Kuwait University
College of Life Sciences
Department of Information Science

Lab Manual

ISC 240Programming and Problem Solving

Prepared by:

Dr. Hanady Abdulsalam

Eng. Aisha Al-Noori

Eng. Aisha Al-Houti

Revised Spring 2015-2016

Table of Contents

Laboratory Hardware and Software/Tools Requirements

In this lab the students will be using Java compiler and an Integrated Development Environment. JCreator with the Java Development Kit (JDK) are typically used for this purpose.

Laboratory Schedule

#	Lab Title	Lab activity
1	Introduction to Java	Exercise # 1
2	Primitive data types, Operations and Selection Statements	Exercise # 2
3	Selection Statements	Exercise # 3
4	Loops	Exercise # 4
5	Methods	Exercise # 5
6	Methods	Exercise # 6
7	Arrays	Exercise # 7
8	Arrays	Exercise # 8
9	String and Text I/O	Exercise # 9
10	File Input and Output / Objects and Classes	Exercise # 10
11	Objects and Classes	Exercise # 11
12	Thinking in Objects	Exercise # 12
13	Inheritance and Polymorphism	

Laboratory Policy

- Fallow the laboratory rules listed in appendix ―A‖
- To pass this course, the student must pass the lab-component of the course.
- Cheating in whatever form will result in F grade.
- Attendance will be checked at the beginning of each Lab.
- Absence for three (03) or more unexcused labs will result in a F grade in the Course. An official excuse must be shown in one week following return to classes.
- Every unexcused absence leads to a loss of 0.5 % marks.
- Cheating in Lab Work or Lab Project will result F grade in Lab.
- Late Submission of Home Works & Projects will not be accepted.
- There will be no make-up for any Quiz/Exam/Lab.
- Hard work and dedication are necessary ingredients for success in this course.

Lab Evaluation (it not mandatory)

Activity	Weight
Lab Work (12)	5%
Lab Quizzes (4)	5%
Lab Final	5%
Total	15%

Introduction

This lab is an integral part of the course ISC 240 Programming and Problem Solving. The main objectives of the lab are to experience the applications of the different Programming and Problem Solving techniques taught in the class.

Familiarity with Lab Hardware and Software tools

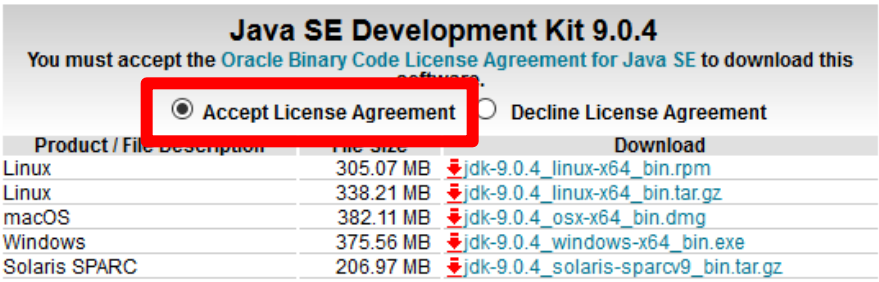
In this lab the students will be using Java compiler and an Integrated Development Environment and they are required to be installed Laboratory Tools Setup

1. Installation of the Java Development Kit:

a. **JDK 9**: go to http://www.oracle.com/technetwork/java/javase/downloads/index.html and download **Java Platform (JDK) 9**.

b. You will be forwarded to the download page.

c. Accept License Agreement.



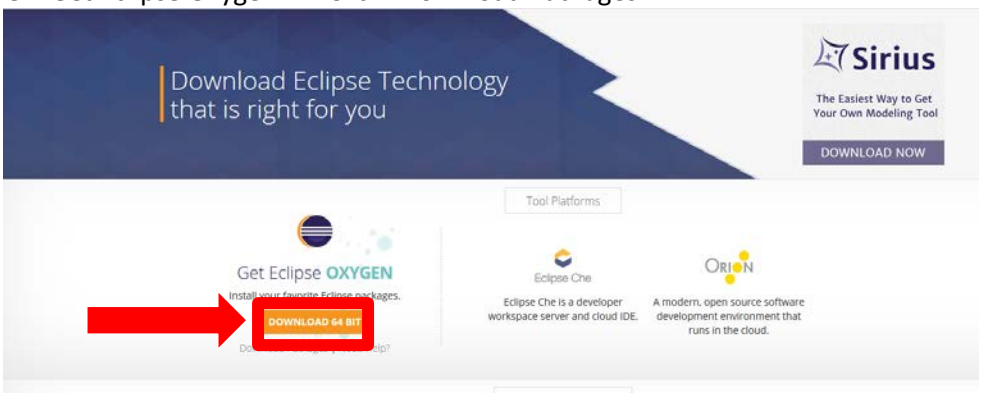
d. Select your platform (**Windows**) and download the executable file.

e. Install the JDK in the default directory.

2. Installation of the Eclipse.

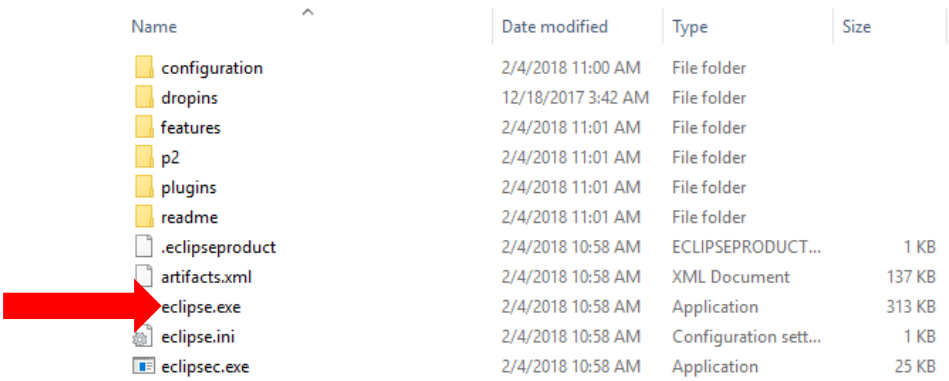
a. You can download it from http://www.eclipse.org/downloads/

b. Under "Get Eclipse Oxygen" ⇒ Click "Download Packages".



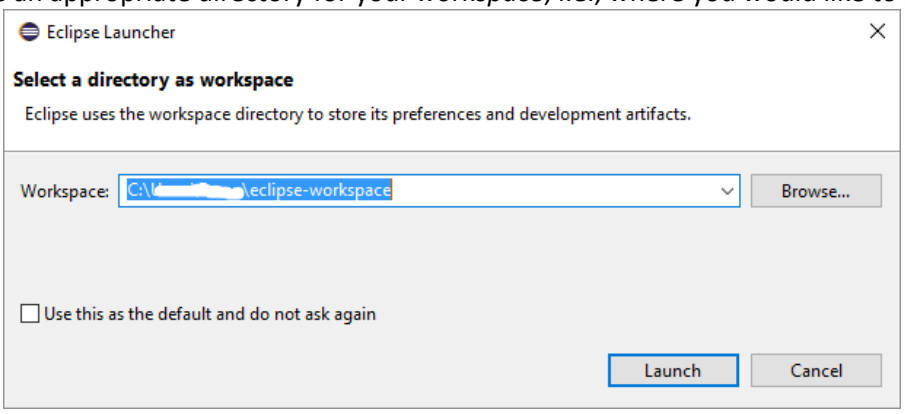
c. For beginners, choose the 2nd (or 3rd) entry "Eclipse IDE for Java Developers"

d. To install Eclipse, simply unzip the downloaded file into a directory of your choice



2. Write, edit, compile, execute and debug a simple Java application program.

1. Launch Eclipse by running "`eclipse.exe`" from the Eclipse installed directory.
2. Choose an appropriate directory for your *workspace*, i.e., where you would like to save your files

