CSC-240 Lesson 4: More on Methods

Assignment 4

These exercises are based on those presented in Chapter 6 as Exercises 6.29 and 6.30/6.31 (but are not identical). These are exercises in both using methods from pre-defined classes, such as **Random**, and in defining methods in your own classes. The complete instructions are below, where you are to implement four classes in the indicated files:

- ***The Coin Tossing simulation***

  Coin.java
  The **Coin** class must be the **enum** class, defining the instances *HEADS* and *TAILS*.

  CoinTossing.java
  The **CoinTossing** class should be a simple class (with a *main* method) that uses the values provided by a **Coin** class, and performs a simulation of the tossing of a coin.

  > There must be a separate (possibly **static**) *flip* method that takes no arguments and returns a value of the **Coin** class. The result returned from the *flip* method is the random result from the flipping of a coin. You may choose which class of which this method is to be a member.

  Write the **CoinTossing** class so that it may do as many tosses as the user requests at each iteration. The user inputs an integral value saying how many additional tosses are to be performed each time (say, something like 1000000), and the program counts how many times each side of the coin appears. The program continues looping asking the user for additional tosss until the user enters “0”, indicating that the user wishes no more tosses of the coin. At this point, the program terminates.

  After each entry from the user, output the total cumulative count (adding up the total from the start of the execution of the program) of each of the possible outcomes, so that it may be determined whether the simulated coin appears to be “fair” (*i.e.*, has approximately as many *HEADS* as *TAILS*). [The program will be tested by asking for something like one million tosses, then about another million to see what the result looks like after a total of near two million tosses, and so on.]

- ***The Guessing Game***

  Guess.java
  The **Guess** class should provide the method *play* to actually play the game. Arrange that the method *play* has no arguments.

  The “game” is that the program is to choose a random number in the range 1 to 1000, and then for the player to attempt to guess that number. At each round, the user enters a guess, and is told whether the number is too high, too low, or correct. If not correct, player may try again, until the player guesses the correct number (you may, if you wish, allow the player to “resign” by entering a zero). Count the number of guesses.

  When the player guesses the correct number, display a message, “`Congratulations, you guessed the number!`” Following that show one of the three following messages.

  - If the number of guesses was fewer than 10, display, “`Either you know the secret or you were lucky!`”
  - If the number of guesses was exactly 10, display, “`Aha! you know the secret!`”
  - Otherwise, display, “`You should be able to do better!`”

  GuessingGame.java
  The **GuessingGame** class should be a simple class (with a *main* method) that creates any needed instance of the **Guess** class, sets it up for one game, and calls the *play* method of the class. When each game ends, the **GuessingGame** class should ask whether the player wishes to play another game. It should then either cause another game to be played, or terminate.

The following *About the Assignment* topic contains further explanation, directions, and hints about this assignment. Please read that topic before starting this assignment.

Important

Please follow the directions of Programming Assignment Identification in submitting your programming solutions.

If you have any questions or concerns about the Assignment or the *About the Assignment* topic, please use the Lesson Question discussion topic, or send a message by the D2L Internal Messaging system if you prefer.

Note: you may submit to the Assignment Submission Folders as many times and as often as you wish, up to the deadline time. Each submission is tagged with the date/time, and so each submission remains separate and distinct. Unless you leave instructions to the contrary, only the most recent of each file with the same name will be viewed for the purposes of grading. Details may be found in the topic *How To Submit and Get Feedback on Assignments* in the *How To* module.

Messages that accompany Assignment Submissions are read, and responded to, ***only*** when assignment submissions are graded (which is after the Assignment Submission Folder closing date/time). If you have a comment or question about an assignment, or a request for assistance, that needs an earlier response, then that comment, question or request should be made or asked *via* an Internal Message or the Discussion board, as these are usually read and answered every day.