**Centennial College**

**COMP 228: Java Programming**

**LAB #2 – Java Methods**

**Student:** Aryan Patel **SIN**: 301226774

Due Date: Week 5

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:

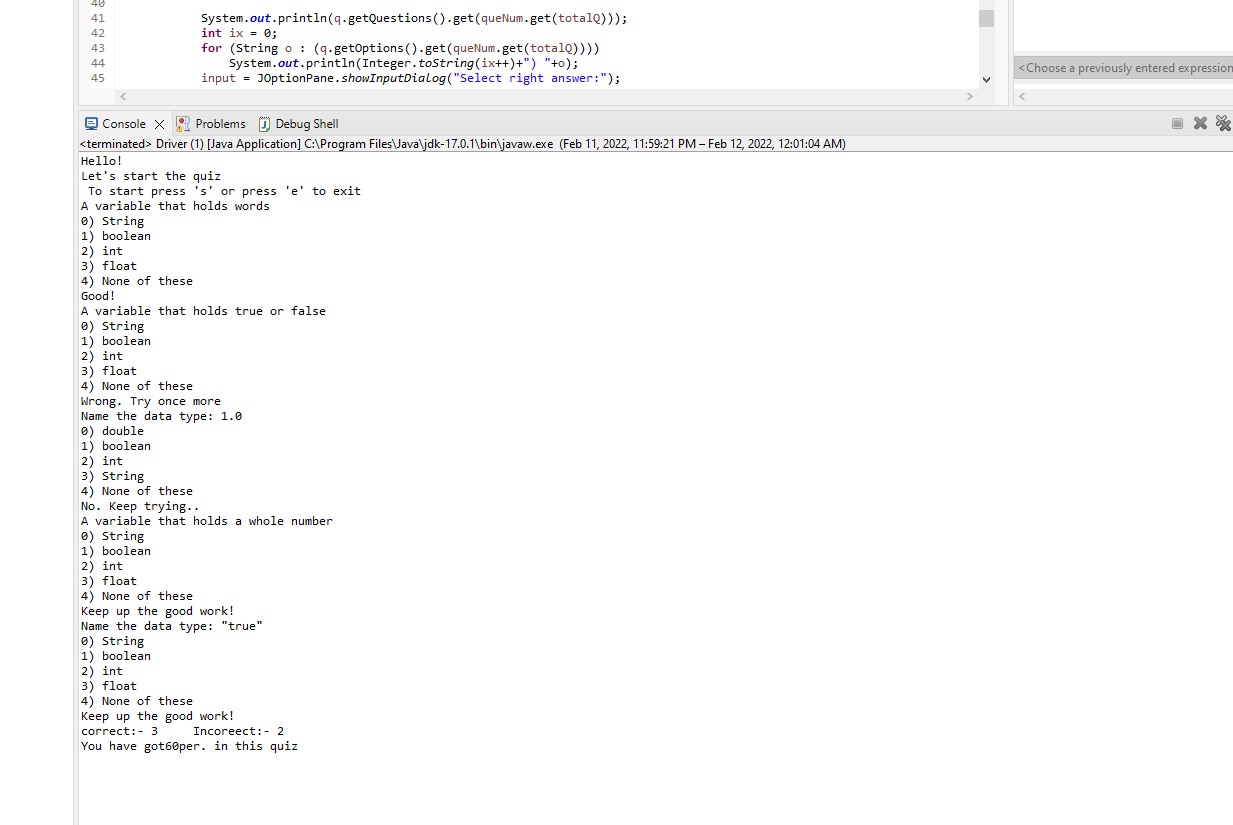
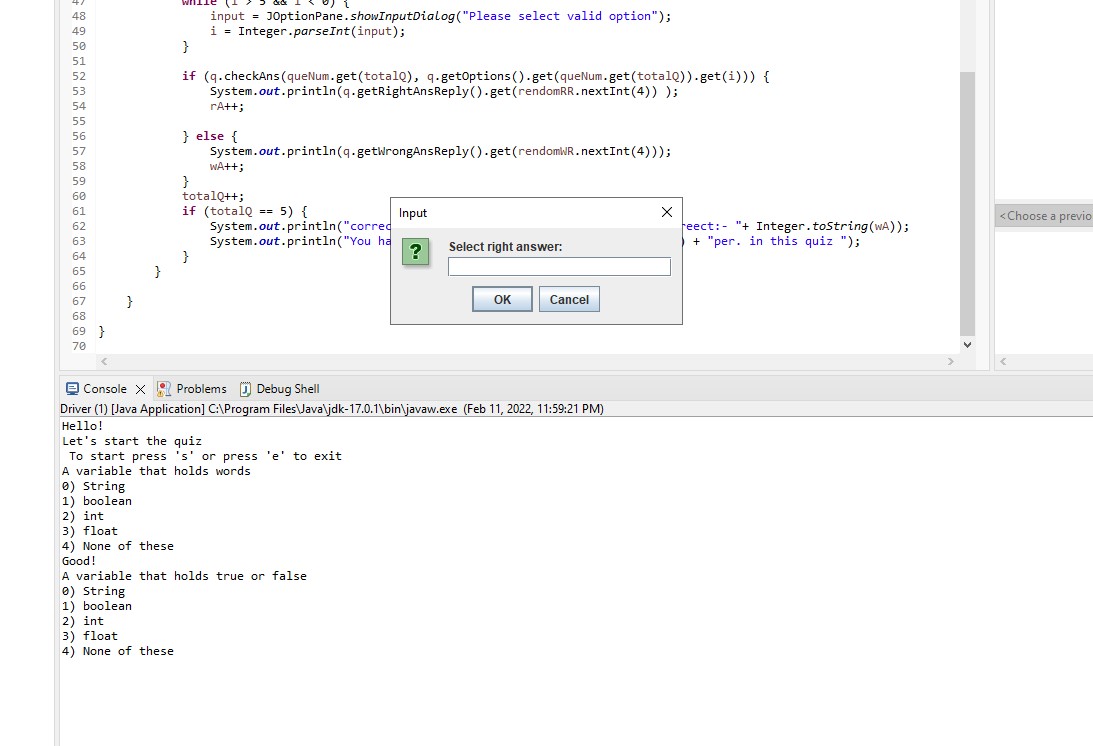
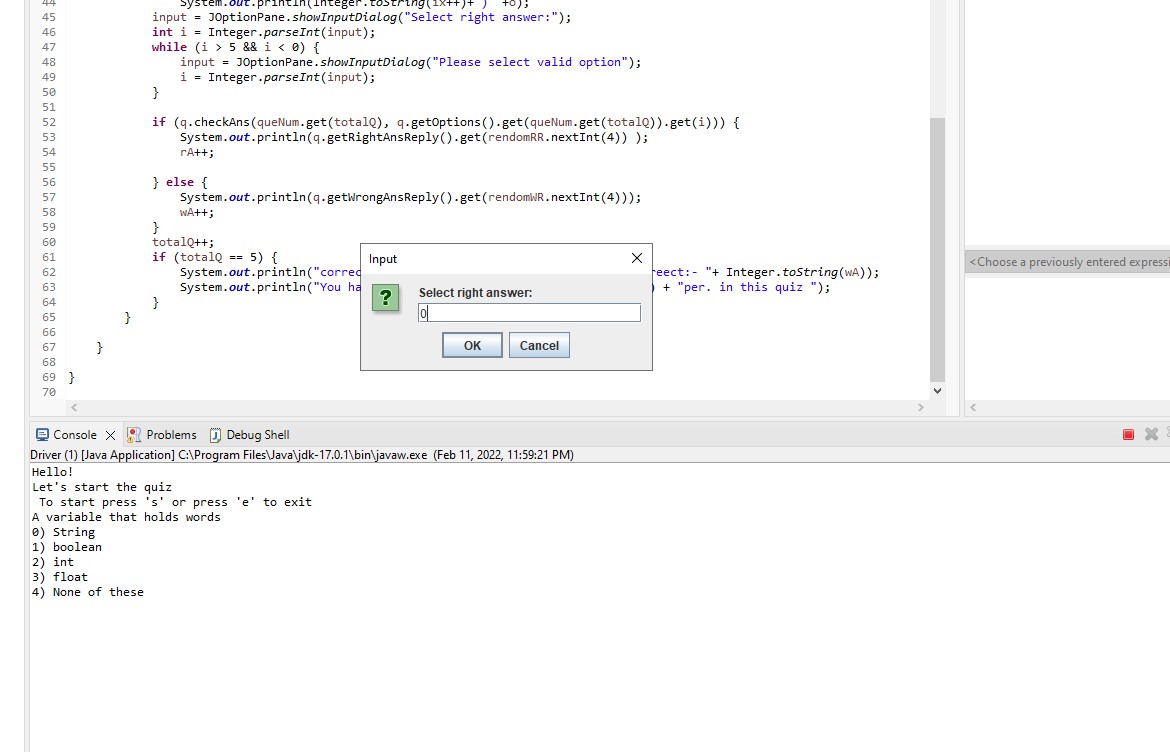
return( "Very good!" );