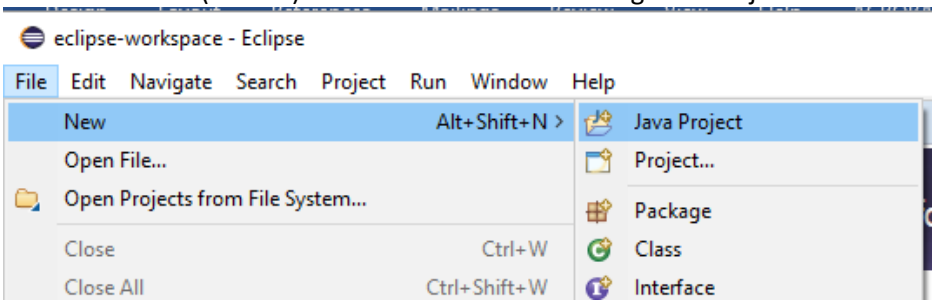


3. For each Java application, you need to create a project to keep all the source files, classes and relevant resources. There are many ways to open this wizard –

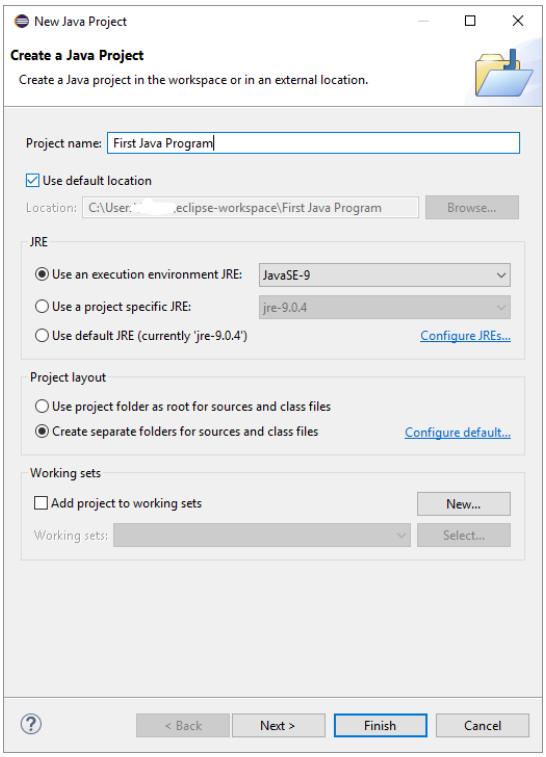
a. By clicking on the File menu and choosing New →Java Project.

b. By right clicking anywhere in the Project Explorer and selecting New → Java Project.

c. By clicking on the New button ( ) in the Tool bar and selecting Java Project.

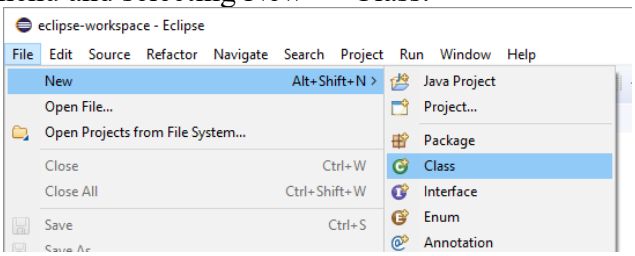


4. The New Java Project Wizard has two pages. On the first page − Enter the Project Name "First Java Program". Click Finish

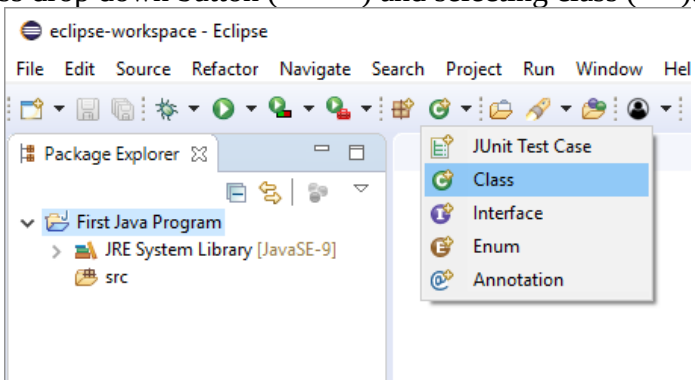


5. You can use the New Java Class wizard to create a Java class. The Java Class wizard can be invoked in different ways −

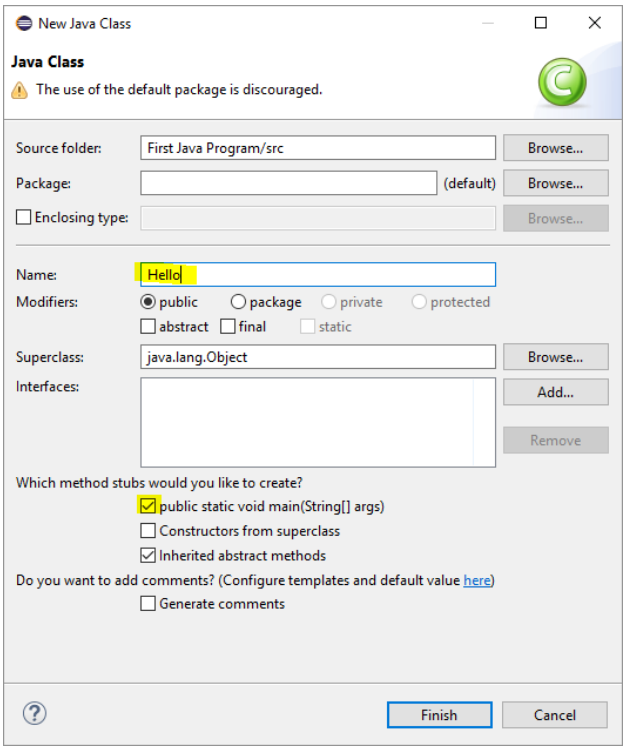
- By clicking on the File menu and selecting New → Class.



- By right clicking in the package explorer and selecting New → Class.
- By clicking on the class drop down button ( ) and selecting class ( ).



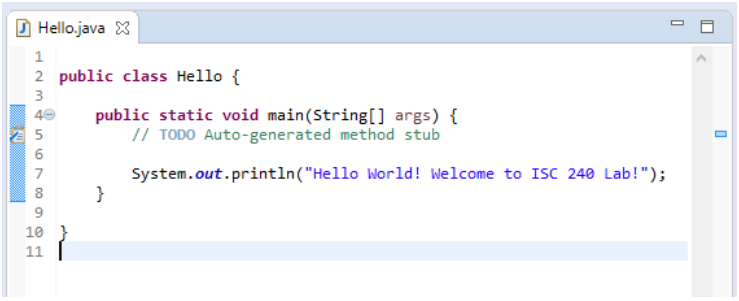
6. The "New Java Class" dialog pops up.



a.In "Source folder", keep the "First Java Program"
b. In "Package", delete the content if it is not empty
c.In "Name", enter "`Hello`"
d. Check "`public static void main(String[] args)`"
e. Don't change the rest. Click "Finish" button.

