

# Lab Manual

**CSC103 – Programming Fundamentals**



**Department of Computer Science  
Islamabad Campus**

**Lab Contents:**

Java Installation, Algorithms, Errors, & Testing; Java Fundamentals-I; Java Fundamentals-II; Selection Control Structures; Repetition Control Structures; Mathematical, Characters and String Methods; Methods and Recursion; One Dimensional Arrays; Two Dimensional(2D) Arrays; Exception Handling; Code Refactoring and using Library Components; File Handling and Testing.

**Student Outcomes (SO)**

S.#	Description
1	Apply knowledge of computing fundamentals, knowledge of a computing specialization, and mathematics, science, and domain knowledge appropriate for the computing specialization to the abstraction and conceptualization of computing models from defined problems and requirements
2	Identify, formulate, research literature, and solve <i>complex</i> computing problems reaching substantiated conclusions using fundamental principles of mathematics, computing sciences, and relevant domain disciplines
3	Design and evaluate solutions for <i>complex</i> computing problems, and design and evaluate systems, components, or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations
4	Create, select, adapt and apply appropriate techniques, resources, and modern computing tools to <i>complex</i> computing activities, with an understanding of the limitations
5	Function effectively as an individual and as a member or leader in diverse teams and in multi-disciplinary settings.

**Intended Learning Outcomes**

Sr.#	Unit #	Course Learning Outcomes	Blooms Taxonomy Learning Level	SO
CLO -5	3-6	Implement a program using basic programming constructs.	<i>Applying</i>	2,4
CLO -6	1-7	Build a medium size application in a team environment.	<i>Creating</i>	2-5

**Lab Assessment Policy**

The lab work done by the student is evaluated using Psycho-motor rubrics defined by the course instructor, viva-voce, project work/performance. Marks distribution is as follows:

Assignments	Lab Mid Term Exam	Lab Terminal Exam	Total
25	25	50	100

**Note: Midterm and Final term exams must be computer based.**

## List of Labs

Lab #	Main Topic	Page #
Lab 01	<a href="#">Java Installation, Algorithms, Errors, and Testing</a>	01
Lab 02	<a href="#">Java Fundamentals-I</a>	10
Lab 03	<a href="#">Java Fundamentals-II</a>	21
Lab 04	<a href="#">Selection Control Structures</a>	31
Lab 05	<a href="#">Repetition Control Structures</a>	40
Lab 06	<a href="#">Mathematical, Characters and String Methods</a>	52
Lab 07	<a href="#">Methods and Recursion</a>	63
Lab 08	<a href="#">One Dimensional Arrays</a>	72
Lab 09	<b>Mid Term Exam</b>	
Lab 10	<a href="#">Two Dimensional(2D) Arrays</a>	82
Lab 11	<a href="#">Exception Handling</a>	88
Lab 12	<a href="#">Code Refactoring and using Library Components</a>	94
Lab 13	<a href="#">File Handling</a>	103
Lab 14	<a href="#">Testing</a>	110
Lab 15	<b>Final Term Exam</b>	

# **Lab 01**

## **Java Installation, Compiling, Executing, Errors, and Testing**

### **Objective:**

The objective of this lab is to give students handon JAVA Installation/JAVA IDE, translating algorithms in Java program and execute it after correcting errors in it.

### **Activity Outcomes:**

On completion of this lab student will be able

- Install JAVA SDK/ JAVA IDE
- Compile a program
- Execute JAVA program using JAVA SDK/JAVA IDE
- Test a program
- Debug program with syntax and logic errors.

### **Instructor Note:**

As a pre-lab activity, read Chapter 01 from the text book “Java How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.

## 1) Useful Concepts

This tutorial is for students who are currently taking a programming fundamentals course.

### Introduction to Java

*Java* is a powerful and versatile programming language, developed by James Gosling in 1995, for developing software running on mobile devices, desktop computers, and servers. It is popular because of its unique feature of writing a program once and run it anywhere.

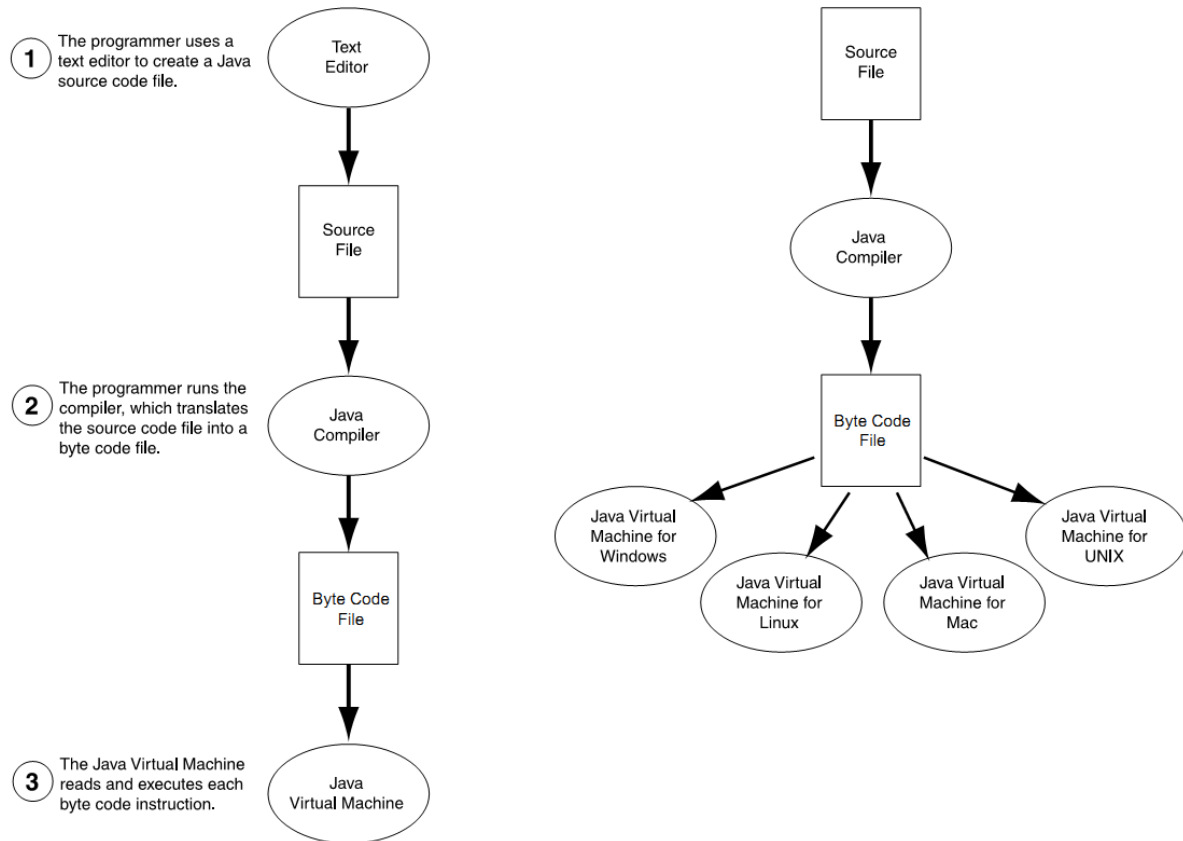
### JDK Versions

You can use the JDK command line utility to write Java programs. The JDK command line utility consists of a set of separate programs, such as compiler and interpreter, each of which is invoked from a command line. Besides the JDK command line utility, there are more than a dozen Java development tools on the market today, including IntelliJ, NetBeans, JBuilder, and Eclipse. These tools support an integrated development environment (IDE) for rapidly developing Java programs. Editing, compiling, building, debugging, and online help are integrated in one graphical user interface. Using these tools effectively will greatly increase your programming productivity.

### The compiler and the Java Virtual Machine

The compiler is the program that translates source code into a language that a computer can understand. This process is different in Java than some other high-level languages. Java translates its source code into byte-code using the `javac` command. Students should be exposed to how the `javac` command gets executed in the particular development environment. Also, since `javac` is a program that can be run at the command prompt if using a Unix environment.

The compiler generates byte-code that is then interpreted by the Java Virtual Machine. This process occurs when the student uses the `java` command to execute the program. Once again, how this command is executed will vary depending on computing environments, and `java` is also a program that can be run at the command prompt. The interpretation of the byte-code by the Java Virtual Machine is the reason Java is considered so portable. The byte-code that is generated by the Java compiler is always the same no matter what the machine or operating system. As long as the Java Virtual Machine is loaded onto a computer, that computer can interpret the byte-code generated by the compiler. This second computer can be a different type or even running a different operating system than the one that originally compiled the source code and the byte-code will still be correctly interpreted.



## 2) Solved Lab Activities

<i>Sr.No</i>	<i>Allocated Time</i>	<i>Level of Complexity</i>	<i>CLO Mapping</i>
<i>Activity 1</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>
<i>Activity 2</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>
<i>Activity 3</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>
<i>Activity 4</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>

**Activity 1:**

*This activity demonstrate the steps to be followed to install Java on the system*

**Solution:**

The JDK can be installed on the following platforms:

- 1- Microsoft Windows
- 2- Linux
- 3- macOS

**Step-1: Download the latest JDK for Windows**

You can download the JDK from [Java SE Development Kit Downloads](#)

**Step-2: Running the JDK Installer**

- 1- Start the JDK 17 installer (or latest) by double-clicking the installer's icon or file name in the download location.
- 2- Follow the instructions provided by the installer.
- 3- After the installation is complete, delete the downloaded file to recover the disk space.

**Step-3: Checking if it is installed**

To check if Java is installed (open command prompt and type following command)

```
java -version
```

## Activity 2:

*Setting Java Path (for using notepad, sublimeText or other text editor)*

### Solution:

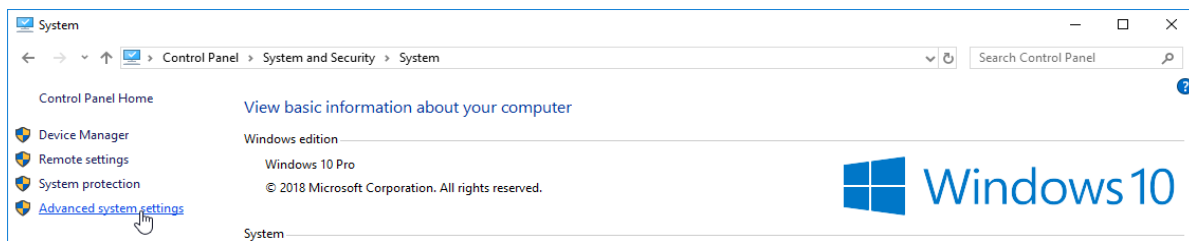
#### Temporary Path

- 1- Open the command prompt
- 2- Copy the path of the *JDK/bin* directory
- 3- Write in command prompt: *set path=copied\_path*

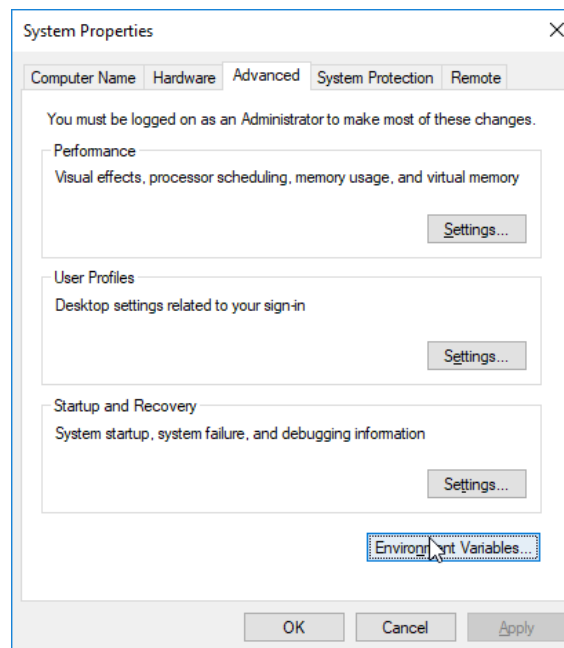
```
C:\Users\Your Name>set path=C:\Program Files\Java\jdk\bin
```

#### Permanent Path using Environment Variables Settings

- 1- Go to "System Properties" (Can be found on Control Panel > System and Security > System > Advanced System Settings)

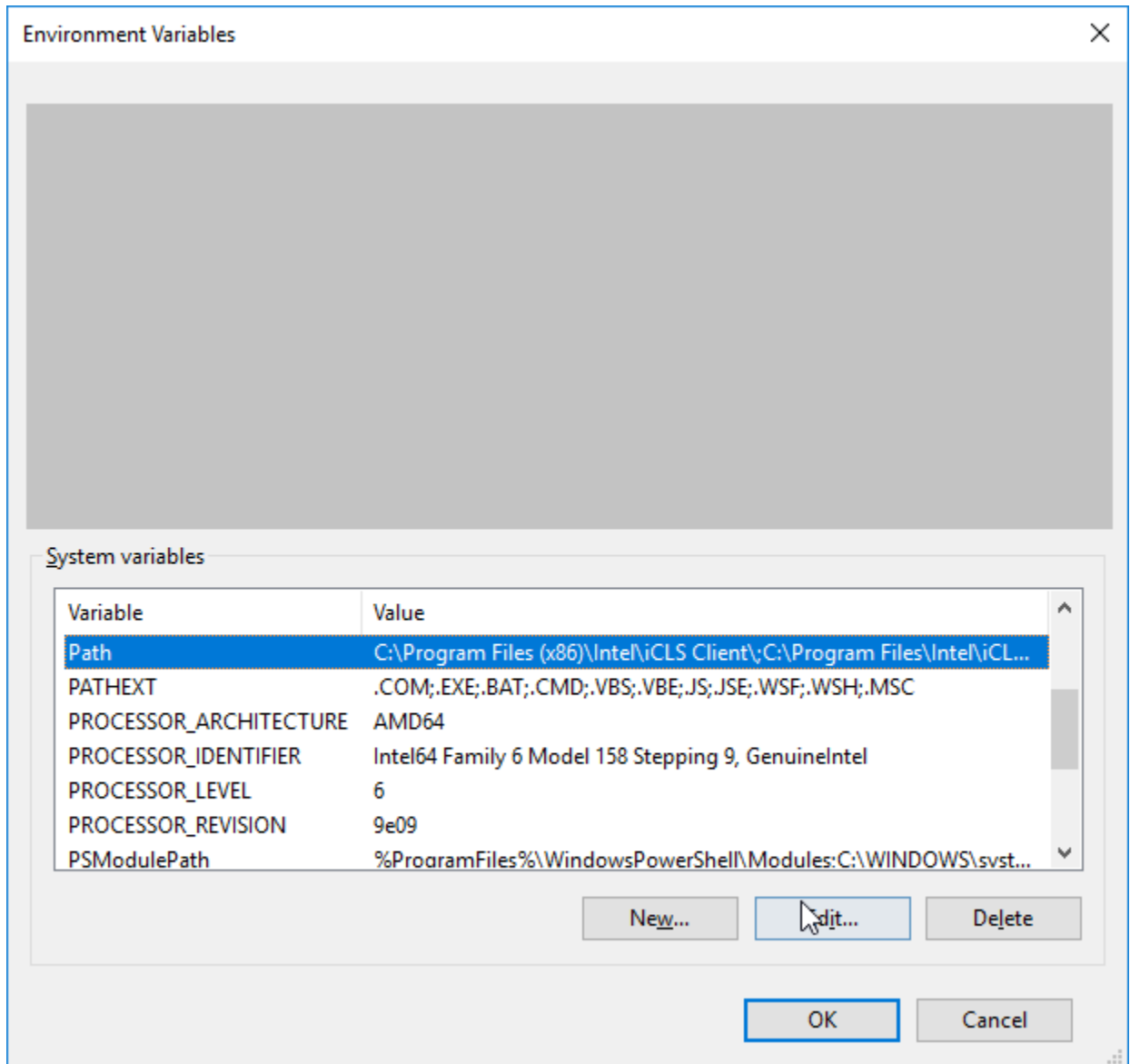


- 2- Click on the "Environment variables" button under the "Advanced" tab

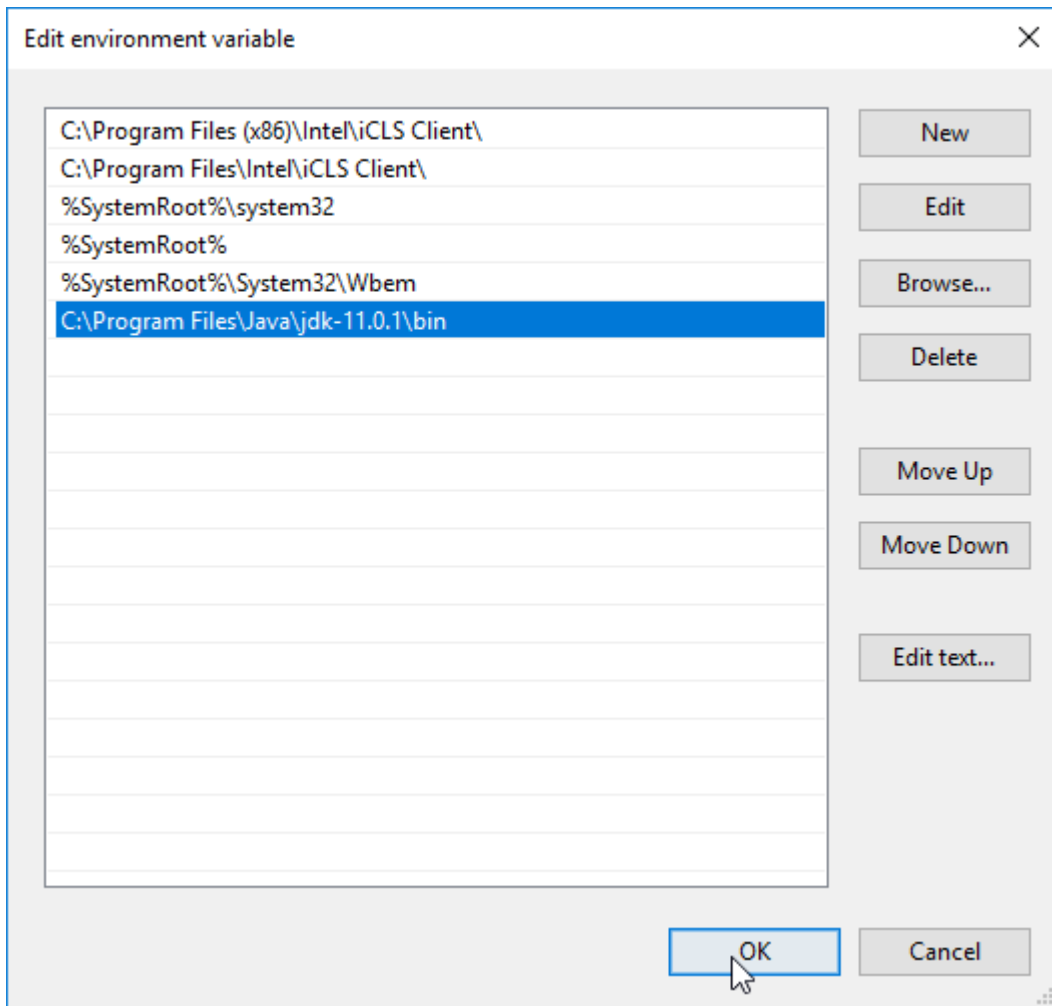




3- Then, select the "Path" variable in *System variables* and click on the "Edit" button



- 4- Click on the "New" button and add the path where Java is installed, followed by \bin. By default, Java is installed in C:\Program Files\Java\jdk-15\bin



- 5- Then, click "OK", and save the settings  
6- Write the following in the command line (cmd.exe):

```
C:\Users\Your Name>java -version
```

If Java was successfully installed, you will see something like this (depending on version):

```
java version "11.0.1" 2018-10-16 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.1+13-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.1+13-LTS, mixed mode)
```

### Activity 3:

#### Writing, Compiling and Executing a JAVA program

#### Solution:

1. Open any text editor (notepad, sublimeText or notepad++) of your choice and write following code in editor

```
//This program calculates the user's gross pay

import java.util.Scanner;    //to be able to read from the keyboard

public class Pay
{
    public static void main(String [] args)
    {
        //create a Scanner object to read from the keyboard
        Scanner keyboard = new Scanner(System.in);

        //identifier declarations
        double hours;        //number of hours worked
        double rate;         //hourly pay rate
        double pay;          //gross pay

        //display prompts and get input
        System.out.print("How many hours did you work? ");
        hours = keyboard.nextDouble();
        System.out.print("How much do you get paid per hour? ");
        rate = keyboard.nextDouble();

        //calculations
        if(hours <= 40)
            pay = hours * rate;
        else
            pay = (hours - 40) * (1.5 * rate) + 40 * rate;

        //display results
        System.out.println("You earned $" + pay);
    }
}
```

2. Save the file using same name as the class name i.e. *Payroll.java*
3. After saving the program, go to your operating system's command prompt and change your current directory or folder to the one that contains the Java program you just created. Then use the following command to compile the program.

```
javac Payroll.java
```

4. You should not receive any error messages.
5. **Execute the Program:** Now enter the following command to run the program.

```
java Payroll
```

6. When this program is executed, it will ask the user for input. You should calculate several

different cases by hand. Since there is a critical point at which the calculation changes, you should test three different cases: the critical point, a number above the critical point, and a number below the critical point. You want to calculate by hand so that you can check the logic of the program. Fill in the chart below with your test cases and the result you get when calculating by hand.

7. Execute the program using your first set of data. Record your result. You will need to execute the program three times to test all your data. Note: you do not need to compile again. Once the program compiles correctly once, it can be executed many times. You only need to compile again if you make changes to the code.

Hours	Rate	Pay (hand calculated)	Pay (program result)

## Activity 4:

### *Debugging a Java Program*

#### **Solution:**

- 1) Open any text editor (notepad, sublimeText or notepad++) of your choice and write following code in editor

```
//This program calculates the total price which includes sales //tax

import java.util.Scanner;

public class SalesTax
{
    public static void main(String[] args)
    {
        //identifier declarations
        final double TAX_RATE = 0.055;
        double price;
        double tax
        double total;
        String item;

        //create a Scanner object to read from the keyboard
        Scanner keyboard = new Scanner(System.in);
```

```

        //display prompts and get input
        System.out.print("Item description:  ");
        item = keyboard.nextLine();
        System.out.print("Item price:  $");
        price = keyboard.nextDouble();

        //calculations
        tax = price + TAX_RATE;
        total = price * tax;

        //display results
        System.out.print(item + "          $");
        System.out.println(price);
        System.out.print("Tax          $");
        System.out.println(tax);
        System.out.print("Total          $");
        System.out.println(total);
    }
}

```

- 2) Save the file as *SalesTax.java*.
- 3) This is a simple Java program that contains errors. Compile the program. You should get a listing of syntax errors. Correct all the syntax errors, you may want to recompile after you fix some of the errors.
- 4) When all syntax errors are corrected, the program should compile. As in the previous exercise, you need to develop some test data. Use the chart below to record your test data and results when calculated by hand.
- 5) Execute the program using your test data and recording the results. If the output of the program is different from what you calculated, this usually indicates a logic error. Examine the program and correct logic error. Compile the program and execute using the test data again. Repeat until all output matches what is expected.

