Java Programming Project 2: Virtual Math Tutor

2020-1
Due: 06/08, 2020 (11:59 p.m.)

Instructions: Please create a Java application to solve the following problem. Submit electronic (Canvas Dropbox) copies of your program source code and **two or more** sample runs to me by the deadline. For the electronic submission of your program, you may simply compress (i.e., zip) your entire project folder (NetBeans, Eclipse, etc.) and upload that to Canvas instead of uploading the individual Java source code files.) Documentation requirements follow the problem specification.

Problem Specification: Your task is to create a GUI-based Java application that will help an elementary school student learn the mathematical operations of addition, subtraction, and multiplication. The program begins by generating a random integer in the range 1 through 3, inclusive, to randomly select one of the three operations. Next, it generates two random integers in the range 1 through 10, inclusive, for the operands. Finally, it displays a math problem of the chosen type. After displaying the problem, the program prompts the student to enter his or her answer. If the student answers incorrectly, the program displays the message "I'm sorry, but no. Please try again". If the student answers correctly, the program displays the message "Very good!", along with the number of attempts to solve the problem. Regardless of whether the student's response was correct or incorrect, the program provides him or her with the opportunity to solve additional problems by clicking on a "New Problem" button. As usual, your program output should resemble the sample runs at the end of this document.

Implementation Requirements and Guidelines:

- A `JLabel` must be used to display the math problem (example below).
  `How much is 5 times 9?`
- A second `JLabel` must be used to display the prompt to the student to enter his or her answer.
- A `JTextField` must be used to input the answer.
  (Note: In the sample run, the program checks the student's answer after s/he enters it in the `JTextField` then presses <enter>. If you prefer that the program check the student's answer after s/he clicks on a button, feel free to add a second button to the interface.)
- A `JLabel` must be used to display the status of the student's answer (correct/incorrect). (You may simply reuse the second `JLabel` if you like.)
- A `JButton` must be provided to allow the student to try a new problem. When the `JButton` is clicked:
  - A new problem must be generated.
  - The prompt "`Please enter your answer.`" must be displayed.
  - The `JTextField` used for input must be cleared.
- Your program must use methods to carry out the primary tasks of the program. Some possible methods include:
  - `public void generateProblem();` // Generates a math problem
  - `public void checkAnswer(int userAnswer);` // Checks student's answer

  etc.

Documentation Requirements:

1) Each program source code file (i.e., Java class) must have a header at the beginning of the class containing the following:
   - Name of author, PSU e-mail address of author, name of course, assignment number and due date,

name of file, purpose of class, compiler/IDE, operating system, and any external references used (e.g., Website)

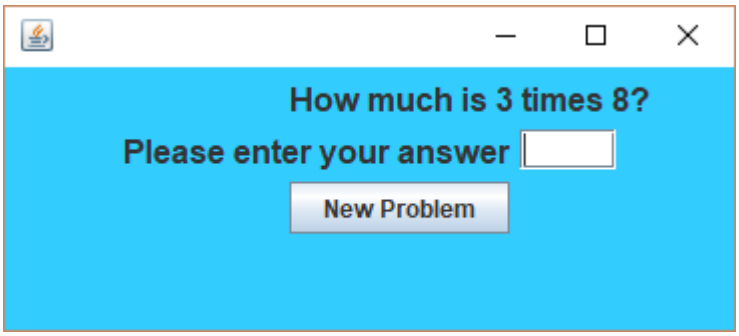
- Example:

```
/*
Author: Samkeun Kim
E-mail: skim@hknu.ac.kr
Course: Java Programming
Assignment: Programming Assignment 2
Due date: 06/08/2020
File: VirtualMathTutor.java
Purpose: Java GUI application that helps an elementary school
student learn addition, subtraction, and multiplication
Compiler/IDE: Java SE Development Kit 8u191/IntelliJ IDEA
Operating system: MS Windows 10
Reference(s): Java 8 API - Oracle Documentation
(http://docs.oracle.com/javase/8/docs/api/);
(Include ALL additional references (Web page, etc.) here.)
*/
```

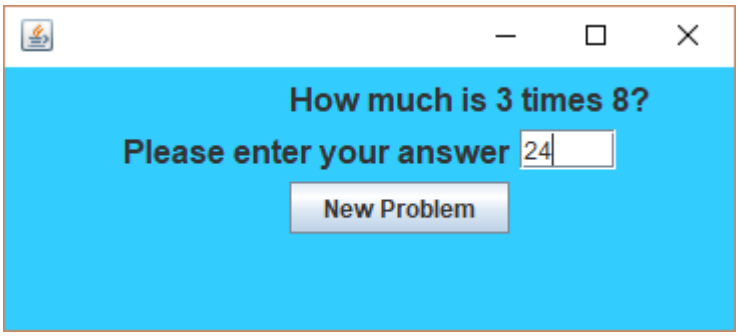
2) The purpose of each method in the source code file(s) must be documented as shown in the example below. I prefer that you use the **javadoc** comment style.

```
/** This method generates a math problem.
*/
public void generateProblem()
{
// Method definition (i.e., body)
}
```