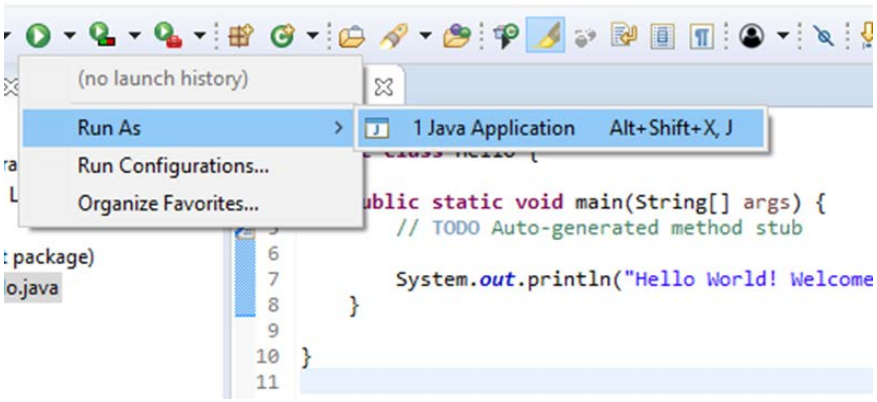
7. The source file "`Hello.java`" opens on the editor panel (the center pane). Enter the following codes:

```
public class Hello {
  public static void main(String[] args) {
        System.out.println("Hello World! Welcome to ISC 240 Lab!");
  }

}
```

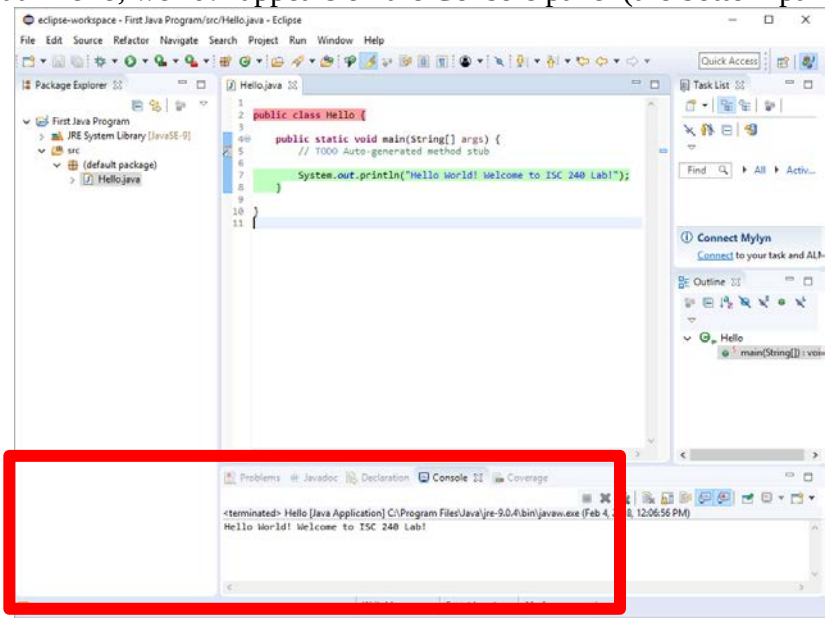


There is no need to *compile* the Java source file in Eclipse explicitly. It is because Eclipse performs the so-called *incremental compilation*, i.e., the Java statement is compiled as and when it is entered.

8. To run the program, right-click anywhere on the source file "`Hello.java`" (or choose "Run" menu) ⇒ Run As ⇒ Java Application.



9. The output "Hello, world!" appears on the Console panel (the bottom pane).



NOTES:

- You should create a NEW Java project for EACH of your Java application.
- Clicking the "Run" button (with a "Play" icon) runs the recently-run program (based on the previous configuration). Try clicking on the "down-arrow" besides the "Run" button.

Correcting Syntax Errors

- Eclipse performs incremented compilation, as and when a source line is entered. It marked a source line with syntax error with a RED CROSS. Point your cursor at the RED CROSS to view the error message.
- You **CANNOT RUN** the program if there is any **syntax error** (marked by a RED CROSS before the filename). Correct all the syntax errors; and RUN the program.

Laboratory #1 – Introduction to Java

1. **Laboratory Objective:**

   The objective of this laboratory is to introduce students to the basic concepts in Java Programming Language as well as the basic forms for the Input and the Output Statements in the Language.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:
   - Install Java compiler and an Integrated Development Environment.
   - Write, edit, compile, execute and debug a simple Java applications program.
   - Recognize and use conventional and standardized programming style and techniques for writing clear and readable programs.
   - Write and execute programs on output statement using **System.out** object method **println**.
   - Use variables to store data.
   - Use arithmetic operators to perform calculations.

3. **Introductory Concepts**
   ---

4. **Laboratory Instructions**

   Recognize and use conventional and standardized programming style and techniques for writing clear and readable programs.

- **Compiler problems:**
  - ☒ **Syntax** of the language is the rules for the correct way to write a program.
  - ☒ If you mistype part of a program, the compiler may issue a **syntaxerror or compilation error**.

  - ☒ **Common programming errors**:
    - × Forgetting one of the delimiters of a multiple-line comment.
    - × Not using the proper uppercase and lowercase letters for an identifier (Java is case sensitive).
    - × It is an error for a public class if the file name is not identical to the class name.
    - × It is an error not to end a file name with the **.java** extension for a file containing a class declaration. If that extension is missing, the Java compiler will not be able to compile the class declaration.
    - × If braces do not occur in matching pairs, the compiler indicates an error.
    - × Omitting the semicolon at the end of a statement.
    - × Splitting a statement in the middle of an identifier or a string.
    - × If a string does not appear between double quote characters on one line in a program.

❒ **Good Programming Practices:**

- ✓ Begin your program with a comment that explains the purpose of program, author and date.
- ✓ Use blank lines and space characters to enhance program readability.
- ✓ By convention, always begin a class name with a capital letter and start each subsequent word in the class name with a capital letter.
- ✓ Make sure when you type an opening left brace **{** , you type another closing right brace **}**.
- ✓ Use indentation (Tab key), to make your program more readable.
- ✓ Following the closing right brace (}) of a method body or class declaration with an end-of-line comment indicating the method or class declaration to which the brace belongs improves program readability.
- ✓ Place a space after each comma (,) in an argument list to make programs more readable.

5. Laboratory Problem Description

Problem 1

```
publicclass Welcome

{

   publicstaticvoid main (String args[])

   {

         System.out.println("welcome to ISC 240 LAB!");

   }

 }
```

Use JCreator to compile and execute the previous program.

Modifying your first java program

a. Use single-line and multiple line comments in your program.
b. Display the same text using string concatenation.

Problem 2

Write an application that declares and initializes three integers then it displays the sum, average and product in the command window. [***Note:*** The calculation of the average in this exercise can be represented as an integer.]

Sample Output

```
For the numbers 10, 20 and 30

Sum is 60

Product is 6000

Average is 20
```

Problem Template

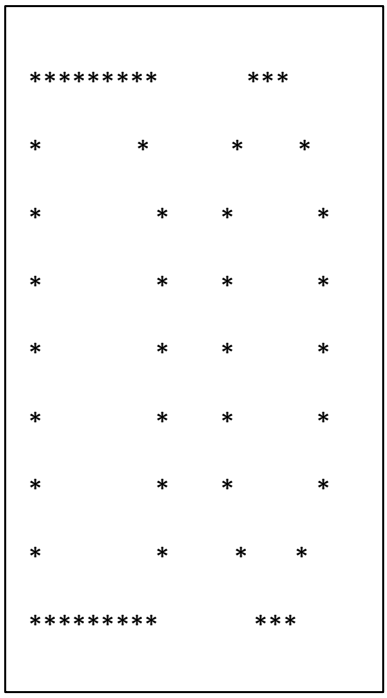
```
// Lab 1: Calculate.java

// performing simple calculations on three integers

public class Calculate

{

   public static void main (String args [ ])

   {

         int number1 = 10; // first number

         int number2 = 20; // second number

         int number3 = 30; // third number

         int average;     // average of the numbers

         int product;     // product of the numbers

         int sum;          // sum of the numbers

         // perform calculations

         sum = number1 + number2 + number3;

         /* write statements to calculate the product and the

            average */

         /* write statements to display the sum, product and

               average */

   }  // end main

}  // end class Calculate
```

6. Laboratory Exercises

Lab Exercise 1:

Write an application that displays a box and an oval using asterisks (*), as shown in the sample output:

Sample Output:



7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Laboratory # 2 – Primitive data types, Operations and Selection Statements

1. **Laboratory Objective:**

   The objective of this laboratory is to introduce students to the basics of declaring Variables, and Constants as well as the use of different Operators in the Java Programming Language.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:

   - Use variables to store data.
   - Use arithmetic operators to perform calculations.
   - Obtain input from the console using the **Scanner** class.
   - Obtain input using the **JOptionPane** input dialog boxes.
   - Display output using the **JOptionPane** message dialog boxes.
   - Implement selection control using **if** and **if-else** statements.