……

}

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.

 (5 marks)

Driver Class

package exercise1;

import java.security.SecureRandom;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collection;

import java.util.Collections;

import java.util.List;

import javax.swing.JOptionPane;

// driver class

public class Driver {

// main method

public static void main(String[] args) {

// TODO Auto-generated method stub

// random number for right answer

SecureRandom rendomRR = new SecureRandom();

// random number for wrong answer

SecureRandom rendomWR = new SecureRandom();

// number of right answer

int rA = 0;

// number of wrong answer

int wA = 0;

List<Integer> queNum = new ArrayList<>(Arrays.asList(0,1,2,3,4));

Collections.shuffle(queNum);

// Quiz object

Quiz q = new Quiz();

// input string

String input = "";

System.out.println("Hello! \nLet's start the quiz\n To start press 's' or press 'e' to exit");

input = JOptionPane.showInputDialog("To start press 's' or press 'e' to exit");

if (input.equalsIgnoreCase("s"))

q.createQuestions();

int totalQ = 0;

// show questions

while (!input.equals("e") && totalQ < 5) {

System.out.println(q.getQuestions().get(queNum.get(totalQ)));

int ix = 0;

for (String o : (q.getOptions().get(queNum.get(totalQ))))

System.out.println(Integer.toString(ix++)+") "+o);

input = JOptionPane.showInputDialog("Select right answer:");

int i = Integer.parseInt(input);

while (i > 5 && i < 0) {

input = JOptionPane.showInputDialog("Please select valid option");

i = Integer.parseInt(input);

}

if (q.checkAns(queNum.get(totalQ), q.getOptions().get(queNum.get(totalQ)).get(i))) {

System.out.println(q.getRightAnsReply().get(rendomRR.nextInt(4)) );

rA++;

} else {

System.out.println(q.getWrongAnsReply().get(rendomWR.nextInt(4)));

wA++;

}

totalQ++;

if (totalQ == 5) {

System.out.println("correct:- "+ Integer.toString(rA) + " Incoreect:- "+ Integer.toString(wA));

System.out.println("You have got" + Integer.toString((100 \* rA) / 5) + "per. in this quiz ");

}

}

}

}

Quiz class

package exercise1;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.List;

public class Quiz {

private HashMap<Integer, String> ans = new HashMap<>();

private HashMap<Integer, String> questions = new HashMap<>();

private HashMap<Integer, List<String>> options = new HashMap<>();

private List<String> rightAnsReply = new ArrayList<>(Arrays.asList("Excellent!","Good!","Keep up the good work!","Nice work!")) ;

private List<String> wrongAnsReply = new ArrayList<>(Arrays.asList("No. Please try again", "Wrong. Try once more", "Don't give up!", "No. Keep trying.."));

