

A Laboratory Manual for

Object Oriented Programming -I

(3140705)

B.E. Semester 4th
(Computer Engineering)



Government Engineering College, Gandhinagar
Gujarat



Directorate of Technical Education,
Gandhinagar, Gujarat

Government Engineering College, Gandhinagar

Certificate

This is to certify that Ms. __Bhabhor sonali_____
__ Enrollment No. _220130107010_____ of B.E. Semester
_4th____Computer Engineering of this Institute (GTU Code: _____) has
satisfactorily completed the Practical / Tutorial work for the subject **Object
Oriented Programming-I(3140705)** for the academic year 2023-24.

Place: _____

Date: _____

Name and Sign of Faculty member

Head of the Department

Preface

Main motto of any laboratory/practical/field work is for enhancing required skills as well as creating ability amongst students to solve real time problem by developing relevant competencies in psychomotor domain. By keeping in view, GTU has designed competency focused outcome-based curriculum for engineering degree programs where sufficient weightage is given to practical work. It shows importance of enhancement of skills amongst the students and it pays attention to utilize every second of time allotted for practical amongst students, instructors and faculty members to achieve relevant outcomes by performing the experiments rather than having merely study type experiments. It is must for effective implementation of competency focused outcome-based curriculum that every practical is keenly designed to serve as a tool to develop and enhance relevant competency required by the various industry among every student. These psychomotor skills are very difficult to develop through traditional chalk and board content delivery method in the classroom. Accordingly, this lab manual is designed to focus on the industry defined relevant outcomes, rather than old practice of conducting practical to prove concept and theory.

By using this lab manual students can go through the relevant theory and procedure in advance before the actual performance which creates an interest and students can have basic idea prior to performance. This in turn enhances pre-determined outcomes amongst students. Each experiment in this manual begins with competency, industry relevant skills, course outcomes as well as practical outcomes (objectives). The students will also achieve safety and necessary precautions to be taken while performing practical.

This manual also provides guidelines to faculty members to facilitate student centric lab activities through each experiment by arranging and managing necessary resources in order that the students follow the procedures with required safety and necessary precautions to achieve the outcomes. It also gives an idea that how students will be assessed by providing rubrics.

Java is a multi-platform, object-oriented, and network-centric language that can be used as a platform. It is a fast, secure, reliable programming language for coding everything from mobile apps and enterprise software to big data applications and server-side technologies. Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages.

Utmost care has been taken while preparing this lab manual however always there is chances of improvement. Therefore, we welcome constructive suggestions for improvement and removal of errors if any.

Practical – Course Outcome matrix

Course Outcomes (COs):						
<ol style="list-style-type: none"> 1. Use various Java constructs, features and libraries for simple problems. 2. Demonstrate how to define and use classes, interfaces, create objects and methods, how to override and overload methods, compile and execute programs. 3. Write a program using exception handling, multithreading with synchronization. 4. Write a program using Files, binary I/O, collection Frameworks for a given problem. 5. Design and develop GUI based applications in a group using modern tools and frameworks. 						
Sr. No.	Objective(s) of Experiment	CO1	CO2	CO3	CO4	CO5
1.	To learn basic java programming constructs.	√				
2.	To learn Arrays and Strings in Java.	√				
3.	To implement basic object-oriented concepts.		√			
4.	To implement inheritance and object-oriented concepts.		√			
5.	To demonstrate the use of abstract classes and interfaces.		√			
6.	To implement packages and exception handling in JAVA application.			√		
7.	To demonstrate I/O from files.				√	
8.	To learn JAVA FX UI Controls.					√
9.	To implement event handling and animation.					√
10.	To learn recursion and generics.				√	
11.	To demonstrate the use of Collection framework.				√	
12.	To demonstrate the use of multithreading.			√		

Industry Relevant Skills

The following industry relevant competency is expected to be developed in the student by undertaking the practical work of this laboratory.

1. Object oriented application development
2. Networking application development
3. GUI based application development

Guidelines for Faculty members

1. Teacher should provide the guideline with demonstration of practical to the students with all features.
2. Teacher shall explain basic concepts/theory related to the experiment to the students before starting of each practical
3. Involve all the students in performance of each experiment.
4. Teacher is expected to share the skills and competencies to be developed in the students and ensure that the respective skills and competencies are developed in the students after the completion of the experimentation.
5. Teachers should give opportunity to students for hands-on experience after the demonstration.
6. Teacher may provide additional knowledge and skills to the students even though not covered in the manual but are expected from the students by concerned industry.
7. Give practical assignment and assess the performance of students based on task assigned to check whether it is as per the instructions or not.
8. Teacher is expected to refer complete curriculum of the course and follow the guidelines for implementation.

Instructions for Students

1. Students are expected to carefully listen to all the theory classes delivered by the faculty members and understand the COs, content of the course, teaching and examination scheme, skill set to be developed etc.
2. Students shall organize the work in the group and make record of all observations.
3. Students shall develop maintenance skill as expected by industries.
4. Student shall attempt to develop related hand-on skills and build confidence.
5. Students shall make a small project/application in Java.
6. Student shall develop the habits of evolving more ideas, innovations, skills etc. apart from those included in scope of manual.
7. Student shall refer technical magazines and books.
8. Student should develop a habit of submitting the experimentation work as per the schedule and s/he should be well prepared for the same.

Common Safety Instructions

Students are expected to

1. Switch on the PC carefully (not to use wet hands)
2. Shutdown the PC properly at the end of your Lab
3. Carefully Handle the peripherals (Mouse, Keyboard, Network cable etc)
4. Use Laptop in lab after getting permission from Teacher

Index
(Progressive Assessment Sheet)

Sr. No.	Objective(s) of Experiment	Page No.	Date of performance	Date of submission	Assessment Marks	Sign. of Teacher with date	Remarks
Total							

1. COURSE OUTCOMES

After learning the course, the students should be able to:

1. Use various Java constructs, features and libraries for simple problems.
2. Demonstrate how to define and use classes, interfaces, create objects and methods, how to override and overload methods, compile and execute programs.
3. Write a program using exception handling, multithreading with synchronization.
4. Write a program using Files, binary I/O, collection Frameworks for a given problem.
5. Design and develop GUI based applications in a group using modern tools and frameworks.

2. TEACHING AND EXAMINATION SCHEME

Teaching Scheme			Credits	Examination Marks				Total Marks
L	T	P	C	Theory Marks		Practical Marks		
				ESE (E)	PA (M)	ESE (V)	PA (I)	
4	0	2	5	70	30	30	20	150

3. SUGGESTED LEARNING RESOURCES

Reference Books:

1. Intro to Java Programming, 10th edition, Y.Daniel Liang, Pearson
2. Object oriented programming with Java , RajkumarBuyya,SThamaraiSelvi, Xingchen Chu, McGrawHill
3. Programming in Java, SachinMalhotra, SaurabhChoudhary, Oxford
4. Programming with JAVA , E Balagurusamy, McGrawHill
5. CORE JAVA volume -I Cay Horstmann, Pearson

Major Equipment: Computer, Laptop

List of Open Source Software/learning website:

<https://docs.oracle.com/javase/tutorial/java/index.html>

<https://www.tutorialspoint.com/JAVA/>

<https://dev.java/learn/>

<https://www.codecademy.com/learn/learn-java>

<https://www.w3schools.com/java/>

Java:

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture.

OpenJDK:

OpenJDK (Open Java Development Kit) is a free and open-source implementation of the Java Platform, Standard Edition (Java SE). It is the result of an effort Sun Microsystems began in 2006. The implementation is licensed under the GPL-2.0-only with a linking exception. Were it not for the GPL linking exception, components that linked to the Java class library would be subject to the terms of the GPL license. OpenJDK is the official reference implementation of Java SE since version 7.

JVM:

The Java Virtual Machine, or JVM, executes live Java applications. Every JRE includes a default JRE, but developers are free to choose another that meets the specific resource needs of their applications.

JRE:

The Java Virtual Machine, or JVM, executes live Java applications. Every JRE includes a default JRE, but developers are free to choose another that meets the specific resource needs of their applications. The Java Virtual Machine, or JVM, executes live Java applications. Every JRE includes a default JRE, but developers are free to choose another that meets the specific resource needs of their applications.

JAVA IDEs:

IDEs typically provide a code editor, a compiler or interpreter and a debugger that the developer accesses through a unified graphical user interface (GUI). Here are a few popular Java IDEs:

Eclipse: a Java-based open source platform that enables the creation of highly customized IDEs from plug-in components built by Eclipse members. The platform is user-friendly for beginners and also suitable for the creation of more sophisticated applications. Eclipse includes a lot of plug-ins that allow developers to develop and test code written in other languages.

NetBeans: a Java-based IDE and underlying application platform framework. In addition to Java, JavaScript and JavaFX, NetBeans supports C/C++, PHP, Groovy, and HTML5.

How to install Java for Windows:

Following are the steps on how to install Java in Windows 10 for JDK 8 free download for 32 bit or JDK8 download for Windows 64 bit and installation

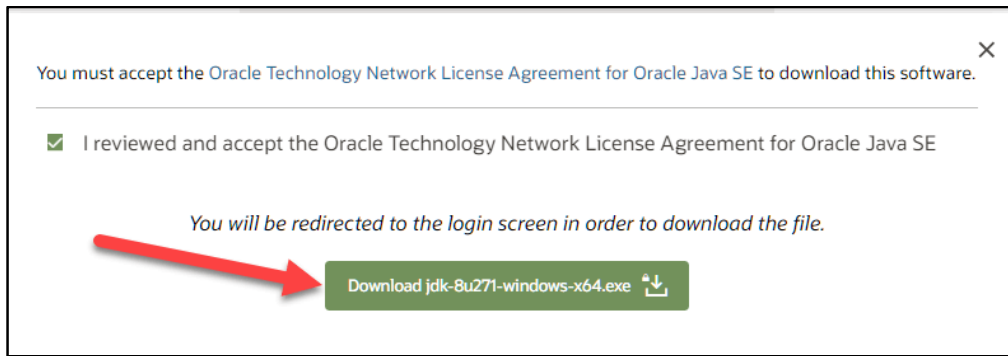
Step 1) Go to <https://www.oracle.com/java/technologies/downloads/>. Click on JDK Download for Java download JDK 8.

Step 2) Next,

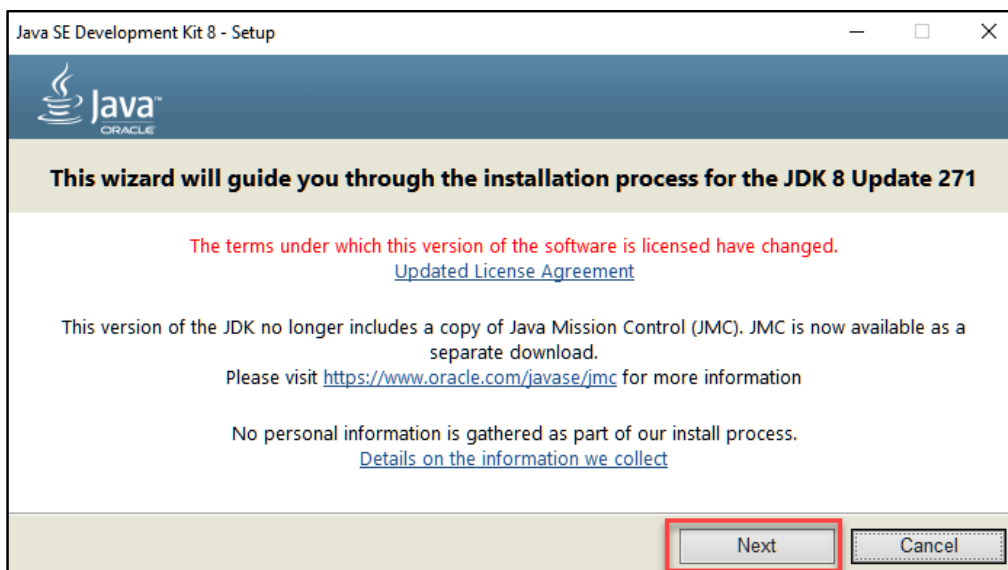
1. Accept License Agreement
2. Download Java 8 JDK for your version 32 bit or JDK download 64 bit.