3. **Introductory Concepts**
   ---

4. **Laboratory Instructions**

☒ **Common programming errors**:

- ✗ All **import** declarations must appear before the first class declaration in the file. Placing import declarations inside a class declaration's body or after a class declaration is a syntax error.
- ✗ Forgetting the left and/or right parenthesis for the condition in an **if** statement is a syntax error.
- ✗ Confusing the equality operator, ==, with the assignment operator, =, can cause a logic error or a syntax error. The equality operator should be read as "is equal to" and the assignment operator should be read as "gets" or "gets the value of". To avoid confusion some people read the equality operator as "double equals" or "equals equals".
- ✗ It is a syntax error if the operators ==, !=, >=, <= contain spaces between their symbols.
- ✗ Reversing the operators !=, >= and <= is a syntax error.
- ✗ Placing a semicolon immediately after the right parenthesis of the condition in an **if** statement is normally a logic error.

❒ **Good Programming Practices:**

- ✓ Place spaces on either side of a binary operator to make it stand out and make the program more readable.
- ✓ Using parenthesis for complex arithmetic expressions, even when the parenthesis is not necessary can make the arithmetic expressions easier to read.
- ✓ Indent an **if** statement body to enhance the program readability.

5. Laboratory Problem Description

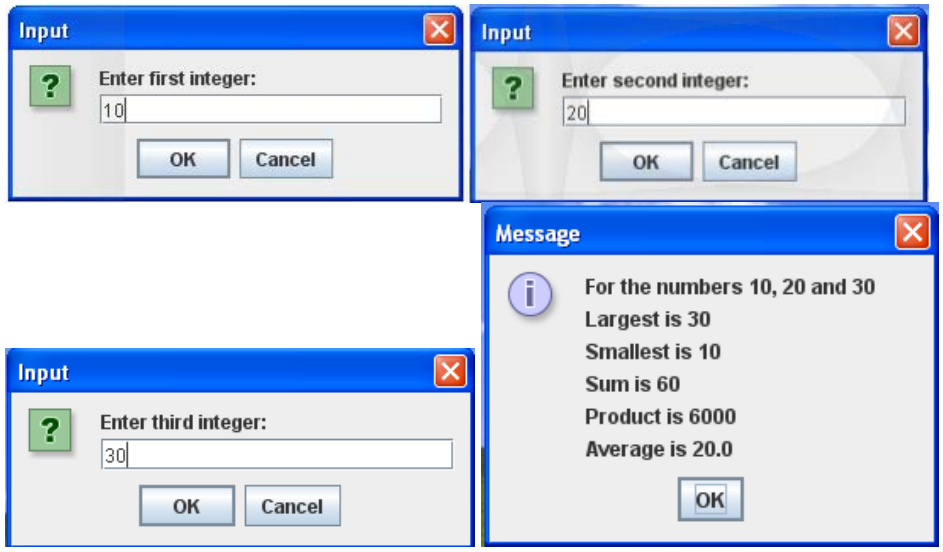
Problem 1

Write a Java application that inputs three integers from the user and displays the sum, average, product, smallest and largest of the numbers. You should implement two versions of your application. The first version should use **Scanner** class to input data from the console and use **System.out.println** to output the results into the command window. The second version should input data using **JOptionPane.showInputDialog** and output the results using **JOptionPane.showMessageDialog**.

Sample Output 1

```
Enter first integer: 10
Enter second integer: 20
Enter third integer: 30

For the numbers 10, 20 and 30
Largest is 30
Smallest is 10
Sum is 60
Product is 6000
Average is 20
```

Sample Output 2



Problem 2

Write an application that reads an integer from the user and determines and prints whether it is odd or even [***Hint:*** Use the reminder operator. An even number is a multiple of 2. Any multiple of 2 leaves a reminder of 0 when divided by 2]

Problem Template

```
// Lab 2: OddEven.java
// Finds if the number is odd or even

import java.util.Scanner;

public class OddEven
{
   public static void main (String args [ ])
   {
         int num;

         // create an object of class Scanner to be used to read
   // data from the keyboard

         // ask the user to enter a number
         // write a statement to read a number and assign it to num

         // determine whether the num is odd or even

         // print the result
   }
}
```

Sample Output

```
Enter a number
5
The number is odd
Enter a number
20
The number is even
```

Problem 3

The general form of the quadratic equation is $ax^2 + bx + c = 0$. Write a program that solves the quadratic equation using the quadratic formula $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. The user should enter the values of a, b and c from a java frame and the two possible values of x should be displayed in a java frame. Note: [you may use some of class Math methods such as **Math.sqrt()** and **Math.pow()**].

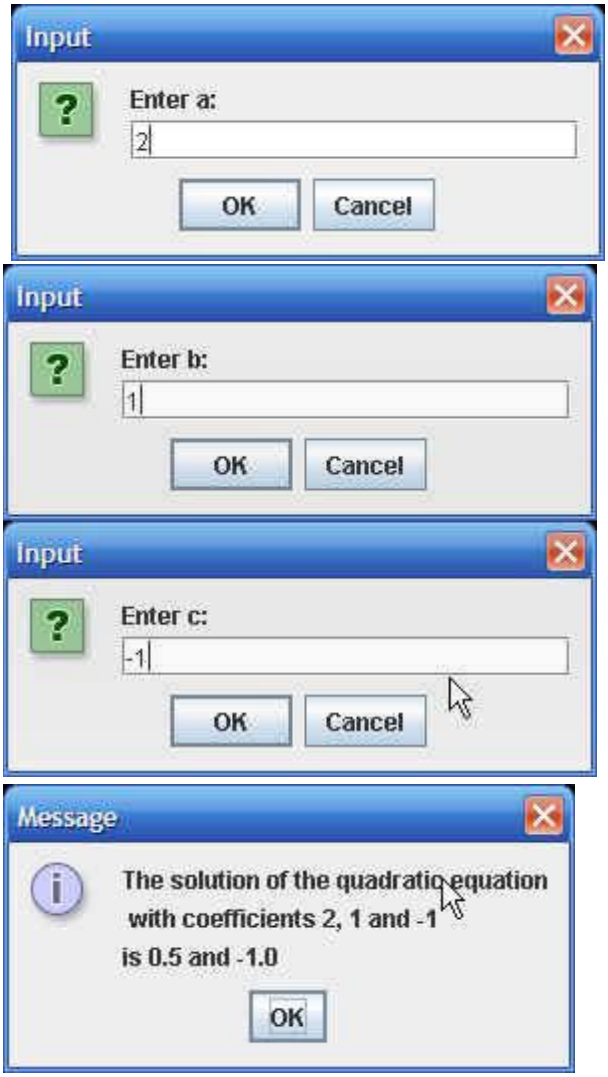
Template

```
import javax.swing.JOptionPane;

class QuadraticGUI
{
  public static void main (String args[])
  {
    // declare three variables (a, b, & c) of type integer
    // declare two variables (x1 and x2) of type double

    // use JOptionPane.showInputDialog("Enter a: ")
    // to read the string from the user
    // you have to convert that string into integer
   // use Integer.parseInt to convert it

    // do the above steps again for b and c

    // find the values of x1 & x2 according to the equation
    JOptionPane.showMessageDialog(null,"The solution of “+
    “the quadraticequation\n with coefficients " + a +
", " + b + " and " + c + "\nis " + x1 + " and " + x2);
    System.exit(0);
  }
}
```

Sample Output

Some values of x could be complex numbers, so the output will be NaN. Use the following values of a, b and c to test your program:

a	b	c
2	1	-1
16	8	-3
-1	2	2
1	14	44



6. Laboratory Exercises

Lab Exercise 2

Write a Java application that reads three integers and determines and prints whether

a. their product positive or negative.
b. their sum is multiple of 7.

