COSC 4301 – MODERN PROGRAMMING
Project 9 – Software as a Service (SaaS)

Complete this software project as a simple software product instead of as a service (SaaS).

As computer costs decline, it becomes feasible for every student, regardless of economic circumstance, to have a computer and use it in school. This creates exciting possibilities for improving the educational experience of all students worldwide, as suggested by the following exercise.

The use of computers in education is referred to as Computer-Assisted Instruction (CAI). Write a program that will help an elementary school student learn to evaluate integer arithmetic expressions. Use a *SecureRandom* object to produce positive one-digit integers for the expression. The program should then prompt the student with a question based on the following:

Difficulty level
1. Basic – 2 Operands and 1 Operator
2. Intermediate – 3 Operands and 2 Operators
3. Advanced – 4 Operands and 3 Operators

Operators selected a random by the program.
* Multiplication Operator
% Mod Operator **(**This is not an arithmetic operator, but must be included in the expression)
+ Addition Operator
- Subtraction Operator
(Note: No division operator)

The student then inputs the answer. Next, the program checks the student's answer. If it's correct, display a message from the *possible responses to a correct answer* below, and ask another question. If the answer is wrong, display a message from the *possible responses to an incorrect answer below*, display the same question again, and let the student try it repeatedly until the student finally gets it right.

A separate method should be used to generate each new question. This method should be called once when the application begins execution and each time the student answers the question correctly.

After the student gets five questions correctly, give him/her the opportunity to try a more difficult level or exit the program. The student might choose to continue, but after every correct answer, thereafter, give him/her another opportunity to try a more difficult level or exit the program.

If the student decides to try a more difficult level, let him/her answer five questions correctly before allowing him/her to try a more difficult level.

If the student is at the advanced level, give him/her the opportunity to exit the program, do not give him/her an opportunity to try a less difficult level.

After the student chooses to exit the program, calculate the percentage of questions that were answered correctly in each level and display it. If the percentage in the **Basic Level** is lower than 80%, display the percentage and this message; **"Please ask your teacher for extra help**".

Log all the student's interaction with the program in a file named Project9-Output.txt.

Possible responses to a correct answer: (Selected at random)

Excellent!
Very good!
Nice work!
Way to go!
Keep up the good work!

Possible responses to an incorrect answer: (Selected at random)

That is incorrect!
No. Please try again!
Wrong, Try once more!
No. Don't give up!
Incorrect. Keep trying!

Write the main method in a file named **Project9.java**.

No input, processing, or output should happen in the main method. All work should be delegated to other non-static methods.

Test your program and make sure it works correctly. Create a folder named, **YourFullName_Project9**. Copy your source codes and the output file to the folder. **Zip the folder, as a ".zip" file, and upload it to Blackboard**.

All classes in this project must be public, non-static and not nested in other classes.

Every method in your program should be limited to performing a single, well-defined task, and the name of the method should express that task effectively. All methods should be non-static unless it is absolutely necessary for it to be static.

Before you upload your project to Blackboard:

- Ensure that your code conforms to the style expectations set out in class and briefly discussed below.

- Make sure your variable names and methods are descriptive and follow standard capitalization conventions.

- Put comments wherever necessary. Comments at the top of each module should include your name, file name, and a description of the module. Comments at the beginning of methods describe what the method does, what the parameters are, and what the return value is. Use comments elsewhere to help your reader follow the flow of your code. **See the ProjectTemplate.java file for more details**.

- *Program readability and elegance are as important as correctness.* After you have written your method, read and re-read it to eliminate any redundant lines of code, and to make sure variables and methods names are intuitive and relevant.

Read the assignment very carefully to ensure that you have followed all instructions and satisfied all requirements.

You will not receive more than 50% of the total credit for this program if it is not written or does not work as expected.