**Centennial College**

**COMP 228: Java Programming**

**LAB #2 – Java Methods**

**Student:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Due Date: Week 4

Purpose: The purpose of this Lab assignment is to:

* Practice the use Java classes, Java methods, and other concepts taught.

References: Learning materials for week 3 and 4, textbook, and other references (if any)

This material provides the necessary information you need to complete the exercises.

Be sure to read the following general instructions carefully:

This lab should be completed individually by all the students.

YOU NEED TO SUBMIT THE FOLLOWING 2 DOCUMENTS IN THE DROPBOX TITLED LAB2:

1. THE FIRST ONE IS A WORD DOCUMENT. USE THIS DOCUMENT AND ADD SCREEN SHOTS OF THE RUNNING STATE OF EACH EXERCISE (If there are more than 1 exercise). DO NOT DELETE THE QUESTIONS. THE SCREEN SHOTS SHOULD FOLLOW EACH QUESTION AND COVER ALL THE ASPECTS/FUNCTIONALITIES OF EACH EXERCISE. AFTER THE SCREEN SHOTS PLEASE COPY THE CODE FROM THE CODE WINDOW AND PASTE THE COMPLETE CODE. DO NOT GIVE ME SCREEN SHOTS OF THE CODE. DO NOT ZIP THIS FILE AND KEEP IT SEPARATE FROM YOUR ZIPPED PROGAM FILE.
2. SUBMIT ALSO ONE ZIPPED PROJECT FILE THAT CONTAINS ALL THE EXERISES SEPARATELY INTO THE SAME DROP BOX.

You must name your Eclipse/IntelliJ project according to the following rule:

**YourFullName\_COMP228Labnumber**

Example: **JohSmith\_COMP228Lab2**

Each exercise should be placed in a separate package (if there are more than 1 exercise) named *exercise1*, *exercise2*, etc.

Submit your assignment in a **zip file** that is named according to the following rule:

**YourLastName\_COMP228Labnumber.zip**

Example: **JohSmith\_COMP228Lab2.zip**

Apply the naming conventions for variables, methods, classes, and packages:

- *variable names* start with a *lowercase* character

- *classes* start with an *uppercase* character

- **packages** use only *lowercase* characters

- *methods* start with a *lowercase* character

**Exercise 1:**

Write a Java application that simulates a test. The test contains **at least five** questions about first three lectures of this course. Each question should be a multiple-choice question with 4 options.

Design a **Test** class. Use programmer-defined methods to implement your solution. For example:

* create a method to simulate the questions – *simulateQuestion*
* create a method to check the answer – *checkAnswer*
* create a method to display a random message for the user – *generateMessage*
* create a method to interact with the user - *inputAnswer*

Display the questions using methods of ***JOptionPane*** class. Use a loop to show all the questions.

For each question:

* If the user finds the right answer, display a random congratulatory message (“Excellent!”,”Good!”,”Keep up the good work!”, or “Nice work!”).
* If the user responds incorrectly, display an appropriate message and the correct answer (“No. Please try again”, “Wrong. Try once more”, “Don't give up!”, “No. Keep trying..”).
* Use random-number generation to choose a number from 1 to 4 that will be used to select an appropriate response to each answer.
* Use a switch statement to issue the responses, as in the following code:

switch ( randomObject.nextInt( 4 ) )

{

case 0:

return( "Very good!" );

……

}

At the end of the test display the number of correct and incorrect answers, and the percentage of the correct answers.

Your main class will simply create a Test object and start the test by calling **inputAnswer** method.

A screenshot of a computer

Description automatically generatedOutput 1 -

Output 2-

A screenshot of a computer

Description automatically generated

OUTPUT 3 -

A screenshot of a computer

Description automatically generated

OUTPUT 4 -

A screenshot of a computer

Description automatically generated

OUTPUT 5-

A screenshot of a computer

Description automatically generated

OUTPUT 6 –

A screenshot of a computer

Description automatically generated

TEST CLASS –

**package** exercise1;

**import** javax.swing.JOptionPane;

**import** java.util.Random;

**public** **class** Test {

**private** **final** String[] questions = {

"Question 1: What is the keyword used to define a class in Java?",

"Question 2: What does the keyword 'static' mean in Java?",

"Question 3: How is encapsulation achieved in Java?",

"Question 4: What is an Interface in Java?",

"Question 5: How is polymorphism achieved in Java?"

};

**private** **final** String[][] options = {

{"1. class", "2. def", "3. Class", "4. ClassDef"},

{"1. It makes a method or variable belong to the class, not an instance", "2. It makes a method or variable immutable", "3. It prevents a method from being overridden", "4. It indicates that a variable can be accessed from anywhere"},

{"1. By using public variables and methods", "2. By using package-private variables and methods", "3. By using private variables and methods, and public getters/setters", "4. By using static variables and methods"},

{"1. A class that cannot be instantiated", "2. A contract that specifies a group of related methods with empty bodies", "3. A parent class from which other classes inherit", "4. A class that can only contain static methods"},

{"1. By using only methods that are declared in the current class", "2. By using the 'poly' keyword", "3. By using interfaces and/or extending classes and overriding methods", "4. By using only static methods"}

};

**private** **final** **int**[] correctAnswers = {1, 1, 3, 2, 3};

**private** **int** correct\_count = 0;

**private** **int** incorrect\_count = 0;

**private** Random random = **new** Random();

**public** **void** simulateQuestion(**int** questionIndex) {

String message = questions[questionIndex] + "\n";