Sample Output

```
Enter three integers
2
6
20
the product 240 is positive
the sum 28 is a multiple of 7
```

7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Laboratory # 3 – Selection Statements

1. **Laboratory Objective:**

   The objective of this laboratory is to train students on how and when to use different selection statements.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:

   - Use Boolean expressions to control selection statements.
   - Write expressions using the conditional operators.
   - Implement selection control using **if** and **if-else** statements.
   - Implement selection control using **switch** statements.

3. **Introductory Concepts**
   ---

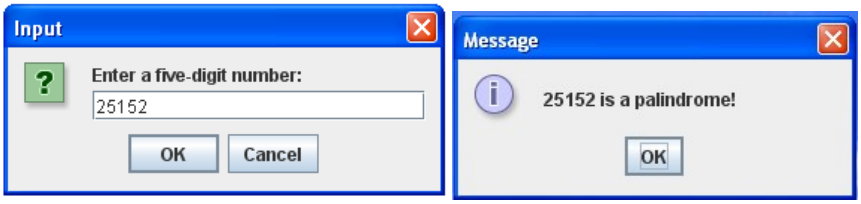
4. **Laboratory Instructions**

5. **Laboratory Problem Description**

   Problem 1

   A palindrome is a sequence of characters that reads the same backward as forward. For example, each of the following five-digit integers is a palindrome: 12321, 55555, 45554 and 11611. Write an application that reads in a five-digit integer and determines whether it is a palindrome. If the number is not five digits long, display an error message and terminate the program. You should input data using **JOptionPane.showInputDialog** and output the results using **JOptionPane.showMessageDialog**.

   Sample Output



Problem-Solving Tips

1. Check the number of digits in the value input by the user. Use an **if** statement to determine whether the user input contains the proper number of digits. If not, terminate the program.
2. Use division and remainder calculations to obtain the separate digits. For a five-digit number to be a palindrome the first and fifth digits must be the same and the second and fourth digits must be the same.

Problem 2

Write a Java application that inputs two integers from the command line then it displays the following menu:

1. Add
2. Subtract
3. Multiply
4. Divide

The application will ask the user to input a menu option then it will display the appropriate output. Make sure that denominator in the division is not zero.

Sample Output

```
Enter first integer: 4

Enter second integer: 3

1. Add
2. Subtract
3. Multiply
4. Divide
Enter your choice: 3

4 * 3 = 12
```

6. Laboratory Exercises

Lab Exercise 3

Write a program that sorts three integers in decreasing order. The integers should be entered from the command line.

Sample Output

```
Enter first integer: 9

Enter second integer: 20

Enter third integer: 2

The three integers in decreasing order are  20, 9 and 2
```

7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Laboratory # 4 – Loops

1. **Laboratory Objective:** The objective of this laboratory is to train students on how and when to use different Loops in Java programs.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:

   - Use **while**, **do-while**, and **for** loop statements to control the repetition of statements.
   - Use Boolean expressions to control loop statements.
   - Write nested loops.
   - Implement program control with **break** and **continue**.

3. **Introductory Concepts**

   ---

4. **Laboratory Instructions**

☒ **Common programming errors**:

- × Not providing, in the body of a **while** statement, an action that eventually causes the condition in the while to become false normally results in a logic error called an *infinite loop*, in which the loop never terminates.
- × Using the value of a local variable before it is initialized results in a compilation error. All local variables must be initialized before their values are used in expressions.
- × Assuming that integer division rounds (rather than truncates) can lead to incorrect results. For example, 7 ÷ 4, which yields 1.75 in conventional arithmetic, truncates to 1 in integer arithmetic, rather than rounding to 2.
- × Choosing a sentinel value that is also a legitimate data value is a logic error.
- × Omitting the braces that delimit a block can lead to logic errors, such as infinite loops. To prevent this problem, some programmers enclose the body of every control statement in braces even if the body contains only a single statement.
- × Because floating-point values may be approximate, controlling loops with floating-point variables may result in imprecise counter values and inaccurate termination tests.
- × Using an incorrect relational operator or an incorrect final value of a loop counter in the loop-continuation condition of a repetition statement can cause an off-by-one error.
- × Using commas instead of the two required semicolons in a for header is a syntax error.
- × When a for statement's control variable is declared in the initialization section of the for's header, using the control variable after the for's body is a compilation error.
- × Placing a semicolon immediately to the right of the right parenthesis of a for header makes that for's body an empty statement. This is normally a logic error.

- ✗ Not using the proper relational operator in the loop-continuation condition of a loop that counts downward (e.g., using i <= 1 instead of i >= 1 in a loop counting down to 1) is usually a logic error.
- ✗ Forgetting a break statement when one is needed in a switch is a logic error.
- ✗ In expressions using operator &&, a condition—we will call this the dependent condition—may require another condition to be true for the evaluation of the dependent condition to be meaningful. In this case, the dependent condition should be placed after the other condition, or an error might occur. For example, in the expression (i != 0) && (10 / i == 2), the second condition must appear after the first condition, or a divide-by-zero error might occur.

❒ **Good Programming Practices:**

- ✓ In a sentinel-controlled loop, the prompts requesting data entry should explicitly remind the user of the sentinel value.
- ✓ Unlike binary operators, the unary increment and decrement operators should be placed next to their operands, with no intervening spaces.
- ✓ Place blank lines above and below repetition and selection control statements, and indent the statement bodies to enhance readability.

5. Laboratory Problem Description

Problem 1

Write a program that reads an unspecified number of integers terminated by 0. The program calculated and prints the maximum and minimum among these numbers. Input data using **JOptionPane.showInputDialog**and output the results using **JOptionPane.showMessageDialog**.

Problem 2

Write a menu-driven java program that draws square and triangle. The program displays a menu of choices and prompts the user to select a menu item. It uses a switch statement to process the user selection, read the size of the shape and the character used to draw the selected shape. The menu should be displayed until the user choose to exit the program.

Sample Output

```
Choose from the menu:
1. Draw Triangle
2. Draw Square
3. Exit
Enter your choice: 1
Enter the size of the triangle: 7
Enter the character: #
#
##
###
####
#####
######
#######
Choose from the menu:
1. Draw Triangle
2. Draw Square
3. Exit
Enter your choice: 2
Enter the side of the square: 4
Enter the character: @
@@@@
@@@@
@@@@
@@@@
```

```
Choose from the menu:

1. Draw Triangle

2. Draw Square

3. Exit

Enter your choice: 3

Bye ... !
```

6. Laboratory Exercises

Lab Exercise 4

Write Java application that inputs a series of 10 integers and determines and prints the largest integer. Your program should use at least the following three variables:

a) **counter**: A counter to count to 10 (i.e., to keep track of how many numbers have been input and to determine when all 10 numbers have been processed).
b) **number**: The integer most recently input by the user.
c) **largest**: The largest number found so far.

Sample Output

```
Enter 10 integers

Enter number: 9

Enter number: -40

Enter number: 44

Enter number: 121

Enter number: 9

Enter number: 22

Enter number: 21

Enter number: -32

Enter number: 456

Enter number: 1

The largest is 456
```

7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Laboratory # 5 – Methods

1. **Laboratory Objective:** The objective of this laboratory is to train students on how to use methods as an important Java Programming Tool in modular programming.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:

   - Create methods, invoke methods, and pass arguments to a method.
   - Determine the scope of variables

3. **Introductory Concepts**

   ---

4. **Laboratory Instructions**

☒ **Common programming errors**:

- × Declaring method parameters of the same type as float x, y instead of float x, float y is a syntax error-a type is required for each parameter in the parameter list.
- × Declaring a method outside the body of a class declaration or inside the body of another method is a syntax error.
- × Omitting the return-value-type in a method declaration is a syntax error.
- × Placing a semicolon after the right parenthesis enclosing the parameter list of a method declaration is a syntax error.
- × Redeclaring a method parameter as a local variable in the method's body is a compilation error.
- × Forgetting to return a value from a method that should return a value is a compilation error. If a return value type other than void is specified, the method must contain a return statement that returns a value consistent with the method's return-value-type. Returning a value from a method whose return type has been declared void is a compilation error.

