

## **CMPSC 221**

OO Programming with Web-Based Applications

Spring 2021

Due Date: Wednesday, March 3<sup>rd</sup>, 2021 (11:59 p.m. ET)

---

**Name:** \_\_\_\_\_

**Instructions:** Please create a Java application to solve the following problem. Submit electronic (Canvas Dropbox) copies of your project files (specified below) and **two or more** sample runs to me by the deadline. For the electronic submission of your project: 1) place your Java source code file(s), Java class file(s), sample runs, and a JAR file of your program in a folder; 2) compress the folder; and 3) upload the compressed archive to Canvas. I recommend that you also place a “ReadMe” file with instructions for executing your JAR file in the folder. Documentation requirements follow the problem specification.

**Important Note:** Please be sure to upload your Project 2 files to Canvas by the deadline. If you miss the deadline, you WILL automatically receive a grade of 0 (zero).

---

1. Create a Java application that implements a binary math tutor. Specifically, you are to create a Java program that helps a Computer Science major learn to add, subtract, and multiply 6-bit binary unsigned integers. Before the student can begin practicing, however, the program must create a set of 15 randomly generated problems: 5 addition, 5 subtraction, and 5 multiplication. Each problem must consist of two different 6-bit binary unsigned integers in the range  $000001_2$  ( $1_{10}$ ) through  $111111_2$  ( $63_{10}$ ), inclusive. Once the problem set has been created, the program proceeds by displaying a welcome message to the student in a message dialog box. After the student dismisses the message dialog box, an input dialog box displaying a problem and prompting the student for an answer appears. Depending on the student's answer, a message dialog box informing the student that s/he has answered the problem correctly or incorrectly is displayed. After the student clicks “OK,” an input dialog box asking the student if s/he wants to continue practicing appears. If the student chooses “Yes,” the program displays another problem. If the student chooses “No” or “Cancel,” the program thanks the student for using the binary math tutor and ends. Your program output should resemble the sample runs at the end of this document.

### **Implementation Requirements:**

- Your math tutor program **must** consist of a minimum of 15 randomly generated problems (5 addition, 5 subtraction, 5 multiplication) stored in an array or `ArrayList`. As stated above, the operands must be different. In the case of subtraction, the first operand must be greater than the second.
- Each problem **must** be chosen randomly. Also, each problem may be chosen only **once**.
- You **must** use the dialog boxes provided by the `JOptionPane` class.
- Your program **must** ignore all leading and trailing spaces in the students' responses.
- Your program **must** correctly handle the case in which the user clicks “Cancel” so that the program does not crash.
- You **must** use a minimum of two classes to implement the math tutor program: a “driver” containing `main` and a `ProblemAndAnswer` class.
- The problems and their corresponding answers **must** be stored as an array or `ArrayList` of `ProblemAndAnswer` objects. You may **not** use parallel arrays in place of an array or `ArrayList` of `ProblemAndAnswer` objects.
- A `ProblemAndAnswer` object **must** have at least two instance variables, one to store the problem and one to store the answer.

- **Required methods for the ProblemAndAnswer class include:**
  - `public ProblemAndAnswer();` // A default constructor that initializes the problem and answer to the empty string
  - `public void setProblem(String problem);` // A “set” method that assigns the problem
  - `public void setAnswer(String answer);` // A “set” method that assigns the answer
  - `public String getProblem();` // A “get” method that retrieves the problem
  - `public String getAnswer();` // A “get” method that retrieves the answer
- **Recommended methods for the ProblemAndAnswer class include:**
  - `public boolean checkAnswer(String response);` // A method to check if the student’s answer is correct or incorrect
  - `public void displayProblem();` // A method to display the problem to the student

References:

- Savitch, 6<sup>th</sup> Edition: Chapters 1 – 4 (basics), Chapter 5 (static methods and static variables), Chapter 6 (arrays)
  - Deitel & Deitel, 9<sup>th</sup> Edition: Chapter 3, Section 8 (dialog boxes)
-