Solaris SPARC 64-bit	88.75 MB	jdk-8u271-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.42 MB	jdk-8u271-solaris-x64.tar.Z
Solaris x64	92.52 MB	jdk-8u271-solaris-x64.tar.gz
Windows x86	154.48 MB	jdk-8u271-windows-i586.exe
Windows x64	166.79 MB	jdk-8u271-windows-x64.exe

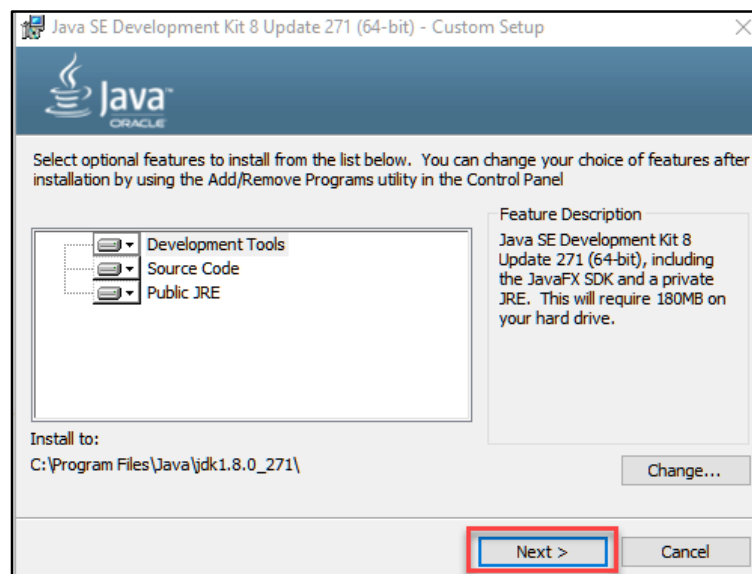
Step 3) When you click on the Installation link the popup will be open. Click on I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE development kit and you will be redirected to the login page. If you don't have an oracle account you can easily sign up by adding basics details of yours.



Step 4) Once the Java JDK 8 download is complete, run the exe for install JDK. Click Next



Step 5) Select the PATH to install Java in Windows... You can leave it Default. Click next.



Step 6) Once you install Java in windows, click Close



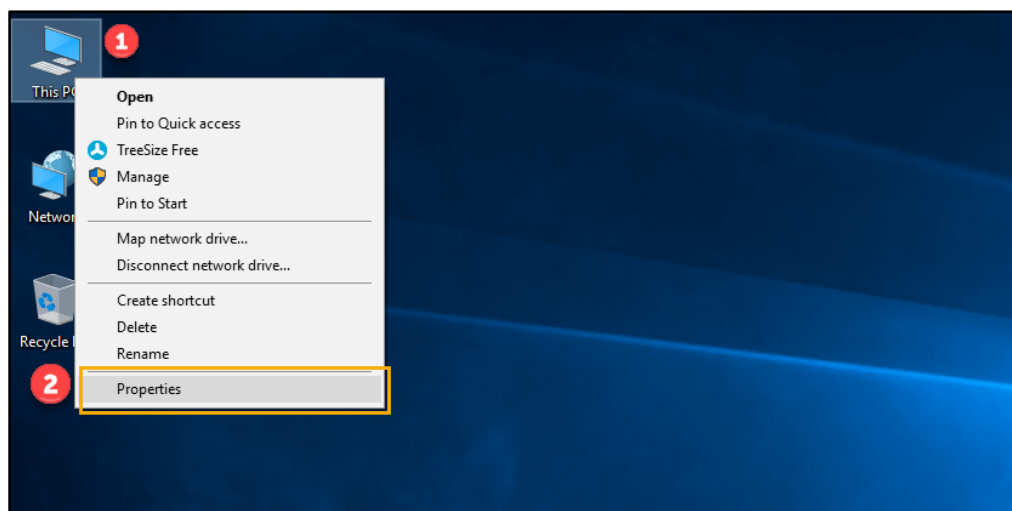
How to set Environment Variables in Java: Path and Classpath:

The PATH variable gives the location of executables like javac, java etc. It is possible to run a program without specifying the PATH but you will need to give full path of executable like **C:\Program Files\Java\jdk1.8.0_271\bin\javac A.java** instead of simple **javac A.java**

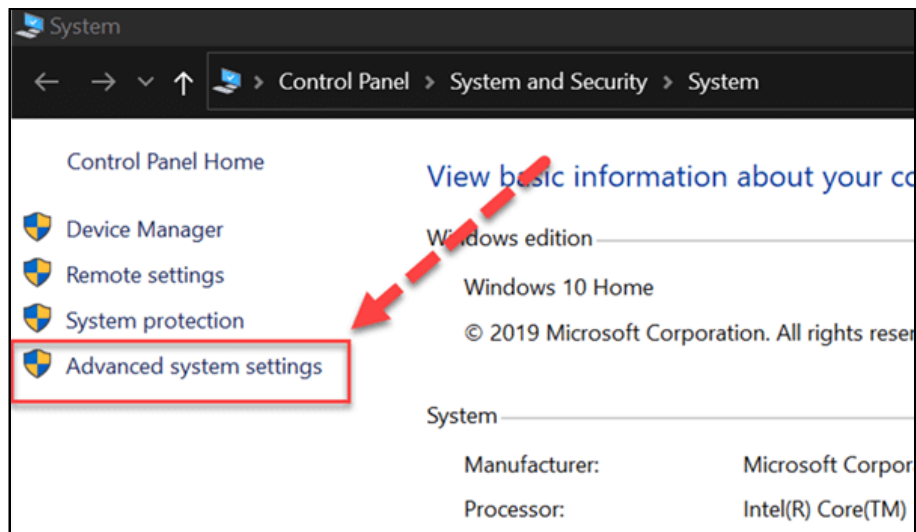
The CLASSPATH variable gives location of the Library Files.

Let's look into the steps to set the PATH and CLASSPATH

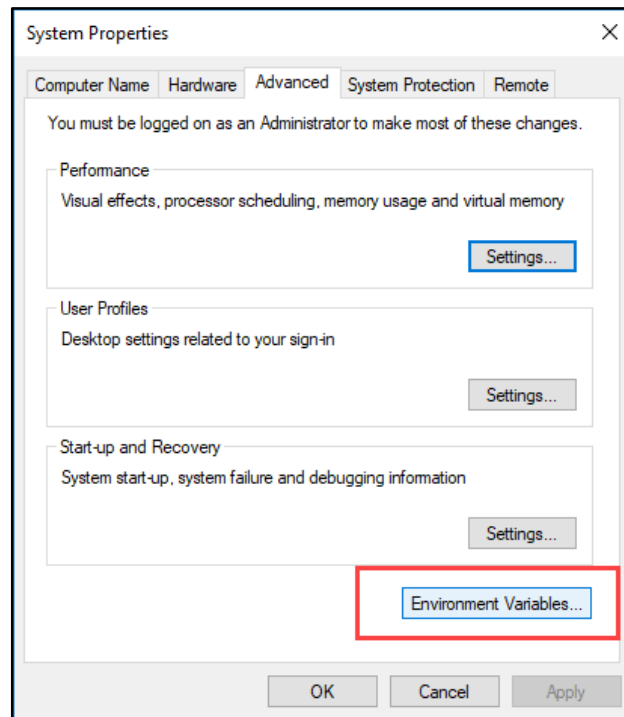
Step 1) Right Click on the My Computer and Select the properties



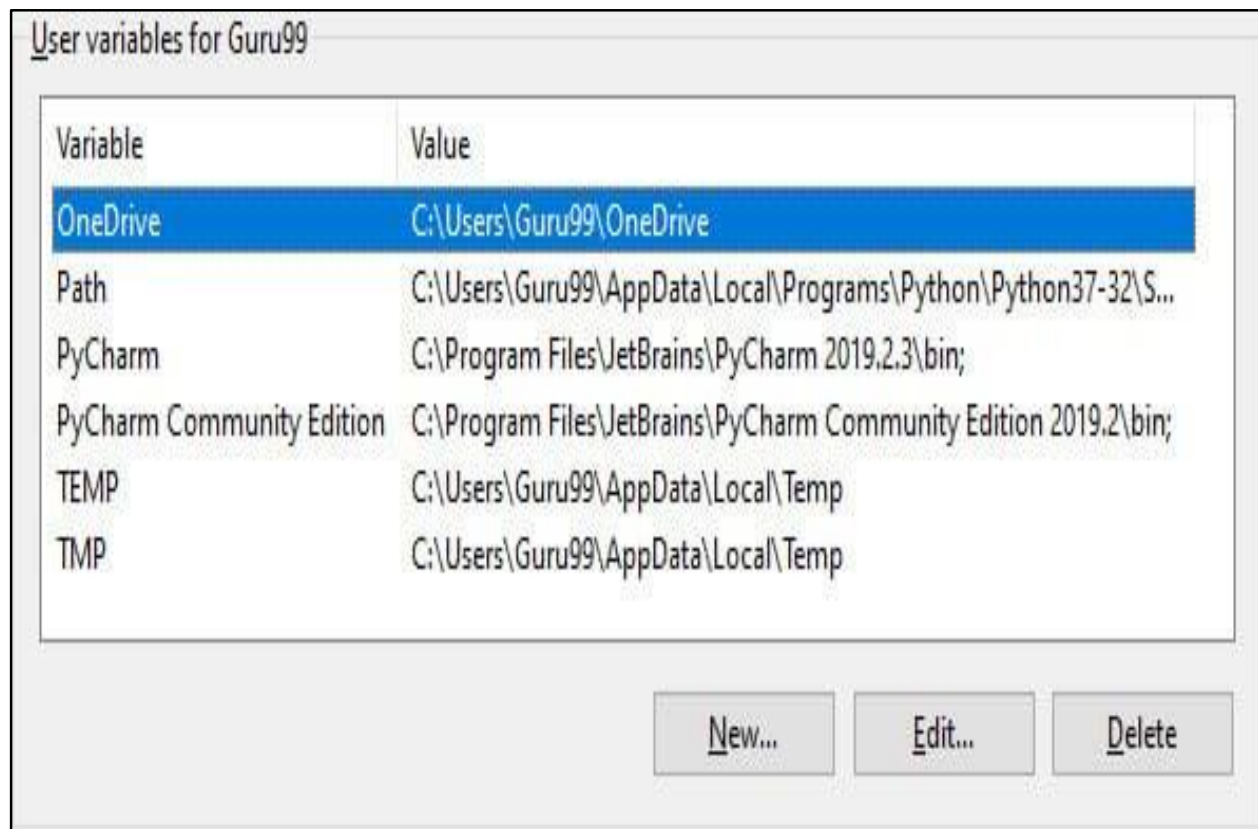
Step 2) Click on advanced system settings



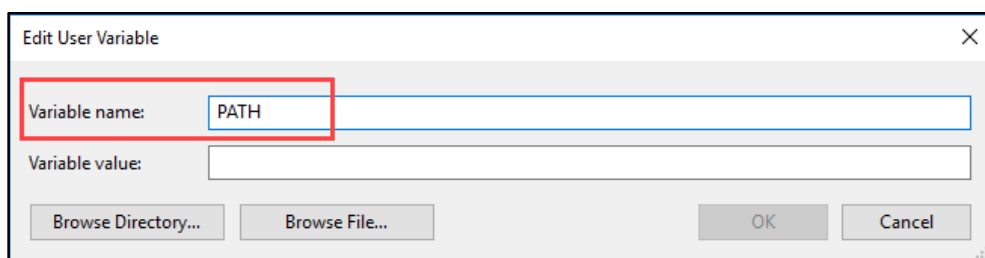
Step 3) Click on Environment Variables to set Java runtime environment



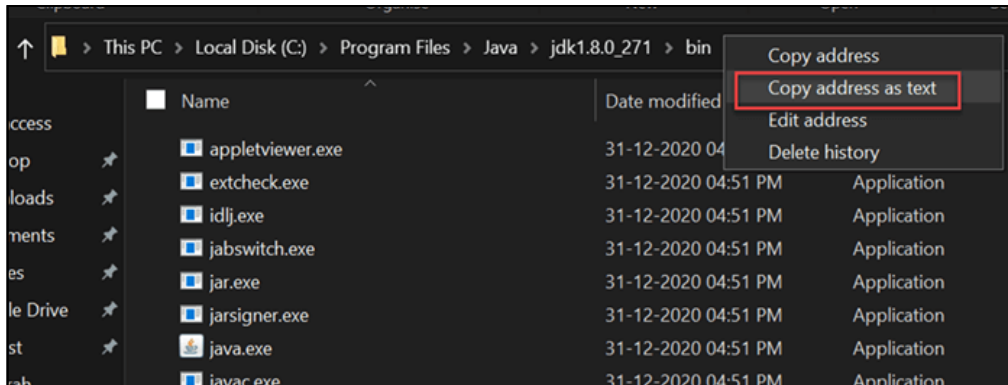
Step 4) Click on new Button of User variables



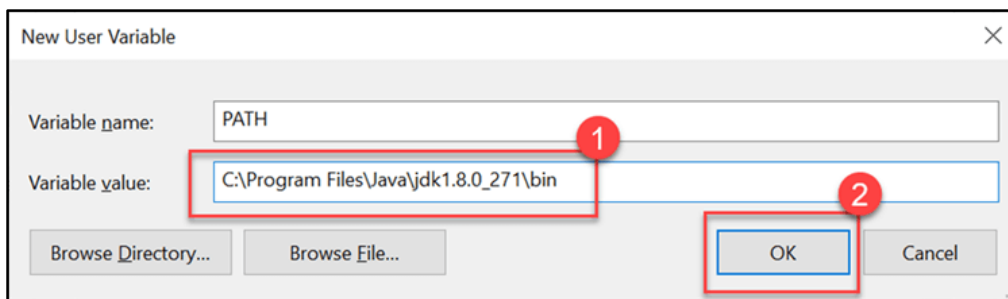
Step 5) Type PATH in the Variable name.