// create questions, options and answers

public void createQuestions()

{

this.questions.put(0,"A variable that holds a whole number");

this.questions.put(1,"A variable that holds words");

this.questions.put(2,"A variable that holds true or false");

this.questions.put(3,"Name the data type: 1.0");

this.questions.put(4,"Name the data type: \"true\"");

this.options.put(0, Arrays.asList("String","boolean","int","float","None of these"));

this.options.put(1, Arrays.asList("String","boolean","int","float","None of these"));

this.options.put(2, Arrays.asList("String","boolean","int","float","None of these"));

this.options.put(3, Arrays.asList("double","boolean","int","String","None of these"));

this.options.put(4, Arrays.asList("String","boolean","int","float","None of these"));

this.ans.put(0,"int");

this.ans.put(1,"String");

this.ans.put(2,"boolean");

this.ans.put(3,"double");

this.ans.put(4,"String");

}

// check the answer of question

public boolean checkAns(int questionId, String answer)

{

if(this.ans.get(questionId) == answer)

return true;

else return false;

}

// get all questions

public HashMap<Integer, String> getQuestions() {

return questions;

}

// get options

public HashMap<Integer, List<String>> getOptions() {

return options;

}

// get wrong ans reply

public List<String> getWrongAnsReply() {

return wrongAnsReply;

}

// get right ans reply

public List<String> getRightAnsReply() {

return rightAnsReply;