Item	Price	Tax	Total (calculated)	Total (output)

### 3) Graded Lab Tasks

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

#### Lab Task 1

*Every student is required to make installation on his / her personal computer before next lab*

# **Lab 02**

## **Java Fundamentals-I**

### **Objective:**

This lab is designed to give you practice with some of the fundamentals concepts in Java.

### **Activity Outcomes:**

On completion of this lab student will be able

- Identify basic elements in java program (comments, reserve words, identifiers etc.)
- Use variables and constants in program development
- Communicate with the user by using the Scanner class
- Create a program from scratch by translating a pseudocode algorithm

### **Instructor Note:**

As a pre-lab activity, read Chapter 02 from the text book “Java How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.

## 1) Useful Concepts

To write meaningful programs, you must learn the programming language's special symbols, words, and syntax rules. The syntax rules tell you which statements (instructions) are legal, or accepted by the programming language, and which are not. You must also learn the semantic rules, which determine the meaning of the instructions. The programming language's rules, symbols, special words, and their meanings enable you to write programs to solve problems.

### Comments

**Single-line comments** begin with `//` and can be placed anywhere in the line. Multiline comments

```
System.out.println("7 + 8 = " + (7 + 8)); //prints: 7 + 8 = 15
```

**Multiple-line comments** are enclosed between `/*` and `*/`.

```
/*  
You can include comments that can  
occupy several lines.  
*/
```

### Special Symbols

The following are some of the special symbols:

```
+      -      *      /  
.      ;      ?      ,  
<=     !=     ==     >=
```

### Reserved Words (Keywords)

Reserved words are also called keywords used by programming language

`int, float, double, char, void, public, static, throws, return`

### Identifiers

Identifiers are names of things, such as variables, constants, and methods, that appear in programs. Some identifiers are predefined; others are defined by the user. A Java identifier consists of letters, digits, the underscore character (`_`), and the dollar sign (`$`) and must begin with a letter, underscore, or the dollar sign

#### User defined identifiers

```
First, conversion,  
payRate, counter1,  
$Amount
```

#### Pre-defined identifiers

```
print, println, and  
printf, nextInt,  
nextDouble, next, and  
nextLine
```

### Data Types

A set of values together with a set of operations on those values.

Data Type	Values	Storage (in bytes)
<code>char</code>	0 to 65535 ( $= 2^{16} - 1$ )	2 (16 bits)
<code>byte</code>	-128 ( $= -2^7$ ) to 127 ( $= 2^7 - 1$ )	1 (8 bits)
<code>short</code>	-32768 ( $= -2^{15}$ ) to 32767 ( $= 2^{15} - 1$ )	2 (16 bits)
<code>int</code>	-2147483648 ( $= -2^{31}$ ) to 2147483647 ( $= 2^{31} - 1$ )	4 (32 bits)
<code>long</code>	-9223372036854775808 ( $= -2^{63}$ ) to 9223372036854775807 ( $= 2^{63} - 1$ )	8 (64 bits)

**Variables:** A memory location whose content may change during program execution.

#### Syntax

```
dataType identifier1, identifier2, ..., identifierN;
```

**Named constant:** A memory location whose content is not allowed to change during program execution.

#### Syntax

```
static final dataType IDENTIFIER = value;
```

#### Putting Data into Variables

```
variable = expression;
```

### Declaring and Initializing Variables

In order to use variables in program, you need to declare and initialize it properly.

```
int first;
int second;
char ch;
double x;
first = 13;
second = 10;
ch = ' ';
x = 12.6;
```

You can also declare and initialize at the same line

```
int first = 13;
int second = 10;
char ch = ' ';
double x = 12.6;
double y = 123.456;
```

#### Declaring String variable

```
String name;
```



## Reading Data using the Scanner class

To put data into variables from the standard input device, Java provides the class `Scanner`. Using this class, we first create an input stream object and associate it with the standard input device. The following statement accomplishes this:

```
Scanner console = new Scanner(System.in);
```

To use the scanner class first you need to import following package

```
Syntax: import packageName.*;
```

```
import java.util.Scanner;
```

<code>nextByte()</code>	Reads an integer of the byte type.
<code>nextShort()</code>	Reads an integer of the short type
<code>nextInt()</code>	Reads an integer of the int type.
<code>nextLong()</code>	Reads an integer of the long type.
<code>nextFloat()</code>	Reads a number of the float type.
<code>nextDouble()</code>	Reads a number of the double type.
<code>next().charAt(0)</code>	Read a character
<code>next()</code>	Read a string
<code>nextLine()</code>	Read a line

## Displaying Output

In Java, output on the standard output device is accomplished by using the standard output object **`System.out`**. The object **`System.out`** has access to two methods, **`print`** and **`println`**, to output a string on the standard output device

### Syntax

```
System.out.print(expression);
```

```
System.out.println(expression);
```

```
System.out.println();
```

## Displaying Multiple items with the + operator

String concatenation operator appends one string to another

```
System.out.println("This is " + "one string.");
```

This statement will print:

```
This is one string
```

```
number = 5;
```

```
System.out.println("The value is " + number);
```

```
The value is 5
```

### Escape Sequences

	Escape Sequence	Description
<code>\n</code>	Newline	Cursor moves to the beginning of the next line
<code>\t</code>	Tab	Cursor moves to the next tab stop
<code>\b</code>	Backspace	Cursor moves one space to the left
<code>\r</code>	Return	Cursor moves to the beginning of the current line (not the next line)
<code>\\</code>	Backslash	Backslash is printed
<code>\'</code>	Single quotation	Single quotation mark is printed
<code>\"</code>	Double quotation	Double quotation mark is printed

## 2) Solved Lab Activities

<i>Sr.No</i>	<i>Allocated Time</i>	<i>Level of Complexity</i>	<i>CLO Mapping</i>
<i>Activity 1</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>
<i>Activity 2</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>
<i>Activity 3</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>
<i>Activity 4</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>
<i>Activity 5</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>

### Activity 1:

*This program illustrates how data in the variables are manipulated. Translate the instruction as given below*

### Solution:

```
public class Activity1{  
  
    public static void main(String [] args){  
  
        // Declare an variable num1 of type int  
  
        // Declare an variable num2 of type int  
  
        // Declare an variable sale of type double  
  
        // Declare an variable first of type char  
  
        // Assign 4 value to num1  
  
        // Display the value in num1 variable on output screen  
  
        // Store the result of 4 * 5 - 11 expression in num2 variable  
  
        // Display the value in num2 variable on output screen  
  
        // Store the result of 0.02 * 1000 expression in sale variable  
  
        // Display the value in sale variable on output screen  
  
        // Assign 'D' value to a variable first  
  
        // Display the value in first variable on output screen  
  
    }  
}
```

```

public class Activity1{

    public static void main(String [] args){

        int num1;

        int num2;

        double sale;

        char first;

        num1 = 4;

        System.out.println("num1 = " + num1);

        num2 = 4 * 5 - 11;

        System.out.println("num2 = " + num2);

        sale = 0.02 * 1000;

        System.out.println("sale = " + sale);

        first = 'D';

        System.out.println("first = " + first);

    }

}

```

## Output

```

num1 = 4
num2 = 9
sale = 20.0
first = D

```

## Activity 2:

*This program illustrates how input statements work.*

## Solution:

```

import java.util.Scanner;

public class Activity2{

    public static void main(String [] args){
        Scanner console = new Scanner(System.in);
        int feet;
        int inches;
    }
}

```

```

        System.out.print("Enter two integers separated by spaces.");
        feet = console.nextInt();
        inches = console.nextInt();
        System.out.print("feet = " + feet);
        System.out.print(" inches = " + inches);
    }
}

```

## Output

```

Enter two integers separated by spaces. 23 7
feet = 23 inches = 7

```

## Activity 3:

*This program illustrates how to read strings and numeric data*

## Solution:

```

import java.util.*;
public class Activity3{
    static Scanner console = new Scanner(System.in);
    public static void main(String [] args){
        String firstName;
        String lastName;
        int age;
        double weight;
        System.out.println("Enter first name, last name, " + "age, and
weight separated " + "by spaces.");
        firstName = console.next();
        lastName = console.next();
        age = console.nextInt();
        weight = console.nextDouble();
        System.out.print("Name: " + firstName+ " " + lastName);
        System.out.print(" Age: " + age);
        System.out.print(" Weight: " + weight);
    }
}

```

## Output

```

Enter first name, last name, age, and weight separated by spaces.
Sheila Mann 23 120.5
Name: Sheila Mann Age: 23 Weight: 120.5

```

#### Activity 4:

*This program demonstrates the close relationship between characters and integers*

#### Solution:

```
public class Activity4{
    public static void main(String[] args){
        char letter;
        letter = 65;
        System.out.println(letter);
        letter = 66;
        System.out.println(letter);
    }
}
```

#### Output

```
A
B
```

#### Activity 5:

*This program illustrate the concept of constants used in java*

#### Solution

```
import java.util.Scanner;
public class ComputeAreaWithConstant {
    public static void main(String[] args) {

        final double PI = 3.14159; // Declare a constant

        Scanner input = new Scanner(System.in);

        System.out.print("Enter a number for radius: ");
        double radius = input.nextDouble();

        double area = radius * radius * PI;
        System.out.println("The area for the circle of radius " +
radius + " is " + area);
    }
}
```

#### Output

```
Enter a number for radius: 1
The area for the circle of radius is 3.14159
```

### 3) Graded Lab Tasks

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

#### Lab Task 1

*Consider the following program segment*

```
//import classes
public class Activity1
{
    public static void main(String [] args)
    {
        //variable declaration
        //executable statements
    }
}
```

- Write Java statements that declare the following variables: **num1** , **num2** , and **num3** , and **average** of type **int** .*
- Write Java statements that store **125** into **num1** , **28** into **num2** , and **-25** into **num3** .*
- Write a Java statement that stores the average of **num1** , **num2** , and **num3** into **average**.*
- Write Java statements that output the values of **num1** , **num2** , **num3** , and **average** .*
- Compile and run your program*

#### Lab Task 2

*Consider the following Java program in which the statements are in the incorrect order. Rearrange and format the statements so that it prompts the user to input the length and width of a rectangle and output the area and perimeter of the rectangle.*

```
public class Activity2
{
    Scanner console = new Scanner(System.in);
    import java.util.*;
    {
        public static void main(String[] args)
        int width;
        System.out.print("Enter the length: ");
        width = console.nextInt();
        System.out.println();
        int length;
```

```

System.out.print("Enter the width: ");
length = console.nextInt();
System.out.println();
area = length * width;
System.out.println("Area = " + area);
System.out.println("Perimeter = " + perimeter);
perimeter = 2 * (length + width);
int area;
int perimeter;
}

```

### Lab Task 3

Consider the following program segment:

```

//import classes
public class LabTask3{
    public static void main(String [] args){
        //variable declaration
        //executable statements
    }
}

```

- a) Write a Java statement that imports the class Scanner.
- b) Write a Java statement that declares **console** to be a Scanner object for inputting data from the standard input device.
- c) Write Java statements that declare and initialize the following named constants: **SECRET** of type int initialized to 1; **RATE** of type double initialized to 12.50.
- d) Write Java statements that declare the following variables: num1 , num2 , and newNum of type int ; name of type String; hoursWorked and wages of type double.
- e) Write Java statements that prompt the user to input two integers and store the first number into num1 and the second number into num2.
- f) Write a Java statement(s) that outputs the value of num1 and num2 , indicating which is num1 and which is num2. For example, if num1 is 8 and num2 is 5 , then the output is:  

**The value of num1 = 8 and the value of num2 = 5.**
- g) Write a Java statement that multiplies that value of num1 by 2 , adds the value of num2 to it, and then stores the result in newNum . Then write a Java statement that outputs the value of newNum
- h) Write a Java statement that updates the value of newNum by adding the value of the named constant **SECRET** . Then, write a Java statement that outputs the value of newNum with an appropriate message.
- i) Write Java statements that prompt the user to enter a person's last name and then store the last name into the variable name.



- j) Write Java statements that prompt the user to enter a decimal number between 0 and 70 and then store the number entered into `hoursWorked`.
- k) Write a Java statement that multiplies that value of the named constant `RATE` with the value of `hoursWorked` and stores the result into the variable `wages`.
- l) Write Java statements that produce the following output:

```
Name: //output the value of the variable name
Pay Rate: $ //output the value of the named constant RATE
Hours Worked: //output the value of the variable hoursWorked
Salary: $ //output the value of the variable wages
```

For example, if the value of `name` is "Rainbow" and `hoursWorked` is 45.50, then the output is:

```
Name: Rainbow
Pay Rate: $12.50
Hours Worked: 45.50
Salary: $568.75
```

- m) Write a Java program that tests each of the Java statements in parts (a)—(l). Place the statements at the appropriate place in the preceding Java program segment. Test run your program (twice) on the following input data:
- `num1 = 13, num2 = 28; name = "Mustafa"; hoursWorked = 48.30`
  - `num1 = 32, num2 = 15; name = "Shakeel"; hoursWorked = 58.45`

# **Lab 03**

## **Java Fundamentals-II**

### **Objective:**

This lab is designed to give you practice with some of the fundamentals concepts in Java.

### **Activity Outcomes:**

On completion of this lab student will be able

- Demonstrate the concept of literals.
- Write expressions using various operators (arithmetic, increment/decrement).
- Demonstrate the type conversion used in program.
- Display formatted output on screen.

### **Instructor Note:**

As a pre-lab activity, read Chapter 02 from the text book “Java How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.

## 1) Useful Concepts

**Literals:** A literal is a constant value that appears directly in a program.

20	Integer literal
12L or 123L	Force integer literal to be treated as a long
"Today we sold "	String literal
12.3	Floating point literal
12.3f	Force a double literal to be treated as a float

### Scientific and E notation

Decimal Notation	Scientific Notation	E Notation
247.91	$2.4791 \cdot 10^2$	2.4791E2
0.00072	$7.2 \cdot 10^{-4}$	7.2E-4
2,900,000	$2.9 \cdot 10^6$	2.9E6

### Arithmetic Operators:

Operator	Meaning	Type	Example
+	Addition	Binary	<code>total = cost + tax;</code>
-	Subtraction	Binary	<code>cost = total - tax;</code>
*	Multiplication	Binary	<code>tax = cost * rate;</code>
/	Division	Binary	<code>salePrice = original / 2;</code>
%	Modulus	Binary	<code>remainder = value % 3;</code>

An **arithmetic expression** is constructed by using arithmetic operators and numbers.

### Precedence of arithmetic operators (highest to lowest)

Highest Precedence →	- (unary negation)
	* / %
Lowest Precedence →	+ -

### Associativity of arithmetic operators

Operator	Associativity
- (unary negation)	Right to left
* / %	Left to right
+ -	Left to right

### Combined assignment operators

Operator	Example Usage	Equivalent To
<code>+=</code>	<code>x += 5;</code>	<code>x = x + 5;</code>
<code>-=</code>	<code>y -= 2;</code>	<code>y = y - 2;</code>
<code>*=</code>	<code>z *= 10;</code>	<code>z = z * 10;</code>
<code>/=</code>	<code>a /= b;</code>	<code>a = a / b;</code>
<code>%=</code>	<code>c %= 3;</code>	<code>c = c % 3;</code>

## Increment and Decrement Operators

Operator	Name	Description	Example (assume i = 1)
<code>++var</code>	preincrement	Increment <b>var</b> by 1, and use the new <b>var</b> value in the statement	<code>int j = ++i;</code> <code>// j is 2, i is 2</code>
<code>var++</code>	postincrement	Increment <b>var</b> by 1, but use the original <b>var</b> value in the statement	<code>int j = i++;</code> <code>// j is 1, i is 2</code>
<code>--var</code>	predecrement	Decrement <b>var</b> by 1, and use the new <b>var</b> value in the statement	<code>int j = --i;</code> <code>// j is 0, i is 0</code>
<code>var--</code>	postdecrement	Decrement <b>var</b> by 1, and use the original <b>var</b> value in the statement	<code>int j = i--;</code> <code>// j is 1, i is 0</code>

## Conversion between primitive Data Types

Java performs some conversions between data types automatically, but does not automatically perform any conversion that can result in the loss of data

## Primitive data type ranking

double  
float  
long  
int  
short  
byte

Highest Rank  
↑  
Lowest Rank

**Cast operators:** The cast operator lets you manually convert a value

Statement	Description
<code>littleNum = (short)bigNum;</code>	The cast operator returns the value in <code>bigNum</code> , converted to a <code>short</code> . The converted value is assigned to the variable <code>littleNum</code> .
<code>x = (long)3.7;</code>	The cast operator is applied to the expression <code>3.7</code> . The operator returns the value <code>3</code> , which is assigned to the variable <code>x</code> .
<code>number = (int)72.567;</code>	The cast operator is applied to the expression <code>72.567</code> . The operator returns <code>72</code> , which is used to initialize the variable <code>number</code> .
<code>value = (float)x;</code>	The cast operator returns the value in <code>x</code> , converted to a <code>float</code> . The converted value is assigned to the variable <code>value</code> .
<code>value = (byte)number;</code>	The cast operator returns the value in <code>number</code> , converted to a <code>byte</code> . The converted value is assigned to the variable <code>value</code> .

## Formatting Console Output

You can use the `System.out.printf` method to display formatted output on the console.

<i>Format Specifier</i>	<i>Output</i>	<i>Example</i>
<b>%b</b>	a Boolean value	true or false
<b>%c</b>	a character	'a'
<b>%d</b>	a decimal integer	200
<b>%f</b>	a floating-point number	45.460000
<b>%e</b>	a number in standard scientific notation	4.556000e+01
<b>%s</b>	a string	"Java is cool"

<i>Example</i>	<i>Output</i>
<b>%5c</b>	Output the character and add four spaces before the character item, because the width is 5.
<b>%6b</b>	Output the Boolean value and add one space before the false value and two spaces before the true value.
<b>%5d</b>	Output the integer item with width at least 5. If the number of digits in the item is < 5, add spaces before the number. If the number of digits in the item is > 5, the width is automatically increased.
<b>%10.2f</b>	Output the floating-point item with width at least 10 including a decimal point and two digits after the point. Thus, there are 7 digits allocated before the decimal point. If the number of digits before the decimal point in the item is < 7, add spaces before the number. If the number of digits before the decimal point in the item is > 7, the width is automatically increased.
<b>%10.2e</b>	Output the floating-point item with width at least 10 including a decimal point, two digits after the point and the exponent part. If the displayed number in scientific notation has width less than 10, add spaces before the number.
<b>%12s</b>	Output the string with width at least 12 characters. If the string item has fewer than 12 characters, add spaces before the string. If the string item has more than 12 characters, the width is automatically increased.

## 2) Solved Lab Activities

<i>Sr.No</i>	<i>Allocated Time</i>	<i>Level of Complexity</i>	<i>CLO Mapping</i>
<i>Activity 1</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>
<i>Activity 2</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>
<i>Activity 3</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>
<i>Activity 4</i>	<i>15 mins</i>	<i>Low</i>	<i>CLO-5</i>
<i>Activity 5</i>	<i>15 mins</i>	<i>Medium</i>	<i>CLO-5</i>

### Activity-1:

*This program illustrate the uses E notation*

#### Solution:

```
public class Activity1{
    public static void main(String[] args){
        double distance, mass;
        distance = 1.495979E11;
        mass = 1.989E30;
        System.out.println("The sun is " + distance + " meters
away.");
        System.out.println("The sun's mass is " + mass + "
kilograms.");
    }
}
```

#### Output

```
The sun is 1.495979E11 meters away.
The sun's mass is 1.989E30 kilograms.
```

### Activity-2:

*This program calculates hourly wages plus overtime.*

#### Solution:

```
public class Activity2{
    public static void main(String[] args){
        double regularWages;
        double basePay = 25;
        double regularHours = 40;
        double overtimeWages;
        double overtimePay = 37.5;
        double overtimeHours = 10;
```

```

        double totalWages;
        regularWages = basePay * regularHours;
        overtimeWages = overtimePay * overtimeHours;
        totalWages = regularWages + overtimeWages;
        System.out.println("Wages for this week are $" + totalWages);
    }
}

```

## Output

```
Wages for this week are $1375.0
```

## Activity-3:

*This program calculates the amount of pay that will be contributed to a retirement plan if 5%, 8%, or 10% of monthly pay is withheld.*

## Solution:

```

public class Activity3{
    public static void main(String[] args){
        double monthlyPay = 6000.0;
        double contribution;
        // Calculate and display a 5% contribution.
        contribution = monthlyPay * 0.05;
        System.out.println("5 percent is $" + contribution + " per month.");
        // Calculate and display an 8% contribution.
        contribution = monthlyPay * 0.08;
        System.out.println("8 percent is $" + contribution + " per month.");
        // Calculate and display a 10% contribution.
        contribution = monthlyPay * 0.1;
        System.out.println("10 percent is $" + contribution + " per
month.");
    }
}

```

## Output

```

5 percent is $300.0 per month.
8 percent is $480.0 per month.
10 percent is $600.0 per month.

```

**Activity-4:**

*This program displays a variety of floating-point numbers in a column with their decimal points aligned.*

**Solution:**

```
public class Activity4{
    public static void main(String[] args){
        // Declare a variety of double variables.
        double num1 = 127.899;
        double num2 = 3465.148;
        double num3 = 3.776;
        double num4 = 264.821;
        double num5 = 88.081;
        double num6 = 1799.999;
        // Display each variable in a field of
        // 8 spaces with 2 decimal places.
        System.out.printf("%8.2f\n", num1);
        System.out.printf("%8.2f\n", num2);
        System.out.printf("%8.2f\n", num3);
        System.out.printf("%8.2f\n", num4);
        System.out.printf("%8.2f\n", num5);
        System.out.printf("%8.2f\n", num6);
    }
}
```

**Output**

```
127.90
3465.15
  3.78
264.82
 88.08
1800.00
```



**Activity-5:**

*This program displays the sales tax with two digits after the decimal point*

**Solution:**

```
import java.util.Scanner;

public class Activity5 {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        System.out.print("Enter purchase amount: ");

        double purchaseAmount = input.nextDouble();

        double tax = purchaseAmount * 0.06;

        System.out.println("Sales tax is $" + (int)(tax * 100) /
100.0);

    }

}
```

**Output**

```
Enter purchase amount: 197.55
Sales tax is $11.85
```

### 3) Graded Lab Tasks

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

#### Lab Task 1

*Suppose you want to develop a program that changes a given amount of money into smaller monetary units. The program lets the user enter an amount as a double value representing a total in dollars and cents, and outputs a report listing the monetary equivalent in the maximum number of dollars, quarters, dimes, nickels, and pennies, in this order, to result in the minimum number of coins*