Step 6) Copy the path of bin folder which is installed in JDK folder.



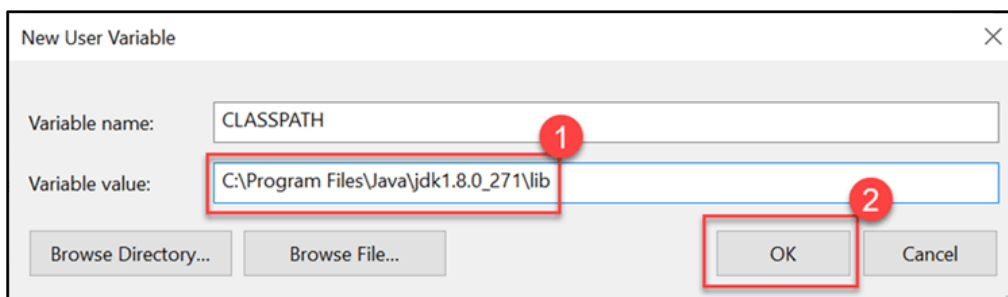
Step 7) Paste Path of bin folder in Variable value. Click on OK Button.



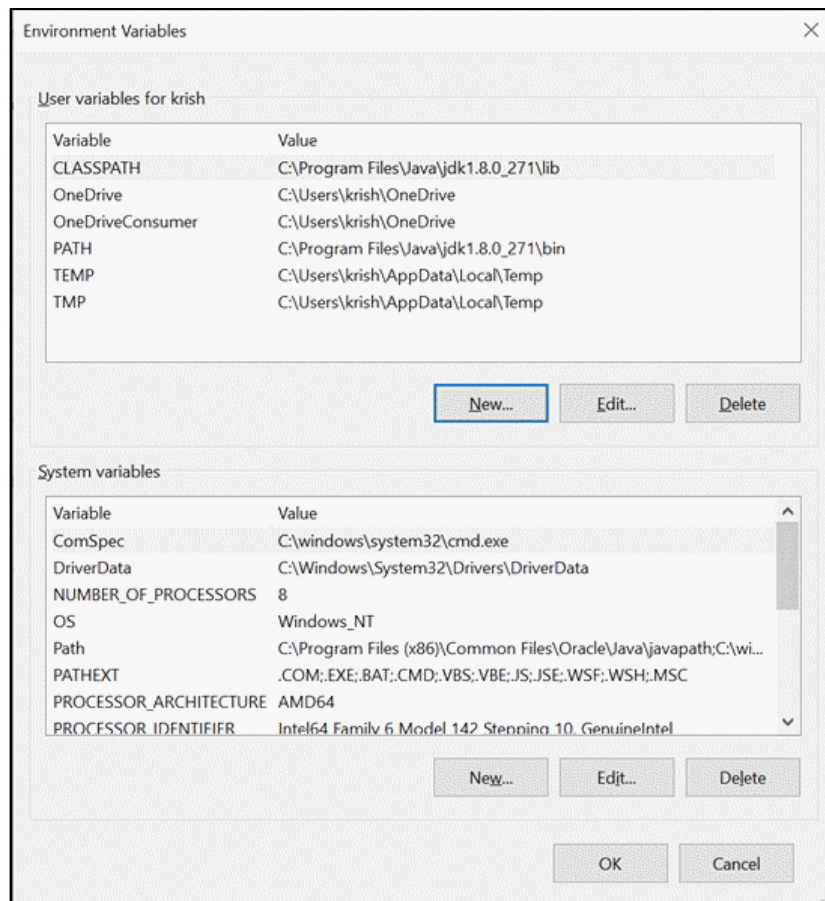
Note: In case you already have a PATH variable created in your PC, edit the PATH variable to
PATH = <JDK installation directory>\bin;%PATH%;

Here, %PATH% appends the existing path variable to our new value

Step 8) You can follow a similar process to set CLASSPATH.



Step 9) Click on OK button



Step 10) Go to command prompt and type javac commands.

If you see a screen like below, Java is installed.

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Guru99>javac
Usage: javac <options> <source files>
where possible options include:
  @<filename>           Read options and filenames from file
  -Akey[-value]         Options to pass to annotation processors
  --add-modules <module>(<module>)*
                        Root modules to resolve in addition to the initial modules, or all modules
                        on the module path if <module> is ALL-MODULE-PATH.
  --boot-class-path <path>, -bootclasspath <path>
                        Override location of bootstrap class files
  --class-path <path>, -classpath <path>, -cp <path>
                        Specify where to find user class files and annotation processors
  -d <directory>        Specify where to place generated class files
  -deprecation           Output source locations where deprecated APIs are used
  --enable-preview       Enable preview language features. To be used in conjunction with either -source or --release.
  -encoding <encoding>  Specify character encoding used by source files
  -endorseddirs <dirs>  Override location of endorsed standards path
  -extdirs <dirs>       Override location of installed extensions
  
```


Experiment No: 1

AIM: To learn basic JAVA programming constructs.

Date:

CO mapped: CO-1

Objectives: (a) To learn and understand the different basic structures in java, such as syntax, logics, libraries and proper indentation.

Background:

Java Variables

A variable is a container that holds the value while the Java program is executed. A variable is assigned with a data type. Variable is a name of a memory location. There are three types of variables in java: local, instance, and static.

Data Types in Java

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

Primitive data types: The primitive data types include boolean, char, byte, short, int, long, float and double.

There are 8 types of primitive data types:

- boolean data type
- byte data type
- char data type
- short data type
- int data type
- long data type
- float data type
- double data type

Non-primitive data types: The non-primitive data types include Classes, Interfaces, and Arrays.

Operators in Java

Operator in Java is a symbol that is used to perform operations. For example: +, -, *, / etc.

There are many types of operators in Java which are given below:

- Unary Operator,
- Arithmetic Operator,
- Shift Operator,
- Relational Operator,
- Bitwise Operator,
- Logical Operator,
- Ternary Operator and
- Assignment Operator.

Java Control Statements

Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear. However, Java provides statements that can be used to control the flow of Java code. Such statements are called control flow statements. It is one of the fundamental features of Java, which provides a smooth flow of program.

Java provides three types of control flow statements.

- Decision Making statements
 - if statements
 - switch statement
- Loop statements
 - do while loop
 - while loop
 - for loop
 - for-each loop
- Jump statements
 - break statement
 - continue statement

Practical questions:

1. Install JDK and IDE in your system. Write down the steps of installation with screenshots.
2. Write a Program that displays Welcome to Java, Learning Java Now and Programming is fun.

Code:

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("220130107094 - Megh Patel");  
        System.out.println("Date: 25-02-24");  
        System.out.println("Welcome to Java");  
        System.out.println("Learning Java Now");  
        System.out.println("Programming is Fun");  
        System.out.println("Time: 12:20 PM");  
    }  
}
```

Output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent
220130107094 - Megh Patel
Date: 25-02-24
Welcome to Java
Learning Java Now
Programming is Fun
Time: 12:20 PM

Process finished with exit code 0
```

3. Write a program that solves the following equation and displays the value x and y:
- $3.4x + 50.2y = 44.5$ $2.1x + .55y = 5.9$ (Assume Cramer's rule to solve equation)
 - $ax + by = e$ $x = \frac{ed - bf}{ad - bc}$ $cx + dy = f$ $y = \frac{af - ec}{ad - bc}$

Code:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 25-02-24");
        System.out.println("Time: 12:30 PM");

        Scanner input = new Scanner(System.in);
        System.out.println("Values from Equation:- 1 :");
        System.out.print("Enter value of a : ");
        double a = input.nextDouble();
        System.out.print("Enter value of b : ");
        double b = input.nextDouble();
        System.out.print("Enter value of e : ");
        double e = input.nextDouble();

        System.out.println("Values from Equation:- 2 :");
        System.out.print("Enter value of c : ");
        double c = input.nextDouble();
        System.out.print("Enter value of d : ");
        double d = input.nextDouble();
        System.out.print("Enter value of f : ");
        double f = input.nextDouble();

        double x = ((e * d) - (b * f)) / ((a * d) - (b * c));
        double y = ((a * f) - (e * c)) / ((a * d) - (b * c));

        System.out.println(" X = " + x + " Y = " + y);
    }
}
```

```
}
```

Output:

```
Values from Equestion:- 1 :  
Enter value of a : 1  
Enter value of b : 2  
Enter value of e : 3  
Values from Equestion:- 2 :  
Enter value of c : 4  
Enter value of d : 5  
Enter value of f : 6  
X = -1.0 Y = 2.0  
  
Process finished with exit code 0
```

4. Write a program that reads a number in meters, converts it to feet, and displays the result.

Code:

```
import java.util.Scanner;  
  
public class Main {  
    public static void main(String[] args) {  
        System.out.println("220130107094 - Megh Patel");  
        System.out.println("Date: 25-02-24");  
        System.out.println("Time: 12:30 PM");  
        Scanner input = new Scanner(System.in);  
        System.out.print("Enter Value in Meters :");  
        double meter = input.nextDouble();  
        double feet = meter * 3.28084;  
        System.out.print(meter + " Meters = " + feet + " Feets");  
    }  
}
```

Output:

```
Enter Value in Meters : 12  
12.0 Meters = 39.37008 Feets  
Process finished with exit code 0
```

5. Body Mass Index (BMI) is a measure of health on weight. It can be calculated by taking your weight in kilograms and dividing it by the square of your height in meters. Write a

program that prompts the user to enter weight in pounds and height in inches and displays the BMI. Note:- 1 pound=.45359237 Kg and 1 inch=.0254 meters.

Code:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 25-02-24");
        System.out.println("Time: 01:00 PM");
        Scanner input = new Scanner(System.in);
        System.out.print("Enter Your weight in Pound :");
        double pound = input.nextDouble();
        System.out.print("Enter Your Height in Inch :");
        double inch = input.nextDouble();
        double BMI = (pound * 0.45359237) / ((inch * 0.0254) * (inch * 0.0254));
        System.out.print("BMI = " + BMI);
    }
}
```

Output:

```
Enter Your weight in Pound : 100
Enter Your Height in Inch : 70
BMI = 14.348358768146106
Process finished with exit code 0
```

6. Write a program that prompts the user to enter three integers and display the integers in decreasing order.

Code:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 25-02-24");
        System.out.println("Time: 01:05 PM");

        Scanner input = new Scanner(System.in);
        System.out.print("Enter 1st Integer :");
        int a = input.nextInt();
        System.out.print("Enter 2nd Integer :");
        int b = input.nextInt();
        System.out.print("Enter 3rd Integer :");
        int c = input.nextInt();
        if(a > b){
            if(a > c){
```

```

        System.out.println("Decreasing Order : " + a + " " + c + " " + b);
    }else{
        System.out.println("Decreasing Order : " + c + " " + a + " " + b);
    }
}else{
    if(b>c){
        System.out.println("Decreasing Order : " + b + " " + c + " " + a);
    }else{
        System.out.println("Decreasing Order : " + c + " " + b + " " + a);
    }
}
}
}
}

```

Output:

```

Enter 1st Integer : 1
Enter 2nd Integer : 2
Enter 3rd Integer : 3
Decreasing Order : 3 2 1

Process finished with exit code 0

```

7. Write a program that prompts the user to enter a letter and check whether a letter is a vowel or constant.

Code:

```

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 25-02-24");
        System.out.println("Time: 01:15 PM");

        Scanner input = new Scanner(System.in);
        System.out.print("Enter Character : ");
        char ch = input.next().charAt(0);
        switch (Character.toLowerCase(ch)) {
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u':
                System.out.print(ch + " is vowel");
                break;
            default:
                System.out.print(ch + " is consonent");
        }
    }
}

```

```

    }
}
}

```

Output:

```

Enter Character : M
M is consonent
Process finished with exit code 0

```

Additional programs:

8. A cashier has currency notes of denominations 1, 2, 5, 10, 50 and 100. If the amount to be withdrawn is input through the keyboard, find the total number of currency notes of each denomination the cashier will have to give to the withdrawer.