5. **Laboratory Problem Description**

Problem 1

Write a method that returns an integer reversed. Use the following method header:

public static int reverse(int number)

Test your method in the **main**. Input data using **JOptionPane.showInputDialog** and output the results using **JOptionPane.showMessageDialog**.

Problem 2

Write a method to compute the following series:

$$1-\frac{1}{x^2}+\frac{1}{x^4}-\frac{1}{x^6}+\cdots+\frac{1}{x^n}$$

Use the following method header:**public static double computeSeries(int x, int n)**

Note that **n** should be an even number, if it is not you should find the sum of the terms up to **n-1**. Test your method in the **main**.

Sample Output

```
Enter the value of x

2

Enter the value of n

5

the value of the series is 0.813
```

Problem 3

Write a method that returns the sum of the digits in a given integer. Use the following method header:
public static int sumOfDigits(int number)
Test your method in the **main**. Input data using **JOptionPane.showInputDialog** and output the results using **JOptionPane.showMessageDialog**.

6. Laboratory Exercises

Lab Exercise 5

Write a method **minimum** that takes three floating-point numbers and returns the smallest. Incorporate the method into an application that reads three values from the user, determines the smallest value and displays the result.

Sample Output

```
Enter first number: 2.54

Enter second number: 3

Enter third number: 10.8

Minimum is 2.54
```

7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Laboratory # 6 – Methods

1. **Laboratory Objective:** The objective of this laboratory is to train students on how to use methods as an important Java Programming Tool in modular programming.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:

   - Create methods, invoke methods, and pass arguments to a method.
   - Determine the scope of variables

3. **Introductory Concepts**

   ---

4. **Laboratory Instructions**

   ---

5. **Laboratory Problem Description**

   Write a class that plays "Guess the number" as follows: Your class should use method **generateNumber()**to choose the number to be guessed by selecting a random integer in the range 1 to 100. The class then uses **userGuess()**to displays the prompt **—G**uess a number between 1 and 100**”**. Then the player inputs a first guess. If the player's guess is incorrect, your method should check if the guessed number is between 1 and 100 ask the user to enter number in the correct range and read the guessed number again, or if the guessed number is greater than the correct number display **“Too high. Try again”** and read the number again, or if the guessed number is less than the correct number display **“Too low. Try again”** and read the number again. Keep reading guessed numbers until the user gets the correct number when the user does so display **“Congratulations. You guessed the number”**.

   Sample Output

   ```
   Guess a number in the range 1 to 100

   20

   Too low. Try again.

   40

   Too high. Try again.
   ```

```
35

Too low. Try again.

37

Too low. Try again.

39

Congratulations. You guessed the number!
```

6. Laboratory Exercises

Lab Exercise 6

Write a method that returns the distance of two points using the following method header:

public static double distance (double x1, double y1, double x2, double y2)

the formula for computing the distance between two points is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

In the **main** method, you should prompt the use to enter the x and y coordinates of the two points. Then pass the four values to the method to display the output.

Sample Output

```
First point

Enter x1: 2

Enter y1: 4

Second point

Enter x2: 3

Enter y2: 6

the distance between the two points is 2.236
```

7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Laboratory # 7 – Arrays

1. **Laboratory Objective:** The objective of this laboratory is to train students on how to use Arrays data structure in the context of the Java Programming Language.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:

   - Declare an array, initialize an array and refer to individual elements of an array.
   - Use for-each (enhanced for) loops.
   - Declare, create, and initialize an array using an array initializer.
   - Copy contents from one array to another.
   - Develop and invoke methods with array arguments and return values.

3. **Introductory Concepts**

   ---

4. **Laboratory Instructions**

   ☒ **Common programming errors**:

   - Using a value of type **long** as an array index results in a compilation error. An index must be an **int** value or a value of a type that can be promoted to **int**—namely, **byte**, **short** or **char**, but not **long**.
   - In an array declaration, specifying the number of elements in the square brackets of the declaration (e.g., **int c[12];**) is a syntax error.
   - Declaring multiple array variables in a single declaration can lead to subtle errors. Consider the declaration **int[] a, b, c;**. If **a**, **b** and **c** should be declared as array variables, then this declaration is correct—placing square brackets directly following the type indicates that all the identifiers in the declaration are array variables. However, if only **a** is intended to be an array variable, and **b** and **c** are intended to be individual **int** variables, then this declaration is incorrect—the declaration **int a[], b, c;** would achieve the desired result.
   - The **new** operator is not used in the array initialize syntax. Using an array initializer, you have to declare, create, and initialize the array all in one statement. Splitting it would cause a syntax error.
   - Accessing an array out of bound causes a run-time error.
   - Programmers often mistakenly reference the first element in an array with index 1, but it should be 0. This is called the off-by-one error.

❒ **Good Programming Practices:**

- ✓ For readability, declare only one variable per declaration. Keep each declaration on a separate line, and include a comment describing the variable being declared.
- ✓ Constant variables also are called named constants or read-only variables. Such variables often make programs more readable than programs that use literal values (e.g., 10)—a named constant such as **ARRAY_LENGTH** clearly indicates its purpose, whereas a literal value could have different meanings based on the context in which it is used.

5. Laboratory Problem Description

Problem 1

Write a program that declares, creates, and initializes an array of integers using array initializer. Copy the array into two different arrays, using a loop and then using **System.copyarray** method. Display the two arrays using **for**-each (enhanced **for**) loops.

Sample Output

```
the source array

2  4  6  8  10

after copying the array using a loop

2  4  6  8  10

after copying the array using System.copyarray method

2  4  6  8  10
```

Problem 2

Write a method that computes the standard deviation of **n** numbers. The formula for computing the standard deviation is:

$$deviation = \sqrt{\frac{\sum_{i=1}^{n}(x_i - mean)^2}{n-1}} \qquad mean = \frac{\sum_{i=1}^{n} x_i}{n}$$

To compute **deviation** with this formula, you have to store the individual numbers using an array, so that they can be used after the **mean** is obtained. Use **{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}** to test the method.

Your program should contain the following methods:

public static double deviation(double[] x)

public static double mean(double[] x)

public static void printArray(double[] x)

Sample Output

```
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0

The mean is 5.5

The standard deviation is 3.028
```

Problem 3

Write a program that declares two arrays A and B of type integer. You have to use array initialize to initialize array B, and use **JOptionPane.showInputDialog()** to read elements of array A. your program should then find the sum of the two arrays A + B and store the result in array C. finally you should print the contents of array C using **JOptionPane.showMessageDialog().**

Note: remember that **JOptionPane.showInputDialog()**reads string from the user so you have to convert it to integer using **Integer.parsInt()**.

Hint: for any manipulation with the arrays you need to use for loop to go through the elements of the array using the index.

6. Laboratory Exercises

Lab Exercise 7

Write a method that finds the largest element in an array of integers. Test your method in **main**. Use **{10, 22, 4, 5, 10, -120, 2, 22}** to test the method.

Sample Output

```
for the list:  10  22  4  5  10  -120  2  22

the max element is 22
```

7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Laboratory # 8 – Arrays

1. **Laboratory Objective:** The objective of this laboratory is to train students on how to use Arrays data structure in the context of the Java Programming Language.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:

   - Declare a method with variable-length argument line.
   - Use the methods in the **Arrays** class.
   - Declare and create two-dimensional arrays.