**for** (**int** i = 0; i < options[questionIndex].length; i++) {

message += options[questionIndex][i] + "\n";

}

JOptionPane.*showMessageDialog*(**null**, message);

}

**public** **boolean** checkAnswer(**int** questionIndex, **int** answer) {

**if** (answer == correctAnswers[questionIndex]) {

correct\_count++;

JOptionPane.*showMessageDialog*(**null**, generateMessage(**true**));

**return** **true**;

} **else** {

incorrect\_count++;

JOptionPane.*showMessageDialog*(**null**, generateMessage(**false**) + " The correct answer is: " +

options[questionIndex][correctAnswers[questionIndex]]);

**return** **false**;

}

}

**public** String generateMessage(**boolean** isCorrect) {

**int** randomIndex = random.nextInt(4);

**switch** (randomIndex) {

**case** 0:

**return** isCorrect ? "Excellent!" : "No. Please try again";

**case** 1:

**return** isCorrect ? "Good!" : "Wrong. Try once more";

**case** 2:

**return** isCorrect ? "Keep up the good work!" : "Don't give up!";

**case** 3:

**return** isCorrect ? "Nice work!" : "No. Keep trying...";

**default**:

**return** "";

}

}

**public** **void** inputAnswer() {

**for** (**int** i = 0; i < questions.length; i++) {

simulateQuestion(i);

**int** answer = 0;

**boolean** validInput = **false**;

**while** (!validInput) {

**try** {

answer = Integer.*parseInt*(JOptionPane.*showInputDialog*("Enter the number of your answer:"));

validInput = **true**;

} **catch** (NumberFormatException e) {

JOptionPane.*showMessageDialog*(**null**, "Invalid input. Please enter a number.");

}

}

checkAnswer(i, answer);

}

**double** percentage = (**double**) correct\_count / questions.length \* 100;

JOptionPane.*showMessageDialog*(**null**, "Test completed!\n" +

"Correct answers: " + correct\_count + "\n" +

"Incorrect answers: " + incorrect\_count + "\n" +

"Percentage: " + percentage + "%");

}

}

MAIN CLASS –

**package** exercise1;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Test test = **new** Test();

test.inputAnswer();

}

}

(5 marks)

**Exercise 2:**

Design a Lotto class with one array instance variable to hold three random integer values (from 1 to 9). Include a constructor that randomly populates the array for a lotto object. Also, include a method in the class to return the array.

Use this class to simulate a simple lotto game in which the user chooses a number between 3 and 27. *The user runs the lotto up to 5 times and each time the sum of lotto numbers is calculated*. *If the number chosen by the user matches the sum*, the user wins and the game ends. *If the number does not match the sum within five rolls*, the computer wins.

Use methods of JOptionPane class to interact with the user.

(3 marks)

OUTPUT 1 -

A screenshot of a computer

Description automatically generated

OUTPUT 2-

A screenshot of a computer

Description automatically generated

LOTTO CLASS-

**package** exercise2;

**import** java.util.Random;

**import** javax.swing.JOptionPane;

**public** **class** Lotto {

**private** **int**[] values;

**public** Lotto() {

values = **new** **int**[3];

Random random = **new** Random();

**for** (**int** i = 0; i < values.length; i++) {

values[i] = random.nextInt(9) + 1;

}

}

**public** **int**[] getValues() {

**return** values;

}

**public** **int** getSum() {

**int** sum = 0;

**for** (**int** value : values) {

sum += value;

}

**return** sum;

}

**public** **static** **void** main(String[] args) {

String input = JOptionPane.*showInputDialog*("Enter a number between 3 and 27:");

**int** userNumber = Integer.*parseInt*(input);

**boolean** win = **false**;

**for** (**int** i = 0; i < 5; i++) {

Lotto lotto = **new** Lotto();

**int** sum = lotto.getSum();

**if** (sum == userNumber) {

win = **true**;

**break**;

}

}

**if** (win) {

JOptionPane.*showMessageDialog*(**null**, "Congratulations! You won the game!");

} **else** {

JOptionPane.*showMessageDialog*(**null**, "Sorry! The computer won the game.");

}

}

}

**Exercise 3:**

Write a Java class that implements a set of three overloaded static methods. The methods should have different set of parameters and perform similar functionalities. Call the methods within main method and display the results. (2 marks)

A screenshot of a computer

Description automatically generated

Code –

**package** exercise3;

**public** **class** OlMethod {

**public** **static** **int** multiply(**int** num1, **int** num2) {

**return** num1 \* num2;

}

**public** **static** **int** multiply(**int** num1, **int** num2, **int** num3) {

**return** num1 \* num2 \* num3;

}

**public** **static** **double** multiply(**double** num1, **double** num2) {

**return** num1 \* num2;

}

**public** **static** **void** main(String[] args) {

System.***out***.println("Multiplication of two integers: " + *multiply*(4, 2));

System.***out***.println("Multiplication of three integers: " + *multiply*(2, 3, 4));

System.***out***.println("Multiplication of two doubles: " + *multiply*(2.5, 3.5));

}

}

**Evaluation:**

|  |  |
| --- | --- |
| **Functionality** |  |
| Correct implementation of classes (instance variable declarations, constructors, getter and setter methods, etc.) | 40% |
| Correct implementation of driver classes (declaring and creating objects, calling their methods, interacting with user, displaying results) | 40% |
| Comments, correct naming of variables, methods, classes, etc. | 5% |
| **Friendly input/output** | 15% |
| **Total** | 100% |