Code:

```

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 25-02-24");
        System.out.println("Time: 08:00 PM");
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the amount to be withdrawn : ");
        int amt = input.nextInt();
        int hundred = amt/100;
        amt = amt%100;
        int fifty = amt/50;
        amt = amt%50;
        int ten = amt/10;
        amt = amt%10;
        int fifer = amt/5;
        amt = amt%5;
        int two = amt/2;
        amt = amt%2;
        int one = amt;
        System.out.println("Number of Hundred Notes : " +hundred);
        System.out.println("Number of Fifty Notes : " +fifty);
        System.out.println("Number of Ten Notes : " +ten);
        System.out.println("Number of Five Notes : " +fifer);
        System.out.println("Number of Two Notes : " +two);
        System.out.println("Number of Onw Notes : " +one);
    }
}

```

Output:

```
Enter the amount to be withdrawn : 718
Number of Hundred Notes : 7
Number of Fifty Notes : 0
Number of Ten Notes : 1
Number of Five Notes : 1
Number of Two Notes : 1
Number of Onw Notes : 1

Process finished with exit code 0
```

9. If a five-digit number is input through the keyboard, write a program to print a new number by adding one to each of its digits. For example, if the number that is input is 12391 then the output should be displayed as 23502.

Code:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 25-02-24");
        System.out.println("Time: 08:14 PM");
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the Five digits Number : ");
        int num = input.nextInt();
        System.out.println("Original Digits : "+num);
        int a,sum = 0;
        a = (num/10000) + 1;
        num = num%10000;
        sum = sum + (a*10000);

        a = (num/1000) + 1;
        num = num%1000;
        sum = sum + (a*1000);

        a = (num/100) + 1;
        num = num%100;
        sum = sum + (a*100);

        a = (num/10) + 1;
        num = num%10;
        sum = sum + (a*10);

        a = num + 1;
        sum = sum + a;
    }
}
```



```

        System.out.print("After Adding One Digits to Each Bit Answer is : "+sum);
    }
}

```

Output:

```

Enter the Five digits Number : 23502
Original Digits : 23502
After Adding One Digits to Each Bit Answer is : 34613
Process finished with exit code 0

```

10. If lengths of three sides of a triangle are input through the keyboard, write a program to print the area of the triangle.

Code:

```

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 25-02-24");
        System.out.println("Time: 08:20 PM");
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the length of 1st side : ");
        double a = scanner.nextDouble();

        System.out.print("Enter the length of 2nt side : ");
        double b = scanner.nextDouble();

        System.out.print("Enter the length of 3rt side : ");
        double c = scanner.nextDouble();

        double s = (a + b + c)/2;
        double area = Math.sqrt(s*(s-a)*(s-b)*(s-c));
        System.out.println("the area of the triangle is : "+area);
    }
}

```

Output:

```
Enter the length of 1st side : 2
Enter the length of 2nt side : 2
Enter the length of 3rt side : 3
the area of the triangle is : 1.984313483298443

Process finished with exit code 0
```

11. Write a program to produce the following patterns.

****	1234
***	123
**	12
*	1
1234	*
567	***
89	*****
0	*****

	*

Code – 1:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 25-02-24");
        System.out.println("Time: 08:25 PM");

        Scanner sc = new Scanner(System.in);

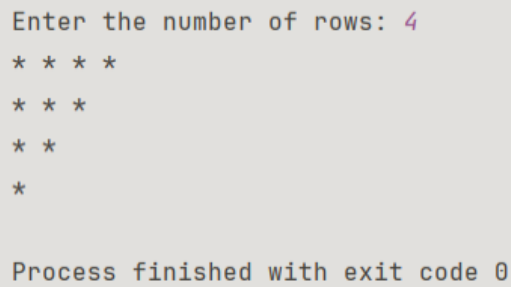
        System.out.print("Enter the number of rows: ");

        int rows = sc.nextInt();

        for (int i= rows-1; i>=0 ; i--)
        {
            for (int j=0; j<=i; j++)
            {
                System.out.print("*" + " ");
            }
        }
    }
}
```

```
        }  
        System.out.println();  
    }  
    sc.close();  
}  
}
```

Output – 1:



```
Enter the number of rows: 4  
* * * *  
* * *  
* *  
*  
  
Process finished with exit code 0
```

Code – 2:

```
import java.util.Scanner;  
  
public class Main {  
    public static void main(String[] args) {  
        System.out.println("220130107094 - Megh Patel");  
        System.out.println("Date: 25-02-24");  
        System.out.println("Time: 08:34 PM");  
  
        Scanner sc = new Scanner(System.in);  
  
        System.out.print("Enter the number of n: ");  
  
        int n = sc.nextInt();  
  
        for (int i = n; i >= 1; i--) {  
            for (int j = 1; j <= i; j++) {  
                System.out.print(j + " ");  
            }  
            System.out.println();  
        }  
    }  
}
```

Output – 2:

```
Enter the number of n: 4
1 2 3 4
1 2 3
1 2
1
Process finished with exit code 0
```

Code – 3:

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 25-02-24");
        System.out.println("Time: 08:36 PM");

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of n: ");

        int n = sc.nextInt();
        int cnt = 1;
        for (int i = n; i >= 1; i--) {
            for (int j = 1; j <= i; j++) {
                System.out.print(cnt + " ");
                cnt++;
            }
            System.out.println();
        }
    }
}
```

Output – 3:

```
Enter the number of n: 4
1 2 3 4
5 6 7
8 9
10
Process finished with exit code 0
```

Code – 4:

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

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("220130107094 - Megh Patel");  
        System.out.println("Date: 25-02-24");  
        System.out.println("Time: 08:40 PM");  
  
        int n, x, j, blank = 1;  
  
        System.out.print("Enter the value for rows: ");  
  
        Scanner s = new Scanner(System.in);  
  
        n = s.nextInt();  
        blank = n - 1;  
        for (j = 1; j <= n; j++) {  
            for (x = 1; x <= blank; x++) {  
                System.out.print(" ");  
            }  
            blank--;  
  
            for (x = 1; x <= 2 * j - 1; x++) {  
                System.out.print("*");  
            }  
            System.out.println("");  
        }  
        blank = 1;  
        for (j = 1; j <= n - 1; j++) {  
            for (x = 1; x <= blank; x++) {  
                System.out.print(" ");  
            }  
            blank++;  
            for (x = 1; x <= 2 * (n - j) - 1; x++) {  
                System.out.print("*");  
            }  
            System.out.println("");  
        }  
    }  
}
```

Output – 4:

```
Enter the value for rows: 4
  *
 ***
*****
*****
 *****
  ***
   *
```

Process finished with exit code 0

Conclusion:

Quiz: (Sufficient space to be provided for the answers)

1. What is the primary purpose of Java??
2. What is the main method in Java used for?
3. How Java Language is Platform Independent?
4. What is JVM and JRE?

Suggested Reference:

1. <https://www.tutorialspoint.com/java/>
2. <https://www.geeksforgeeks.org/>
3. <https://www.w3schools.com/java/>
4. <https://www.javatpoint.com/>

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Criteria	Need Improvement	Good	Excellent	Total
Marks	Design of logic (4) Correct output (4) Mock viva test (2)	Program has significant logic errors. (1) Output has multiple errors. (1) Delayed & only few correct answers (1)	Program has slight logic errors that do not significantly affect the results (2) Output has minor errors. (2) Partially correct response (1)	Program is logically well designed (3) Program displays correct output with no errors (3) All questions responded Correctly (2)	

Signature of Faculty:

Experiment No: 2

AIM: To learn Arrays and Strings in Java.

Date:

CO mapped: CO-1

Objectives:

- a) Array manipulation: Learn how to create, populate, access, and modify arrays in Java.
- b) String manipulation: Understand how to create and manipulate strings, including concatenation, comparison, and extraction of substrings.
- c) Array and String methods: Explore common array and string methods available in Java's standard library.

Background:

Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.

Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on the 1st index, and so on.

There are two types of array.

- Single Dimensional Array
- Multidimensional Array

In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string. For example:

```
char[] ch={'j','a','v','a','t','p','o','i','n','t'};
```

```
String s=new String(ch);
```

is same as:

```
String s="javatpoint";
```

Java String class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

Practical questions:

1. Write a program that generate 6*6 two-dimensional matrix, filled with 0's and 1's , display the matrix, check every row and column have an odd number's of 1's.

Code:

```
import java.sql.SQLOutput;
```

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

```
public class Main {
```

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



```
System.out.println("220130107094 - Megh Patel");

System.out.println("Date: 26-02-24");

System.out.println("time: 08:55 PM");

int [][] numbers = {

    {0,1,0,1,0,1},

    {1,0,1,0,1,0},

    {0,1,0,1,0,1},

    {1,0,1,0,1,0},

    {0,1,0,1,0,1},

    {1,0,1,0,1,0}

};

for (int i=0;i<6;i++){

    for(int j=0;j<6;j++){

        System.out.print(" " + numbers[i][j] + " ");

    } System.out.println();

}

}
```

Output:

```
0 1 0 1 0 1
1 0 1 0 1 0
0 1 0 1 0 1
1 0 1 0 1 0
0 1 0 1 0 1
1 0 1 0 1 0

Process finished with exit code 0
```

2. Write a generic method that returns the minimum elements and their indices in a two dimensional array.

Code:

```

import java.util.*;

class Hello{
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 26-02-24");
        System.out.println("Time: 09:30 PM");

        Scanner input = new Scanner(System.in);
        int a[][] = new int[3][3];
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                a[i][j] = input.nextInt();
            }
        }

        System.out.println("Your enter array is: ");
        for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a.length; j++) {
                System.out.print(" "+a[i][j]+" ");
            }System.out.println();
        }
        FindMin(a,3);
    }
    static void FindMin(int a[], int n){
        int min = a[0][0];
        int x = 0,y = 0;
        for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a.length; j++) {
                if(min >= a[i][j]){
                    min = a[i][j];
                    x = i;
                    y = j;
                }
            }
        }
        System.out.println(+min+" is the min element in the array at the positions " + x + " " + y);
    }
}

```

Output:

```

1 2 3 4 5 6 7 8 9
Your enter array is:
1 2 3
4 5 6
7 8 9
1 is the min element in the array at
the positions 0 0
PS D:\Programmes\Java> 

```

3. Write a method that returns a new array by eliminating the duplicate values in the array.

Code:

```
import java.util.*;

class Hello{
public static void main(String[] args) {
    System.out.println("220130107094 - Megh Patel");
    System.out.println("Date: 27-02-24");
    System.out.println("Time: 09:30 AM");

    Scanner input = new Scanner(System.in);
    int a[] = new int[10];
    for(int i=0;i<10;i++){
        a[i] = input.nextInt();
    }

    System.out.println("Your enter array is: ");
    for (int i = 0; i < 10; i++) {
        System.out.print(a[i] + " ");
    }System.out.println();
    // FindMin(a,3);
    RemoveDuplicate(a,10);
}

static void RemoveDuplicate(int a[],int n){
    for(int i=0;i<n-1;i++){
        for(int j=i+1;j<n;j++){
            if(a[i] == a[j]){
                a[j] = -1;
            }
        }
    }
    for(int i=0;i<n;i++){
        if(a[i] != -1){
            System.out.print(a[i] + " ");
        }
    }System.out.println();
}
}
```

Output:

```
1 22 2 3 2 4 6 5 4 1
Your enter array is:
1 22 2 3 2 4 6 5 4 1
1 22 2 3 4 6 5
PS D:\Programmes\Java>
```

4. Write a program to add, subtract or multiply two 3*3 integer arrays as per choice of user.

Sample Input:

Array 1:

1 2 3

4 5 6

7 8 9

Array 2:

5 6 7

1 2 0

4 3 2

Symbol: +

Sample Output:

6 8 10

5 7 6

11 11 11

Code:

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

```
class ArrayOp {  
    private int[][] array;
```

```
  
    public ArrayOperations(int[][] array) {  
        this.array = array;  
    }
```

```
  
    public int[][] add(int[][] otherArray) {  
        int[][] result = new int[3][3];  
        for (int i = 0; i < 3; i++) {  
            for (int j = 0; j < 3; j++) {  
                result[i][j] = this.array[i][j] + otherArray[i][j];  
            }  
        }  
        return result;  
    }
```

```
  
    public int[][] subtract(int[][] otherArray) {  
        int[][] result = new int[3][3];  
        for (int i = 0; i < 3; i++) {  
            for (int j = 0; j < 3; j++) {  
                result[i][j] = this.array[i][j] - otherArray[i][j];  
            }  
        }  
        return result;  
    }
```

```
  
    public int[][] multiply(int[][] otherArray) {  
        int[][] result = new int[3][3];  
        for (int i = 0; i < 3; i++) {  
            for (int j = 0; j < 3; j++) {
```

```
        for (int k = 0; k < 3; k++) {
            result[i][j] += this.array[i][k] * otherArray[k][j];
        }
    }
}
return result;
}

public void display() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            System.out.print(this.array[i][j] + " ");
        }
        System.out.println();
    }
}

}

public class Main {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 27/02/24");
        System.out.println("Time: 10:30 AM");
        System.out.println();

        Scanner scanner = new Scanner(System.in);
        int[][] array1 = new int[3][3];
        int[][] array2 = new int[3][3];

        System.out.println("Enter elements of first 3x3 array:");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                array1[i][j] = scanner.nextInt();
            }
        }

        System.out.println("Enter elements of second 3x3 array:");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                array2[i][j] = scanner.nextInt();
            }
        }

        ArrayOperations arrayOp1 = new ArrayOperations(array1);
        ArrayOperations arrayOp2 = new ArrayOperations(array2);

        System.out.println("Choose operation:");
        System.out.println("1. Add");
        System.out.println("2. Subtract");
        System.out.println("3. Multiply");
        int choice = scanner.nextInt();

        switch (choice) {
```

```

        case 1:
            int[][] additionResult = arrayOp1.add(array2);
            System.out.println("Addition result:");
            new ArrayOperations(additionResult).display();
            break;
        case 2:
            int[][] subtractionResult = arrayOp1.subtract(array2);
            System.out.println("Subtraction result:");
            new ArrayOperations(subtractionResult).display();
            break;
        case 3:
            int[][] multiplicationResult = arrayOp1.multiply(array2);
            System.out.println("Multiplication result:");
            new ArrayOperations(multiplicationResult).display();
            break;
        default:
            System.out.println("Invalid choice!");
    }

    scanner.close();
}

```

Output:

```

Enter elements of first 3x3 array:
1 2 3 4 5 6 7 8 9
Enter elements of second 3x3 array:
1 2 3 4 5 6 7 8 9
Choose operation:
1. Add
2. Subtract
3. Multiply
3
Multiplication result:
30 36 42
66 81 96
102 126 150
PS D:\Programmes\Java\OOP>

```

5. Write a program to sort an array of 10 elements using selection sort.

Code:

```

import java.util.*;

class Hello{
    public static void main(String args[]) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 27-02-24");
        System.out.println("Time: 12:00 PM");

        Scanner input = new Scanner(System.in);

        System.out.println("Enter the elements of array: ");
    }
}

```

```
int[] arr = new int[10];
for(int i=0;i<10;i++){
    arr[i] = input.nextInt();
}

System.out.println("Unsorted Array: ");
for(int i=0;i<10;i++){
    System.out.print(arr[i] + " ");
}System.out.println();

SelectionSort(arr);
// FindMin(a,3);
// RemoveDuplicate(a,10);
}

public static void SelectionSort(int[] arr) {
    for (int i = 0; i <10-1; i++) {
        int minIndex = i;
        for (int j = i + 1; j <10; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        int temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }

    System.out.println("Sorted Array: ");
    for(int i=0;i<10;i++){
        System.out.print(arr[i] + " ");
    }System.out.println();
}
}
```

Output:

```
Enter the elements of array:
2 4 5 6 6 8 45 18 7 9
Unsorted Array:
2 4 5 6 6 8 45 18 7 9
Sorted Array:
2 4 5 6 6 7 8 9 18 45
PS D:\Programmes\Java> █
```

6. Write a program that prompts the user to enter a string and displays the number of vowels and consonants in the string.

Code:

```
import java.util.*;

class Hello{
    public static void main(String args[]) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 27-02-24");
        System.out.println("Time: 12:45 PM");

        String name;
        int cout =0;
        int cout1=0;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the string: ");
        name = sc.nextLine();
        name=name.toUpperCase();
        for ( int i=0;i<name.length();i++)
        {
            char ch = name.charAt(i);
            if(ch=='A' || ch=='E' || ch=='I' || ch=='O' || ch=='U')
                cout++;
            else if ( Character.isLetter(ch))
                cout1++;
        }
        System.out.println("Number of the vowel is: "+cout+" and number of constant is: "
+cout1);
        // FindMin(a,3);
        // RemoveDuplicate(a,10);
    }
}
```

Output:

```
Enter the string: Megh
Number of the vowel is: 1 and number of constant is: 3
PS D:\Programmes\Java> □
```

7. Write a program that prompts the user to enter two strings and displays the largest common prefix of the two strings.

Code:

```
import java.util.*;

class Hello{
    public static void main(String args[]) {
```



```
System.out.println("220130107094 - Megh Patel");
System.out.println("Date: 28-02-24");
System.out.println("Time: 04:44 PM");

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first string: ");
String str1 = scanner.nextLine();

System.out.print("Enter the second string: ");
String str2 = scanner.nextLine();

int minLength = Math.min(str1.length(), str2.length());

StringBuilder commonPrefix = new StringBuilder();

for (int i = 0; i < minLength; i++) {
    if (str1.charAt(i) == str2.charAt(i)) {
        commonPrefix.append(str1.charAt(i));
    } else {
        break;
    }
}

if (commonPrefix.length() > 0) {
    System.out.println("Largest common prefix: " + commonPrefix.toString());
} else {
    System.out.println("There is no common prefix.");
}
}
```

Output:

```
Enter the first string: Megh
Enter the second string: Megha
Largest common prefix: Megh
PS D:\Programmes\Java> 
```

8. Some websites impose certain rules for passwords. Write a method that checks whether a string is a valid password. Suppose the password rules are as follows: A password must have at least eight characters. A password consists of only letters and digits. A password must contain at least two digits. Write a program that prompts the user to enter a password and displays Valid Password if the rules are followed or Invalid Password otherwise.

Code:

```
import java.util.*;

class Hello{
    public static void main(String args[]) {
        System.out.println("220130107094 - Megh Patel");
    }
}
```

```
System.out.println("Date: 28-02-24");
System.out.println("Time: 05:44 PM");

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a password: ");
String password = scanner.nextLine();

if (isValidPassword(password)) {
    System.out.println("Valid Password");
} else {
    System.out.println("Invalid Password");
}

scanner.close();
}

public static boolean isValidPassword(String password) {
    if (password.length() < 8) {
        return false;
    }

    for (int i = 0; i < password.length(); i++) {
        char ch = password.charAt(i);
        if (!Character.isLetterOrDigit(ch)) {
            return false;
        }
    }

    int digitCount = 0;
    for (int i = 0; i < password.length(); i++) {
        char ch = password.charAt(i);
        if (Character.isDigit(ch)) {
            digitCount++;
        }
    }

    return digitCount >= 2;
}
}
```

Output:

```
Enter a password: Megh82Patel
Valid Password
PS D:\Programmes\Java> □
```

Conclusion:

Quiz: (Sufficient space to be provided for the answers)

4. What are ragged arrays in java and how are they implemented?
5. Differentiate String class and StringBuffer class.
6. How Create a two dimensional array. Instantiate and Initialize it?
7. Explain the various String functions with their syntax.

Suggested Reference:

1. <https://www.tutorialspoint.com/java/>
2. <https://www.geeksforgeeks.org/>
3. <https://www.w3schools.com/java/>
4. <https://www.javatpoint.com/>

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Criteria	Need Improvement	Good	Excellent	Total
Marks	Design of logic (4) Correct output (4) Mock viva test (2)	Program has significant logic errors. (1) Output has multiple errors. (1) Delayed & only few correct answers (1)	Program has slight logic errors that do no significantly affect the results (2) Output has minor errors. (2) Partially correct response (1)	Program is logically well designed (3) Program displays correct output with no errors (3) All questions responded Correctly (2)	

Signature of Faculty:

Experiment No: 3

AIM: To implement basic object-oriented concepts.

Date:

CO mapped: CO-2

Objectives:

- a) To apply fundamental object-oriented principles, such as class design, encapsulation, inheritance, and polymorphism, to improve software modularity, code organization, and maintainability.
- b) Implementing these basic object-oriented concepts in your software development practices will help you create more structured, maintainable, and reusable code, which is essential for building robust and scalable software systems.

Background:

Object means a real-world entity such as a pen, chair, table, computer, watch, etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

- **Object:** Any entity that has a state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical. An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.
- **Class:** Collection of objects is called class. It is a logical entity. A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.
- **Inheritance:** When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.
- **Polymorphism:** If one task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc. In Java, we use method overloading and method overriding to achieve polymorphism. Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.
- **Abstraction:** Hiding internal details and showing functionality is known as abstraction. For example, phone call, we don't know the internal processing. In Java, we use abstract class and interface to achieve abstraction.
- **Encapsulation:** Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines. A java class is an example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Practical questions:

1. Write a Java application which takes several command line arguments, which are supposed to be names of students and prints output as given below: (Suppose we enter 3 names then output should be as follows):

Number of arguments = 3

1: First Student Name is =Tom

2: Second Student Name is =Dick

3: Third Student Name is =Harry

(Hint: An array may be used for converting from numeric values from 1 to 20 into String.)

Code:

```
import java.util.*;

class main31{

    public static void main(String[] args) {

        System.out.println("Megh Patel - 220130107094");

        System.out.println("Date: 29 - 02 - 2024");

        System.out.println("Time: 04:00 PM");

        Scanner input = new Scanner(System.in);

        System.out.println();

        System.out.print("Enter the No. of arguments you want to give: ");

        int Num = input.nextInt();

        String[] str1 = new String[Num];

        for (int i = 0; i < Num; i++) {

            System.out.print("Enter your " + (i+1) + " arguments: ");

            str1[i] = input.next();

        }System.out.println();

        System.out.println("Number of arguments entered by user: " + Num);

        for (int i = 0; i < Num; i++) {

            System.out.println("Your " + (i+1) + " argument is: "+ str1[i]);
```

```
}  
  
}  
  
}
```

Output:

```
Enter the No. of arguments you want to give: 3  
Enter your 1 arguments: Tom  
Enter your 2 arguments: Dick  
Enter your 3 arguments: Harry  
  
Number of arguments entered by user: 3  
Your 1 argument is: Tom  
Your 2 argument is: Dick  
Your 3 argument is: Harry  
PS D:\Programmes\Java\Practicle - 3> █
```

2. Design a class named Rectangle to represent a rectangle. The class contains: Two double data fields named width and height that specify the width and height of the rectangle. The default values are 1 for both width and height.

A no-arg constructor that creates a default rectangle.

A constructor that creates a rectangle with the specified width and height.

A method named getArea() that returns the area of this rectangle.

A method named getPerimeter() that returns the perimeter.

Write a test program that creates two Rectangle objects—one with width 4 and height 40 and the other with width 3.5 and height 35.9. Display the width, height, area, and perimeter of each rectangle in this order.

