ASSIGNMENT #1

REVIEW OF JAVA (PROCEDURAL PROGRAMMING)

WRITE A JAVA PROGRAM TO PLAY THE "MYSTERY NUMBER" GAME.

Mystery Number game: The player is asked to think of a number and keep it to herself/himself. Then the player is asked to do some calculations and give out the resulting from the calculations. Based on the calculation results the mystery number is revealed to the player.

Write a Java program

- to choose a level of difficulty for the game
- to ask the user, with a graphical message box, to think of a number
- to ask the user to perform operations on that number (again the instructions are given graphically)
- to ask the user to give to the program the result of the operations
- to tell the user what his/her mystery number is
- to ask the user whether s/he wants to play again and to repeat the guessing game until the user chooses to stop the process

About your implementation:

- Program procedurally using a top-down design that breaks the problem into smaller problems using methods (functions).
- Use the JOptionPane class for input and output (http://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html).

Come up with a test plan even before you design your program (black-box testing). Refer to section 5.6 in the textbook. As you are coding your program, add more input values to your test plan (white-box testing). You want to have complete code coverage.

A sample test plan format:

Test Case	Input	Expected Output	Actual Output	Comments

Your test plan has inputs, expected outputs, and the reason why you have chosen such inputs. Do not concentrate on erroneous input. However, if the user enters a non-numeric data, you should report the error (graphically) to the user. You do not need to use "Java exceptions" as a way of handling errors. You can simply write a Boolean function that parses the input string and determines whether the input is an integer or not.

Remember that the "button" pressed is also an input value. For instance, if the user presses "Cancel", that is a form of input as well.

A test plan

- lists input values
- lists the expected output corresponding to the input values
- lists the actual results of the program

With a test plan, you can compare and see if the results of your program match the expected outputs that you worked out by hand.

Spend quite a bit of time *designing* your program. Do not just write one long main method that does all the operations. Use several methods. Think of how you are going to represent the "operations" on the mystery number. Can you somehow implement the inverse of those operations to get the "mystery number"? Design your program in such a way that it's easy to add more operations or to delete some.

Submit a single zip (compressed) file containing:

1. A test plan including the 'actual results' produced by your program (write into the "actual results column" what your graphical window shows on the screen). Point out the differences between the "expected outputs" and the "actual results".
2. The Java file that has the source code of your program (e.g. *GuessTheNumber.java*). Note that you should follow the 'Good Programming Practices' as outlined in BrightSpace and in your textbook. Document your code.

Unzipped submissions or submissions containing .class or other unneeded files will be penalized.

Markig Scheme:

[15] Marks for appropriate test cases.

[85] Marks for the program

[45] For program working as required.

[20] Using top-down design and appropriate methods in the program

[20] Programming style:

Internal documentation using javadoc notation,

Descriptive variable and method name,

Consistent, conventional class, methods and variable naming,

Proper indentation and formatting of code, appropriate use of constants