## 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., Web site)
  - Example:

```
/*
Author:          Wanda Kunkle
E-mail:          wmk12@psu.edu
Course:          CMPSC 221
Assignment:      Programming Assignment 2
Due date:        3/3/2021
File:            BinaryMathTutor.java
Purpose:         Java application that implements a binary math tutor
Compiler/IDE:    Java SE Development Kit 14.0.2/IntelliJ IDEA 2020.3.1
Operating
system:          MS Windows 10 Home
Reference(s):    Java 14 API - Oracle Documentation
                  (https://docs.oracle.com/en/java/javase/14/docs/api/);
                  (Include ALL additional references (Web page, etc.) here.)
*/
```

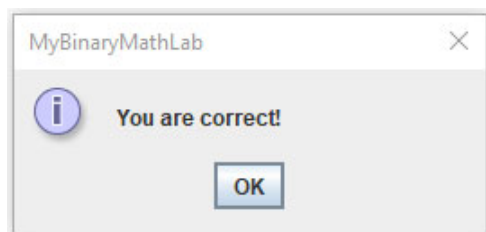
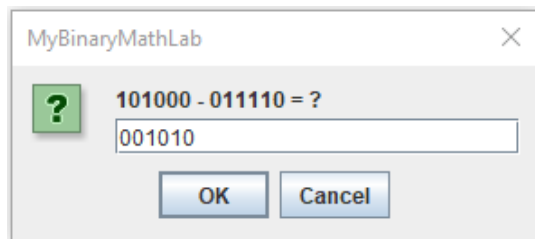
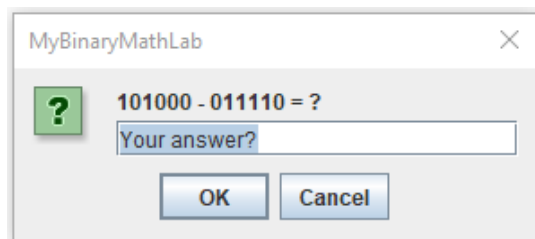
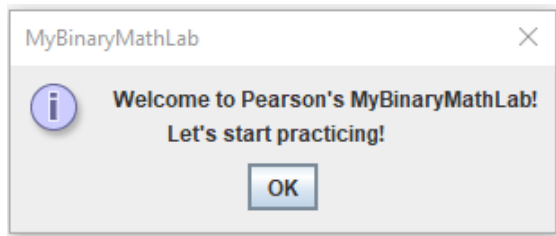
- 2) The purpose of each method in the source code file(s) must be documented as shown in the example below. I prefer that you use the **javadoc** comment style.

```
/** This method checks a given response for correctness.
 *
 * @param response    The response to check
 * @return            True if the response was correct, false otherwise
 */
public boolean checkAnswer(String response)
{
    // Method definition (i.e., body)
}
```

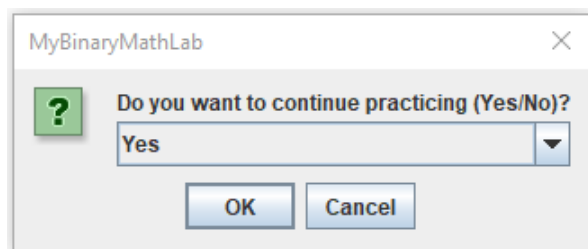
---

Sample run #1 (*Student answers two of three problems correctly, then quits.*):


---



User answers correctly!




MyBinaryMathLab

 **010011 \* 100110 = ?**


Your answer?

MyBinaryMathLab

 **010011 \* 100110 = ?**


1011010010

MyBinaryMathLab

 **You are correct!**


User answers correctly!