3. **Introductory Concepts**

   ---

4. **Laboratory Instructions**

   ---

5. **Laboratory Problem Description**

   Problem 1

   Write a method that accepts an unspecified number of double values and returns the maximum.

   The method header is specified as follows:

   public static double findMax(double… numbers)

   Test your method in **main**.

   Sample Output

   ```
   the max of 2.5, 1.3, 6, 100, 90 is 100.0

   the max of 9, 2, 8, 11, 5 is 11.0
   ```

Problem 2

Write a method that accepts a two dimensional array then it returns a one dimensional array that stores the average of each row. The header of the method is as follows:

public static double[ ] findRowAverage(int[ ][ ] a)

Test your method in **main**. You should fill the two dimensional array with random integers in the range 0 to 99.

Sample Output

```
47   31   98

62   68   70

69    9   93

45   34   42

52    7   65

row 0: average = 58.67

row 1: average = 66.67

row 2: average = 57.00

row 3: average = 40.33

row 4: average = 41.33
```

Problem 3

Write a Java application for performing operations on Matrices. Your application should do the following:

- **a)** Input the elements of the two matrices.
- **b)** Add two matrices.
- **c)** Output the elements of the resulted matrix.

Sample Output

```
Enter the matrices size
# of rows: 2
# of columns: 2
```

```
Enter matrix 1 elements
element ( 0, 0 ): 2
element ( 0, 1 ): 5
element ( 1, 0 ): 5
element ( 1, 1 ): 2

Enter matrix 2 elements
element ( 0, 0 ): 2
element ( 0, 1 ): 1
element ( 1, 0 ): 2
element ( 1, 1 ): 1
matrix 1 + matrix 2 =
4  6
7  3
```

6. Laboratory Exercises

Lab Exercise 8

Write a method that accepts a two dimensional array then it returns a one dimensional array that stores the maximum of each row. The header of the method is as follows:

public static int[ ] findRowMax(int[ ][ ] a)

Test your method in **main**. You should fill the two dimensional array with random integers in the range 0 to 99.

Sample Output

```
5   67   83

19   40    5

18    4   46

22   11   84

64   76   80

row 0: max = 83

row 1: max = 40

row 2: max = 46

row 3: max = 84

row 4: max = 80
```

7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Laboratory # 9 – String and Text I/O

1. **Laboratory Objective:** The objective of this laboratory is to train students on how to write data to a file andhow to process Strings.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:

   - Use the **String** class to process strings and the **Character** class to process characters.
   - Pass arguments to the **main** method from the command line.
   - Write data to a file using the **PrintWriter** class.

3. **Introductory Concepts**

   ---

4. **Laboratory Instructions**

   ---

5. **Laboratory Problem Description**

Problem 1

Write a program that passes a string as a command-line argument and displays the number of uppercase letters and the number of the lower-case letters in the string.

Sample Output

```
C:\>java CountingLetters My name is Aisha

Usage: java CountingLetters string

C:\>java CountingLetters Aisha

the number of uppercase letters is 1

the number of lowercase letters is 4
```

Problem 2

Write a program that passes an unspecified number of integers as separate strings to the **main** method and write the integers and the maximum value in a file.

Sample Output

```
C:\>java Max

Usage: java Max integer1 integer2 integer3 ...

C:\>java Max 25 2 66 14 100 26
```

In the file Maximum.txt:

```
25  2  66  14  100  26

the max is 100
```

6. Laboratory Exercises

Lab Exercise 9

Rewrite Problem 1 such that the output is displayed in a file named **letters.txt**.

Sample Output

```
C:\>java CountingLettersFile My name is Aisha

Usage: java CountingLetters string

C:\>java CountingLettersFile Aisha
```

In file letters.txt:

```
the number of uppercase letters is 1

the number of lowercase letters is 4
```

7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Laboratory # 10 – File Input and Output / Objects and Classes

1. **Laboratory Objective:** The objective of this laboratory is to train students on how to write data to a file and read data from the file. The concept of objects and classes is introduced in this lab.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:

   - Write data to a file using the **PrintWriter** class.
   - Read data from a file using the **Scanner** class.
   - Declare a class and create an object from a class.

3. **Introductory Concepts**

   ---

4. **Laboratory Instructions**

   ☒ **Common programming errors:**

   - ✗ It is a common mistake to put the **void** keyword in front of a constructor. For example,
     Public void Circle()

     {

     }

     In this case, **Circle()** is a method not a constructor.

5. **Laboratory Problem Description**

   Problem 1

   Suppose that a text file **scores.txt** contains an unspecified number of scores. Write a program that reads the scores from the file and write their total and average to text file **grades.txt**. Scores are separated by blanks.

Problem 2

Design a class named **Triangle** to represent a triangle. The class contains:

- Three double data fields named **side1**, **side2**, and **side3** that specify the three sides of the triangle. The default values are **1** for all the side.
- One class variable (static variable) named **noOfTriangles** to count the number of triangles.
- A no-argument constructor that creates a default triangle.
- A constructor that creates a triangle with the specified sides.
- A method named **getArea()** tha returns the area of a triangle.
- A method named **getPerimeter()** that returns the perimeter.

Write a test program that creates two **Triangle** objects. Assign sides **4**, **5**, and **6** to the first object and **1.5**, **2.5** and **3.5** to the second object. Display the properties of both objects and find their areas and perimeters.

6. Laboratory Exercises

Lab Exercise 10

Design a class named **Rectangle** to represent a rectangle. The class contains:

- Two double data fields named **side1** and **side2** that specify the two sides of the rectangle. The default values are **1** for all the side.
- A no-argument constructor that creates a default rectangle.
- A constructor that creates a rectangle with the specified sides.
- A method named **getArea()** tha returns the area of a rectangle.
- A method named **getPerimeter()** that returns the perimeter.

Write a test program that creates two **Rectangle** objects. Assign sides **4** and **5** to the first object and **1.5** and **2.5** to the second object. Display the properties of both objects and find their areas and perimeters.

7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Laboratory # 11 – Objects and Classes

1. **Laboratory Objective:** The objective of this laboratory is to introduce the concept of object-oriented programming in Java.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:

   - Declare a class and create an object from a class.
   - Declare private data fields with appropriate **get** and **set** methods.
   - Store and process objects in arrays.

3. **Introductory Concepts**

   ---

4. **Laboratory Instructions**

   ---

5. **Laboratory Problem Description**

   Design a class named **Triangle** to represent a triangle. The class contains:

   - Three private double data fields named **side1**, **side2**, and **side3** that specify the three sides of the triangle. The default values are **1** for all the side.
   - Public **set** and **get** methods for each of the three data fields.
   - A public no-argument constructor that creates a default triangle.
   - A public constructor that creates a triangle with the specified sides.
   - A public method named **getArea()** tha returns the area of a triangle.
   - A public method named **getPerimeter()** that returns the perimeter.

   Write a test program that creates an array of 5 **Triangle** objects. Initialize the **Triangle** objects by reading the three sides from the command-line. Then, print the three sides, the area and the perimeter of each **Triangle** object.

6. Laboratory Exercises

Lab Exercise 11

Design a class named **Rectangle** to represent a rectangle. The class contains:

- Two private double data fields named **side1** and **side2** that specify the two sides of the rectangle. The default values are **1** for all the sides.
- Public **set** and **get** methods for each of the two data fields.
- A public no-argument constructor that creates a default rectangle.
- A public constructor that creates a rectangle with the specified sides.
- A public method named **getArea()** tha returns the area of a rectangle.
- A public method named **getPerimeter()** that returns the perimeter.

Write a test program that creates an array of 5 **Rectangle** objects. Initialize the **Rectangle** objects by reading the two sides from the command-line. Then, print the two sides, the area and the perimeter of each **Rectangle** object.