*Here are the steps in developing the program:*

- 1. Prompt the user to enter the amount as a decimal number, such as 11.56.*
- 2. Convert the amount (e.g., 11.56) into cents (1156).*
- 3. Divide the cents by 100 to find the number of dollars. Obtain the remaining cents using the cents remainder 100.*
- 4. Divide the remaining cents by 25 to find the number of quarters. Obtain the remaining cents using the remaining cents remainder 25.*
- 5. Divide the remaining cents by 10 to find the number of dimes. Obtain the remaining cents using the remaining cents remainder 10.*
- 6. Divide the remaining cents by 5 to find the number of nickels. Obtain the remaining cents using the remaining cents remainder 5.*
- 7. The remaining cents are the pennies.*
- 8. Display the result.*

*You are required to implement the above steps 1-8 in JAVA language*

#### Lab Task 2

*N students take K apples and distribute them among each other evenly. The remaining (the undivisible) part remains in the basket. How many apples will each single student get? How many apples will remain in the basket?*

*The program reads the numbers N and K. It should print the two answers for the questions above.*

<b>Input:</b>	<b>Output:</b>
6	8
50	2

#### Lab Task 3

*A school decided to replace the desks in three classrooms. Each desk sits two students. Given the number of students in each class, print the smallest possible number of desks that can be purchased.*

*The program should read three integers: the number of students in each of the three classes, a, b and c respectively.*

<b>Input:</b>	<b>Output:</b>
17	28
19	
18	

#### **Lab Task 4**

*Given the integer  $N$  – the number of minutes that is passed since midnight - how many hours and minutes are displayed on the 24h digital clock?*

*The program should print two numbers: the number of hours (between 0 and 23) and the number of minutes (between 0 and 59).*

*For example, if  $N = 150$ , then 150 minutes have passed since midnight - i.e. now is **2:30 am**. So the program should print **2 30**.*

#### **Lab Task 5**

*A milk carton can hold 3.78 liters of milk. Each morning, a dairy farm ships cartons of milk to a local grocery store. The cost of producing one liter of milk is \$0.38, and the profit of each carton of milk is \$0.27. Write a program that does the following:*

- a. Prompts the user to enter the total amount of milk produced in the morning*
- b. Outputs the number of milk cartons needed to hold milk (Round your answer to the nearest integer.)*
- c. Outputs the cost of producing milk*
- d. Outputs the profit for producing milk*

#### **Lab Task 6**

*You found an exciting summer job for five weeks. It pays \$15.50 per hour. Suppose that the total tax you pay on your summer job income is 14%. After paying the taxes, you spend 10% of your net income to buy new clothes and other accessories for the next school year and 1% to buy school supplies. After buying clothes and school supplies, you use 25% of the remaining money to buy savings bonds. For each dollar you spend to buy savings bonds, your parents spend \$0.50 to buy additional savings bonds for you. Write a program that prompts the user to enter the pay rate for an hour and the number of hours you worked each week. The program then outputs the following:*

- a. Your income before and after taxes from your summer job*
- b. The money you spend on clothes and other accessories*
- c. The money you spend on school supplies*
- d. The money you spend to buy savings bonds*
- e. The money your parents spend to buy additional savings bonds for you*

#### **Lab Task 7**

*A cricket game is to be held in a stadium and there are four seating categories available for the audience. Class A seats cost \$20, Class B seats cost \$15, Class C seats cost \$10, and Class D seats cost \$5. You should write a JAVA program that asks how many tickets for each class of seats were sold and finally display the total income generated and income corresponding to ticket sales.*

#### **Lab Task 8**

*Write a program that reads an integer between 0 and 1000 and adds all the digits in the integer. For example, if an integer is 932, the sum of all its digits is 14.*

*Enter a number between 0 and 1000: 999*

*The sum of the digits is 27*

## Lab Task 9

Consider the statements:

```
double x = 75.3987;  
double y = 982.89764;
```

What is the output of the following statements?

```
System.out.printf("%.2f %n", x);  
System.out.printf("%.2f %n", y);  
System.out.printf("%.3f %n", x);  
System.out.printf("%.3f %n", y);
```

## Lab Task 10

Write JAVA statements using **System.out.printf()** statement to display output as given below

Degrees	Radians	Sine	Cosine	Tangent
30	0.5236	0.5000	0.8660	0.5773
60	1.0472	0.8660	0.5000	1.7320

```
public class DemoFormat {  
    public static void main(String[] args) {  
  
        // Display the header of the table using System.out.printf()  
  
        int degrees = 30;  
        double radians = Math.toRadians(degrees);  
        double sin = Math.sin(radians);  
        double cos = Math.cos(radians);  
        double tan = Math.tan(radians);  
  
        // Display the Data of the table using System.out.printf()  
  
        degrees = 60;  
        radians = Math.toRadians(degrees);  
        sin = Math.sin(radians);  
        cos = Math.cos(radians);  
        tan = Math.tan(radians);  
  
        // Display the Data of the table using System.out.printf()  
    }  
}
```

# **Lab 04**

## **Selection Control Structures**

This lab will give you practical implementation of different types of Selection Control Structures.

### **Activity Outcomes:**

On completion of this lab student will be able

- Construct boolean expressions to evaluate a given condition
- Construct if and if-else-if statements to perform a specific task
- Construct a switch statement

### **Instructor Note:**

As a pre-lab activity, read Chapter 04 from the text book “Java How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.

## 1) Useful Concepts

**boolean Data Type:** The boolean data type declares a variable with the value either **true** or **false**.

### Relational Operators

<i>Java Operator</i>	<i>Mathematics Symbol</i>	<i>Name</i>	<i>Example (radius is 5)</i>	<i>Result</i>
<	<	less than	<code>radius &lt; 0</code>	<code>false</code>
<=	≤	less than or equal to	<code>radius &lt;= 0</code>	<code>false</code>
>	>	greater than	<code>radius &gt; 0</code>	<code>true</code>
>=	≥	greater than or equal to	<code>radius &gt;= 0</code>	<code>true</code>
==	=	equal to	<code>radius == 0</code>	<code>false</code>
!=	≠	not equal to	<code>radius != 0</code>	<code>true</code>

### Logical (Boolean) Operators

Operator	Description
!	not
&&	and
	or

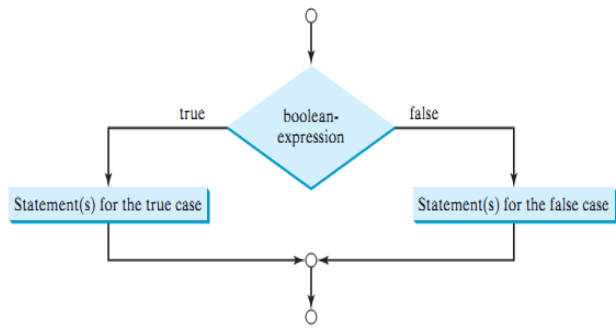
### if Statements

<b>Syntax</b>  <pre>if (boolean-expression) {     statement(s); }</pre>	<pre>graph TD; Entry(( )) --&gt; Decision{boolean-expression}; Decision -- true --&gt; Statement[Statement(s)]; Decision -- false --&gt; Join(( )); Statement --&gt; Join; Join --&gt; Exit(( ))</pre>
---	--

### if-else Statements

### Syntax

```
if (boolean-expression) {  
    statement(s)-for-the-true-case;  
}  
else {  
    statement(s)-for-the-false-  
case;  
}
```



### if-else-if Statement

#### Syntax

```
if (expression_1) {  
    statement  
    statement  
    etc. }  
else if (expression_2) {  
    statement  
    statement  
    etc. }
```

*If expression\_1 is true these statements are executed, and the rest of the structure is ignored.*

*Otherwise, if expression\_2 is true these statements are executed, and the rest of the structure is ignored.*

Insert as many else if clauses as necessary

```
else {  
    statement  
    statement  
    etc. }
```

*These statements are executed if none of the expressions above are true.*

### switch Statements

A switch statement executes statements based on the value of a variable or an expression.

```
switch (switch-expression) {  
    case value1: statement(s) 1;  
        break;  
    case value2: statement(s) 2;  
        break;
```

```
        ...
    case valueN:statement(s)N;
        break;
    default:statement(s)-for-
        default;
}
```

### **The Conditional operator**

You can use the conditional operator to create short expressions that work like if-else statements.

#### **Syntax**

**BooleanExpression ? Value1: Value2;**

#### **Example**

```
System.out.println("Your grade is: " + (score < 60 ? "Fail." : "Pass."));
```



## 2) Solved Lab Activites

<i>Sr.No</i>	<i>Allocated Time</i>	<i>Level of Complexity</i>	<i>CLO Mapping</i>
<i>Activity 1</i>	<i>15 mins</i>	<i>Midum</i>	<i>CLO-5</i>
<i>Activity 2</i>	<i>15 mins</i>	<i>Midum</i>	<i>CLO-5</i>
<i>Activity 3</i>	<i>15 mins</i>	<i>Midum</i>	<i>CLO-5</i>
<i>Activity 4</i>	<i>15 mins</i>	<i>Midum</i>	<i>CLO-5</i>
<i>Activity 5</i>	<i>15 mins</i>	<i>Medium</i>	<i>CLO-5</i>

### Activity-1:

*This program illustrate the uses of logical operator to calculate Leap Year*

#### Solution:

```
import java.util.Scanner;
public class Activity1 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a year: ");
        int year = input.nextInt();
        boolean isLeapYear = (year % 4 == 0 && year % 100 != 0) ||
        (year % 400 == 0);
        System.out.println(year + " is a leap year? " +
        isLeapYear);
    }
}
```

#### Output

```
Enter a year: 2008
2008 is a leap year? True
Enter a year: 1900
1900 is a leap year? False
```

### Activity-2:

*This program prompts the user to enter an integer. If the number is a multiple of 5, the program displays HiFive. If the number is divisible by 2, it displays HiEven*

#### Solution:

```
import java.util.Scanner;
public class Activity2 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
```

```

        System.out.println("Enter an integer: ");
        int number = input.nextInt();
        if (number % 5 == 0)
            System.out.println("HiFive");
        if (number % 2 == 0)
            System.out.println("HiEven");
    }
}

```

### Output

```

Enter an integer: 4
HiEven
Enter an integer: 30
HiFive
HiEven

```

### Activity-3:

*The following program demonstrate **if-else** concept. It determines an employee's weekly wages. If the hours worked exceed 40, then wages include overtime payment*

### Solution:

```

import java.util.*;
public class Activity3{
    static Scanner console = new Scanner(System.in);
    public static void main(String [] args){
        double wages, rate, hours; //Line 1
        System.out.print("Line 2: Enter the working "+ "hours: ");
        hours = console.nextDouble();
        System.out.println();
        System.out.print("Line 5: Enter the pay "+ "rate: ");
        rate = console.nextDouble();
        System.out.println();
        if (hours > 40.0)
            wages = 40.0 * rate +1.5 * rate * (hours - 40.0);
        else
            wages = hours * rate; //Line 11
        System.out.printf("Line 12: The wages are $%.2f %n",wages);
        System.out.println();
    }
}

```

### Output

```

Line 2: Enter working hours: 60
Line 5: Enter pay rate: 10
Line 12: The wages are $700

```

#### Activity-4:

*This program illustrate the usage of **else-if** concept. It calculates the grade based on the score entered by the user*

#### Solution:

```
import java.util.*;
public class Activity4{
    static Scanner console = new Scanner(System.in);
    public static void main(String [] args){
        int score;
        System.out.print("Enter score: ");
        score = console.nextInt();
        if (score >= 90)
            System.out.println("The grade is A");
        else if (score >= 80)
            System.out.println("The grade is B");
        else if (score >= 70)
            System.out.println("The grade is C");
        else if (score >= 60)
            System.out.println("The grade is D");
        else
            System.out.println("The grade is F");
    }
}
```

#### Output

```
Enter score: 80
The grade is B
```

#### Activity-5:

*This program demonstrates the working of switch statement.*

#### Solution:

```
import java.util.Scanner;
public class Activity5{
    public static void main(String[] args){
        int number;
        Scanner input = new Scanner(System.in);
        System.out.print("Enter 1, 2, or 3: ");
        number = input.nextInt();
        switch (number){
            case 1:
                System.out.println("You entered 1.");
                break;
```

```
        case 2:
            System.out.println("You entered 2.");
            break;
        case 3:
            System.out.println("You entered 3.");
            break;
        default:
            System.out.println("That's not 1, 2, or 3!");
    }
}
```

### Output

```
Enter 1, 2, or 3: 2 [enter]
You entered 2.
```

### 3) Graded Lab Tasks

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

#### Lab Task 1

Suppose that  $x$ ,  $y$ , and  $z$  are *int* variables and  $x = 10$ ,  $y = 15$ , and  $z = 20$ . Determine whether the following expressions evaluates to *true* or *false*.

```
!(x > 10)
x <= 5 || y < 15
(x != 5) && (y != z)
x >= z || (x + y >= z)
(x <= y - 2) && (y >= z) || (z - 2 != 20)
```

#### Lab Task 2

Suppose that  $x$ ,  $y$ ,  $z$ , and  $w$  are *int* variables and  $x = 3$ ,  $y = 4$ ,  $z = 7$ , and  $w = 1$ . What is the output of the following statements?

```
System.out.println("x == y: " + (x == y));
System.out.println("x != z: " + (x != z));
System.out.println("y == z - 3: " + (y == z - 3));
System.out.println("!(z > w): " + !(z > w));
System.out.println("x + y < z: " + (x + y < z));
```

#### Lab Task 3

Consider the following code segment. Determine the value of  $b3$

```
boolean b1=true;
boolean b2=false;
boolean b3=(b1==b2);
```

#### Lab Task 4

- Minimum of two numbers:** Given two integers, print the smaller value.
- Minimum of two numbers:** Given two integers, print the smaller value.
- Sign function:** For the given integer  $X$  print *1* if it's positive, *-1* if it's negative, or *0* if it's equal to zero.
- Minimum of three numbers:** Given three integers, print the smallest value.

#### Lab Task 5

**Equal numbers:** Given three integers, determine how many of them are equal to each other. The program must print one of these numbers: 3 (if all are the same), 2 (if two of them are equal to each other and the third is different) or 0 (if all numbers are different).

Sample Input: 10 5 10

Output: 2

### **Lab Task 6**

*Write a program that prompts the user to enter a number within the range of 1 through 10. The program should display the Roman numeral version of that number. If the number is outside the range of 1 through 10, the program should display an error message. The following table shows the Roman numerals for the numbers 1 through 10*

### **Lab Task 7**

*The area of a rectangle is the rectangle's length times its width. Write a program that asks for the length and width of two rectangles. The program should tell the user which rectangle has the greater area, or if the areas are the same.*

### **Lab Task 8**

*The date June 10, 1960, is special because when it is written in the following format, the month times the day equals the year: 6/10/60*

*Design a program that asks the user to enter a month (in numeric form), a day, and a two-digit year. The program should then determine whether the month times the day equals the year. If so, it should display a message saying the date is magic. Otherwise, it should display a message saying the date is not magic.*

### **Lab Task 9**

*Create a change-counting game that gets the user to enter the number of coins required to make exactly one dollar. The program should prompt the user to enter the number of pennies, nickels, dimes, and quarters. If the total value of the coins entered is equal to one dollar, the program should congratulate the user for winning the game. Otherwise, the program should display a message indicating whether the amount entered was more than or less than one dollar.*

### **Lab Task 10**

*Serendipity Booksellers has a book club that awards points to its customers based on the number of books purchased each month. The points are awarded as follows:*

- *If a customer purchases 0 books, he or she earns 0 points.*
- *If a customer purchases 1 book, he or she earns 5 points.*
- *If a customer purchases 2 books, he or she earns 15 points.*
- *If a customer purchases 3 books, he or she earns 30 points.*
- *If a customer purchases 4 or more books, he or she earns 60 points.*

*Write a program that asks the user to enter the number of books that he or she has purchased this month and displays the number of points awarded.*

# **Lab 05**

## **Repetition Control Structures**

### **Objective:**

This lab will give you practical implementation of different types of Repetition Control Structures.

### **Activity Outcomes:**

On completion of this lab student will be able

- Write a while loop and its variation (counter-controlled, sentinel-controlled and flag-controlled)
- Write a do-while loop
- Write a for loop
- Use break and continue statements in loops

### **Instructor Note:**

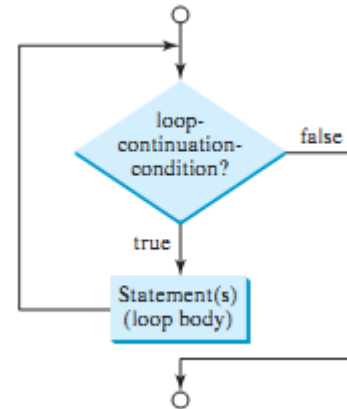
As a pre-lab activity, read Chapter 05 from the text book “Java How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.

## 1) Useful Concepts

**The while Loop:** A **while** loop executes statements repeatedly while the condition is true.

### Syntax

```
while(loop-continuation-condition){  
    // Loop body  
    Statement(s);  
}
```



### Counter-Controlled while Loops

Suppose you know exactly how many times certain statements need to be executed. For example, suppose you know exactly how many pieces of data (or entries) need to be read. In such cases, the while loop assumes the form of a counter-controlled while loop.

```
counter = 0; //initialize the loop control variable  
while (counter < N){ //test the loop control variable  
    ...  
    counter++; //update the loop control variable  
}
```

### Sentinel-Controlled while Loops

Another common technique for controlling a loop is to designate a special value when reading and processing a set of values. This special input value, known as a sentinel value, signifies the end of the input. A loop that uses a sentinel value to control its execution is called a *sentinel-controlled* loop

```
input the first data item into variable //initialize the loop control  
//variable  
while (variable != sentinel){ //test the loop control variable  
    ...  
    input a data item into variable //update the loop control  
    ... //variable  
}
```

### Flag-Controlled while Loops

A flag controlled **while** loop use s a **boolean** variable to control the loop. Suppose **found** is a boolean variable. The flag-controlled **while** loop takes the following form

```
found = false ; //initialize the loop control variable  
while (!found){ //test the loop control variable  
    ...
```



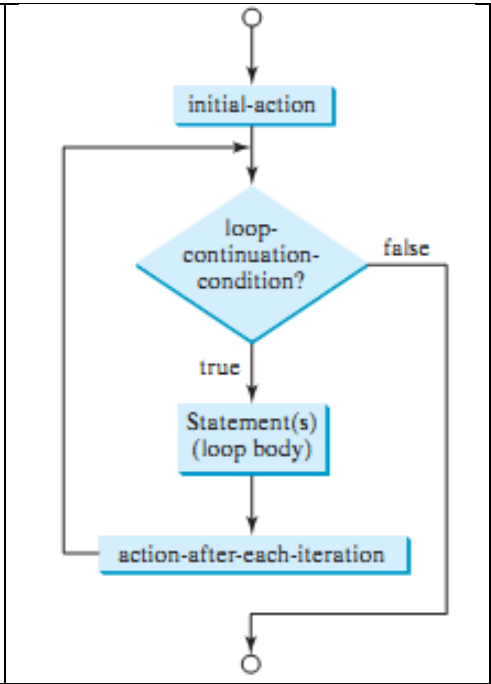
```
    if (logical expression)
        found = true ; //update the loop control variable
    . . .
}
```

**The for Loop**

A **for** loop has a concise syntax for writing loops.

**Syntax**

```
for (initial-action; loop-continuation-
condition; action-after-each-iteration)
{
    // Loop body;
    Statement(s);
}
```

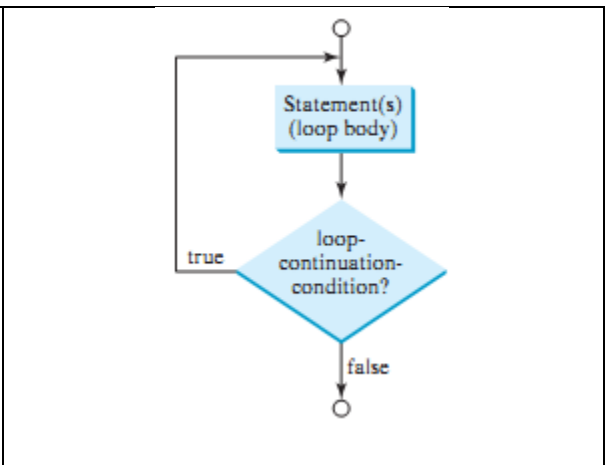


**The do-while Loop**

A **do-while** loop is the same as a while loop except that it executes the loop body first and then checks the loop continuation condition.

**Syntax**

```
do{
    // Loop body;
    Statement(s);
}while(loop-continuation-condition);
```



**Nested Loops**

A loop can be nested inside another loop. Nested loops consist of an outer loop and one or more inner loops. Each time the outer loop is repeated, the inner loops are reentered, and started a new.

### The **break** and **continue** Statements

The **break** statement causes a loop to terminate early. The **continue** statement causes a loop to stop its current iteration and begin the next one.