Code:

```
public class Rectangle {  
    private double width;  
    private double height;  
  
    public Rectangle() {  
        this.width = 1.0;  
        this.height = 1.0;
```

```
}

public Rectangle(double width, double height) {

    this.width = width;

    this.height = height;

}

public double getWidth() {

    return width;

}

public void setWidth(double width) {

    this.width = width;

}

public double getHeight() {

    return height;

}

public void setHeight(double height) {

    this.height = height;

}

public double getArea() {

    return width * height;

}

public double getPerimeter() {

    return 2 * (width + height);

}


public static void main(String[] args) {

    System.out.println("Megh Patel - 220130107094");

    System.out.println("Date: 29-02-2024");

}
```

```
System.out.println("time: 04:44 PM");

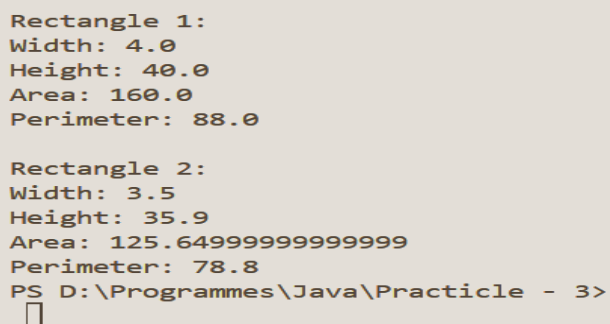
System.out.println();

Rectangle rectangle1 = new Rectangle(4, 40);
Rectangle rectangle2 = new Rectangle(3.5, 35.9);

System.out.println("Rectangle 1:");
System.out.println("Width: " + rectangle1.getWidth());
System.out.println("Height: " + rectangle1.getHeight());
System.out.println("Area: " + rectangle1.getArea());
System.out.println("Perimeter: " + rectangle1.getPerimeter());

System.out.println("\nRectangle 2:");
System.out.println("Width: " + rectangle2.getWidth());
System.out.println("Height: " + rectangle2.getHeight());
System.out.println("Area: " + rectangle2.getArea());
System.out.println("Perimeter: " + rectangle2.getPerimeter());
}
}
```

Output:



```
Rectangle 1:
Width: 4.0
Height: 40.0
Area: 160.0
Perimeter: 88.0

Rectangle 2:
Width: 3.5
Height: 35.9
Area: 125.64999999999999
Perimeter: 78.8
PS D:\Programmes\Java\Practicle - 3>
```

3. Define a class called Cartesian Point, which has two instance variables, x and y. Provide the

methods get X() and get Y() to return the values of the x and y values respectively, a method called move() which would take two integers as parameters and change the values of x and y respectively, a method called display() which would display the current values of x and y. Now overload the method move() to work with single parameter, which would set both x and y to the same values, provide constructors with two parameters and overload to work with one parameter as well. Now define a class called Test Cartesian Point, with the main method to test the various methods in the Cartesian Point class.

Code:

```
public class CartesianPoint {
    private int x;
    private int y;

    public CartesianPoint(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public CartesianPoint(int value) {
        this.x = value;
        this.y = value;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public void move(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public void move(int value) {
        this.x = value;
        this.y = value;
    }

    public void display() {
        System.out.println("Current values: x = " + x + ", y = " + y);
    }

    public static void main(String[] args) {

        System.out.println("Megh Patel - 220130107094");
        System.out.println("Date: 29-02-2024");
    }
}
```

```
System.out.println("time: 05:04 PM");
System.out.println();

CartesianPoint point1 = new CartesianPoint(5, 10);
CartesianPoint point2 = new CartesianPoint(3);

System.out.println("Point 1:");
point1.display();

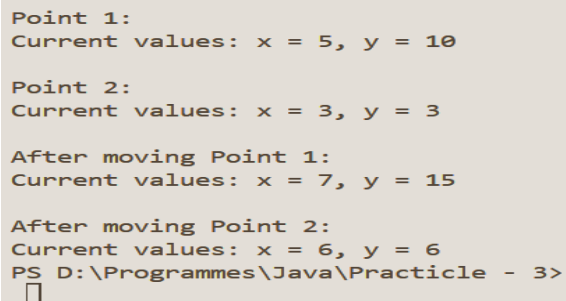
System.out.println("\nPoint 2:");
point2.display();

point1.move(7, 15);
System.out.println("\nAfter moving Point 1:");
point1.display();

point2.move(6);
System.out.println("\nAfter moving Point 2:");
point2.display();
}

}
```

Output:



```
Point 1:
Current values: x = 5, y = 10

Point 2:
Current values: x = 3, y = 3

After moving Point 1:
Current values: x = 7, y = 15

After moving Point 2:
Current values: x = 6, y = 6
PS D:\Programmes\Java\Practicle - 3>
```

4. Create a class Employee which has two private data members name and salary and it has two public member functions named as getData() and putData() where getData() gets name and salary from the user putData() displays name and salary for any user.

Code:

```
import java.util.Scanner;

public class Employee {
    private String name;
    private double salary;

    public void getData(){
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Employee Name: ");
        name = scanner.nextLine();
        System.out.print("Enter Salary "+ name + ": ");
    }
}
```

```

        salary = scanner.nextDouble();
    }

    public void putData(){
        System.out.println();
        System.out.println("Employee Name: "+ name);
        System.out.println("Employee Salary: "+ salary);
    }

    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 07/03/2024");
        System.out.println("Time: 04:21 PM");
        System.out.println();
        Employee e = new Employee();
        e.getData();
        e.putData();
    }
}

```

Output:

```

Enter Employee Name: Megh
Enter Salary Megh: 10000000

Employee Name: Megh
Employee Salary: 1.0E7
PS D:\Programmes\Java\Practicle - 3>

```

5. Define a class Time with hours and minutes as two data members, add necessary member functions to initialize and display data of class. Do not use constructors in a class. Define a member function sum () which adds two Time objects. (Use the statements like T3.sum (T1, T2)).

Code:

```

public class time_prac_5{
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 07/03/2024");
        System.out.println("Time: 04:30 PM");
        System.out.println();

        Time t1 = new Time();
        t1.setData(4,20);
        t1.Display();

        Time t2 = new Time();
        t2.setData(16,50);
        t2.Display();

        Time t3 = new Time();
        t3.Sum(t1, t2);
        t3.Display();
    }
}

```

```

    }

    class Time{
        private int hour = 0;
        private int minutes = 0;
        public void setData(int h, int m){
            this.hour = h;
            this.minutes = m;
        }
        public int getHour(){
            return this.hour;
        }
        public int getMinutes(){
            return this.minutes;
        }
        public void Display(){
            System.out.println("Hour : "+ this.hour + " Minutes : "+ this.minutes);
        }

        public void Sum(Time t1, Time t2){

            this.hour = t1.getHour() + t2.getHour();
            this.minutes = t1.getMinutes() + t2.getMinutes();
            hour = hour + minutes/60;
            minutes = minutes%60;
        }
    }
}

```

Output:

```

Hour : 4 Minutes : 20
Hour : 16 Minutes : 50
Hour : 21 Minutes : 10

```

6. Define Class named Point which represents 2-D Point, i.e P (x, y). Define Default constructor to initialize both data member value 5, Parameterized constructor to initialize member according to value supplied by user and Copy Constructor. Define Necessary Function and Write a program to test class Point.

Code:

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

```

public class Point {
    private int x;
    private int y;

    public Point(){
        x = 5;
        y = 5;
    }
}

```

```
public Point(int x,int y){
    this.x = x;
    this.y = y;
}

public Point(Point oth){
    this.x = oth.x;
    this.y = oth.y;
}

public int getX(){
    return x;
}
public int getY(){
    return y;
}

public void display(){
    System.out.println("Point: (" +x+" , "+y+" )");
}

public static void main(String[] args) {
    System.out.println("220130107094 - Megh Patel");
    System.out.println("Date: 07/03/2024");
    System.out.println("Time: 04:45 PM");
    System.out.println();

    Point defaultPoint = new Point();
    System.out.print("Default Point: ");
    defaultPoint.display();
    System.out.println();

    Point parameterizedPoint = new Point(5, 5);
    System.out.print("Parameterized Point: ");
    parameterizedPoint.display();
    System.out.println();

    Point copyPoint = new Point(parameterizedPoint);
    System.out.print("Copy Point: ");
    copyPoint.display();
    System.out.println();
}
}
```

Output:

```
Default Point: Point: (5 , 5)

Parameterized Point: Point: (5 , 5)

Copy Point: Point: (5 , 5)
```

7. Create a class Account. It has three data member account id, name and balance. Define function to assign value and display value. Define function that search account number

given by the user. If account number exists, print detail of that account. Write a program using array of object. Declare at least 5 account and print details

Code:

```
import java.util.Scanner;

public class account_test_7{
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 07/03/2024");
        System.out.println("Time: 05:00 PM");
        System.out.println();

        Scanner scanner = new Scanner(System.in);
        int id1;
        String name1;
        double balance1;
        System.out.print("Enter Number of Account holders : ");
        int number = scanner.nextInt();
        Account[] a;
        a = new Account[number];
        // Account[] a = new Account[3];
        for(int i = 0; i<number;i++){
            a[i] = new Account();
            System.out.printf("For Customer " + (i+1) + "\n");
            System.out.print("\tEnter id : ");
            id1 = scanner.nextInt();
            System.out.print("\tEnter name : ");
            name1 = scanner.next();
            System.out.print("\tEnter balance : ");
            balance1 = scanner.nextDouble();
            a[i].setData(id1, name1, balance1);
        }
        System.out.print("Enter the Id you want to search : ");
        int search_id = scanner.nextInt();

        for(int i = 0; i<number;i++){
            System.out.print("\n");
            if(a[i].getId()==search_id){
                a[i].getData();
            }
        }
    }
}

class Account{
    private int id;
    private String name;
    private double balance;
```

```

    public void setData(int id1, String name1, double balance1){
        this.id = id1;
        this.name = name1;
        this.balance = balance1;
    }
    public int getId(){
        return id;
    }

    public void getData(){
        System.out.println("Id : "+ this.id + ", \nName : "+ this.name+ ", \nBalance : "+
this.balance);
    }
}

```

Output:

```

Enter Number of Account holders : 1
For Customer 1
    Enter id : 1
    Enter name : Megh
    Enter balance : 10000000
Enter the Id you want to search : 1

Id : 1,
Name : Megh,
Balance : 1.0E7

```

Conclusion:

Quiz: (Sufficient space to be provided for the answers)

1. Explain the concept of encapsulation and why it is important in OOP.
2. What is a class, and how does it relate to objects in OOP?
3. Define and explain static and dynamic binding.
4. Explain the concept of method overloading and method overriding in OOP.

Suggested Reference:

1. <https://www.tutorialspoint.com/java/>
2. <https://www.geeksforgeeks.org/>
3. <https://www.w3schools.com/java/>
4. <https://www.javatpoint.com/>

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Criteria	Need Improvement	Good	Excellent	Total

Marks	Design of logic (4) Correct output (4) Mock viva test (2)	Program has significant logic errors. (1) Output has multiple errors. (1) Delayed & only few correct answers (1)	Program has slight logic errors that do not significantly affect the results (2) Output has minor errors. (2) Partially correct response (1)	Program is logically well designed (3) Program displays correct output with no errors (3) All questions responded Correctly (2)	
--------------	--	--	--	---	--

Signature of Faculty:

Experiment No: 4

AIM: To implement inheritance and object-oriented concepts.

Date:

CO mapped: CO-2

Objectives:

- a) To master the fundamental principles of inheritance and object-oriented concepts, enabling the design and development of efficient, maintainable, and scalable software solutions by leveraging the power of class hierarchies and code reuse.
- b) Implementing these basic object-oriented concepts in your software development practices will help you create more structured, maintainable, and reusable code, which is essential for building robust and scalable software systems.

Background:

Object means a real-world entity such as a pen, chair, table, computer, watch, etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system). The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Terms used in Inheritance

- **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- **Sub Class/Child Class:** Subclass is a class that inherits the other class. It is also called a derived class, extended class, or child class.
- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
- **Reusability:** As the name specifies, reusability is a mechanism that facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

Practical questions:

1. A set of 5 words (strings) will be taken as command line arguments. Write a program to reverse each word and check whether it is palindrome or not using method.

Code:

```
public class Strings{
```

```
public static boolean isPalindrome(String word) {  
  
    int left = 0;  
  
    int right = word.length() - 1;  
  
    while (left < right) {  
  
        if(Character.toLowerCase(word.charAt(left))!=Character.toLowerCase(word.charAt(right))  
        {  
  
            return false;  
  
        }  
  
        left++;  
  
        right--;  
  
    }  
  
    return true;  
}  
  
public static void main(String[] args) {  
  
    System.out.println("Megh Patel - 220130107094");  
  
    System.out.println("Date: 14/03/24");  
  
    System.out.println("Time: 03:25 PM");  
  
    for (String word : args) {  
  
        String reversedWord = new StringBuilder(word).reverse().toString();  
  
        System.out.println(word + " -> " + reversedWord + (isPalindrome(word) ? "  
(Palindrome)" : ""));  
  
    }  
  
}
```

Output:

```
megh -> hgem  
meem -> meem (Palindrome)  
jenil -> linej  
indra -> ardni  
neem -> meen  
PS D:\Programmes\Java\Practicle - 4>
```

2. Define the class BankAccount to represent an account we open with bank. Define the subclasses SavingAccount and FixedDepositAccount. Implement the operations like openAccount(), deposit(), checkBalance(), withdraw() and calInterest() for these classes.