}

}

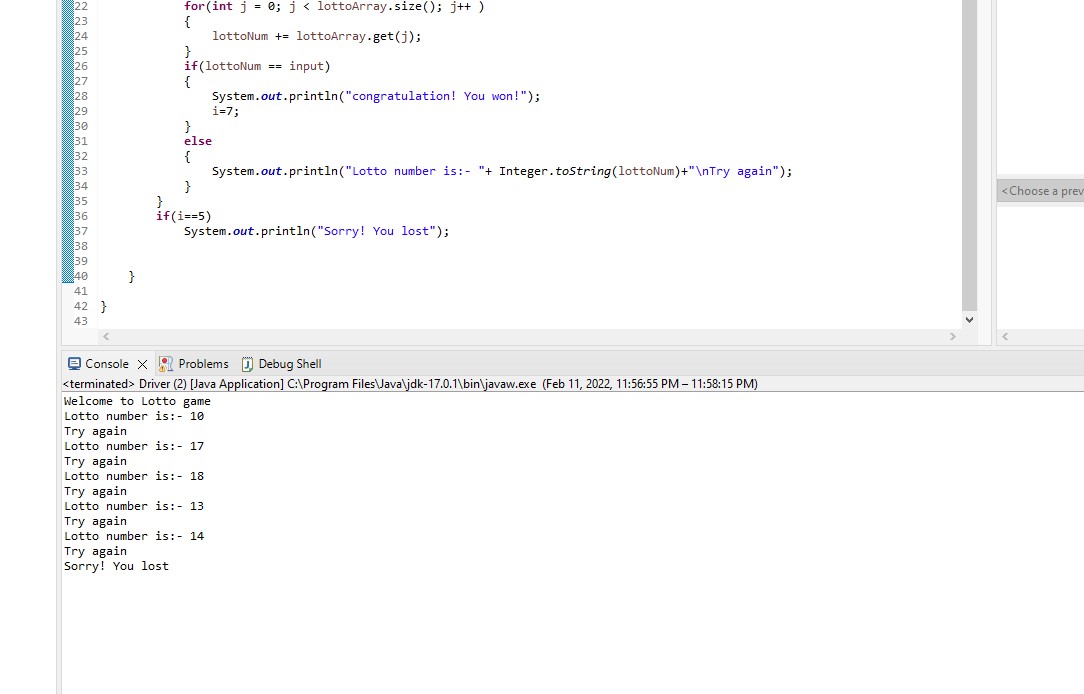
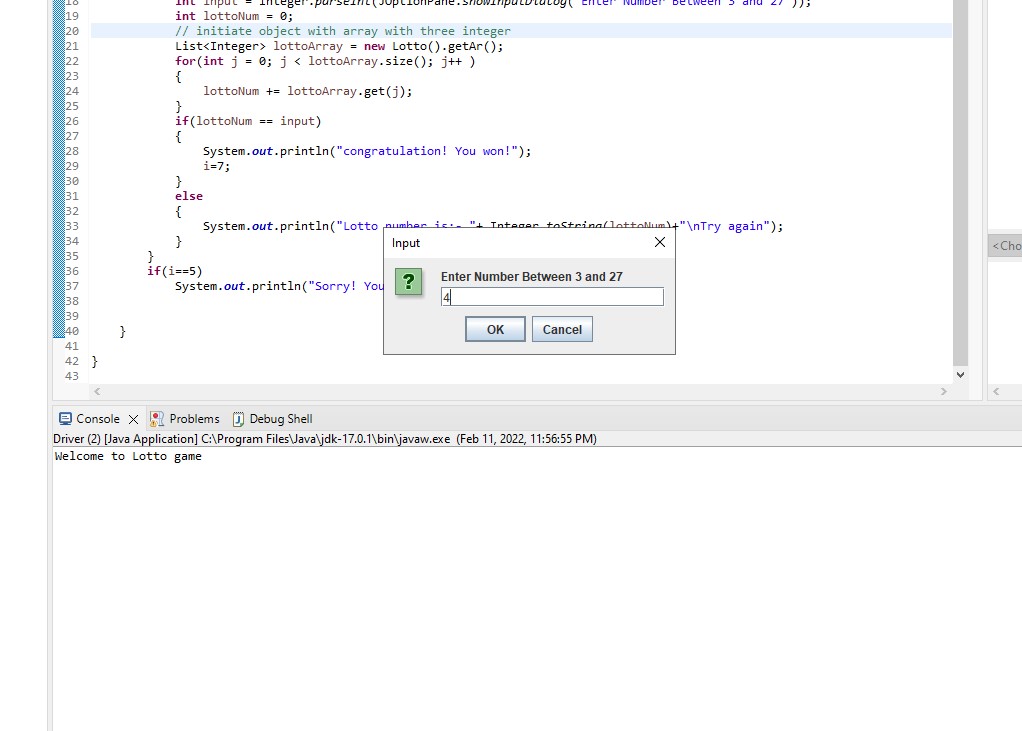
**Exercise 2:**

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)



Driver class

package exercise2;

import java.util.List;

import javax.swing.JOptionPane;

// lotto driver class

public class Driver {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.out.println("Welcome to Lotto game");

int i;

// five round of lotto game

for( i=0;i<5;i++)

{

// input from user

int input = Integer.parseInt(JOptionPane.showInputDialog("Enter Number Between 3 and 27"));

int lottoNum = 0;

// initiate object with array with three integer

List<Integer> lottoArray = new Lotto().getAr();

for(int j = 0; j < lottoArray.size(); j++ )

{

lottoNum += lottoArray.get(j);

}

if(lottoNum == input)

{

System.out.println("congratulation! You won!");

i=7;

}

else

{

System.out.println("Lotto number is:- "+ Integer.toString(lottoNum)+"\nTry again");

}

}

if(i==5)

System.out.println("Sorry! You lost");

}

}

Lotto class

package exercise2;

import java.security.SecureRandom;

import java.util.ArrayList;

import java.util.List;

// lotto class

public class Lotto {

// list

List<Integer> ar = new ArrayList<>();

// constructor

public Lotto()

{

SecureRandom random = new SecureRandom();

ar.add(random.nextInt(1,9));

ar.add(random.nextInt(1,9));

ar.add(random.nextInt(1,9));