## 2) Solved Lab Activites

<i>Sr.No</i>	<i>Allocated Time</i>	<i>Level of Complexity</i>	<i>CLO Mapping</i>
<i>Activity 1</i>	<i>10 mins</i>	<i>Midum</i>	<i>CLO-5</i>
<i>Activity 2</i>	<i>10 mins</i>	<i>Midum</i>	<i>CLO-5</i>
<i>Activity 3</i>	<i>10 mins</i>	<i>Midum</i>	<i>CLO-5</i>
<i>Activity 4</i>	<i>10 mins</i>	<i>Midum</i>	<i>CLO-5</i>
<i>Activity 5</i>	<i>10 mins</i>	<i>Medium</i>	<i>CLO-5</i>
<i>Activity 6</i>	<i>10 mins</i>	<i>Medium</i>	<i>CLO-5</i>
<i>Activity 7</i>	<i>10 mins</i>	<i>Medium</i>	<i>CLO-5</i>
<i>Activity 8</i>	<i>10 mins</i>	<i>Medium</i>	<i>CLO-5</i>
<i>Activity 9</i>	<i>10 mins</i>	<i>Medium</i>	<i>CLO-5</i>

### Activity 1:

*This program illustrate the usage of simple while loop. It prompts the user to enter an answer for a question on addition of two single digits.*

### Solution:

```
import java.util.Scanner;
public class Activity1{
    public static void main(String[] args) {
        int number1 = (int) (Math.random() * 10);
        int number2 = (int) (Math.random() * 10);
        Scanner input = new Scanner(System.in);
        System.out.print("What is " + number1 + " + " + number2 +
"? ");
        int answer = input.nextInt();
        while (number1 + number2 != answer) {
            System.out.print("Wrong answer. Try again. What is "+
number1 + " + " + number2 + "? ");
            answer = input.nextInt();
        }
        System.out.println("You got it!");
    }
}
```

```
}  
}
```

## Output

```
What is 5 + 9? 12  
Wrong answer. Try again. What is 5 + 9? 34  
Wrong answer. Try again. What is 5 + 9? 14  
You got it!
```

## Activity-2:

*This program illustrate the working of counter-controlled while loop. Suppose the input is: 8 9 2 3 90 38 56 8 23 89 7 2 (limit is defined by user). Suppose you want to add these numbers and find their average.*

## Solution:

```
import java.util.*;  
public class Activity2{  
    public static void main(String [] args){  
        Scanner console = new Scanner(System.in);  
        int limit;  
        int number;  
        int sum;  
        int counter; //loop control variable  
        System.out.print("Enter the number of " + "integers in the  
list: ");  
        limit = console.nextInt();  
        System.out.println();  
        sum = 0;  
        counter = 0;  
        System.out.println("Enter " + limit+ " integers."); while  
(counter < limit){  
            number = console.nextInt();  
            sum = sum + number;  
            counter++;  
        }  
        System.out.printf("The sum of the %d "+"numbers= %d%n",  
limit, sum);  
        if (counter != 0)  
            System.out.printf("The average = %d%n", (sum /  
counter));  
        else  
            System.out.println("No input.");  
    }  
}
```

## Output

```
Enter the number of integers in the list: 12
Enter 12 integers.
8 9 2 3 90 38 56 8 23 89 7 2
The sum of the 12 numbers = 335
The average = 27
```

## Activity 3:

*This activity demonstrate the usage of sentinel-controlled while loop. Suppose you want to read some positive integers and average them, but you do not have a preset number of data items in mind. Suppose you choose the number -999 to mark the end of the data.*

## Solution:

```
import java.util.*;
public class Activity3{
    static Scanner input = new Scanner(System.in);
    static final int SENTINEL = -999;
    public static void main(String [] args){
        int number; //variable to store the number
        int sum = 0; //variable to store the sum
        int count = 0; //variable to store the total
        System.out.println("Enter positive integers "+ "ending with
" + SENTINEL);
        number = input.nextInt();
        while (number != SENTINEL)
        {
            sum = sum + number;
            count++;
            number = input.nextInt();
        }
        System.out.printf("The sum of %d " +"numbers = %d\n",
count, sum);
        if (count != 0)
            System.out.printf("The average = %d\n", (sum / count));
        else
            System.out.println("No input.");
    }
}
```

## Output

```
Enter positive integers ending with -999
34 23 9 45 78 0 77 8 3 5 -999
The sum of 10 numbers = 282
```

#### Activity 4:

*This activity demonstrate the usage of flag-controlled while loop The following program randomly generates an integer greater than or equal to 0 and less than 100 . The program then prompts the user to guess the number. If the user guesses the number correctly, the program outputs an appropriate message. Otherwise, the program checks whether the guessed number is less than the random number. If the guessed number is less than the random the number generated by the program, the program outputs the message, “Your guess is lower than the number”; otherwise, the program outputs the message, “Your guess is higher than the number”. The program then prompts the user to enter another number. The user is prompted to guess the random number until the user enters the correct number.*

#### Solution:

```
import java.util.*;
public class Activity4{
    static Scanner input = new Scanner(System.in);
    public static void main(String [] args){
        //declare the variables
        int num; //variable to store the random number
        int guess; //variable to store the number
        //guessed by the user
        boolean done; //boolean variable to control the loop
        num = ( int) (Math.random() * 100);
        done = false ;
        while (!done){
            System.out.print ("Enter an integer greater"+"than or
            equal to 0 and "+"less than 100: ");
            guess = input.nextInt();
            System.out.println();
            if (guess == num){
                System.out.println("You guessed the "+"correct
                number.");
                done = true;
            } //Line 12
            else if (guess < num)
                System.out.println("Your guess is " +"lower than"
                + "the number.\n" + "Guess again!");
            else //Line 15
                System.out.println("Your guess is "+"higher than"
                + "the number.\n"+"Guess again!");
        } //end while
    }
}
```

## Output

```
Enter an integer greater than or equal to 0 and less than 100: 25
Your guess is higher than the number.
Guess again!
Enter an integer greater than or equal to 0 and less than 100: 5
Your guess is lower than the number.
Guess again!
Enter an integer greater than or equal to 0 and less than 100: 10
Your guess is higher than the number.
Guess again!
Enter an integer greater than or equal to 0 and less than 100: 8
Your guess is higher than the number.
Guess again!
Enter an integer greater than or equal to 0 and less than 100: 6
Your guess is lower than the number.
Guess again!
Enter an integer greater than or equal to 0 and less than 100: 7
You guessed the correct number.
```

## Activity 5:

*This activity demonstrate the working of for loop. The following program calculates sum of five numbers entered by the user. Suppose the input is 2 3 4 5 0.*

### Solution:

```
import java.util.Scanner;
public class Activity5 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int number, sum = 0, count;
        for (count = 0; count < 5; count++) {
            number = input.nextInt();
            sum += number;
        }
        System.out.println("sum is " + sum);
    }
}
```

## Output

```
Sum is 14
```

### Activity 6:

*This activity demonstrate the working of do-while loop. Following program keep reading data until the input is 0. The program will display maximum of the numbers. Suppose the input is 2 3 4 5 0.*

#### Solution:

```
import java.util.Scanner;
public class Activity6 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int number, max;
        number = input.nextInt();
        max = number;
        do {
            number = input.nextInt();
            if (number > max)
                max = number;
        }while (number != 0);
        System.out.print("max is " + max+" and number " + number);
    }
}
```

#### Output

```
max is 5 and number 0
```

### Activity-7:

*This program uses nested for loops to display a multiplication table*

#### Solution:

```
public class Activity7{
    public static void main(String[] args){
        System.out.println("          Multiplication Table");
        // Display the number title
        System.out.print("          ");
        for (int j = 1; j <= 9; j++)
            System.out.print("    " + j);
        System.out.println("\n-----");
        // Display table body
        for (int i = 1; i <= 9; i++){
            System.out.print(i + " | ");
            for (int j = 1; j <= 9; j++) {
                // Display the product and align properly
                System.out.printf("%4d", i * j);
            }
            System.out.println();
        }
    }
}
```

```

    }
}
}

```

## Output

Multiplication Table										
	1	2	3	4	5	6	7	8	9	
1	1	2	3	4	5	6	7	8	9	
2	2	4	6	8	10	12	14	16	18	
3	3	6	9	12	15	18	21	24	27	
4	4	8	12	16	20	24	28	32	36	
5	5	10	15	20	25	30	35	40	45	
6	6	12	18	24	30	36	42	48	54	
7	7	14	21	28	35	42	49	56	63	
8	8	16	24	32	40	48	56	64	72	
9	9	18	27	36	45	54	63	72	81	

## Activity-8:

*This program demonstrate the effect of using **break** in a loop.*

## Solution:

```

public class Activity8{
    public static void main(String[] args) {
        int sum = 0;
        int number = 0;
        while (number < 20) {
            number++;
            sum += number;
            if (sum >= 100)
                break;
        }
        System.out.println("The number is " + number);
        System.out.println("The sum is " + sum);
    }
}

```

## Output

```

The number is 14
The sum is 105

```



### Activity 9:

*This program demonstrate the effect of using `continue` statement in a loop.*

### Solution:

```
public class Activity9{
    public static void main(String[] args) {
        int sum = 0;
        int number = 0;
        while (number < 20){
            number++;
            if (number ==10 || number == 11)
                continue;
            sum += number;
        }
        System.out.println("The sum is " + sum);
    }
}
```

### Output

```
The sum is 189
```

### 3) Graded Lab Tasks

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

#### Lab Task 1

- Given two integers  $A$  and  $B$  ( $A \leq B$ ). Print all numbers from  $A$  to  $B$  inclusively.
- Given two integers  $A$  and  $B$ . Print all numbers from  $A$  to  $B$  inclusively, in ascending order, if  $A < B$ , or in descending order, if  $A \geq B$ .
- Sum of  $N$  numbers:**  $N$  numbers are given in the input. Read them and print their sum. The first line of input contains the integer  $N$ , which is the number of integers to follow. Each of the next  $N$  lines contains one integer. Print the sum of these  $N$  integers.
- Sum of Cubes:** For the given integer  $N$  calculate the following sum:

$$1^3 + 2^3 + \dots + N^3$$

#### Lab Task 2

**Factorial:** In mathematics, the factorial of an integer  $n$ , denoted by  $n!$  is the following product:

$$n! = 1 \times 2 \times \dots \times n$$

For the given integer  $n$  calculate the value  $n!$

#### Lab Task 3

**Number of zeros:** Given  $N$  numbers: the first number in the input is  $N$ , after that  $N$  integers are given. Count the number of zeros among the given integers and print it. You need to count the number of numbers that are equal to zero, not the number of zero digits.

Input: 5 0 700 0 200 2	Output: 2
Input: 6 0 0 0 0 0 0	Output: 6

#### Lab Task 4

**The length of Sequence:** Given a sequence of non-negative integers, where each number is written in a separate line. Determine the length of the sequence, where the sequence ends when the integer is equal to 0. Print the length of the sequence (not counting the integer 0). The numbers following the number 0 should be omitted.

Input: 1 2 3 4 5 6 7 0 1 2 3	Output: 7
------------------------------	-----------

#### Lab Task 5

**The maximum of the Sequence:** A sequence consists of integer numbers and ends with the number 0. Determine the largest element of the sequence.

## Lab Task 6

**The index of the maximum of a sequence:** A sequence consists of integer numbers and ends with the number 0. Determine the index of the largest element of the sequence. If the highest element is not unique, print the index of the first of them.

Input: 1 2 3 2 1 0

Output: 3

## Lab Task 7

**The number of even elements of the sequence:** Determine the number of even elements in the sequence ending with the number 0.

## Lab Task 8

**The number of elements that are greater than the previous one:** A sequence consists of integer numbers and ends with the number 0. Determine how many elements of this sequence are greater than their neighbours above.

Input: 1 5 2 4 3 0

Output: 2

## Lab Task 9

Write a program to print following :

```
i) *****
*****
*****
*****
*****

ii) *
**
***
****
*****

iii)
**
***
****
*****

iv) *
***
*****
*****
*****
*****

v) 1
222
33333
4444444
555555555
```

The program asks the user to enter the number which pattern he/she wants to print. The loop should ask the user whether he or she wishes to perform the operation again. If so, the loop should repeat; otherwise it should terminate.

## Lab Task 10

Write a program that prompts the user to enter the year and first day of the year and displays the calendar table for the year on the console. For example, if the user entered the year 2013, and 2 for Tuesday, January 1, 2013, your program should display the calendar for each month in the year, as follows:

January 2013							December 2013						
Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5	1	2	3	4	5	6	7
6	7	8	9	10	11	12	8	9	10	11	12	13	14
13	14	15	16	17	18	19	15	16	17	18	19	20	21
20	21	22	23	24	25	26	22	23	24	25	26	27	28
27	28	29	30	31			29	30	31				

## Lab 06

### Mathematical, Characters and String Methods

#### Objective:

This lab will give you practical implementation of predefined mathematical, characters and String methods.

#### Activity Outcomes:

On completion of this lab student will be able

- Use pre-defined **Math** class methods
- Use pre-defined **Character** class methods
- Use pre-defined **String** class methods

#### Instructor Note:

As a pre-lab activity, read Chapter 06 and Chapter 16 from the text book “Java How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.

## 1) Useful Concepts

### Common Mathematical Functions

Java provides many useful methods in the Math class for performing common mathematical functions.

#### Trigonometric Methods

<i>Method</i>	<i>Description</i>
<code>sin(radians)</code>	Returns the trigonometric sine of an angle in radians.
<code>cos(radians)</code>	Returns the trigonometric cosine of an angle in radians.
<code>tan(radians)</code>	Returns the trigonometric tangent of an angle in radians.
<code>toRadians(degree)</code>	Returns the angle in radians for the angle in degree.
<code>toDegree(radians)</code>	Returns the angle in degrees for the angle in radians.
<code>asin(a)</code>	Returns the angle in radians for the inverse of sine.
<code>acos(a)</code>	Returns the angle in radians for the inverse of cosine.
<code>atan(a)</code>	Returns the angle in radians for the inverse of tangent.

#### Exponent Methods

<i>Method</i>	<i>Description</i>
<code>exp(x)</code>	Returns e raised to power of x ( $e^x$ ).
<code>log(x)</code>	Returns the natural logarithm of x ( $\ln(x) = \log_e(x)$ ).
<code>log10(x)</code>	Returns the base 10 logarithm of x ( $\log_{10}(x)$ ).
<code>pow(a, b)</code>	Returns a raised to the power of b ( $a^b$ ).
<code>sqrt(x)</code>	Returns the square root of x ( $\sqrt{x}$ ) for $x \geq 0$ .

#### The Rounding Methods

<i>Method</i>	<i>Description</i>
<code>ceil(x)</code>	x is rounded up to its nearest integer. This integer is returned as a double value.
<code>floor(x)</code>	x is rounded down to its nearest integer. This integer is returned as a double value.
<code>rint(x)</code>	x is rounded up to its nearest integer. If x is equally close to two integers, the even one is returned as a double value.
<code>round(x)</code>	Returns (int)Math.floor(x + 0.5) if x is a float and returns (long)Math.floor(x + 0.5) if x is a double.

#### The min, max, and abs Methods

The **min** and **max** methods return the minimum and maximum numbers of two numbers (**int**, **long**, **float**, or **double**). The **abs** method returns the absolute value of the number (**int**, **long**, **float**, or **double**).

#### The random Method

This method generates a random **double** value greater than or equal to 0.0 and less than 1.0 ( $0 \leq \text{Math.random()} < 1.0$ )

`(int) (Math.random() * 10)` Returns a random integer between 0 and 9.

`50 + (int) (Math.random() * 50)` Returns a random integer between 50 and 99.

General Form,

`a + Math.random() * b` Returns a random number between a and a + b, excluding a + b.

### Methods in the Character Class

<i>Method</i>	<i>Description</i>
<code>isDigit(ch)</code>	Returns true if the specified character is a digit.
<code>isLetter(ch)</code>	Returns true if the specified character is a letter.
<code>isLetterOfDigit(ch)</code>	Returns true if the specified character is a letter or digit.
<code>isLowerCase(ch)</code>	Returns true if the specified character is a lowercase letter.
<code>isUpperCase(ch)</code>	Returns true if the specified character is an uppercase letter.
<code>toLowerCase(ch)</code>	Returns the lowercase of the specified character.
<code>toUpperCase(ch)</code>	Returns the uppercase of the specified character.

### The String Type

A string is a sequence of characters. Class **String** is used to represent strings in Java.

### Declaring and Initializing String Variables

The String type is not a primitive type. It is known as a reference type

```
String message = "Welcome to Java";
```

Any Java class can be used as a reference type for a variable. The variable declared by a reference type is known as a reference variable that references an object. Here, **message** is a reference variable that references a string object with contents **Welcome to Java**

### Simple Methods for String Class

<i>Method</i>	<i>Description</i>
<code>length()</code>	Returns the number of characters in this string.
<code>charAt(index)</code>	Returns the character at the specified index from this string.
<code>concat(s1)</code>	Returns a new string that concatenates this string with string s1.
<code>toUpperCase()</code>	Returns a new string with all letters in uppercase.
<code>toLowerCase()</code>	Returns a new string with all letters in lowercase
<code>trim()</code>	Returns a new string with whitespace characters trimmed on both sides.

### Reading a String from the Console

To read a string from the console, invoke the **next()** method on a **Scanner** object. For example, the following code reads three strings from the keyboard

```
Scanner input = new Scanner(System.in);  
System.out.print("Enter three words separated by spaces: ");  
String s1 = input.next();
```

```
String s2 = input.next();
String s3 = input.next();
System.out.println("s1 is " + s1);
System.out.println("s2 is " + s2);
System.out.println("s3 is " + s3);
```

```
Enter three words separated by spaces: Welcome to Java
s1 is Welcome
s2 is to
s3 is Java
```

You can use the **nextLine()** method to read an entire line of text. The **nextLine()** method reads a string that ends with the Enter key pressed

```
Scanner input = new Scanner(System.in);
System.out.println("Enter a line: ");
String s = input.nextLine();
System.out.println("The line entered is " + s);
```

```
Enter a line: Welcome to Java
The line entered is Welcome to Java
```

## Comparing Strings

<i>Method</i>	<i>Description</i>
<code>equals(s1)</code>	Returns true if this string is equal to string <code>s1</code> .
<code>equalsIgnoreCase(s1)</code>	Returns true if this string is equal to string <code>s1</code> ; it is case insensitive.
<code>compareTo(s1)</code>	Returns an integer greater than 0, equal to 0, or less than 0 to indicate whether this string is greater than, equal to, or less than <code>s1</code> .
<code>compareToIgnoreCase(s1)</code>	Same as <code>compareTo</code> except that the comparison is case insensitive.
<code>startsWith(prefix)</code>	Returns true if this string starts with the specified prefix.
<code>endsWith(suffix)</code>	Returns true if this string ends with the specified suffix.
<code>contains(s1)</code>	Returns true if <code>s1</code> is a substring in this string.

## Obtaining Substrings

<i>Method</i>	<i>Description</i>
<code>substring(beginIndex)</code>	Returns this string's substring that begins with the character at the specified <code>beginIndex</code> and extends to the end of the string, as shown in Figure 4.2.
<code>substring(beginIndex, endIndex)</code>	Returns this string's substring that begins at the specified <code>beginIndex</code> and extends to the character at index <code>endIndex - 1</code> , as shown in Figure 4.2. Note that the character at <code>endIndex</code> is not part of the substring.

## Finding a Character or a Substring in a String

<i>Method</i>	<i>Description</i>
<code>indexOf(ch)</code>	Returns the index of the first occurrence of <code>ch</code> in the string. Returns -1 if not matched.
<code>indexOf(ch, fromIndex)</code>	Returns the index of the first occurrence of <code>ch</code> after <code>fromIndex</code> in the string. Returns -1 if not matched.
<code>indexOf(s)</code>	Returns the index of the first occurrence of string <code>s</code> in this string. Returns -1 if not matched.
<code>indexOf(s, fromIndex)</code>	Returns the index of the first occurrence of string <code>s</code> in this string after <code>fromIndex</code> . Returns -1 if not matched.
<code>lastIndexOf(ch)</code>	Returns the index of the last occurrence of <code>ch</code> in the string. Returns -1 if not matched.
<code>lastIndexOf(ch, fromIndex)</code>	Returns the index of the last occurrence of <code>ch</code> before <code>fromIndex</code> in this string. Returns -1 if not matched.
<code>lastIndexOf(s)</code>	Returns the index of the last occurrence of string <code>s</code> . Returns -1 if not matched.
<code>lastIndexOf(s, fromIndex)</code>	Returns the index of the last occurrence of string <code>s</code> before <code>fromIndex</code> . Returns -1 if not matched.

## 2) Solved Lab Activites

<i>Sr.No</i>	<i>Allocated Time</i>	<i>Level of Complexity</i>	<i>CLO Mapping</i>
<b>Activity 1</b>	<b>15 mins</b>	<b>Midum</b>	<b>CLO-5</b>
<b>Activity 2</b>	<b>15 mins</b>	<b>Midum</b>	<b>CLO-5</b>
<b>Activity 3</b>	<b>15 mins</b>	<b>Midum</b>	<b>CLO-5</b>
<b>Activity 4</b>	<b>15 mins</b>	<b>Midum</b>	<b>CLO-5</b>

### Activity-1:

*This activity demonstrate usage of Math class functions. Following program prompts the user to enter the x- and y-coordinates of the three corner points in a triangle and then displays the three angles.*

### Solution:

```
public class Activity1{
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // Prompt the user to enter three points
        System.out.print("Enter three points: ");
        double x1 = input.nextDouble();
        double y1 = input.nextDouble();
        double x2 = input.nextDouble();
        double y2 = input.nextDouble();
        double x3 = input.nextDouble();
        double y3 = input.nextDouble();
        // Compute three sides
        double a = Math.sqrt((x2 - x3) * (x2 - x3) + (y2 - y3) * (y2 - y3));
        double b = Math.sqrt((x1 - x3) * (x1 - x3) + (y1 - y3) * (y1 - y3));
        double c = Math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) *
```



```

        (y1 - y2));
        // Compute three angles
        double A = Math.toDegrees(Math.acos((a * a - b * b - c * c)
/ (-2 * b * c)));
        double B = Math.toDegrees(Math.acos((b * b - a * a - c * c)
/ (-2 * a * c)));
        double C = Math.toDegrees(Math.acos((c * c - b * b - a *
a) / (-2 * a * b)));
        // Display results
        System.out.println("The three angles are " + Math.round(A *
100) / 100.0 + " " + Math.round(B * 100) / 100.0 + " " +
Math.round(C * 100) / 100.0);
    }
}

```

## Output

```

Enter three points: 1 1 6.5 1 6.5 2.5
The three angles are 15.26 90.0 74.74

```

## Activity-2:

*This activity illustrate useful methods of Character class.*

## Solution:

```

public class Activity2{
    public static void main(String[] args) {
        System.out.println("isDigit('a') is " + Character.isDigit('a'));
        System.out.println("isLetter('a') is " + Character.isLetter('a'));
        System.out.println("isLowerCase('a') is "+
Character.isLowerCase('a'));
        System.out.println("isUpperCase('a') is "+
Character.isUpperCase('a'));
        System.out.println("toLowerCase('T') is "+
Character.toLowerCase('T'));
        System.out.println("toUpperCase('q') is "+
Character.toUpperCase('q'));
    }
}

```

## Output

```
isDigit('a') is false
isLetter('a') is true
isLowerCase('a') is true
isUpperCase('a') is false
toLowerCase('T') is t
toUpperCase('q') is Q
```

## Activity-3:

This activity uses compareTo() method to display two cities, entered by user, in alphabetical order.