Code;

```
class BankAccount{
    long account_no;
    String account_name;
    double balance;

    public BankAccount(long account_no,String account_name){
        this.account_no = account_no;
        this.account_name = account_name;
        this.balance = 0.0;
    }

    public void openAccount(){
        System.out.println("Account opened Successfully");
    }

    public void deposit(double amount){
        balance += amount;
        System.out.println("Amount " + amount + " deposited successfully");
    }

    public double checkBalance(){
        return balance;
    }

    public void withdraw(double amount){
        if(amount >= 0 && amount < balance){
            balance -= amount;
            System.out.println("Amount " + amount + " withdrawn successfully");
        }
    }
}

class SavingsAccount extends BankAccount{
    private double interestRate;

    public SavingsAccount(long account_no,String account_name,double interestRate){
        super(account_no, account_name);
        this.interestRate = interestRate;
    }

    double calInterest(){
        return balance*interestRate;
    }
}
```

```

class FixedDepositAccount extends BankAccount{
    private int tenure;
    public FixedDepositAccount(long account_no,String account_name,int tenure){
        super(account_no, account_name);
        this.tenure = tenure;
    }

    public double calInterest() {
        return 0.0;
    }
}

public class Bank {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 14/03/24");
        System.out.println("Time: 03:40PM");
        System.out.println();

        SavingsAccount savingsAccount = new SavingsAccount(1234567890, "Megh", 0.05);
        savingsAccount.openAccount();
        savingsAccount.deposit(1000.00);
        System.out.println("Current Balance: " + savingsAccount.checkBalance());
        System.out.println("Interest Earned: " + savingsAccount.calInterest());
        System.out.println();

        FixedDepositAccount fdAccount = new FixedDepositAccount(987654321, "Jenil", 12);
        fdAccount.openAccount();
        fdAccount.deposit(5000.00);
        System.out.println("Current Balance: " + fdAccount.checkBalance());
        System.out.println("Interest Earned (placeholder): " + fdAccount.calInterest());
    }
}

```

Output:

```

Account opened Successfully
Amount 1000.0 deposited successfully
Current Balance: 1000.0
Interest Earned: 50.0

Account opened Successfully
Amount 5000.0 deposited successfully
Current Balance: 5000.0
Interest Earned (placeholder): 0.0
PS D:\Programmes\Java\Practicle - 4> 

```

- Write a program that finds area of any shape by overloading area () method for Square, Rectangle, Triangle and Square.

Code:

```

import java.lang.Math;
class Square{
    double length;

    public Square(double length){

```

```
        this.length = length;
    }

    public double area(){
        return length*length;
    }
}

class Triangle{
    double a,b,c;

    public Triangle(double a,double b,double c){
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public double area(){
        double s = (a+b+c)/2;
        double sq = (s*(s-a)*(s-b)*(s-c));
        return Math.sqrt(sq);
    }
}

class Rectangle{
    double length,width;
    public Rectangle(double length,double width){
        this.length = length;
        this.width = width;
    }

    public double area(){
        return length*width;
    }
}

public class Area {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 14/03/2024");
        System.out.println("time: 03:50 PM");
        System.out.println();
        Square s = new Square(10);
        System.out.println("Area of Square is: " + s.area());

        Triangle t = new Triangle(10, 20, 30);
        System.out.println("Area of Triangle is: " + t.area());

        Rectangle r = new Rectangle(10, 20);
        System.out.println("Area of Rectangle is: " + r.area());
    }
}
```

Output:

```
Area of Square is: 100.0
Area of Triangle is: 0.0
Area of Rectangle is: 200.0
PS D:\Programmes\Java\Practicle - 4>
```

4. Write a program that finds Volume of any shape by overloading volume () method for Cube, Rectangular Cube and Sphere.

Code:

```
class Cube{
    double a;

    public Cube(double length){
        this.a = length;
    }

    public double Volume(){
        return a*a*a;
    }
}

class RectangularCube{
    double a,b,c;

    public RectangularCube(double a,double b,double c){
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public double Volume(){
        return a*b*c;
    }
}

class Sphere{
    double r,pi;
    public Sphere(double r,double pi){
        this.r = r;
        this.pi = pi;
    }

    public double Volume(){
        return (4/3)*r*pi;
    }
}

public class Volume {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 14/03/24");
        System.out.println("time: 04:05 PM");
    }
}
```

```

    System.out.println();

    Cube c = new Cube(3);
    System.out.println("Volume of Cube is: " + c.Volume());

    RectangularCube rc = new RectangularCube(1, 2, 3);
    System.out.println("Volume of RectangularCube is: " + rc.Volume());

    Sphere s = new Sphere(2, 22.7);
    System.out.println("Volume of Sphere is: " + s.Volume());
}
}

```

Output:

```

Volume of Cube is: 27.0
Volume of RectangularCube is: 6.0
Volume of Sphere is: 45.4
PS D:\Programmes\Java\Practicle - 4>

```

5. Write a Program to maintain employee's information. Program should illustrate Inheritance concept. (Use your imagination to create class or subclass used for employee).

Code:

```

class Person{
    String Name;
    int Age;

    public Person(String Name,int Age){
        this.Name = Name;
        this.Age = Age;
    }
}

class Intern extends Person{
    double Stipend;
    String Role;
    public Intern(double Stipend,String Role,String Name,int Age){
        super(Name,Age);
        this.Stipend = Stipend;
        this.Role = Role;
    }

    public void details(){
        System.out.println("Details of Intern is: ");
        System.out.println("Name: " + Name);
        System.out.println("Age: " + Age);
        System.out.println("Stipend: " + Stipend);
        System.out.println("Role: " + Role);
    }
}

```

```
class Freelancer extends Person{
    double Payment;
    String Work;
    public Freelancer(double Payment,String Work,String Name,int Age){
        super(Name,Age);
        this.Payment = Payment;
        this.Work = Work;
    }

    public void details(){
        System.out.println("Details of Freelancer is: ");
        System.out.println("Name: " + Name);
        System.out.println("Age: " + Age);
        System.out.println("Payment: " + Payment);
        System.out.println("Work: " + Work);
    }
}

class Employee extends Person{
    double Salary;
    String Designation;
    int Experience;
    public Employee(double Salary,String Designation,int Experience,String Name,int Age){
        super(Name, Age);
        this.Salary = Salary;
        this.Designation = Designation;
        this.Experience = Experience;
    }

    public void details(){
        System.out.println("Details of Employee is: ");
        System.out.println("Name: " + Name);
        System.out.println("Age: " + Age);
        System.out.println("Salary: " + Salary);
        System.out.println("Designation: " + Designation);
        System.out.println("Experience: " + Experience);
    }
}

public class EmployeeManagement {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 14/03/2024");
        System.out.println("Time: 04:30 PM");
        System.out.println();

        Person P1 = new Person("A", 20);
        Person P2 = new Person("B", 22);
        Person P3 = new Person("C", 21);
        Person P4 = new Person("D", 24);
        Person P5 = new Person("E", 25);
    }
}
```



```

    Intern I1 = new Intern(40000, "Jr.Java Developer", P1.Name, P1.Age);
    Intern I2 = new Intern(40000, "Jr.Java Developer", P3.Name, P3.Age);
    Freelancer FL1 = new Freelancer(25000, "Software Installation", P2.Name, P2.Age);
    Employee E1 = new Employee(100000, "Sr.Developer", 4,P4.Name,P4.Age);
    Employee E2 = new Employee(100000, "Sr.Developer", 5,P5.Name,P5.Age);

    I1.details();
    FL1.details();
    E1.details();
}
}

```

Output:

```

Details of Intern is:
Name: A
Age: 20
Stipend: 40000.0
Role: Jr.Java Developer
Details of Freelancer is:
Name: B
Age: 22
Payment: 25000.0
Work: Software Installation
Details of Employee is:
Name: D
Age: 24
Salary: 100000.0
Designation: Sr.Developer
Experience: 4
PS D:\Programmes\Java\Practicle - 4> 

```

6. Create a base class Shape. Use this class to store two double type values that could be used to compute area of any shape. Derive two specific classes called Triangle and Rectangle from the base shape. Add to the base a member function getdata() to initialize base class data member and another member function display_area() to compute and display the area of figures. (Use Method Overriding).

Code;

```

abstract class Shape{
    double a,b;

    public void getData(){
        System.out.print("Enter a: ");
        a = Double.parseDouble(System.console().readLine());
        System.out.print("Enter b: ");
        b = Double.parseDouble(System.console().readLine());
    }

    public abstract void display_Area();
}

class Triangle extends Shape{

    public void display_Area() {
        double area = 0.5*a*b;
    }
}

```

```
        System.out.println("Area of Triangle is: " + area);
    }
}
```

```
class Rectangle extends Shape{
    public void display_Area(){
        double area = a*b;
        System.out.println("Area of Rectangle is: " + area);
    }
}
```

```
public class Base {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 14/03/24");
        System.out.println("time: 04:45 PM");
        System.out.println();

        Triangle t = new Triangle();
        t.getData();
        t.display_Area();

        Rectangle r = new Rectangle();
        r.getData();
        r.display_Area();

    }
}
```

Output:

```
PS D:\Programmes\Java\Practicle - 4> cd "d:\Programmes\Java\Practicle - 4\" ; if ($?) { javac Base.java } ; if ($?) { java
Base }
220130107094 - Megh Patel
Date: 14/03/24
time: 04:45 PM

Enter a: 1
Enter b: 1
Area of Triangle is: 0.5
Enter a: 1
Enter b: 1
Area of Rectangle is: 1.0
PS D:\Programmes\Java\Practicle - 4> █
```

Conclusion:

Quiz: (Sufficient space to be provided for the answers)

1. What is inheritance in java? Explain different types of inheritance with proper example.

2. Explain the use of final and Super keyword in JAVA
3. Define polymorphism with its need.
4. Explain about Encapsulation, Abstraction.

Suggested Reference:

1. <https://www.tutorialspoint.com/java/>
2. <https://www.geeksforgeeks.org/>
3. <https://www.w3schools.com/java/>
4. <https://www.javatpoint.com/>

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Criteria	Need Improvement	Good	Excellent	Total
Marks	Design of logic (4) Correct output (4) Mock viva test (2)	Program has significant logic errors. (1) Output has multiple errors. (1) Delayed & only few correct answers (1)	Program has slight logic errors that do no significantly affect the results (2) Output has minor errors. (2) Partially correct response (1)	Program is logically well designed (3) Program displays correct output with no errors (3) All questions responded Correctly (2)	

Signature of Faculty

Experiment No: 5

AIM: To demonstrate the use of abstract classes and interfaces.

Date:

CO mapped: CO-2

Objectives:

- a) To understand the purpose and usage of abstract classes and interfaces in object-oriented programming. Develop the ability to design and implement abstract classes and interfaces effectively to promote code reusability, ensure consistent behavior in class hierarchies, and facilitate the development of flexible and extensible software systems.
- b) Abstract classes and interfaces are important OOP concepts that allow you to define common contracts and behaviors for classes. Achieving this objective will enable you to use these tools to create more modular and maintainable software, especially when dealing with class hierarchies and multiple implementations.

Background:

A class that is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Points to Remember

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

An interface in Java is a blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not the method body. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Practical questions:

1. Describe abstract class called Shape which has three subclasses say Triangle, Rectangle, Circle. Define one method area() in the abstract class and override this area() in these three subclasses to calculate for specific object, i.e., area() of Triangle subclass should calculate area of triangle etc. Same for Rectangle and Circle.