}

// get array of three value

public List<Integer> getAr() {

return ar;

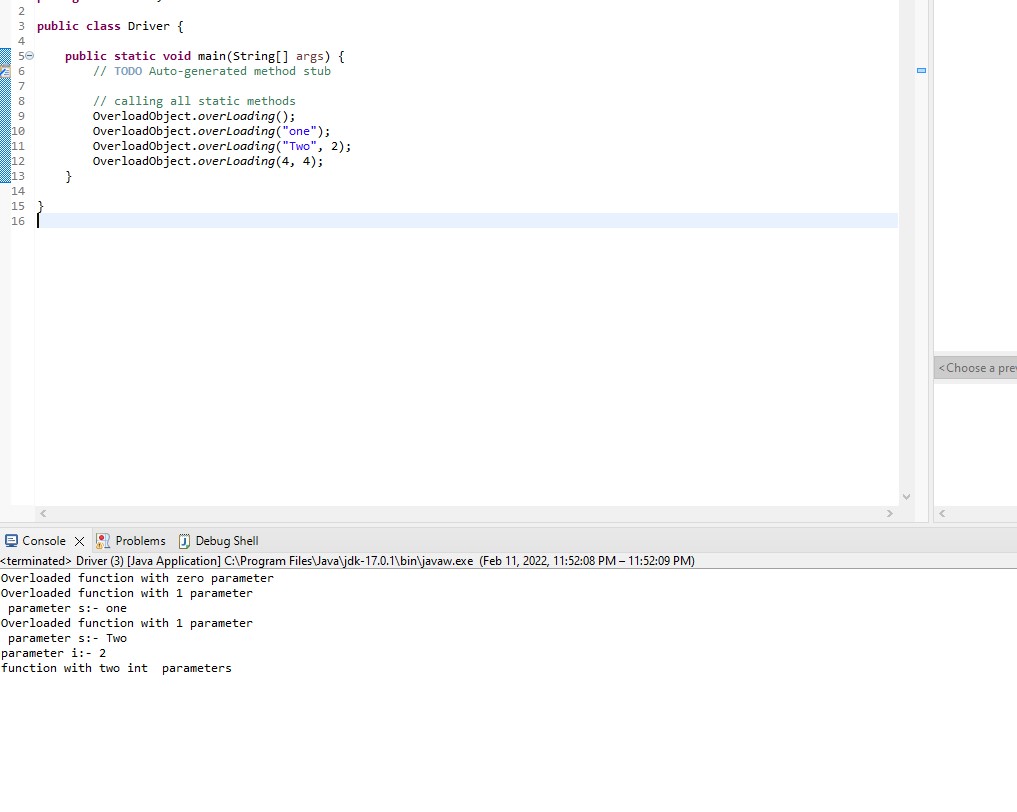
}

}

**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)



Driver class

**package** exercise3;

**public** **class** Driver {

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

// **TODO** Auto-generated method stub

// calling all static methods

OverloadObject.*overLoading*();

OverloadObject.*overLoading*("one");

OverloadObject.*overLoading*("Two", 2);

OverloadObject.*overLoading*(4, 4);

}

}

Overloading class with static overloaded methods

**package** exercise3;

**public** **class** OverloadObject {

// four different overloaded static methods

**public** **static** **void** overLoading()

{

System.***out***.println("Overloaded function with zero parameter");

}

**public** **static** **void** overLoading(String s)

{

System.***out***.println("Overloaded function with 1 parameter\n parameter s:- "+ s);

}

**public** **static** **void** overLoading(String s, **int** i)

{

System.***out***.println("Overloaded function with 1 parameter\n parameter s:- "+ s+"\nparameter i:- "+Integer.*toString*(i));

}

**public** **static** **void** overLoading(**int** i, **int** s)

{

System.***out***.println("function with two int parameters ");

}

}

**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% |