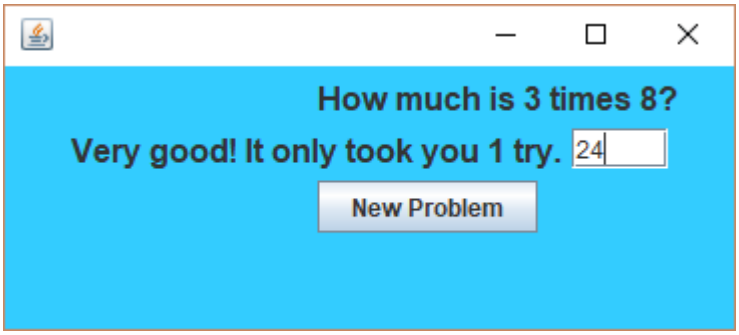
Sample run #1 *(Student answers correctly after one try.)*:



Program displays problem



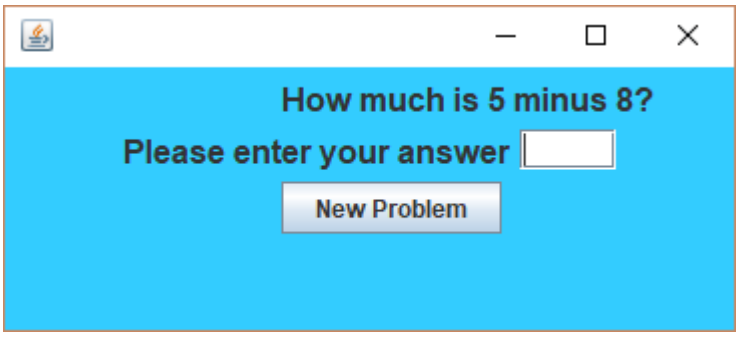
Student enters answer



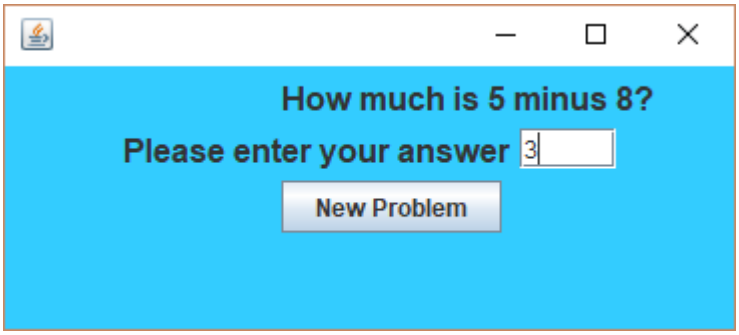
Student hits <enter> to check answer

Student clicks “New Problem”, then … (see next run)

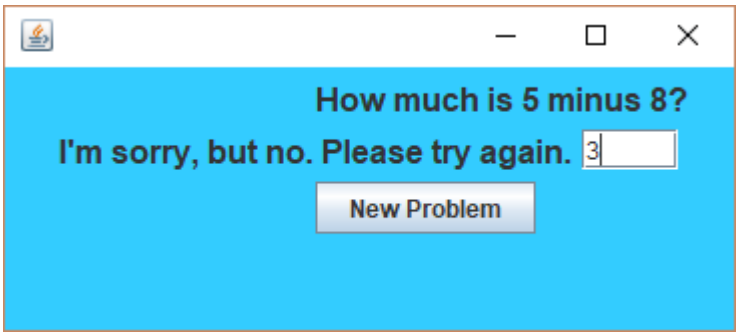
Sample run #2 *(Student answers correctly after multiple tries.)*:



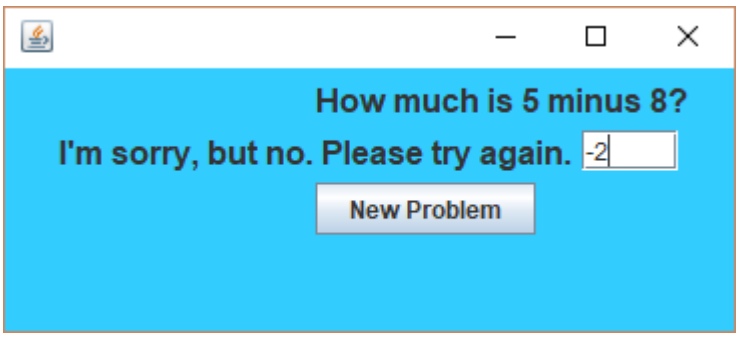
Program displays problem



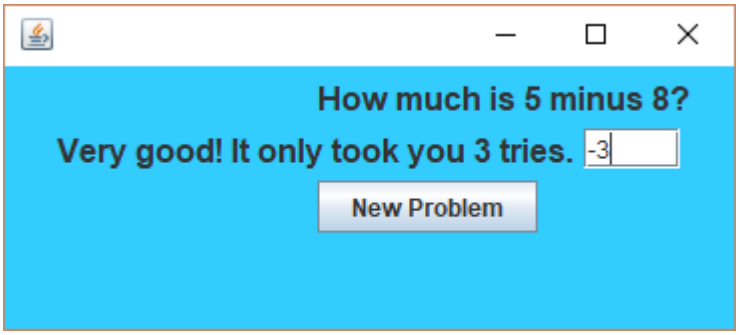
Student enters answer



Student hits <enter> to check answer



Student enters new answer, then hits <enter> again to check new answer



Student enters new answer, then hits <enter> again to check new answer