### Solution:

```
import java.util.Scanner;
public class Activity3 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // Prompt the user to enter two cities
        System.out.print("Enter the first city: ");
        String city1 = input.nextLine();
        System.out.print("Enter the second city: ");
        String city2 = input.nextLine();
        if (city1.compareTo(city2) < 0)
            System.out.println("The cities in alphabetical order
            are " + city1 + " " + city2);
        else
            System.out.println("The cities in alphabetical order
            are " + city2 + " " + city1);
    }
}
```

## Output

```
Enter the first city: New York
Enter the second city: Boston
The cities in alphabetical order are Boston New York
```

## Activity-4:

*This program illustrates how various String methods work.*

### Solution:

```
public class Activity4{
    public static void main(String [] args){
        String sentence;
```

```

String str1;
String str2;
String str3;
int index;
sentence = "Now is the time for the birthday party";
System.out.println("sentence = \"" + sentence + "\"");
System.out.println("The length of sentence = " +
sentence.length());
    System.out.println("The character at index 16 in " +
"sentence = " + sentence.charAt(16));
    System.out.println("The index of first t in sentence = " +
sentence.indexOf(' t'));
    System.out.println("The index of for in sentence = " +
sentence.indexOf("for"));
    System.out.println("sentence.substring(0, 6) = \"" +
sentence.substring(0, 6) + "\"");
    System.out.println("sentence.substring(7, 12) = \"" +
sentence.substring(7, 12) + "\"");
    System.out.println("sentence.substring(7, 22) = \"" +
sentence.substring(7, 22) + "\"");
    System.out.println("sentence.substring(4, 10) = \"" +
sentence.substring(4, 10) + "\"");
    str1 = sentence.substring(0, 8);
    System.out.println("str1 = \"" + str1 + "\"");
    str2 = sentence.substring(2, 12);
    System.out.println("str2 = \"" + str2 + "\"");
    System.out.println("sentence in uppercase = \"" + sentence.toUpperCase() + "\"");
    index = sentence.indexOf("birthday");
    str1 = sentence.substring(index, index + 14);
    System.out.println("str1 = \"" + str1 + "\"");
    System.out.println("sentence.replace('t', 'T') = \"" +
sentence.replace('t', 'T') + "\"");

```

```
}  
}
```

## Output

```
sentence = "Now is the time for the birthday party"  
The length of sentence = 38  
The character at index 16 in sentence = f  
The index of first t in sentence = 7  
The index of for in sentence = 16  
sentence.substring(0, 6) = "Now is"  
sentence.substring(7, 12) = "the t"  
sentence.substring(7, 22) = "the time for th"  
sentence.substring(4, 10) = "is the"  
str1 = "Now is t"  
str2 = "w is the t"  
sentence in uppercase = "NOW IS THE TIME FOR THE BIRTHDAY PARTY"  
str1 = "birthday party"  
sentence.replace('t', 'T') = "Now is The Time for The birThday parTy"
```

### 3) Graded Lab Tasks

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

#### Lab Task 1

The great circle distance is the distance between two points on the surface of a sphere. Let  $(x_1, y_1)$  and  $(x_2, y_2)$  be the geographical latitude and longitude of two points. The great circle distance between the two points can be computed using the following formula:

$$d = radius \times \arccos(\sin(x_1) \times \sin(x_2) + \cos(x_1) \times \cos(x_2) \times \cos(y_1 - y_2))$$

Write a program that prompts the user to enter the latitude and longitude of two points on the earth in degrees and displays its great circle distance. The average earth radius is 6,371.01 km. Note that you need to convert the degrees into radians using the `Math.toRadians` method since the Java trigonometric methods use radians. The latitude and longitude degrees in the formula are for north and west. Use negative to indicate south and east degrees. Here is a sample run:

```
Enter point 1 (latitude and longitude) in degrees: 39.55, -116.25  
Enter point 2 (latitude and longitude) in degrees: 41.5, 87.37  
The distance between the two points is 10691.79183231593 km
```

#### Lab Task 2

- a) Write a program that receives an ASCII code (an integer between 0 and 127) and displays its character. Here is a sample run

```
Enter an ASCII code: 69
The character for ASCII code 69 is E
```

- b) Write a program that receives a character and displays its Unicode. Here is a sample run:

```
Enter a character: E
The Unicode for the character E is 69
```

### Lab Task 3

- a) Write a program that prompts the user to enter an integer between 0 and 15 and displays its corresponding hex number. Here are some sample runs:

```
Enter a decimal value (0 to 15): 11
The hex value is B
Enter a decimal value (0 to 15): 5
The hex value is 5
Enter a decimal value (0 to 15): 31
31 is an invalid input
```

- b) Write a program that prompts the user to enter a hex digit and displays its corresponding binary number. Here is a sample run

```
Enter a hex digit: B
The binary value is 1011
Enter a hex digit: G
G is an invalid input
```

### Lab Task 4

Write a program that displays a random uppercase letter using the **Math.random()** method.

### Lab Task 5

Write a program that checks whether a string is a palindrome. A string is a palindrome if it reads the same forward and backward. The words “mom,” “dad,” and “noon,” for instance, are all palindromes. Sample run:

```
Enter a string: noon
noon is a palindrome
Enter a string: moon
moon is not a palindrome
```

### Lab Task 6

Given a string consisting of exactly two words separated by a space. Print a new string with the first and second word positions swapped (the second word is printed first). **This task should not use loops and if.**

*Sample Run:*

```
Input: Hello, world!  
Correct Answer: world! Hello,
```

### Lab Task 7

*Given a string that may or may not contain a letter of interest. Print the index location of the first and last appearance of **f**. If the letter **f** occurs only once, then output its index. If the letter **f** does not occur, then do not print anything.*

```
Input: office  
Correct Answer: 1 2
```

### Lab Task 8

*Given a string in which the letter **h** occurs at least twice. Remove from that string the first and the last occurrence of the letter **h**, as well as all the characters between them.*

*Sample Run:*

```
Input: In the hole in the ground there lived a hobbit  
Correct Answer: In tobbit
```

### Lab Task 9

*Given a string. Replace every occurrence of the letter **h** by the letter **H**, except for the first and the last ones.*

```
Input: In the hole in the ground there lived a hobbit  
Correct Answer: In the Hole in tHe ground tHere lived a hobbit
```

### Lab Task 10

*You are given a string.*

*In the first line, print the third character of this string.*

*In the second line, print the second to last character of this string.*

*In the third line, print the first five characters of this string.*

*In the fourth line, print all but the last two characters of this string.*

*In the fifth line, print all the characters of this string with even indices (remember indexing starts at 0, so the characters are displayed starting with the first).*

*In the sixth line, print all the characters of this string with odd indices (i.e. starting with the second character in the string).*

*In the seventh line, print all the characters of the string in reverse order.*

*In the eighth line, print every second character of the string in reverse order, starting from the last one.*

*In the ninth line, print the length of the given string.*

**Sample Run:**

```
Input: Hello  
1
```

```
1
Hello
Hel
Hlo
el
olleH
olH
5
```

# **Lab 07**

## **Methods and Recursion**

### **Objective:**

This lab will give you practical implementation of predefined mathematical, characters and String methods.

### **Activity Outcomes:**

On completion of this lab student will be able

- Modularize a program by writing own methods
- Call user-defined methods in program
- Pass different types of arguments
- Develop program using recursion

### **Instructor Note:**

As a pre-lab activity, read Chapter 06 from the text book “Java How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.



## 1) Useful Concepts

It is usually a better approach to divide a large code in small methods. A method is a small piece of code used to perform a specific purpose. Methods are defined first then called whenever needed. A program may have as many methods as required. Similarly a method may be called as many times as required.

User-defined methods in Java are classified into two categories:

- **Value-returning methods**—methods that have a return data type. These methods return a value of a specific data type using the **return** statement.
- **Void methods**—methods that do not have a return data type. These methods do not use a **return** statement to return a value.

### Defining a Method

A method definition consists of its method name, parameters, return value type, and body.

#### Syntax

```
modifier returnValueType methodName(list of parameters){  
    // Method body;  
}
```

### Calling a Method

Calling a method executes the code in the method. Two ways to call a method.

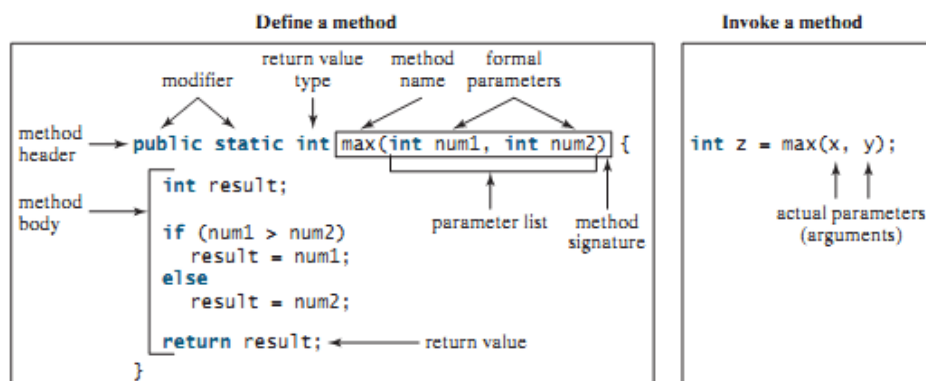
- a) **Value Returning Method** should be called as follows

```
int larger = max(3, 4); or System.out.println(max(3, 4));
```

- b) **Void methods** should be called as a statement

```
System.out.println("Welcome to Java");
```

### Defining and Invoking a Method



### Passing Arguments by Values

The arguments are passed by value to parameters when invoking a method. When calling a method, you need to provide arguments, which must be given in the same order as their respective parameters in the method signature. This is known as *parameter order association*.

When you invoke a method with an argument, the value of the argument is passed to the parameter. This is referred to as *pass-by-value*. If the argument is a variable rather than a literal value, the value of the variable is passed to the parameter. The variable is not affected, regardless of the changes made to the parameter inside the method

### Overloading Methods

Overloading methods enables you to define the methods with the same name as long as their signatures are different. The Java compiler determines which method to use based on the method signature.

### Recursion

Recursion is a technique that leads to elegant solutions to problems that are difficult to program using simple loops.

All recursive methods have the following characteristics:

- The method is implemented using an **if-else** or a **switch** statement that leads to different cases.
- One or more base cases (the simplest case) are used to stop recursion.
- Every recursive call reduces the original problem, bringing it increasingly closer to a base case until it becomes that case.

In general, to solve a problem using recursion, you break it into subproblems. Each sub-problem is the same as the original problem but smaller in size. You can apply the same approach to each subproblem to solve it recursively.

## 2) Solved Lab Activities

<i>Sr.No</i>	<i>Allocated Time</i>	<i>Level of Complexity</i>	<i>CLO Mapping</i>
<i>Activity 1</i>	<i>10 mins</i>	<i>Medium</i>	<i>CLO-5</i>
<i>Activity 2</i>	<i>10 mins</i>	<i>Medium</i>	<i>CLO-5</i>
<i>Activity 3</i>	<i>10 mins</i>	<i>Medium</i>	<i>CLO-5</i>
<i>Activity 4</i>	<i>10 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 5</i>	<i>10 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 6</i>	<i>10 mins</i>	<i>High</i>	<i>CLO-5</i>

### Activity 1:

*This activity demonstrate defining and calling a method. In this program a method `max()`, is defined which accepts two numbers as parameter(`int` type) and it returns a maximum value (`int` type).*

### Solution:

```
public class Activity1{
    /** Main method */
    public static void main(String[] args) {
        int i = 5;
        int j = 2;
        int k = max(i, j); //Method calling
        System.out.println("The maximum of " + i + " and " + j + "
        is " + k);
    }
    /** Return the max of two numbers - Method Definition*/
    public static int max(int num1, int num2) {
        int result;
        if (num1 > num2)
            result = num1;
        else
            result = num2;
        return result; // returning the value
    }
}
```

### Output

```
The maximum of 5 and 2 is 5
```

### Activity 2:

*This activity illustrate concept of void method. Following is a program that defines a method named `printGrade` and invokes it to print the grade for a given score.*

### Solution:

```
public class Activity2{
    public static void main(String[] args){
        System.out.print("The grade is ");
        printGrade(78.5);
        System.out.print("The grade is ");
        printGrade(59.5);
    }
    public static void printGrade(double score){
        if(score >= 90.0){
            System.out.println('A');
        }
    }
}
```

```

    }
    else if(score >= 80.0){
        System.out.println('B');
    }
    else if (score >= 70.0){
        System.out.println('C');
    }
    else if (score >= 60.0){
        System.out.println('D');
    }
    else{
        System.out.println('F');
    }
}
}

```

### Output

```

The grade is C
The grade is F

```

### Activity 3:

*This activity demonstrates the effect of passing by value. Following program creates a method for swapping two variables. The swap method is invoked by passing two arguments.*

### Solution:

```

public class Activity3 {
    /** Main method */
    public static void main(String[] args){
        // Declare and initialize variables
        int num1 = 1;
        int num2 = 2;
        System.out.println("Before invoking the swap method, num1
        is " + num1 + " and num2 is " + num2);
        // Invoke the swap method to attempt to swap two variables
        swap(num1, num2);
        System.out.println("After invoking the swap method, num1
        is" + num1 + " and num2 is " + num2);
    }

    /** Swap two variables */
    public static void swap(int n1, int n2) {
        System.out.println("\tInside the swap method");
        System.out.println("\t\tBefore swapping, n1 is " + n +
        " and n2 is " + n2);
        // Swap n1 with n2
    }
}

```

```

        int temp = n1;
        n1 = n2;
        n2 = temp;
        System.out.println("\t\tAfter swapping, n1 is " + n1+ " and
        n2 is " + n2);
    }
}

```

## Output

```

Before invoking the swap method, num1 is 1 and num2 is 2
Inside the swap method
    Before swapping, n1 is 1 and n2 is 2
    After swapping, n1 is 2 and n2 is 1
After invoking the swap method, num1 is 1 and num2 is 2

```

## Activity 4:

*This activity demonstrate the concept of method overloading. In the following program three methods are created. The first finds the maximum integer, the second finds the maximum double, and the third finds the maximum among three double values. All three methods are named **max**.*

## Solution:

```

public class Activity4 {
    /** Main method */
    public static void main(String[] args) {
        // Invoke the max method with int parameters
        System.out.println("The maximum of 3 and 4 is " + max(3, 4));
        // Invoke the max method with the double parameters
        System.out.println("The maximum of 3.0 and 5.4 is "+
        max(3.0,5.4));
        // Invoke the max method with three double parameters
        System.out.println("The maximum of 3.0, 5.4, and 10.14 is "
        + max(3.0, 5.4, 10.14));
    }
    /** Return the max of two int values */
    public static int max(int num1, int num2){
        if (num1 > num2)
            return num1;
        else
            return num2;
    }
    /** Find the max of two double values */
    public static double max(double num1, double num2){
        if (num1 > num2)
            return num1;
    }
}

```

```

        else
            return num2;
    }
    /** Return the max of three double values */
    public static double max(double num1, double num2, double num3){
        return max(max(num1, num2), num3);
    }
}

```

## Output

```

The maximum of 3 and 4 is 4
The maximum of 3.0 and 5.4 is 5.4
The maximum of 3.0, 5.4, and 10.14 is 10.14

```

## Activity 5:

The purpose of this activity is to show the working of recursion. Following is a complete program that prompts the user to enter a nonnegative integer and displays the factorial for the number

## Solution:

```

import java.util.Scanner;
public class ComputeFactorial{
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a nonnegative integer: ");
        int n = input.nextInt();
        System.out.println("Factorial of " + n + " is " +
            factorial(n));
    }
    public static long factorial(int n) {
        if (n == 0) // Base case
            return 1;
        else
            return n * factorial(n - 1); // Recursive call
    }
}

```

## Output

```

Enter a nonnegative integer: 4
Factorial of 4 is 24

```

## Activity 6:

*This activity gives a complete program that prompts the user to enter an index and computes the Fibonacci number for that index using the recursion*

### Solution:

```
import java.util.Scanner;
public class ComputeFibonacci {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter index for a Fibonacci number: ");
        int index = input.nextInt();
        // Find and display the Fibonacci number
        System.out.println("The Fibonacci number at index " + index
            + " is " + fib(index));
    }
    /** The method for finding the Fibonacci number */
    public static long fib(long index) {
        if (index == 0) // Base case
            return 0;
        else if (index == 1) // Base case
            return 1;
        else // Reduction and recursive calls
            return fib(index - 1) + fib(index - 2);
    }
}
```

### Output

```
Enter index for a Fibonacci number: 1
The Fibonacci number at index 1 is 1
```

```
Enter index for a Fibonacci number: 6
The Fibonacci number at index 6 is 8
```

```
Enter index for a Fibonacci number: 7
The Fibonacci number at index 7 is 13
```

## 3) Graded Lab Tasks

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

### Lab Task 1

- Write a method that computes the sum of the digits in an integer. Use the following method header: `public static int sumDigits(long n)`. For example, `sumDigits(234)` returns 9 (2 + 3 + 4).
- Write a method with the following header to display an integer in reverse order:

```
public static void reverse(int number)
```

For example, **reverse(3456)** displays **6543**. Write a test program that prompts the user to enter an integer and displays its reversal.

## Lab Task 2

Write the methods with the following headers

*// Return the reversal of an integer, i.e., reverse(456) returns 654*

```
public static int reverse(int number)
```

*// Return true if number is a palindrome*

```
public static boolean isPalindrome(int number)
```

Use the reverse method to implement **isPalindrome**. A number is a palindrome if its reversal is the same as itself. Write a test program that prompts the user to enter an integer and reports whether the integer is a palindrome.

## Lab Task 3

Write a method with the following header to display three numbers in increasing order:

```
public static void displaySortedNumbers(double num1, double num2,  
double num3)
```

Write a test program that prompts the user to enter three numbers and invokes the method to display them in increasing order.

## Lab Task 4

Write a method that returns the number of days in a year using the following header:

```
public static int numberOfDaysInAYear(int year)
```

Write a test program that displays the number of days in year from 2000 to 2020.

## Lab Task 5

Write a method that counts the number of letters in a string using the following header:

```
public static int countLetters(String s)
```

Write a test program that prompts the user to enter a string and displays the number of letters in the string.

## Lab Task 6

Write a function **capitalize(lower\_case\_word)** that takes the lower case word and returns the word with the first letter capitalized. Eg., **System.out.println(capitalize("word"))** should print the word **Word**. Then, given a line of lowercase ASCII words (text separated by a single space), print it with the first letter of each word capitalized using the your own function **capitalize()**.

## Lab task 7

Write a method that displays an n-by-n matrix using the following header:

```
public static void printMatrix(int n)
```



Each element is 0 or 1, which is generated randomly. Write a test program that prompts the user to enter  $n$  and displays an  $n$ -by- $n$  matrix. Here is a sample run:

```
Enter n: 3
0 1 0
0 0 0
1 1 1
```

### Lab Task 8

Write a Java method to count all vowels in a string. Here is sample run

```
Enter a string: Welcome to Java
Number of Vowels in the string: 6
```

### Lab Task 9

Given a positive real number  $a$  and a non-negative integer  $n$ . Calculate  $a^n$  without using loops, `**` operator or the built in function `math.pow()`. Instead, use recursion and the relation  $a^n = a \cdot a^{n-1}$ . Print the result. Form the method `power (a, n)`.

### Lab Task 10

Print following patterns using recursion.

```

      *
    * *
  * * *
* * * *
* * * * *

      *
    * *
  * * *
* * * *
* * * * *

      *
    * *
  * * *
* * * *
* * * * *

* * * * *
* * * *
* * *
* *
*
```

a) Given a sequence of integers. Print the sequence in reverse order using recursion.

```
Input: 1234567      Output: 7654321
```

b) Given an integer number. Convert it into its equivalent Binary number using recursion.

c) Implement Binary Search algorithm using the recursion.[To perform this task students must have taken the Arrays Lab]

# Lab 08

## One Dimensional Arrays

### Objective:

In this lab, students will know about the basic concepts of Array data structure, storing different types of data in arrays, creating arrays of integers, double, Strings and passing arrays to the methods.

### Activity Outcomes:

The activities provide hands - on practice with the following topics

- Creating arrays
- Accessing indexes/location of variables in arrays
- Accessing maximum, minimum numbers in arrays
- Different array's operation
- Passing arrays to methods

### Instructor Note:

As pre-lab activity, read Chapter 7 from the text book “Java How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.

## 1) Useful Concepts

Often you will have to store a large number of values during the execution of a program. Suppose, for instance, that you need to read 100 numbers, compute their average, and find out how many numbers are above the average. Your program first reads the numbers and computes their average, then compares each number with the average to determine whether it is above the average. In order to accomplish this task, the numbers must all be stored in variables. You have to declare 100 variables and repeatedly write almost identical code 100 times. Writing a program this way would be impractical. So, how do you solve this problem?

An efficient, organized approach is needed. Java and most other high-level languages provide a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. In the present case, you can store all 100 numbers into an array and access them through a single array variable.

Array is a data structure that represents a collection of the same types of data.

The array elements are accessed through the index. The array indices are 0-based, i.e., it starts from 0 to `arrayRefVar.length-1`.

1. **Declaring arrays:** An array can be declared as below.

```
elementType[] arrayRefVar;
```

The `elementType` can be any data type, and all elements in the array will have the same data type. For example, the following code declares a variable `myList` that references an array of double elements.

```
double[] myArray;
```

2. **Creating arrays:** Once array is declared, it can be created as below.

```
arrayRefVar = new elementType[arraySize];
```

In our case it can be as below.

```
myArray = new double[10];
```

3. We can also define arrays as below.

```
double[] myArray = new double[10];
```

## 2) Solved Lab Activities

<i>Sr.No</i>	<i>Allocated Time</i>	<i>Level of Complexity</i>	<i>CLO Mapping</i>
<i>Activity 1</i>	<i>15 mins</i>	<i>Medium</i>	<i>CLO-5</i>
<i>Activity 2</i>	<i>15 mins</i>	<i>Medium</i>	<i>CLO-5</i>
<i>Activity 3</i>	<i>15 mins</i>	<i>Medium</i>	<i>CLO-5</i>
<i>Activity 4</i>	<i>15 mins</i>	<i>Medium</i>	<i>CLO-5</i>

### Activity 1:

*Write a Java program to accept an array of 10 integer values from user and find the largest and second largest values.*

### Solution:

```
1 import java.util.Scanner;
2 public class ArrayExample
3 {
4     public static void main(String args[])
5     {
6         Scanner input=new Scanner(System.in);
7         int[] array=new int[10];
8         System.out.println("Enter array elements...")
9
10        for(int i=0;i<10;i++)
11        {
12            array[i]=input.nextInt();
13        }
14        int largest=array[0];
15        for(int i=0;i<10;i++)
16        {
17            if(array[i]>largest)
18                largest=array[i];
19        }
20        int largest2;
21        if(array[0]==largest)
22            largest2=array[1];
23        else
24            largest2=array[0];
25        for(int i=0;i<10;i++)
26        {
27            if(array[i]!=largest)
28                if(array[i]>largest2)
29                    largest2=array[i];
30        }
31        System.out.println("Largest="+largest);
32        System.out.println("2nd Largest="+largest2);
33    }
```

### Output

```
Enter array elements...
9
1
22
11
99
2
8
3
7
6
Largest=99
2nd Largest=22

Process completed.
```

## Activity 2:

*Write a Java program to accept 5 integer values from user. Pass this array to a method to find the sum of the array.*

## Solution:

```
1 //Program to find sum of the given integer array
2 import java.util.Scanner;
3 class SumArray
4 {
5     public static void main(String args[])
6     {
7         Scanner input=new Scanner(System.in);
8         int[] array=new int[5];
9         System.out.println("Enter array elements...");
10        for(int i=0;i<array.length;i++)
11        {
12            array[i]=input.nextInt();
13        }
14        System.out.println("Sum of the given array is... "+sum(array));
15    }
16    static int sum(int[] arr)
17    {
18        int s=0;
19        for(int x:arr)
20            s+=x;
21        return s;
22    }
23 }
```

**Output:**

```
Enter array elements...  
1  
2  
3  
4  
5  
Sum of the given array is... 15  
Process completed.
```

**Activity 3:**

*Write a program which takes 50 characters as input in array and counts the occurrences of each character.*

```
E.g.  A occurs 2 times  
      Y occurs 1 time  
      E occurs 1 time  
      O occurs 2 times  
      K occurs 1 time  
      @ occurs 1 time  
..... So on...
```

**Solution:**

```

1  import java.util.Scanner;
2
3  public class numberofoccurances {
4
5      public static void main(String[] args) {
6          char[] characters = new char[10];
7          int size = 0;
8          int element = characters[0];
9          char[] unique = new char[size];
10         int counter = 0;
11         Scanner input = new Scanner(System.in);
12
13         /////Input in array/////
14         for (int i = 0; i < 10; i++) {
15             System.out.print("A[" + i + "] = ");
16             characters[i] = input.next().charAt(0);
17         }
18         for (int i = 0; i < 10; i++) {
19             for (int j = 0; j < 10; j++) {
20
21                 if (characters[i] == characters[j]) {
22                     counter = counter + 1;
23                 }
24             }
25             System.out.println(characters[i] + " occurs " + counter + " times");
26             counter = 0;
27         }
28     }
29 }

```

## Output

```

run:
A[0] = 5
A[1] = 7
A[2] = 4
A[3] = 2
A[4] = 1
A[5] = 4
A[6] = 6
A[7] = 8
A[8] = 9
A[9] = 5
5 occurs 2 times
7 occurs 1 times
4 occurs 2 times
2 occurs 1 times
1 occurs 1 times
4 occurs 2 times
6 occurs 1 times
8 occurs 1 times
9 occurs 1 times
5 occurs 2 times
BUILD SUCCESSFUL (total time: 24 seconds)

```

#### Activity 4:

Create a menu driven program, with the following functionalities: (Note: all of the functionalities should be created in a single program with following menu options)

1. Input elements in array. (details of this functionality is given in Step a)
  2. Search element and its location. (details of this functionality is given in Step b)
  3. Find largest & smallest value in the array. (details of this functionality is given in Step c)
  4. Copy data. (details of this functionality is given in Step d)
- a) Input 10 elements in the array and display the array. (Note: this should call two methods `Input(int Array[ ] )` and `display(int A[ ] )` )
- b) Search element is in the array then print "Element found" along with its location. (Note: this should call two methods `Input(int Array[ ])` and `search(intsearchkey, int Array[ ])`. You should call the same `Input()` method that is called in step a )
- c) Find the largest and the smallest element in array. Then place largest element on 0<sup>th</sup> index and smallest element on the last index 9<sup>th</sup>. (Note: this should call three methods previously used `Input(int Array[])` , `Largest(int Array[])` and `Smallest (int Array[])`)
- d) Copy the contents of one array into another.(Note: this should call two methods `Input(int Array[])` and `copydata(int Array[], intcopiedArray[])` ).