MyBinaryMathLab

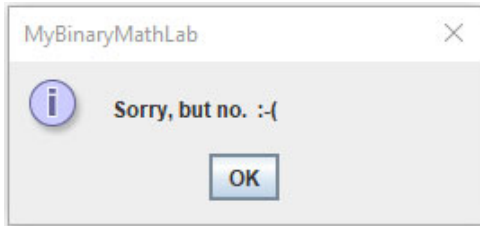
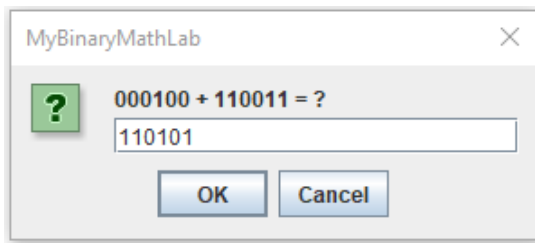
 **Do you want to continue practicing (Yes/No)?**

Yes

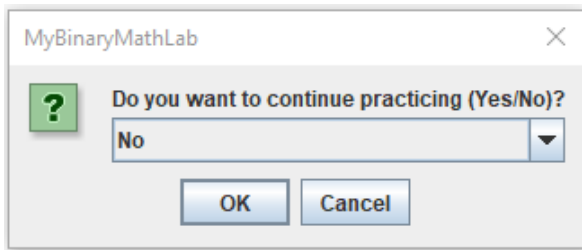
MyBinaryMathLab

 **000100 + 110011 = ?**

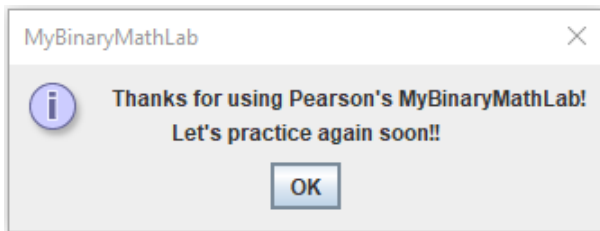
Your answer?



User answers incorrectly.



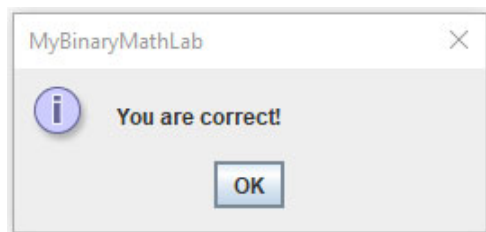
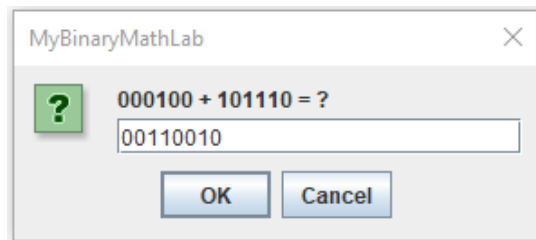
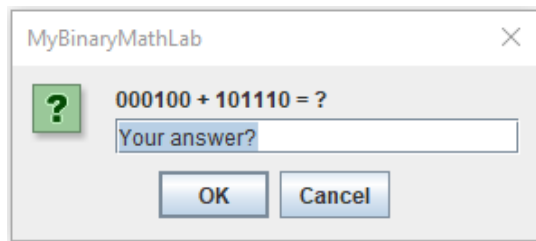
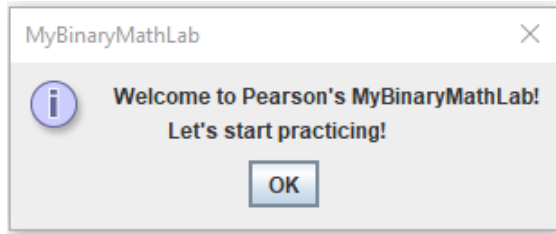
User decides to quit.



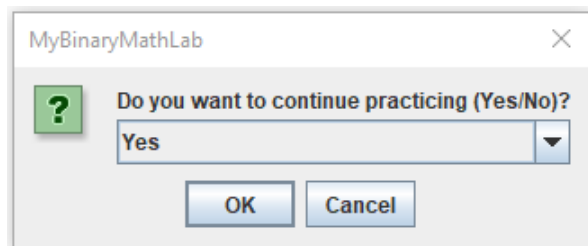
---

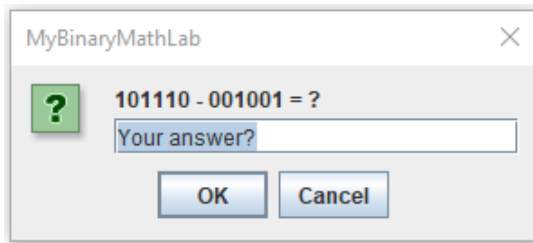
Sample run #2 (Student answers one of two problems correctly and clicks Cancel twice.):

