**. So, make sure you implement this functionality.

3) You must provide all the controls mentioned in the specifications. Each missing control will cost you **10 points.**

4) You **MUST provide the aDie class**. If not, you lose **50 points**. So, create that class appropriately, as discussed in class!

    a. The 2 different constructors must exist. Lose **10 points per missing constructor**. You should NOT lose any points under this requirement.

    b. You **MUST provide the aDie** class and it **MUST inherit from the Random class**! Failure to do so will have you lose **30 points**.

    c. **CHANGE:** After analyzing the pros and cons of different approaches in class you should now understanding that inheriting **DIRECTLY** from the Random class is not a good solution. As such, we discussed the benefits of creating an intermediate class, called **aRandomVariable**, that would contain a **Static Random member**. It is from the aRandomVariable that you can derive your aDie class from, as well as any other class similar to aDie (e.g. aCoin, aDeckOfCards, aRouletteWheel, etc…). **YOU MUST EXPLAIN, in your comments about the class, why this is a good approach!** If there is no explanation, you will lose 30 point! This is where you really show that you understand this design and why it is better than the alternatives.

5) If your code doesn't compile cleanly (no errors, no warnings) you will receive a 10% to 20% deduction depending on the severity of the warning.

6) Your executable must run. If not, you will automatically receive a grade of 0.

7) If your code doesn't display the histogram appropriately you will suffer a penalty of at least 50% depending on the cause of the error.

You will submit your project through canvas. You will "Clean" the project (use the Build->Clean Solution), zip your project and then submit the zipped file. The project should be named yourNameProject1.

This is an individual project. Each student is to submit their own project and report. Please ask your TA if there are any questions.

You will be given ample time to get this project done, but as always, you should start it early so that you can have the time to fix any problem you will surely encounter. My advice is that you start early and not wait until even the last week or days.