#### Solution:

```
1
2 import java.util.Scanner;
3
4 public class ArraysImplementation {
5     // Main Method
6
7     public static void main(String[] args) {
8         // Menu Option place
9         Scanner input = new Scanner(System.in);
10        System.out.println(" Enter a number to choose menu : ");
11        System.out.println(" 1 : For input and display the Array \n 2 : For searching Elemnt in an Array ");
12        System.out.println(" 3 : For largest and Smallest element in the Array ");
13        System.out.println(" 4 : For copy data to another Array");
14
15        int a = input.nextInt();
16
17        // Condition checking
18        switch (a) {
19            // For input and Display
20            case 1: {
21                int[] a1 = new int[10];
22                int[] array = (Input(a1));
23                display(array);
24                break;
25            }
26            case 2: {
27                int[] a1 = new int[10];
28                int[] array1 = (Input(a1));
29                System.out.println(" Enter number for search");
30                int searchkey = input.nextInt();
31                search(array1, searchkey);
32                break;
33            }
34        }
35    }
36 }
```



```

35 // For Largest and Smallest in the Array
36     case 3: {
37         int[] a1 = new int[1];
38         int[] array11 = (Input(a1));
39         largest(array11);
40         smallest(array11);
41         break;
42     }
43 // For Copying to new Array
44     case 4: {
45         int[] a1 = {1};
46         int[] array1 = (Input(a1));
47         int[] copied = new int[10];
48         copydata(array1, copied);
49         break;
50     }
51     default:
52         System.out.println(" You Enter invalid number");
53 }
54 }
55 // Input array Method
56
57 public static int[] Input(int Array[]) {
58     int[] array1 = new int[10];
59     Scanner input = new Scanner(System.in);
60     for (int i = 0; i < 10; i++) {
61         System.out.println(" A[" + i + "] = ");
62         array1[i] = input.nextInt();
63     }
64     return array1;
65 }
66 // Display array Method
67
68 public static void display(int[] A) {
69     for (int j = 0; j < 10; j++) {
70         System.out.println(" A[" + j + "]" + A[j]);
71     }
72 }
73
74 // Searching element in Array Method
75 public static void search(int[] array, int x) {
76     int flag = 0;
77
78     while (flag == 0) {
79         for (int j = 0; j < 10; j++) {
80             if (array[j] == x) {
81                 System.out.println(" Element Found A[" + j + "]" + array[j]);
82                 flag++;
83             }
84         }
85         if (flag == 0) {
86             System.out.println(" Element is not found ");
87             break;
88         }
89     }
90 }
91

```

```

91
92 // Finding largest element in array Method
93 public static void largest(int[] array) {
94     int k = array[0];
95     for (int i = 0; i < 10; i++) {
96         if (array[i] >= k) {
97             k = array[i];
98         }
99     }
100     System.out.println("Largest value at A[0] " + k);
101 }
102 // Finding Smallest element in array Method
103
104 public static void smallest(int[] array) {
105     int k = array[9];
106     for (int i = 0; i < 10; i++) {
107         if (k >= array[i]) {
108             k = array[i];
109         }
110     }
111     System.out.println("Smallest value at A[9] " + k);
112 }
113 // Copying data to new array Mehtod
114
115 public static void copydata(int array[], int copied[]) {
116
117     for (int j = 0; j < 10; j++) {
118         copied[j] = array[j];
119     }
120     for (int k = 0; k < 10; k++) {
121         System.out.println(" New copied array is B[" + k + "]" + copied[k]);
122     }
123 }
124 }

```

## Output

```

run:
A[0] = 5
A[1] = 7
A[2] = 4
A[3] = 2
A[4] = 1
A[5] = 4
A[6] = 6
A[7] = 8
A[8] = 9
A[9] = 5
5 occurs 2 times
7 occurs 1 times
4 occurs 2 times
2 occurs 1 times
1 occurs 1 times
4 occurs 2 times
6 occurs 1 times
8 occurs 1 times
9 occurs 1 times
5 occurs 2 times
BUILD SUCCESSFUL (total time: 24 seconds)

```

### 3) Graded Lab Tasks

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

#### Lab Task 1

*10 students were asked to rate the quality of food in the student cafeteria, on a scale of 1 to 10 (1 means awful and 10 means excellent). Place the forty responses in an integer array and summarize the results of the poll.*

#### Lab Task 2

*Write a program which performs the following tasks:*

- *Initialize an integer array of 10 elements in main( )*
- *Pass the entire array to a function modify( )*
- *In modify( ) multiply each element of array by 3*
- *return the control to main( ) and print the new array elements in main( )*

#### Lab Task 3

*Write a program to copy the contents of one array into another in the reverse order.*

**Lab 09**  
**Mid-Term Exam**

# **Lab 10**

## **Two Dimensional(2D) Arrays**

### **Objective:**

The objective of this lab is to give hands-on experience of two dimensional arrays and its usage in developing the program.

### **Activity Outcomes:**

On the completion of this lab, students will be able to

- Create 2D arrays
- Process 2D arrays
- Pass 2D array as parameter to a method
- Apply different operations on 2D Array

### **Instructor Note:**

As pre-lab activity, read Chapter 7 from the text book “Java How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.

## 1) Useful Concepts

A multidimensional array is an array of arrays. Each element of a multidimensional array is an array itself. For example,

```
int[][] a = new int[3][4];
```

Here, we have created a multidimensional array named **a**. It is a 2-Dimensional array, that can hold a maximum of 12 elements,

Column 1

Column 2

Column 3

Column 4

Row 1

a[0][0]

a[0][1]

a[0][2]

a[0][3]

Row 2

a[1][0]

a[1][1]

a[1][2]

a[1][3]

Row 3

a[2][0]

a[2][1]

a[2][2]

a[2][3]

Initializing 2D Array

```
int[][] a = {  
    {1, 2, 3},  
    {4, 5, 6},  
    {7, 8, 9},  
};
```

The same can be written as below as well.

```
int[][] a = {{1,2,3}, {4,5,6}, {7,8,9} };
```

*Remember, Java uses zero-based indexing, that is, indexing of arrays in Java starts with 0 and not 1.*

## 2) Solved Lab Activites

Sr.No	Allocated Time	Level of Complexity	CLO Mapping
Activity 1	15 mins	High	CLO-5
Activity 2	15 mins	High	CLO-5
Activity 3	15 mins	High	CLO-5
Activity 4	15 mins	High	CLO-5

### Activity 1:

*Write a Java program to accept a 3X4 array from user and find sum of each row.*

**Solution:**

```

1 //Program to find sum of each row in a 2D array
2 import java.util.Scanner;
3 class SumArrayRows
4 {
5     public static void main(String args[])
6     {
7         Scanner input=new Scanner(System.in);
8         int[][] array=new int[3][4];
9         for(int i=0;i<3;i++)
10        {
11            System.out.println("Enter array elements of row # "+(i+1));
12            for(int j=0;j<4;j++)
13                array[i][j]=input.nextInt();
14        }
15        System.out.println("The given 2D array is...");
16        for(int i=0;i<3;i++)
17        {
18            for(int j=0;j<4;j++)
19                System.out.print(array[i][j]+"\\t");
20            System.out.println();
21        }
22        int sum;
23        for(int i=0;i<3;i++)
24        {
25            sum=0;
26            for(int j=0;j<4;j++)
27                sum+=array[i][j];
28            System.out.println("Sum of row "+(i+1)+" is "+sum);
29        }
30 }

```

## Output

```

Enter array elements of row # 1
1
2
3
4
Enter array elements of row # 2
11
22
33
44
Enter array elements of row # 3
1
3
5
7
The given 2D array is...
1   2   3   4
11  22  33  44
1   3   5   7
Sum of row 1 is 10
Sum of row 2 is 110
Sum of row 3 is 16

Process completed.

```

## Activity 2:

Write a Java program to initialize a 3X3 2D array with natural numbers starting from 11 and count the number of prime numbers in the given array.

➤ **NOTE:** You are required to send each array element to a user-defined method `isPrime` to know that whether the given number is prime or not.

## Solution:

```
1 //Program program to initialize a 3X3 2D array with natural numbers starting
2 //from 11 and count the number of prime numbers in the given array
3 class LabExample{
4     public static void main(String args[]){
5         Scanner input=new Scanner(System.in);
6         int[][] array={{11,12,13},{14,15,16},{17,18,19},{20,21,22}};
7         System.out.println("The given 2D array is...");
8         for(int i=0;i<3;i++){
9             for(int j=0;j<3;j++){
10                 System.out.print(array[i][j]+"\\t");
11                 System.out.println();
12             }
13             System.out.println("The prime numbers in 2D array are...");
14             int n=0;
15             for(int i=0;i<3;i++){
16                 for(int j=0;j<3;j++){
17                     if(isPrime(array[i][j])==true)
18                     {
19                         System.out.println(array[i][j]);
20                         n++;
21                     }
22                 }
23                 System.out.println("Total Prime Numbers =" +n);
24             }
25         }
26         static boolean isPrime(int n){
27             boolean prime=true;
28             for(int i=2;i<=n/2;i++){
29                 if(n%i==0){
30                     prime=false;
31                     break;
32                 }
33             }
34             return prime;
35         }
```

## Output

```
The given 2D array is...
11 12 13
14 15 16
17 18 19
The prime numbers in 2D array are..
11
13
17
19
Total Prime Numbers =4
Process completed.
```



### Activity 3:

Write a Java program to initialize a 3x4 integer array. Pass this array to a user-defined method transpose and return the transpose of the given matrix. Display the returned matrix in main method.

### Solution:

```
1 //Program to find tranpose of the matrix
2 class MatrixTransposeExample{
3     public static void main(String args[]){
4         int matrix1[][]={{1,2,3,4},{5,6,7,8},{9,10,11,12}}; //3 rows and 4 columns
5
6         //creating another matrix to store transpose of a matrix
7         int matrix2[][]=new int[4][3]; //4 rows and 3 columns
8
9         System.out.println("Original matrix is:");
10        for(int i=0;i<3;i++){
11            for(int j=0;j<4;j++){
12                System.out.print(matrix1[i][j]+"\\t");
13                System.out.println();
14            }
15            matrix2=transpose(matrix1);
16            System.out.println("Transpose of given matrix is:");
17            for(int i=0;i<4;i++){
18                for(int j=0;j<3;j++){
19                    System.out.print(matrix2[i][j]+"\\t");
20                    System.out.println();
21                }
22            }
23        static int[][] transpose(int[][] matrix){
24            int[][]trans=new int[4][3];
25            for(int i=0;i<4;i++){
26                for(int j=0;j<3;j++){
27                    trans[i][j]=matrix[j][i];
28                }
29            }
30            return trans;
31        }
```

### Output

```
Original matrix is:
1  2  3  4
5  6  7  8
9  10 11 12
Transpose of given matrix is:
1  5  9
2  6  10
3  7  11
4  8  12

Process completed.
```

#### Activity 4:

Write a Java program to accept a 4x4 integer array from your user and find sum of the diagonal.

#### Solution:

```
1 //Program to find sum of the diagonal
2 import java.util.Scanner;
3 class Diagonal{
4     public static void main(String args[]){
5         Scanner input=new Scanner(System.in);
6         int matrix[][]=new int[4][4];
7
8         for(int i=0;i<4;i++){
9             System.out.println("Enter array elements of row # "+(i+1));
10            for(int j=0;j<4;j++){
11                matrix[i][j]=input.nextInt();
12            }
13
14            System.out.println("Original matrix is:");
15            for(int i=0;i<4;i++){
16                for(int j=0;j<4;j++){
17                    System.out.print(matrix[i][j]+"\\t");
18                }
19                System.out.println();
20            }
21            int sum=0;
22            for(int i=0;i<4;i++){
23                for(int j=0;j<4;j++){
24                    if(i==j){
25                        sum+=matrix[i][j];
26                        break;
27                    }
28                }
29            }
30            System.out.println("Sum of diagonal is "+sum);
31        }
32    }
```

#### Output:

```

Enter array elements of row # 1
11
22
33
44
Enter array elements of row # 2
44
33
22
11
Enter array elements of row # 3
5
5
5
5
Enter array elements of row # 4
6
6
6
6
Original matrix is:
11 22 33 44
44 33 22 11
5 5 5 5
6 6 6 6
Sum of diagonal is 55

Process completed.

```

### 3) Graded Lab Tasks

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

#### Lab Task 1

Write a Java program to accept a 3x4 integer array from user. You are required to:

1. Find the row having maximum sum
2. Find the column having maximum sum

#### Lab Task 2

Write a Java program to accept a 3x4 integer array from user. You are required to find the row or column having maximum number of prime numbers.

#### Lab Task 3

Write a program to accept a 3x4 matrix from user having integer values. Accept another 4x3 matrix from user having integer values. You are required to multiply these two matrices and display the result.

# Lab 11

## Exception Handling

### **Objective:**

The purpose of this lab to give practical implementation of Exception Handling and its associated concepts.

### **Activity Outcomes:**

On the completion of this lab, students will be able to

- Demonstrate the concept of exception handling.
- Use try-throw-catch block, and purpose of finally clause in a program.
- Catch multiple exceptions in a program.

### **Instructor Note:**

As pre-lab activity, read Chapter 11 from the text book “Java How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.

## 1) Useful Concepts

An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e. at run time, that disrupts the normal flow of the program's instructions. Exception Handling in Java is one of the effective means to handle the runtime errors so that the regular flow of the application can be preserved. Java Exception Handling is a mechanism to handle runtime errors such as **ArrayIndexOutOfBoundsException**, **ClassNotFoundException**, **IOException** etc.

An exception can occur for many reasons e.g. invalid user input, device failure, code errors or opening an unavailable file etc.

Exceptions can be Categorized into two Ways:

1. **Built-in Exceptions:** Built-in exceptions are the exceptions that are available in Java libraries. There are two types of built-in exceptions.
  - **Checked Exception**  
Checked exceptions are called compile-time exceptions because these exceptions are checked at compile-time by the compiler.
  - **Unchecked Exception**  
The unchecked exceptions are just opposite to the checked exceptions. The compiler will not check these exceptions at compile time. In simple words, if a program throws an unchecked exception, and even if we didn't handle or declare it, the program would not give a compilation error.
2. **User-Defined Exceptions:** Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, users can also create exceptions which are called 'user-defined Exceptions'.

## 2) Solved Lab Activities

<i>Sr.No</i>	<i>Allocated Time</i>	<i>Level of Complexity</i>	<i>CLO Mapping</i>
<i>Activity 1</i>	<i>15 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 2</i>	<i>15 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 3</i>	<i>15 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 4</i>	<i>15 mins</i>	<i>High</i>	<i>CLO-5</i>

### Activity 1:

Write a Java program to keep accepting two integer values from user and find their sum, subtraction, multiplication and division depending on given choice. You need to handle the divide by zero exception such that if the second value is 0 then this exception must be handled.

### Solution:

```
1 import java.util.Scanner;
2 class ExceptionHandlingExample
3 {
4     public static void main(String args[])
5     {
6         Scanner input=new Scanner(System.in);
7         int a, b, choice;
8         do
9         {
10             System.out.print("Enter 1st value "); a=input.nextInt();
11             System.out.print("Enter 2nd value "); b=input.nextInt();
12             System.out.print("1:Add\n2:Subtract\n3:Multiply\n4:Divide\n5:Quit\nEnter your choice: ");
13             choice=input.nextInt();
14             try
15             {
16                 switch(choice)
17                 {
18                     case 1: System.out.println(a+"*"+b+"="+ (a*b));
19                             break;
20                     case 2: System.out.println(a+"-"+b+"="+ (a-b));
21                             break;
22                     case 3: System.out.println(a+"*"+b+"="+ (a*b));
23                             break;
24                     case 4: System.out.println(a+"/"+b+"="+ (a/b));
25                             break;
26                     case 5: break;
27                     default:
28                         System.out.println("You have entered an invalid choice");
29                 }
30             }
31             catch(ArithmeticException e)
32             {
33                 System.out.println("It is divide by ZERO exception");
34                 System.out.println("Try Again...");
35             }
36         }while(choice!=5);
37     }
38 }
39 }
```

### Output:

```

Enter 1st value 10
Enter 2nd value 2
1:Add
2:Subtract
3:Multiply
4:Divide
5:Quit
Enter your choice: 4
10/2=5
Enter 1st value 10
Enter 2nd value 0
1:Add
2:Subtract
3:Multiply
4:Divide
5:Quit
Enter your choice: 4
It is divide by ZERO exception ←
Try Again...
Enter 1st value 11
Enter 2nd value 2
1:Add
2:Subtract
3:Multiply
4:Divide
5:Quit
Enter your choice: 5

Process completed.

```

## Activity 2:

*Write a Java program to accept an integer value from user and find its factorial. You need to make sure that if the user enters an invalid value (other than integer value) then appropriate message should be displayed to handle this exception and keep accepting the input until valid integer is entered.*

## Solution:

```

1  //Program to handle InputMismatchException
2  import java.util.*;
3  class ExceptionHandlingExample{
4      public static void main(String args[]){
5          while(true){
6              try{
7                  Scanner input=new Scanner(System.in);
8                  System.out.print("Enter an integer value ");
9                  int n=input.nextInt();
10                 System.out.println("Factorial = "+factorial(n));
11                 break;
12             }
13             catch(InputMismatchException ex){
14                 System.out.println("You have entered an invalid input");
15                 System.out.println("Try Again");
16             }
17         }
18     }
19     static int factorial(int n){
20         int f=1;
21         for(int i=1;i<=n;i++){
22             f=f*i;
23         }
24         return f;
25     }

```

## Output

```
Enter an integer value ali
You have entered an invalid input
Try Again
Enter an integer value 1.2
You have entered an invalid input
Try Again
Enter an integer value 5
Factorial = 120

Process completed.
```

## Activity 3:

*This activity shows that how multiple exceptions can be handled by using a single try and multiple catch blocks.*

*In this activity the following exceptions are handled:*

- `InputMismatchException`
- `ArrayIndexOutOfBoundsException`

## Solution:

```
1 //Program to handle multiple exceptions
2 import java.util.*;
3 class ExceptionHandlingExample{
4     public static void main(String args[]){
5         Scanner input=new Scanner(System.in);
6         int array[]={11,5,8,3,21,7,66,33,31,90};
7         int choice=1;
8         while(choice==1){
9             try{
10                 System.out.print("Enter an integer value to search in array: ");
11                 int n=input.nextInt();
12                 System.out.print("Enter the index of the array: ");
13                 int index=input.nextInt();
14                 if(array[index]==n)
15                     System.out.println("You Won");
16                 else
17                     System.out.println("You Lost");
18                 System.out.println("Do you want to continue... (Press 1 for YES and 0 for NO)");
19                 choice=input.nextInt();
20                 if(choice==0)
21                     break;
22             }
23             catch(InputMismatchException ex){
24                 System.out.println("You have entered an invalid input type");
25                 System.out.println("Try Again");
26                 input.nextLine();
27             }
28             catch(ArrayIndexOutOfBoundsException ex2){
29                 System.out.println("You have entered an invalid index");
30                 System.out.println("Try Again");
31                 input.nextLine();
32             }
33         }
34     }
35 }
36 }
```



## Output

```
Enter an integer value to search in array: 21
Enter the index of the array: 4
You Won
Do you want to continue... (Press 1 for YES and 0 for NO)
1
Enter an integer value to search in array: abc
You have entered an invalid input type ←
Try Again
Enter an integer value to search in array: 11
Enter the index of the array: 22
You have entered an invalid index ←
Try Again
Enter an integer value to search in array: 21
Enter the index of the array: 2
You Lost
Do you want to continue... (Press 1 for YES and 0 for NO)
0
Process completed.
```

## Activity 4:

*The following code shows that how nested try statements can be used.*

## Solution:

```
1 class Example4{
2     public static void main(String[] args) {
3         try {
4             int arr[]={5,0,1,2};
5             try {
6                 int x=arr[3]/arr[1];
7             }
8             catch(ArithmeticException ae){
9                 System.out.println("divide by zero");
10            }
11            arr[4]=3;
12        }
13        catch(ArrayIndexOutOfBoundsException e) {
14            System.out.println("array index out of bound exception");
15        }
16    }
17 }
```

## Output:

```
divide by zero
array index out of bound exception
Process completed.
```

### 3) Graded Lab Tasks

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

#### **Lab Task 1**

*Write a Java program to accept two integer values from user and display their sum. If the user enters an invalid input then display a message “You have entered an invalid input, type integers only”. User should be asked to enter the input again. Keep accepting the values until valid inputs are entered.*

#### **Lab Task 2**

*Write a Java program to accept an array of 10 integer values from user. If the user enters a wrong input then ask him to enter the input again. Once all 10 integer values are entered then accept an index and display the value at that index. If the user has entered a wrong index then your program should catch this exception.*

# Lab 12

## Code Refactoring and using Library Components

### Objective:

The objective of this lab is to provide hands-on activities related to the code refactoring and of library components in Java using NetBeans.