---

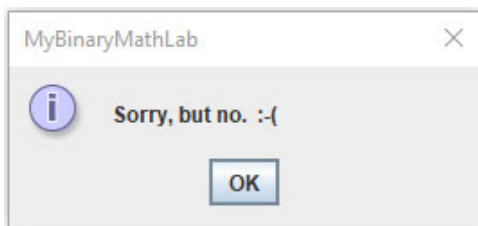
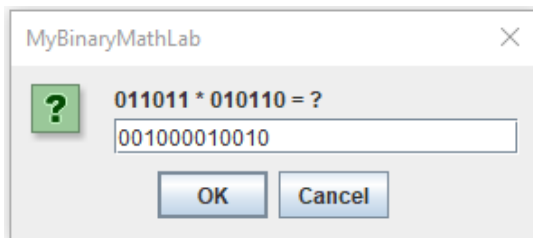
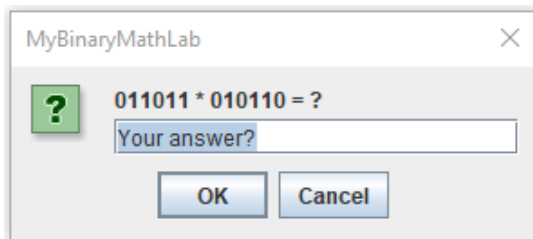
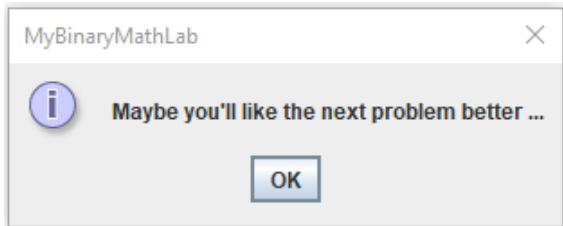


User answers correctly!



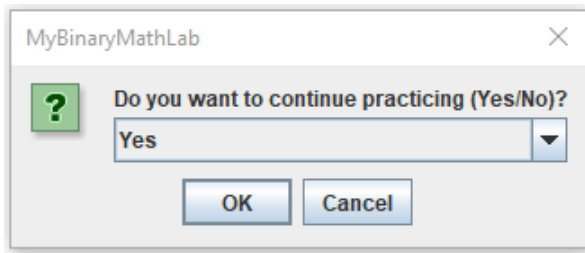


User clicks Cancel.

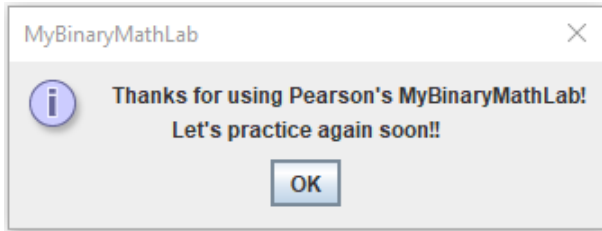


User answers incorrectly.





User clicks Cancel.



-----  
Command-line example to create an archive called WordSearch.jar with one class file:  
-----

(OS: Windows 10 Home; JDK: jdk-14.0.2)

Class file (in default package):  
KeyWordCode\_2021.class

First create manifest.txt (contains one line specifying where main method is):  
Main-Class: KeyWordCode\_2021

Create \*.jar file (assumes all files are in root directory):  
C:\>"C:\Program Files\Java\jdk-14.0.2\bin\jar.exe" --create --verbose --file  
WordSearch.jar --manifest manifest.txt KeyWordCode\_2021.class

Execute \*.jar file: (assumes all files are in root directory)  
C:\>"C:\Program Files\Java\jdk-14.0.2\bin\java.exe" -jar WordSearch.jar

NOTE: If you are using an IDE (e.g., IntelliJ), you may be able to create the JAR file from within the IDE.