Lab 9 Name:_________________________ Checked:_____

Objectives:

Practice using the conditional operator; **switch** statements; **do** and **for** loops. Explore the use of dialog boxes (JOptionPane) and learn how to implement a simple experimental setup to measure response times using `System.currentTimeMillis()` (current system time in milliseconds).

Preparation: Response time experiment – accessing the system clock

1. Download and run `ResponseTimeExperiment.java` (shown on next page).

2. Run the program a few times to observe its function.
   Note the use of `System.currentTimeMillis()` to find the current time at two different points during the run, and thus obtain the amount of time elapsed between issuing of a question and the input (answer) from the user.

3. Refer to the code shown on the next page. Modify the code enclosed in the boxes, as follows:
   - Using the conditional operator, compute the value of `outcome` in just one line of code (replace the if/else).
   - Display the result ("Correct!" or "Incorrect.") using a single `System.out.println` statement (again, use the conditional operator).

4. Test your program to ensure that it functions exactly as before.

5. Upload and submit `ResponseTimeExperiment.java` through [blackboard](https://blackboard.com) under “Lab 9 Prep”.

Villanova University  CSC 1051  www.csc.villanova.edu/~map/1051  Dr. Papalaskari
// ResponseTimeExperiment.java
// Measure response time for addition problems.
// M A Papalaskari

import java.util.Scanner;
import java.util.Random;

public class ResponseTimeExperiment
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        Random rand = new Random();

        System.out.print("Please enter your name: ");
        String name = in.nextLine();

        System.out.println("Hello " + name
               + ". Please answer as fast as you can.
               + "\n\nHit <ENTER> when ready for the question.");

        in.nextLine();   // wait for user to hit <ENTER>

        int a = rand.nextInt(100);
        int b = rand.nextInt(100);

        long startTime = System.currentTimeMillis();

        System.out.print(a + " + " + b + " = ");
        String response = in.nextLine();
        int number = Integer.parseInt(response);

        long endTime = System.currentTimeMillis();

        int outcome;
        if  (number == a + b)
            outcome = 1;
        else
            outcome = 0;

        long reactionTime = endTime - startTime;

        if (outcome == 1)
            System.out.println("Correct!");
        else
            System.out.println("Incorrect.");

        System.out.println("That took " + reactionTime
               + " milliseconds");
        System.out.println("Thank you " + name + ", goodbye.");
    }
}
Part A: switch statement

Write a program to input a number \( n \) that symbolizes a version of the Mac OS X 10.\( n \) software. Output the name of that version of the Mac OS X 10.\( n \) software. For example, if the user inputs 8, then the program should output “Mountain Lion.” [You will need to do a little googling to find the names of the different mac software. They run from 1-10 and most of them are cats, but more recently they are joined by mountains.]

Part B. Dialog boxes and do loops

Run EvenOdd.java and get familiar with how it works. Modify it so that it performs the function of the GPA calculator (from Lab 4b – version that asks each time whether to repeat). You will need to modify the prompts and other interaction (for example, "Enter quality points: " instead of "Enter an integer: ") and to input an additional number for the credits.

Part C. for-loops

1. Examine the code of ResponseTimeExperiment.java printed on the previous page and annotate the code with the number to match each step in the algorithm:

   **Algorithm:**
   1. input name
   2. print personalized welcome message & instructions
   3. \( a = \) random int
   4. \( b = \) random int
   5. \( \text{startTime} = \) current time
   6. print question using \( a, b \)
   7. input answer
   8. \( \text{endTime} = \) current time
   9. outcome = 1 or 0 (answer is correct or incorrect)
   10. reactionTime = endTime – startTime
   11. print outcome as “Correct” or “Incorrect”
   12. print goodbye
2. We will now modify the ResponseTimeExperiment to repeat the experiment four times, computing the number of correct answers and the average response time of the user, and printing these results at the end. See the algorithm below – some steps have been added (highlighted below).

**Algorithm:**
1. input name
2. print personalized welcome message & instructions

A. numCorrect = 0
B. totalTime = 0

C. repeat 4 times:
   3. a = random int
   4. b = random int
   5. startTime = current time
   6. print question using a, b
   7. input answer
   8. endTime = current time
   9. outcome = 1 or 0 (answer is correct or incorrect)
   10. reactionTime = endTime – startTime
   11. print outcome as “Correct” or “Incorrect”
   D. numCorrect = numCorrect + outcome
   E. totalTime = totalTime + reactionTime

F. averageTime = totalTime/4
G. print averageTime
12. print goodbye

- Annotate the code with the new steps to prepare to implement the algorithm. Be sure to mark clearly, with the appropriate letter, the exact position where code must be added.

3. Add some code and restructure ResponseTimeExperiment.java to implement the new steps, using your notes from 2, above. Be sure to use a for-loop to implement the repetition of the experiment.

**Part D. Dialog boxes, revisited**

Convert the program from part A or part C to use dialog boxes for input/output.
Lab 9 Comments  Name:_________________  Checked: ______

Comments on this lab, please:

What was the most valuable thing you learned in this lab?

What did you like best about this lab?

Was there any particular problem?

Do you have any suggestions for improving this lab as an effective learning experience?