7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Laboratory # 12 – Thinking in Objects

1. **Laboratory Objective:** The objective of this laboratory is to introduce the concept of object-oriented programming in Java.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:

   - Design programs using the object-oriented paradigm.
   - Use the keyword **this** to refer to the calling object itself.

3. **Introductory Concepts**
   ---

4. **Laboratory Instructions**

   ➢ **Programming Tips**

   - If a class has multiple constructors, it is better to implement them using **this(arg-list)** as much as possible. In general, a constructor with no or fewer arguments using **this(arg-list)**. This often simplifies coding and makes the class easier to read and maintain.
   - Java requires that the **this(arg-list)** statement appear first in the constructor before any other statements.

5. **Laboratory Problem Description**

   Design a class named **Course**. The class contains:

   - A private data field named **courseName** of type **String**.
   - A private data field named **students** of type **String[]**. The size of the array is **100**.
   - A private data field named **numberOfStudents** of type **int**.
   - A no-argument constructor that creates a default **Course** and sets the **courseName** to the default value **"ISC 100-51"**.
   - A constructor that creates a **Course** with the specified **courseName**.
   - Three **get** methods for data fields **courseName**, **students[]** and **numberOfStudents**.
   - A method named **addStudent(String)** that adds a student to the **Course**.
   - A method named **dropStudent(String)** that drops a student from the **Course**.
   - A method named **clear()** that removes all students from the **Course**.

   Write a test class to test your methods.

6. Laboratory Exercises

Lab Exercise 12

Add a new method named **search(String)** that returns a **String** telling if the specified student takes the course or not. Test your method in the **main**.

7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Laboratory # 13 – Inheritance and Polymorphism

1. **Laboratory Objective:** The objective of this laboratory is to introduce the concept of inheritance and polymorphism in Java.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:

   - Develop a subclass from a superclass through inheritance.
   - Understand polymorphism.
   - Restrict access to data and methods using the **protected** visibility modifier.

3. **Introductory Concepts**

   ---

4. **Laboratory Instructions**

   ---

5. **Laboratory Problem Description**

   Problem 1

   Write an inheritance hierarchy for classes **CircularShape**, **Circle** and **Cylinder** as shown in figure 1. Specify the instance variables and methods for each class. Write a program that instantiates objects of your classes and test your methods.

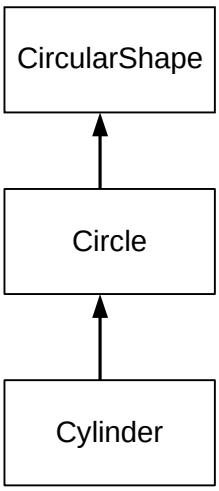


Figure 1

Formulas for Circle and Cylinder:

- **Circle:**
  Area = πr^2

  Circumference = $2 \pi r$

- **Cylinder:**
  Surface Area = $2 \pi r h$

  Volume = $\pi r^2 h$

Problem 2

Create a hierarchy for **class CircularShape** as shown in Figure 1. The class attributes and methods of each class are:

- Class **CirculareShape** should have **radius** (instance variable) of type integer and **pi** (class variable and constant) of type double. It should have two methods the first method to set the radius (**void setRadius(int)**), the second to get the radius (**int getRadius()**).

- Class **Circle** should have the **x** and **y** coordinates of the center both of type integer. It should also have five methods: **void setCenter(int , int)**, **int getX()**, **int getY()**, **double findArea()**,and **double findCircumference()**.

- Class **ThreeDimension** should have **type** (String) of the shape. It has also two methods: **void setType(String)** and **String getType()**.

- Class **Sphere** should have a constructor (**Sphere()**) that sets the type of the sphere to **―Sphere‖** and two methods to find the surface area (**double findSurfaceArea()**) and the volume (**double findVolume()**).

- Class **Cylinder** should have **height** of type integer, a constructor (**Cylinder()**) that sets the type of the cylinder to **―Cylinder‖** and three methods one to set the height (**void setHeight(int)**) and the others to find the surface area (**double findSurfaceArea()**) and the volume (**double findVolume()**).

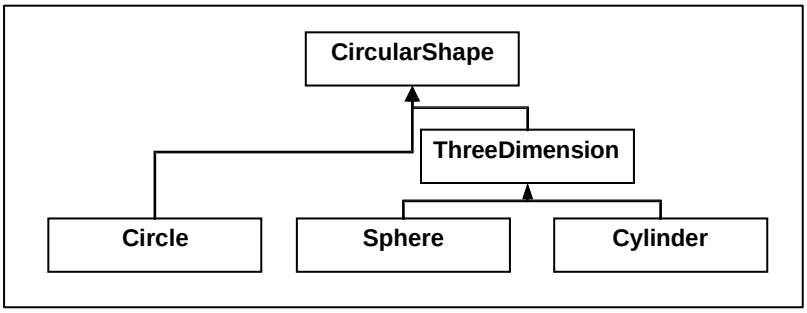


Figure 2

Write a program to test your class hierarchy according to the following:

1. Create objects for classes **Circle**, **Sphere** and **Cylinder**.
2. Set the radius of the circle (radius = 2) and set x and y coordinate of the center (the center is the origin). Print the radius, center, area and circumference of the circle.
3. Set the radius of the sphere (radius = 2). Print the shape type, surface area and volume of the sphere.
4. Set the radius and the height of the cylinder (radius = 2, height = 3). Print the shape type, surface area and volume of the cylinder.

- **Formulas for circular shapes:**

  - **Circle:**
    Area = πr^2
    Circumference = $2 \pi r$

  - **Sphere:**
    Surface Area = $4 \pi r^2$
    Volume = $4/3 \pi r^3$

  - **Cylinder:**
    Surface Area = $2 \pi r h$
    Volume = $\pi r^2 h$

Sample Output:

```
Circular shapes are Circle, Sphere and Cylinder
Shape is Circle
radius = 2
Center ( 0 , 0 )
Area = 12.56636
Circumference = 12.56636
```

```
Shape is Sphere
Surface Area = 50.26544
Volume = 33.51029333333333
Shape is Cylinder
Surface Area = 37.699079999999995
Volume = 37.699079999999
```

6. Laboratory Exercises

---

7. Homeworks

Reference

Introduction to Java programming, Liang, seventh edition

Appendix—A‖

Rules to fallow by Computer Lab Users

- The loud conversations / discussion that disturbing the other users is prohibited.
- Audio CDs or applications with audio output may only be used with headphones with minimum volume that it should not be disturb other users.
- All cell phones are to be turned off or set to silent while in the lab. If you receive a phone call, you should exit the lab before answering your cell phone.

- Do not bring food or beverages inside the lab.
  - Any file saved on the computer hard drive will be deleted without notice. Students should save their work onto an external storage device such as USB drive or CD.
  - Changing hardware and software configurations in the computer labs is prohibited. This includes modifications of the settings, modification of system software, unplugging equipment, etc.
  - Open labs are reserved for academic use only. Use of lab computers for other purposes, such as personal email, non-academic printing, instant messaging, playing games, and listening to music is not permitted.
  - Please leave the computer bench ready for the next patron. Leave the monitor on the login screen, and do not forget to take goods related to you. While leaving computer bench please push the chair inside the computer bench.
  - Users are responsible for their own personal belongings and equipment. Do not leave anything in the Computer Lab unattended for any length of time. The Computer Labs staffs are not responsible for lost or stolen items.
  - Users are not allowed to clear paper jams in the printer by themselves.
  - Operate the lab equipments with care.
- After using white-board the user must clean for other user.

Thanks for your cooperation.

Information Science Department

Appendix—B‖

Certification

LABORARTORY MANUAL FOR COURSE ISC 240
(Programming and Problem Solving)

#	Instructor name	Remarks	Signature	Date
1	Dr. Hanady Abdulsalam			
2	Dr. Paul Manual			
3	Eng. Aisha Al-Houti			
4	Eng. Aisha Al-Noori			