### Activity Outcomes:

Upon the completion of activities, students will be able to

- Apply code refactoring in NetBeans.
- Debug the code in NetBeans.
- Use Library Components and their APIs in developing a program.

### Instructor Note:

As pre-lab activity, read the text given in following links.

1. <https://www.altexsoft.com/blog/engineering/code-refactoring-best-practices-when-and-when-not-to-do-it/>
2. <https://www.c-sharpcorner.com/UploadFile/0b8ab6/refactoring-in-netbeans-ide/>

## 1) Useful Concepts

Code refactoring is a process that involves editing and cleaning up previously written software code without changing the function of the code at all. The basic purpose of code refactoring is to make the code more efficient and maintainable. This is key in reducing technical cost since it's much better to clean up the code now than pay for costly errors later. Code refactoring, which improves readability, makes the QA and debugging process go much more smoothly. And while it doesn't remove bugs, it can certainly help prevent them in the future.

Debugging is the process of analyzing how your program runs, how it generates data in order to find defects and issues in your code. These errors or defects are referred to as "bugs", hence the term "debugging. A debugger is a tool that is typically used to allow the user to view the execution state and data hold by variables as the application is running.

The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects. Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion. Java Collection means a single unit of objects. Java Collection framework provides many interfaces (**Set**, **List**, **Queue**, **Deque**) and classes (**ArrayList**).

## 2) Solved Lab Activites

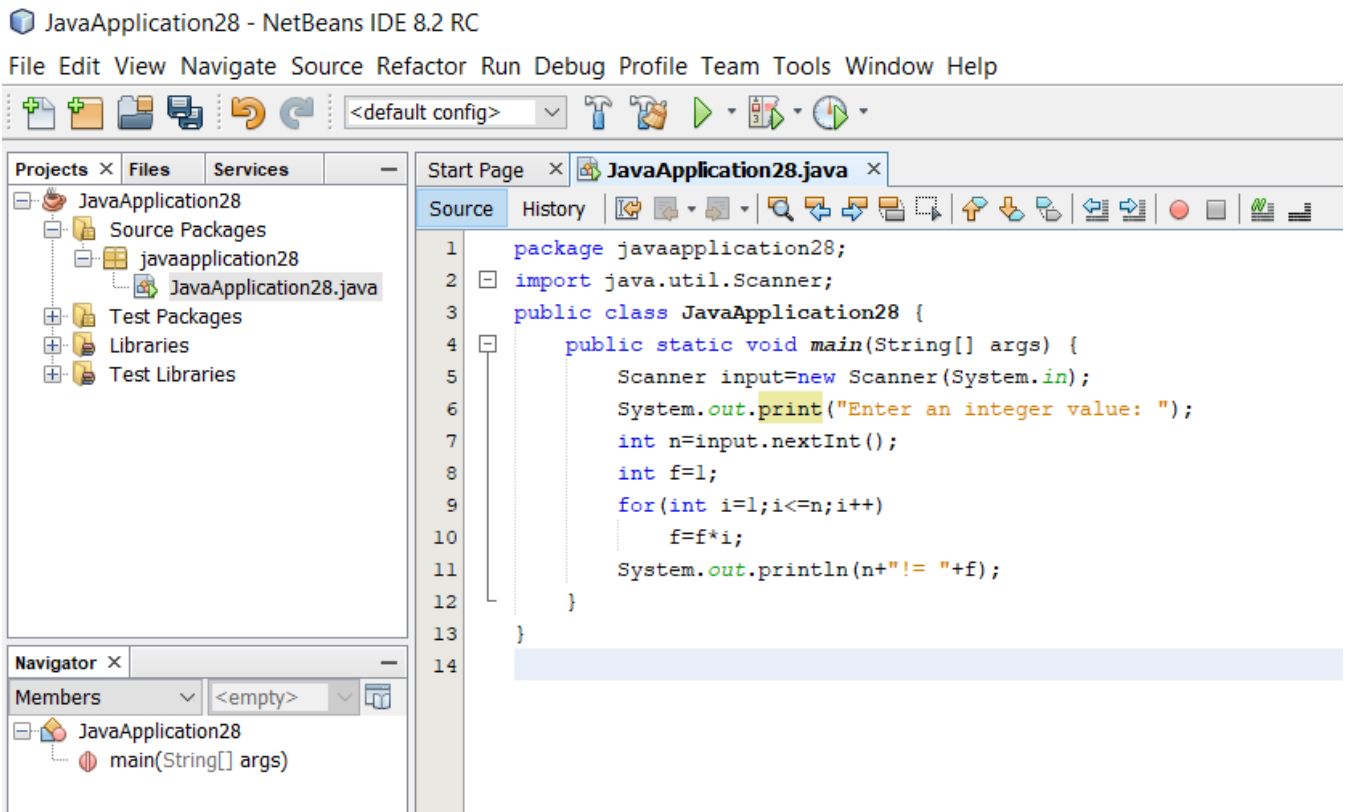
<i>Sr.No</i>	<i>Allocated Time</i>	<i>Level of Complexity</i>	<i>CLO Mapping</i>
<i>Activity 1</i>	<i>20 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 2</i>	<i>20 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 3</i>	<i>20 mins</i>	<i>High</i>	<i>CLO-5</i>

### Activity 1:

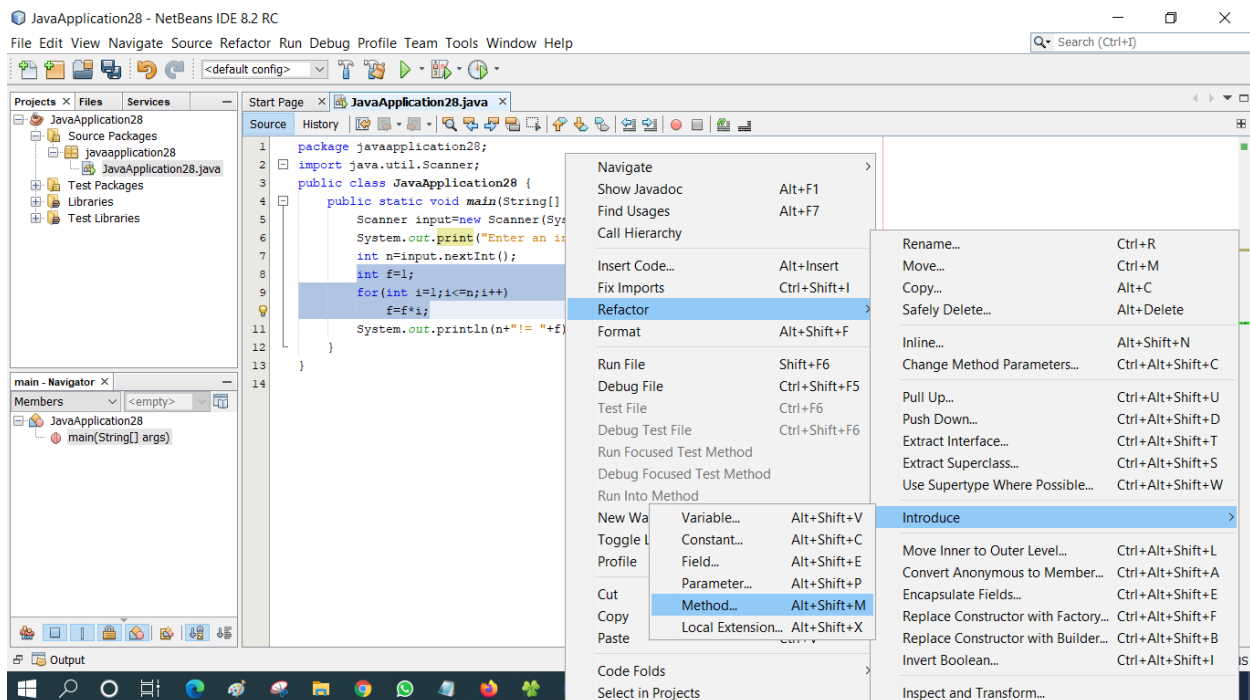
*This activity shows that how to use NetBeans for code refactoring. In this activity a code is converted/refactored into a method using the built-in feature of NetBeans.*

### Solution:

**Step1:** Write the code as shown below in NetBeans.



**Step2:** Select line 8 to 10 to refactor this segment into a method → Right Click → Select 'Refactor' → Select 'Introduce' → Select 'Method' → Select 'public' → Type 'factorial' as method name → Click 'Ok'



After performing the above steps, the code will look like below.

```

1  package javaapplication28;
2  import java.util.Scanner;
3  public class JavaApplication28 {
4      public static void main(String[] args) {
5          Scanner input=new Scanner(System.in);
6          System.out.print("Enter an integer value: ");
7          int n=input.nextInt();
8          int f=factorial(n);
9          System.out.println(n+"!="+"f);
10     }
11
12     public static int factorial(int n) {
13         int f=1;
14         for(int i=1;i<=n;i++)
15             f=f*i;
16         return f;
17     }
18 }
19

```

## Output:

JavaApplication28 - NetBeans IDE 8.2 RC

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

The screenshot displays the NetBeans IDE interface. The top toolbar includes icons for file operations and a configuration dropdown set to "<default config>". The left sidebar contains the "Projects" and "Files" tabs, showing a project named "JavaApplication28" with sub-packages "Source Packages" and "Test Packages". The "Source Packages" tab is active, showing the file "JavaApplication28.java". Below this is the "Navigator" tab, which shows the "Members" of the "JavaApplication28" class, including "factorial(int n) : int" and "main(String[] args)". The main editor window shows the source code of "JavaApplication28.java". The code defines a package, imports "java.util.Scanner", and contains two methods: "main" and "factorial". The "main" method prompts the user to enter an integer value, reads it, and calculates its factorial. The "factorial" method is a static recursive function. The bottom of the IDE shows the "Output" window, which displays the results of running the application. The output shows the prompt "Enter an integer value: 5", the result "5!= 120", and a successful build message.

```
1 package javaapplication28;
2 import java.util.Scanner;
3 public class JavaApplication28 {
4     public static void main(String[] args) {
5         Scanner input=new Scanner(System.in);
6         System.out.print("Enter an integer value: ");
7         int n=input.nextInt();
8         int f=factorial(n);
9         System.out.println(n+"!= "+f);
10    }
11
12    public static int factorial(int n) {
13        int f=1;
14        for(int i=1;i<=n;i++)
15            f=f*i;
16        return f;
17    }
18 }
19
```

Output - JavaApplication28 (run)

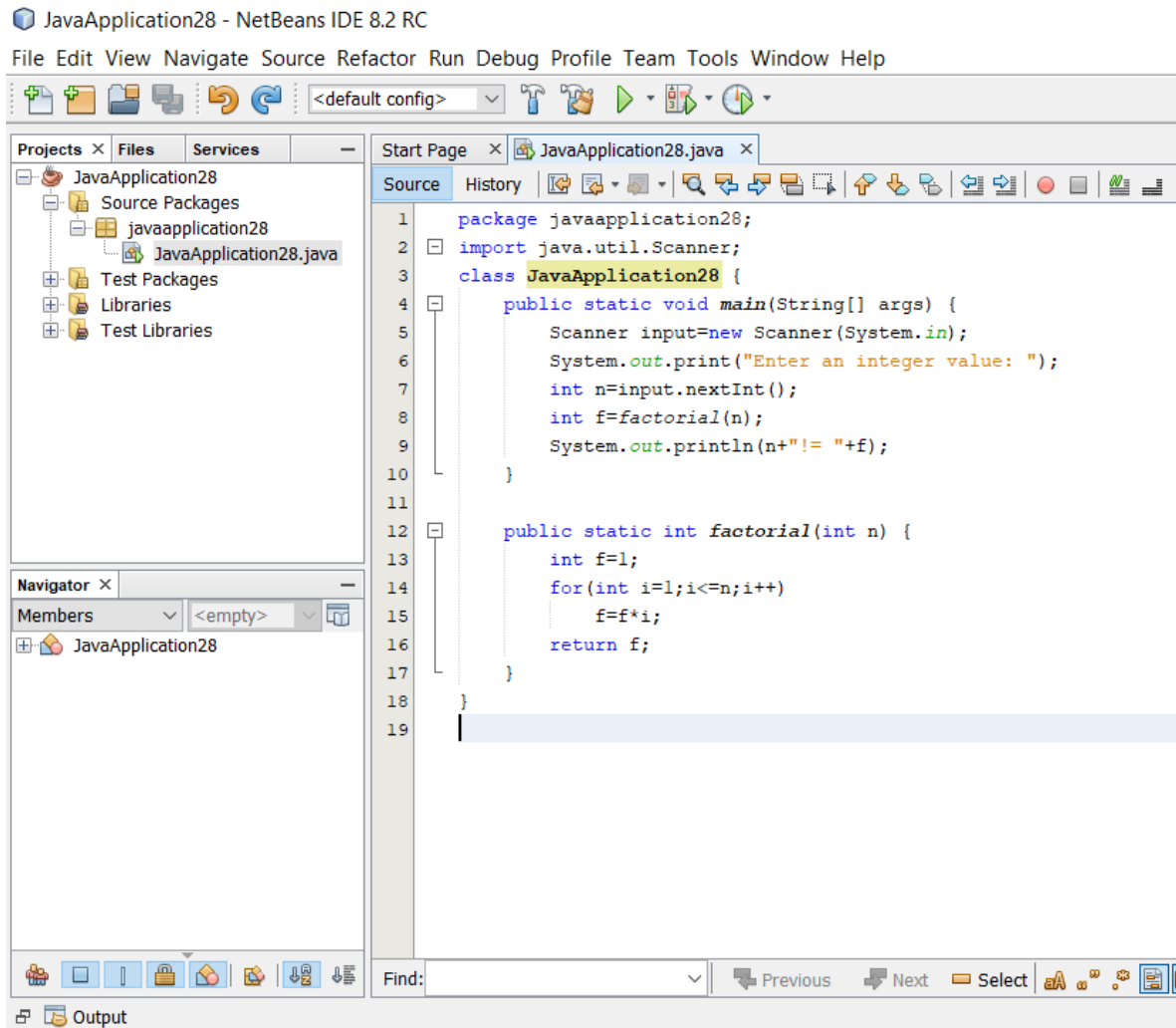
```
run:
Enter an integer value: 5
5!= 120
BUILD SUCCESSFUL (total time: 5 seconds)
```

## Activity 2:

*This activity shows that how to use debugging feature of NetBeans. A debugger is a tool that is typically used to allow the user to view the execution state and data hold by variables as the application is running.*

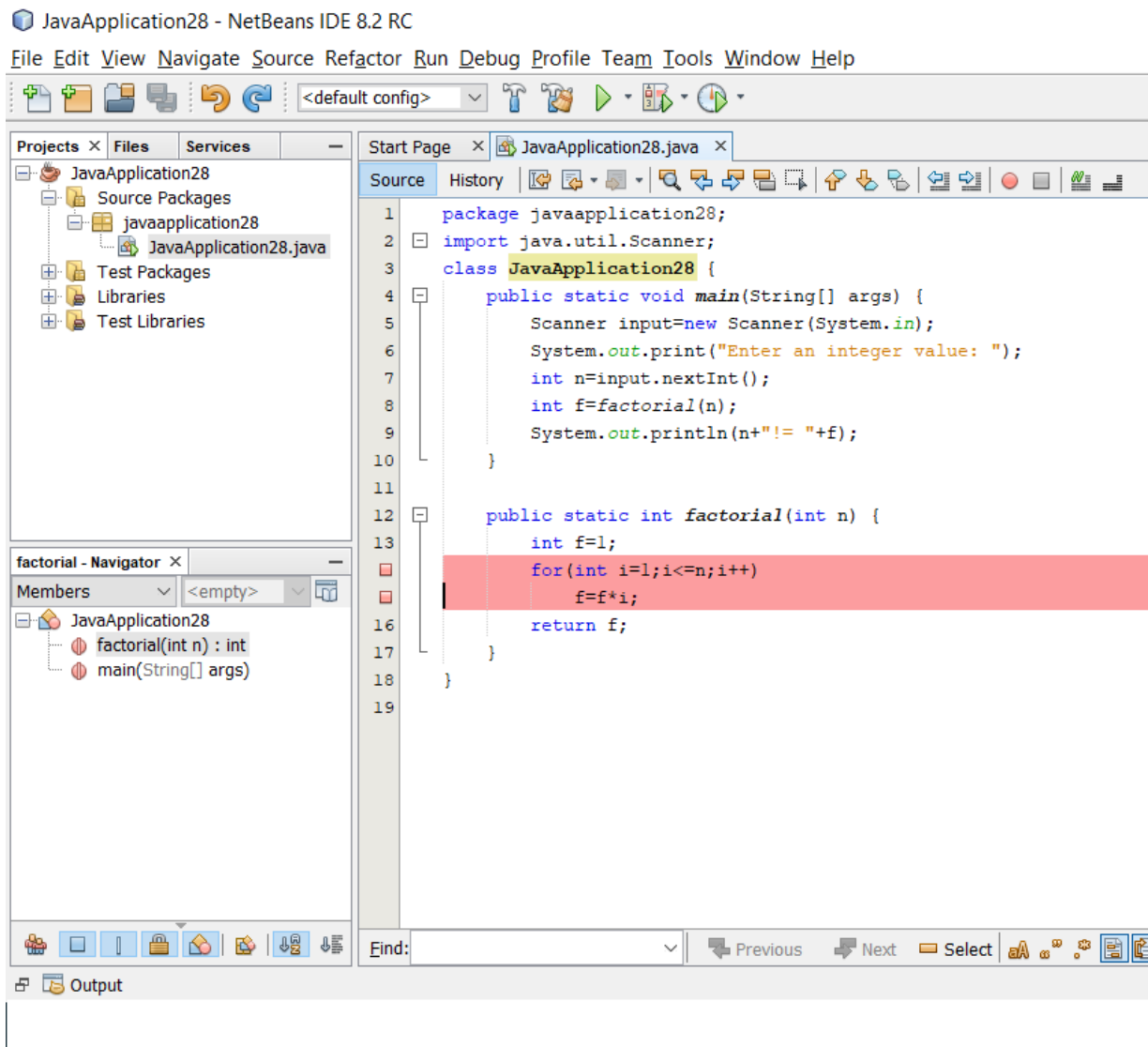
### Solution:

**Step 1:** Write the following code in NetBeans.



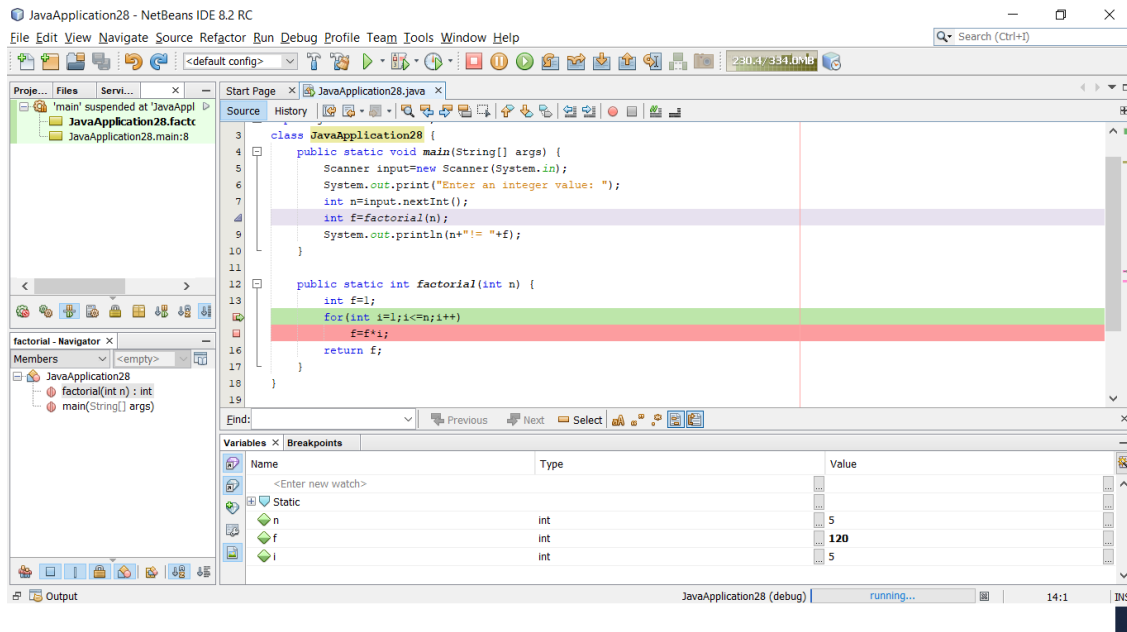


**Step 2:** Click on line number 14 and 15 to insert the line break. It should look as below:



**Step 3:** Press Ctrl+Shift+F5 to debug the program. It will run the program and show you the values of each variable during execution. Press F8 for 'Step Over' to see the values of next iteration.

