ICT-4361 Homework 3a

Purpose

This exercise to to provide you an opportunity to extend the previous exercise, and to understand static and instance methods and attributes.
Extending our dice exercise will show how to create multiple constructors, and apply another layer of abstraction on top of our class. We will be moving our dice class to handle configurable sides, and allowing different dice to have different numbers of sides.
We will be using multiple Dice objects and compute the total number showing.

Dice with different numbers of sides

What to Hand In

Please hand in a listing for each program requested, formatted in an easy-to-read style, and following our Java conventions.

Ensure your name, and the name of the file is available in a comment at the top of each submitted file (excepting screenshots).

Also, ensure that you have a sample of the output from the program.

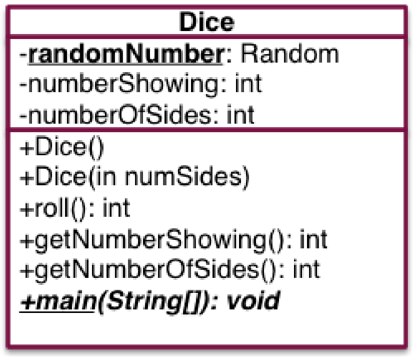
If your program fails to compile, hand in your error listing as your output.

For each question asked, provide one or two sentences summarizing your answer. Please be both complete and succinct.

Submit the files comprising your submission in a single "zip" or similar archive through the assignment's submission button. File types of .java, .txt, .jpg, and .png are preferred within the archive.

Problems

I. Different Dice for different folks .
1. Modify your `Dice` class to add a new private int field, `numberOfSides`.
2. Create (or modify) a default constructor for the Dice class that sets the number of sides to 6. It should set the numberOfSides field for that instance of dice to 6.
3. Add a new constructor that takes an integer argument that allows you to create a different number of sides for a dice. It should set the numberOfSides field for that instance of dice to the argument.
4. Make the `Random` instance a *static attribute*. That is, you will have one instance of the `Random` class that services all the Dice objects.
5. Modify your `roll()` method to take into account the `numberOfSides` on the Dice object.
   You should not add any parameters to Dice `roll()` method.
6. Modify (or create) the `main` method of the `Dice` class, ensuring you account for the new range of totals, and test the new constructor.



Detailed class diagram

7. Capture the output of this run.

II. We totally get it!
   1. Create a new Dice class client called `TwoDice`, with a `main` method.
   2. In the main method of `TwoDice`, create two Dice objects, called `diceOne` and `diceTwo`.
   3. Initialize `diceOne` using the zero-parameter constructor. Initialize `diceTwo` with the integer constructor, creating a dice with four sides.
   4. Create an int array called `totals` that is large enough to have an index equal to the highest sum of the two dice's `numberShowing`.
   5. Set up a loop to run 100,000 times (or more), with the maximum being the value of a variable in the main method.
   6. In each iteration of the loop, roll each dice, and add one to the value of `totals[sum]`, where sum represents `diceOne.getNumberShowing() + diceTwo.getNumberShowing()`
   7. After the loop has completed, the descriptions of the Dice should be output, as well as the contents of the totals array. Each index represents a possible total, and each value in the array represents how many times that total occurred.
   8. Capture the output of this run.

III. Imagine that we wished to change the Dice class to allow for Dice not sequentially numbered from one to the number of faces (say, a backgammon doubling cube). Describe (in a short text paragraph) what would need to change to support this approach.

Backgammon doubling cube: 6-sides with values 2, 4, 8, 16, 32, 64

Notes

- The new fields in Dice should have `private` visibility.
- Be sure to have public getters available for the numberShowing and numberOfSides fields.
- Remember to initialize your instance of `Random`!

Evaluation

Criteria	Weight
New instance attribute	15%
Default constructor	15%
Integer constructor	15%
New static attribute	15%
Problem I main program and output	15%
Problem II main program and output	15%
Short textual answer to question III (in a text document, or comment in the code)	10%