## Output



## Activity 3:

*This activity demonstrates the use of a built-in Java collection called `ArrayList`. Java `ArrayList` class uses a dynamic array for storing the elements. It is like an array, but there is no size limit. We can add or remove elements anytime. So, it is much more flexible than the traditional array. It is found in the `java.util` package.*

## Solution:

```
1 import java.util.Scanner;
2 import java.util.ArrayList;
3 import java.util.Collections;
4
5 class ArrayListExample {
6     static Scanner input=new Scanner(System.in);
7     static ArrayList<Integer> al=new ArrayList<Integer>();
8     public static void main(String[] args) {
9         boolean bool=true;
10        while(bool){
11            int choice=displayChoices();
12            switch(choice){
13                case 1:
14                    inputData();
15                    break;
16                case 2:
17                    displayData();
18                    break;
19                case 3:
20                    al.clear();
21                case 4:
22                    findValueAtGivenIndex();
23                    break;
24                case 5:
25                    findIndexOfGivenValue();
26                    break;
```

```

27         case 6:
28             removeData();
29             break;
30         case 7:
31             System.out.println("Current size of array list is "+al.size());
32             break;
33         case 8:
34             Collections.sort(al);
35             System.out.println("Data sorted in ascending order");
36             break;
37         case 9:
38             Collections.sort(al, Collections.reverseOrder());
39             System.out.println("Data sorted in deslscending order");
40             break;
41         case 0:
42             bool=false;
43             break;
44         default:
45             System.out.println("Invalid Choice");
46     }
47 }
48 }

```

## Output

```

1: Input data
2: Display data
3: Clear data
4: Find data at particular index
5: Find index of given data
6: Remove the data at given index
7: Current size of array list
8: Sort data in ascending order
9: Sort data in descending order
0: Exit
Enter your choice: 1
Enter an integer value: 1
Enter an integer value: 5
Enter an integer value: 2
Enter an integer value: 4
Enter an integer value: 3
Enter an integer value: 0
1: Input data
2: Display data
3: Clear data
4: Find data at particular index
5: Find index of given data
6: Remove the data at given index
7: Current size of array list
8: Sort data in ascending order
9: Sort data in descending order
0: Exit
Enter your choice: |

```

```

1: Input data
2: Display data
3: Clear data
4: Find data at particular index
5: Find index of given data
6: Remove the data at given index
7: Current size of array list
8: Sort data in ascending order
9: Sort data in descending order
0: Exit
Enter your choice: 2
Current data in array list is...
1
5
2
4
3
1: Input data
2: Display data
3: Clear data
4: Find data at particular index
5: Find index of given data
6: Remove the data at given index
7: Current size of array list
8: Sort data in ascending order
9: Sort data in descending order
0: Exit
Enter your choice: |

```

```

Enter your choice: 8
Data sorted in ascending order
1: Input data
2: Display data
3: Clear data
4: Find data at particular index
5: Find index of given data
6: Remove the data at given index
7: Current size of array list
8: Sort data in ascending order
9: Sort data in descending order
0: Exit
Enter your choice: 2
Current data in array list is...
1
2
3
4
5
1: Input data
2: Display data
3: Clear data
4: Find data at particular index
5: Find index of given data
6: Remove the data at given index
7: Current size of array list
8: Sort data in ascending order
9: Sort data in descending order
0: Exit
Enter your choice: |

```

```

Enter your choice: 9
Data sorted in deslscending order
1: Input data
2: Display data
3: Clear data
4: Find data at particular index
5: Find index of given data
6: Remove the data at given index
7: Current size of array list
8: Sort data in ascending order
9: Sort data in descending order
0: Exit
Enter your choice: 2
Current data in array list is...
5
4
3
2
1
1: Input data
2: Display data
3: Clear data
4: Find data at particular index
5: Find index of given data
6: Remove the data at given index
7: Current size of array list
8: Sort data in ascending order
9: Sort data in descending order
0: Exit
Enter your choice: |

```

```
1: Input data
2: Display data
3: Clear data
4: Find data at particular index
5: Find index of given data
6: Remove the data at given index
7: Current size of array list
8: Sort data in ascending order
9: Sort data in descending order
0: Exit
Enter your choice: 4
Enter Index Number 44
You entered a wrong index
1: Input data
2: Display data
3: Clear data
4: Find data at particular index
5: Find index of given data
6: Remove the data at given index
7: Current size of array list
8: Sort data in ascending order
9: Sort data in descending order
0: Exit
Enter your choice: |
```

### 3) Graded Lab Tasks

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

#### Lab Task 1

*Write a Java program to keep accepting roll number, name and marks of students from user until 0 is entered against roll number. Display name of roll number of student having maximum marks.*

#### Lab Task 2

*Rewrite the above program to display the data of students in descending order according to their marks such that data of student having maximum marks should come at the top and the data of student having minimum marks should come at the bottom.*

# Lab 13

## File Handling

### Objective:

The objective of this lab is to provide hands-on activities related to the File Handling concepts.

### Activity Outcomes:

Upon the completion of this lab, students will be able to

- Write data to files
- Read data from files
- Use streams to implement file handling

### Instructor Note:

As pre-lab activity, read Chapter 17 from the text book “Java How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.

## 1) Useful Concepts

The input and output shown so far has used the pre-defined “standard” streams System.in and System.out. Obviously in many real applications it is necessary to access named files. In many programming languages there is a logical (and practical) distinction made between files that will contain text and those while will be used to hold binary information. Files processed as binary are thought of as sequences of 8-bit bytes, while ones containing text model sequences of characters.

## 2) Solved Lab Activities

<i>Sr.No</i>	<i>Allocated Time</i>	<i>Level of Complexity</i>	<i>CLO Mapping</i>
<i>Activity 1</i>	<i>15 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 2</i>	<i>15 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 3</i>	<i>15 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 4</i>	<i>15 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 5</i>	<i>15 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 6</i>	<i>15 mins</i>	<i>High</i>	<i>CLO-5</i>

### Activity 1:

*Write a Java program to store a character array in a text file. Display all the characters stored in a text file.*

### Solution:

```
1 import java.io.*;
2 class FileStreamTest {
3
4     public static void main(String args[]) {
5
6         try {
7             char ch[] = {'j','a','v','a'};
8             OutputStream os = new FileOutputStream("test.txt");
9             for(int i = 0; i < ch.length ; i++) {
10                 os.write( ch[i] );
11             }
12             os.close();
13
14             InputStream is = new FileInputStream("test.txt");
15             int size = is.available();
16
17             for(int i = 0; i < size; i++) {
18                 System.out.print((char)is.read() + " ");
19                 //System.out.print(is.read() + " ");
20             }
21             is.close();
22         } catch(IOException e) {
23             System.out.print("Exception");
24         }
25     }
26 }
```

### Output

```
j a v a
Process completed.
```

## Activity 2:

*Write a Java program to keep accepting characters from user and store in a file. Display all the characters stored in a text file.*

## Solution:

```
1 import java.io.*;
2 class FileStreamTest {
3     public static void main(String args[]) {
4         InputStreamReader cin = null;
5         try {
6             cin = new InputStreamReader(System.in);
7             OutputStream os = new FileOutputStream("test.txt");
8             System.out.println("Enter characters, 'q' to quit.");
9             char c;
10            do {
11                c = (char) cin.read();
12                os.write( c);
13            } while(c != 'q');
14            os.close();
15            System.out.println("The given data is as below");
16            InputStream is = new FileInputStream("test.txt");
17            int size = is.available();

18            for(int i = 0; i < size; i++) {
19                System.out.print((char)is.read());
20            }
21            is.close();
22        } catch(IOException e) {
23            System.out.print("Exception");
24        }
25    }
26 }
```

## Output:

```
Enter characters, 'q' to quit.
Muzaffar Iqbal
The given data is as below
Muzaffar Iq
Process completed.
```

### Activity 3:

*This activity shows that how you can write data of different type to a file using `PrintWriter` class.*

### Solution:

```
1  import java.io.*;
2  class WriteFile {
3      public static void main(String[] args)
4      {
5          int rno=101;
6          String name="Muzaffar";
7          int marks=88;
8          try {
9              FileOutputStream fos = new FileOutputStream("d:\\myfile.txt");
10             PrintWriter writer=new PrintWriter(fos);
11
12             writer.println(rno);
13             writer.println(name);
14             writer.println(marks);
15             System.out.println("Successfully written.");
16             writer.close();
17         }
18         catch (IOException e) {
19             System.out.println("An error has occurred.");
20         }
21     }
22 }
```

### Output

```
Successfully written.
Process completed.
```



#### Activity 4:

*This activity shows that how to read data saved in file in previous activity.*

#### Solution:

```
import java.io.*;
import java.util.Scanner;

public class ReadFile {
    public static void main(String[] args)
    {
        try {
            File Obj = new File("myfile.txt");
            Scanner Reader = new Scanner(Obj);
            while (Reader.hasNext()) {
                int rno=Reader.nextInt();
                System.out.println("R. No="+rno);
                String name=Reader.next();
                System.out.println("Name="+name);
                int marks=Reader.nextInt();
                System.out.println("Marks="+marks);
            }
            Reader.close();
        }
        catch (FileNotFoundException e) {
            System.out.println("An error has occurred.");
        }
    }
}
```

#### Output

```
R. No=101
Name=Muzaffar
Marks=88

Process completed.
```

## Activity 5:

*This activity shows that how to read data from user and save in file.*

## Solution:

```
1 import java.io.*;
2 import java.util.Scanner;
3 class WriteFile {
4     public static void main(String[] args)
5     {
6         Scanner input=new Scanner(System.in);
7         int rno;
8         String name;
9         int marks;
10        try {
11            FileOutputStream fos = new FileOutputStream("d:\\myfile.txt");
12            PrintWriter writer=new PrintWriter(fos);
13            while(true)
14            {
15                System.out.print("Enter Roll Number ");
16                rno=input.nextInt();
17                if(rno==0)
18                    break;
19                System.out.print("Name ");
20                name=input.next();
21                System.out.print("Enter Marks ");
22                marks=input.nextInt();
23                writer.println(rno);
24                writer.println(name);
25                writer.println(marks);
26            }
27            System.out.println("Successfully written.");
28            writer.close();
29        }
30        catch (IOException e) {
31            System.out.println("An error has occurred.");
32        }
33    }
34 }
```

## Output:

```
Enter Roll Number 101
Name Muzaffar
Enter Marks 88
Enter Roll Number 123
Name Iqbal
Enter Marks 61
Enter Roll Number 132
Name Farooq
Enter Marks 99
Enter Roll Number 0
Successfully written.

Process completed.
```

## Activity 6:

*This activity shows that how to read data from file saved in previous activity.*

### Solution:

```
import java.io.*;
import java.util.Scanner;

public class ReadFile {
    public static void main(String[] args)
    {
        try {
            File Obj = new File("myfile.txt");
            Scanner Reader = new Scanner(Obj);
            while (Reader.hasNext()) {
                int rno=Reader.nextInt();
                System.out.println("R. No="+rno);
                String name=Reader.next();
                System.out.println("Name="+name);
                int marks=Reader.nextInt();
                System.out.println("Marks="+marks);
            }
            Reader.close();
        }
        catch (FileNotFoundException e) {
            System.out.println("An error has occurred.");
        }
    }
}
```

### Output:

```
R. No=101
Name=Muzaffar
Marks=88
R. No=123
Name=Iqbal
Marks=61
R. No=132
Name=Farooq
Marks=99

Process completed.
```

### **3) Graded Lab Tasks**

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

#### **Lab Task 1**

*Write a Java program to accept 10 integer values from user. Save this data in a file.*

#### **Lab Task 2**

*Write a Java program to accept all the data stored in a file in the previous task. Sort this data in ascending order and display on screen.*

#### **Lab Task 3**

*Write a Java program to accept all the data stored in a file in the previous task (Lab Task1). Remove all the prime numbers from this data and save again in the same file.*

# Lab 14

## Testing

### Objective:

The objective of this lab is to develop and execute different test cases in order to test the program.

### Activity Outcomes:

Upon the completion of this lab, students will be able to

- Develop test cases using JUnit Testing Framework in Java
- Use JUnit in NetBeans

### Instructor Note:

As pre-lab activity, read the [JUnit](https://www.vogella.com/tutorials/JUnit/article.html) article given on the link[<https://www.vogella.com/tutorials/JUnit/article.html>]

## 1) Useful Concepts

JUnit is a Regression Testing Framework used by developers to implement unit testing in Java, and accelerate programming speed and increase the quality of code. JUnit Framework can be easily integrated with either of the following:

- NetBeans
- Eclipse
- IntelliJ
- Maven

### Features of JUnit Test Framework

JUnit test framework provides the following important features:

- Fixtures
- Test suites
- Test runners
- JUnit classes

### Fixtures

Fixtures is a fixed state of a set of objects used as a baseline for running tests. The purpose of a test fixture is to ensure that there is a well-known and fixed environment in which tests are run so that results are repeatable. It includes:

- setUp() method, which runs before every test invocation.
- tearDown() method, which runs after every test method.

### JUnit Classes

JUnit classes are important classes, used in writing and testing JUnits. Some of the important classes are:

- Assert – Contains a set of assert methods.
- TestCase – Contains a test case that defines the fixture to run multiple tests.
- TestResult – Contains methods to collect the results of executing a test case.

## 2) Solved Lab Activities

<i>Sr.No</i>	<i>Allocated Time</i>	<i>Level of Complexity</i>	<i>CLO Mapping</i>
<i>Activity 1</i>	<i>20 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 2</i>	<i>20 mins</i>	<i>High</i>	<i>CLO-5</i>
<i>Activity 3</i>	<i>20 mins</i>	<i>High</i>	<i>CLO-5</i>

### Activity 1:

*Write a Java method to accept a value and return its factorial. Write the main method to check the factorial method. You are also required to use JUnit for writing and testing unit testing for the defined factorial method.*

### Solution:

**Step 1:** Write the java application having factorial method as below

```

1  package javaapplication29;
2  import java.util.Scanner;
3  public class JavaApplication29 {
4      public static void main(String[] args) {
5          Scanner input=new Scanner(System.in);
6          System.out.print("Enter a number: ");
7          int n=input.nextInt();
8          System.out.println(n+"!= "+factorial(n));
9      }
10
11     public static int factorial(int n)
12     {
13         int f=1;
14         for(int i=1;i<=n;i++)
15             f=f*i;
16         return f;
17     }
18 }

```

**Step 2:** Select 'Create/Update Tests' from Tools menu and write the test cases for the factorial method as shown below

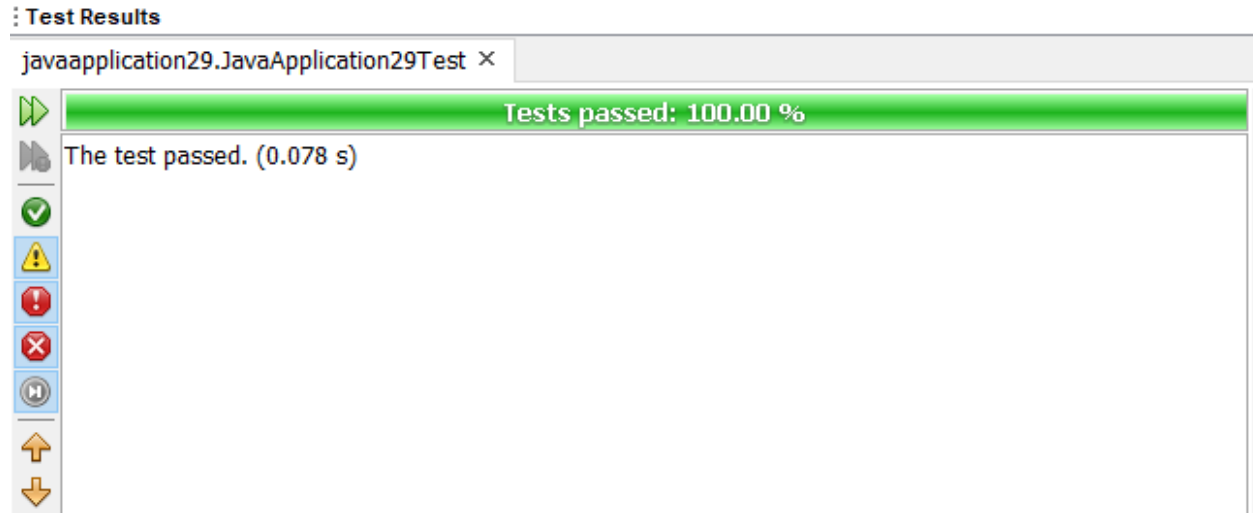
```

1  package javaapplication29;
2  import org.junit.After;
3  import org.junit.Test;
4  import static org.junit.Assert.*;
5  import org.junit.Before;
6  public class JavaApplication29Test {
7      JavaApplication29 obj=new JavaApplication29();
8      @Test
9      public void testFactorial() {
10         int n=5;
11         int f=JavaApplication29.factorial(n);
12         assertEquals(120,f);
13         assertEquals(3628800,JavaApplication29.factorial(10));
14     }
15 }

```

## Output

Press Ctrl+F6 to get the output as given below.



### Activity 2:

*Write a Java program to define the following methods and write the appropriate test case for them:*

```
boolean isOdd(int)  
boolean isEven(int)  
boolean isPrime(int)
```

### Solution:

**Step 1:** Write the java application having the above mentioned method as below



```

1  package javaapplication29;
2  public class JavaApplication29 {
3      public static void main(String[] args) {
4
5      }
6      public static boolean isOdd(int n){
7          if(n%2==1)
8              return true;
9          else
10             return false;
11     }
12     public static boolean isEven(int n){
13         if(n%2==0)
14             return true;
15         else
16             return false;
17     }
18     public static boolean isPrime(int n){
19         boolean prime=true;
20         for(int i=2;i<n/2;i++)
21             if(n%i==0){
22                 prime=false;
23                 break;
24             }
25         return prime;
26     }
27 }

```

**Step 2:** Select 'Create/Update Tests' from Tools menu and write the test cases for the factorial method as shown below...

```

1 package javaapplication29;
2 import junit.framework.Assert;
3 import org.junit.After;
4 import org.junit.Test;
5 import static org.junit.Assert.*;
6 import org.junit.Before;
7 public class JavaApplication29Test {
8     JavaApplication29 obj=new JavaApplication29();
9     @Test
10    public void testIsEven() {
11        assertTrue(JavaApplication29.isEven(2));
12        assertTrue(JavaApplication29.isEven(4));
13        assertTrue(JavaApplication29.isEven(6));
14        assertTrue(JavaApplication29.isEven(8));
15        assertTrue(JavaApplication29.isEven(10));
16    }
17    @Test
18    public void testIsOdd() {
19        assertTrue(JavaApplication29.isOdd(1));
20        assertTrue(JavaApplication29.isOdd(3));
21        assertTrue(JavaApplication29.isOdd(5));
22        assertTrue(JavaApplication29.isOdd(7));
23        assertTrue(JavaApplication29.isOdd(9));
24    }
25    @Test
26    public void testIsPrime() {
27        assertTrue(JavaApplication29.isPrime(3));
28        assertTrue(JavaApplication29.isPrime(5));
29        assertTrue(JavaApplication29.isPrime(7));
30        assertTrue(JavaApplication29.isPrime(11));
31        assertTrue(JavaApplication29.isPrime(13));
32        assertTrue(JavaApplication29.isPrime(17));
33        assertTrue(JavaApplication29.isPrime(19));
34    }
35 }

```

## Output

Press Ctrl+F6 to get the output

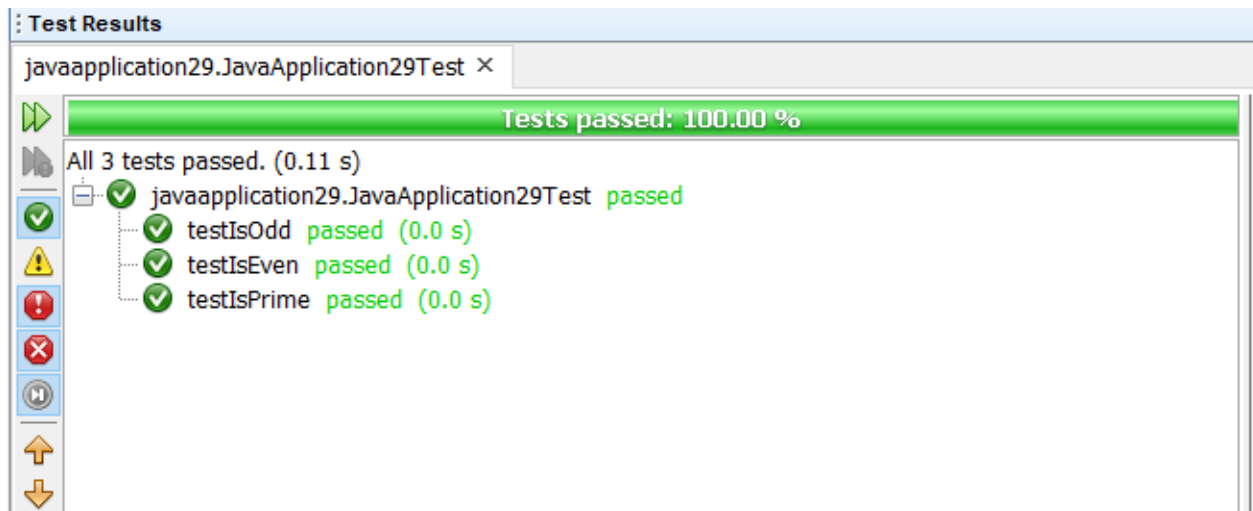
**Test Results**

javaapplication29.JavaApplication29Test ×

Tests passed: 100.00 %

All 3 tests passed. (0.11 s)

- ✓ javaapplication29.JavaApplication29Test passed
  - ✓ testIsOdd passed (0.0 s)
  - ✓ testIsEven passed (0.0 s)
  - ✓ testIsPrime passed (0.0 s)



### Activity 3:

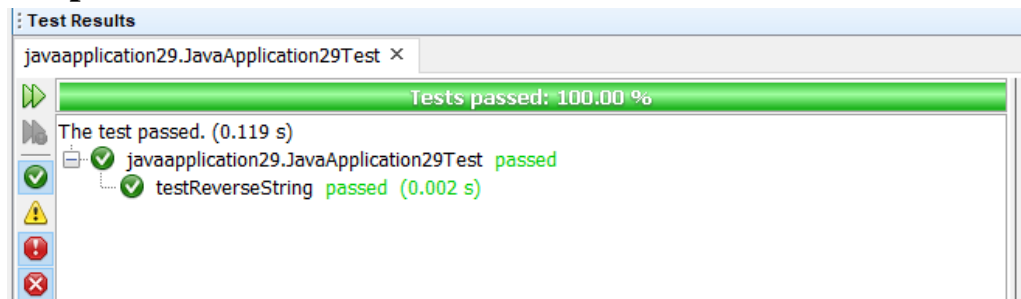
Write a Java method to accept a string and return the string in reverse order. You are required to write necessary test cases for this method.

### Solution:

```
1 package javaapplication29;
2 public class JavaApplication29 {
3     public static void main(String[] args) {
4         System.out.println(reverseString("abc"));
5     }
6     public static String reverseString(String str){
7         char ch[]=str.toCharArray();
8         String rev="";
9         for(int i=ch.length-1;i>=0;i--){
10             rev+=String.valueOf(ch[i]);
11         }
12         return rev;
13     }
14 }

1 package javaapplication29;
2 import junit.framework.Assert;
3 import org.junit.After;
4 import org.junit.Test;
5 import static org.junit.Assert.*;
6 import org.junit.Before;
7 public class JavaApplication29Test {
8     JavaApplication29 obj=new JavaApplication29();
9     @Test
10     public void testReverseString() {
11         assertEquals("cba",JavaApplication29.reverseString("abc") );
12         assertEquals("54321",JavaApplication29.reverseString("12345") );
13     }
14 }
```

### Output:



### 3) Graded Lab Tasks

*Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.*

#### Lab Task 1

*Write a Java method as given below to accept day, month and year and check that whether the given date is valid or not*

**`boolean checkDate(int, int, int)`**

*You are required to write necessary test cases to test the above method.*

#### Lab Task 2

*Write a Java method as given below to accept an integer and check that whether the given number is a perfect number or not.*

**`Boolean isPerfectNumber(int)`**

*You are required to write necessary test cases to test the above method.*

#### Lab Task 3

*Write a Java method as given below to accept an integer and return the day of the week.*

**`String dayOfTheWeek(int)`**

*You are required to write necessary test cases to test the above method.*