Code:

```
abstract class Shape {  
  
    double a, b;  
  
    public void getData() {
```

```
        System.out.print("Enter a: ");

        a = Double.parseDouble(System.console().readLine());

        System.out.print("Enter b: ");

        b = Double.parseDouble(System.console().readLine());

    }

    public abstract void display_Area();

}

class Triangle extends Shape {

    public void display_Area() {

        double area = 0.5 * a * b;

        System.out.println("Area of Triangle is: " + area);

    }

}

class Rectangle extends Shape {

    public void display_Area() {

        double area = a * b;

        System.out.println("Area of Rectangle is: " + area);

    }

}

class Circle extends Shape {

    public void display_Area() {

        System.out.println("Area of Circle is: " + Math.PI * a * a);

    }

}

public class Base {
```

```
public static void main(String[] args) {  
  
    System.out.println("220130107094 - Megh Patel");  
  
    System.out.println("Date: 14/03/24");  
  
    System.out.println("time: 05:00 PM");  
  
    System.out.println();  
  
    Triangle t = new Triangle();  
  
    t.getData();  
  
    t.display_Area();  
  
    Rectangle r = new Rectangle();  
  
    r.getData();  
  
    r.display_Area();  
  
    Circle c = new Circle();  
  
    c.getData();  
  
    c.display_Area();  
  
}  
}
```

Output:

```
Enter a: 1  
Enter b: 1  
Area of Triangle is: 0.5  
Enter a: 1  
Enter b: 2  
Area of Rectangle is: 2.0  
Enter a: 1  
Enter b: 3  
Area of Circle is: 3.141592653589793  
PS D:\Programmes\Java\Practicle - 4> █
```

2. Write a program that demonstrates the instance of operator. Declare interfaces I1 and I2. Interface I3 extends both of these interfaces. Also declare interface I4. Class X implements I3. Class W extends X and implements I4. Create an object of class W. Use the instance of operator to test if that object implements each of the interfaces and is of type X.

Code;

```
interface I1 {
```

```
}

interface I2 {

}

interface I3 extends I1, I2 {

}

interface I4 {

}

class X implements I3 {

}

class W extends X implements I4 {

}


public class Instance {

    public static void main(String[] args) {

        System.out.println("220130107094 - Megh Patel");

        System.out.println("Date: 21/03/24");

        System.out.println("time: 03:40 PM");

        System.out.println();

        W obj = new W();

        System.out.println("obj instanceof I1: " + (obj instanceof I1));

        System.out.println("obj instanceof I2: " + (obj instanceof I2));

        System.out.println("obj instanceof I3: " + (obj instanceof I3));

        System.out.println("obj instanceof I4: " + (obj instanceof I4));

        System.out.println("obj instanceof X: " + (obj instanceof X));

        System.out.println("obj instanceof W: " + (obj instanceof W));

    }

}
```

Output:

```
obj instanceof I1: true
obj instanceof I2: true
obj instanceof I3: true
obj instanceof I4: true
obj instanceof X: true
obj instanceof W: true
PS D:\Programmes\Java\Practicle - 4> █
```

- Write a java program to implement an interface called Exam with a method Pass (int mark) that returns a boolean. Write another interface called Classify with a method Division (int average) which returns a String. Write a class called Result which implements both Exam and Classify. The Pass method should return true if the mark is greater than or equal to 50 else false. The Division method must return "First" when the parameter average is 60 or more, "Second" when average is 50 or more but below 60, "No division" when average is less than 50.

Code;

```
interface Exam {
    boolean Pass(int mark);
}

interface Classify {
    String Division(int average);
}

class Result implements Exam, Classify {
    public boolean Pass(int mark) {
        return mark >= 50;
    }

    public String Division(int average) {
        if (average >= 60) {
            return "First";
        } else if (average >= 50) {
            return "Second";
        } else {
            return "No division";
        }
    }
}

public class StudentMark {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 21/03/24");
        System.out.println("time: 04:10PM");
        System.out.println();

        Result result = new Result();

        int marks;
        System.out.print("Enter marks: ");
```



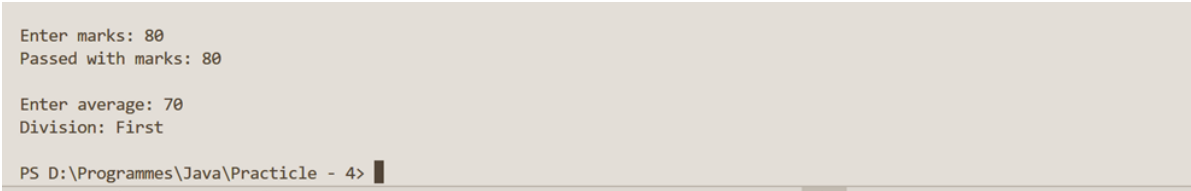
```
marks = Integer.parseInt(System.console().readLine());

if (result.Pass(marks)) {
    System.out.println("Passed with marks: " + marks);
    System.out.println();

    int average;
    System.out.print("Enter average: ");
    average = Integer.parseInt(System.console().readLine());

    System.out.println("Division: " + result.Division(average));
} else {
    System.out.println("Failed with marks: " + marks);
}
System.out.println();
}
```

Output:



```
Enter marks: 80
Passed with marks: 80

Enter average: 70
Division: First
```

PS D:\Programmes\Java\Practicle - 4>

Conclusion:

Quiz: (Sufficient space to be provided for the answers)

1. Explain how interfaces promote the concept of multiple inheritance in OOP.
2. What is an interface, and how does it differ from an abstract class?
3. When would you choose to use an abstract class over an interface, and vice versa, in your software design?
4. Can a class implement multiple interfaces? If so, what benefits does this provide?
5. Can you declare an interface method static? Justify your answer.

Suggested Reference:

1. <https://www.tutorialspoint.com/java/>
2. <https://www.geeksforgeeks.org/>
3. <https://www.w3schools.com/java/>
4. <https://www.javatpoint.com/>

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Criteria	Need Improvement	Good	Excellent	Total
Marks	Design of logic (4) Correct output (4) Mock viva test (2)	Program has significant logic errors. (1) Output has multiple errors. (1) Delayed & only few correct answers (1)	Program has slight logic errors that do not significantly affect the results (2) Output has minor errors. (2) Partially correct response (1)	Program is logically well designed (3) Program displays correct output with no errors (3) All questions responded Correctly (2)	

Signature of Faculty:

Experiment No: 6

AIM: To implement packages and exception handling in JAVA application.

Date:

CO mapped: CO-3

Objectives:

To effectively implement packages and exception handling in a Java application, organizing code into logical modules for improved maintainability, and ensuring robust error handling to enhance the application's reliability and user experience.

Background:

A java package is a group of similar types of classes, interfaces, and sub-packages. Package in java can be categorized in two forms, built-in package, and user-defined package. There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc. Here, we will have the detailed learning of creating and using user-defined packages.

Exception Handling in Java is one of the powerful mechanisms to handle runtime errors so that the normal flow of the application can be maintained. In this practical, we will learn about Java exceptions, their types, and the difference between checked and unchecked exceptions.

Advantage of Java Package

- 1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.
- 2) Java package provides access protection.
- 3) Java package removes naming collision.

Practical questions:

1. Write a program in Java to develop user defined exception for “Divide by Zero” error.

Code:

```
public class DividebyZero{  
  
    public static void main(String[] args) {  
  
        System.out.println("220130107094 - Megh Patel");  
  
        System.out.println("Date: 21/03/24");  
  
        System.out.println("Time: 04:20 PM");  
  
        System.out.println();  
    }  
}
```

```

    int a = 5;

    int b = 0;

    try{

        int result = a/b;

        System.out.println("Program Execute without Exception and result: " + result);

    }

    catch(Exception e){

        System.out.println(e);

    }

}

```

Output:

```

java.lang.ArithmeticException: / by zero
PS D:\Programmes\Java\Practical - 6>

```

2. Write a program in Java to demonstrate throw, throws, finally, multiple try block and multiple catch exception.

Code:

```

class MyException extends Exception {
    public MyException(String message) {
        super(message);
    }
}

public class Demonstrate {
    public static void risk(int num) throws MyException, ArithmeticException {
        if (num < 0) {
            throw new MyException("Negative number should not allow.");
        } else if (num == 0) {
            throw new ArithmeticException("Division by zero is not allow.");
        }
    }

    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 21/03/24");
        System.out.println("Time: 04:45 PM");
        System.out.println();
    }
}

```

```

    try {
        risk(-1);
    } catch (MyException e) {
        System.out.println("Caught MyException: " + e.getMessage());
    } catch (ArithmeticException e) {
        System.out.println("Caught ArithmeticException");
    } finally {
        System.out.println("Hi ! I am finally - 1. I am always executes.");
        System.out.println();
    }

    try {
        risk(0);
    } catch (MyException e) {
        System.out.println("Caught MyException");
    } catch (ArithmeticException e) {
        System.out.println("Caught ArithmeticException: " + e.getMessage());
    } finally {
        System.out.println("Hi ! I am finally - 2. I am always executes.");
        System.out.println();
    }
}
}

```

Output:

```

Caught MyException: Negative number should not allow.
Hi ! I am finally - 1. I am always executes.

Caught ArithmeticException: Division by zero is not allow.
Hi ! I am finally - 2. I am always executes.

```

3. Write a small application in Java to develop Banking Application in which user deposits the amount Rs 1000.00 and then start withdrawing ofRs 400.00, Rs 300.00 and it throws exception "Not Sufficient Fund" when user withdraws Rs 500 thereafter.

```

public class BankAccount {

    private double balance = 1000.00;

    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: Rs." + amount + ". New balance: Rs." + balance);
    }

    public void withdraw(double amount) throws InsufficientFundsException {
        if (amount > balance) {
            throw new InsufficientFundsException("Not Sufficient Funds");
        }
        balance -= amount;
        System.out.println("Withdrew: Rs." + amount + ". New balance: Rs." + balance);
    }
}

```

```

public static void main(String[] args) {
    System.out.println("220130107094 - Megh Patel");
    System.out.println("Date: 21/03/24");
    System.out.println("Time: 05:10 PM");
    System.out.println();
    BankAccount account = new BankAccount();
    account.deposit(1000.00);

    try {
        account.withdraw(400.00);
        account.withdraw(300.00);
        account.withdraw(500.00);
    } catch (InsufficientFundsException e) {
        System.out.println("Error: ");
    }
}

class InsufficientFundsException extends Exception {
    public InsufficientFundsException(String message) {
        super(message);
    }
}

```

Output:

```

Deposited: Rs.1000.0. New balance: Rs.2000.0
Withdrew: Rs.400.0. New balance: Rs.1600.0
Withdrew: Rs.300.0. New balance: Rs.1300.0
Withdrew: Rs.500.0. New balance: Rs.800.0
PS D:\Programmes\Java\Practical - 6>

```

- Write an application that contains a method named average () has one argument that is an array of strings. It converts these to double values and returns their average. The method generates a NullPointerException, if an array elements is null or a NumberFormatException, if an element is incorrectly formatted. Include throws statement in method declaration.

Code:

```

public class StringAverage {
    public static double average(String[] values) throws NullPointerException,
        NumberFormatException {
        if (values == null) {
            throw new NullPointerException("Array cannot be null");
        }

        double sum = 0.0;
        int count = 0;
        for (String value : values) {
            if (value == null) {
                throw new NullPointerException("Array element cannot be null");
            }
            try {
                double num = Double.parseDouble(value);
            }

```

```

        sum += num;
        count++;
    } catch (NumberFormatException e) {
        System.out.println("Error parsing element: " + value);
    }
}

return sum / count;
}

public static void main(String[] args) {
    System.out.println("220130107094 - Megh Patel");
    System.out.println("Date: 27/03/24");
    System.out.println("Time: 03:45 PM");
    System.out.println();
    String[] a1 = { "10.5", "20.2", "33", "null", "hello" };
    String[] a2 = { "4.5", "nine", "7.8", "9 " };

    try {
        System.out.println("Average of - a1: " + average(a1));
        System.out.println("Average of - a2: " + average(a2));
    } catch (NullPointerException | NumberFormatException e) {
        System.out.println("Error: " + e);
    }
}
}

```

Output:

```

Error: java.lang.NullPointerException: Array element cannot be null
PS D:\Programmes\Java\Practical - 6>

```

```

Error parsing element: null
Error parsing element: hello
Average of - a1: 21.233333333333334
Error parsing element: nine
Average of - a2: 7.1000000000000005
PS D:\Programmes\Java\Practical - 6>

```

5. Write an application that generates custom exception if first argument from command line argument is 0.

Code:

```

class NullArgumentException extends Exception {
    public NullArgumentException(String message) {
        super(message);
    }
}

public class CheckFirstArg {

    public static void main(String[] args) throws NullArgumentException {
        System.out.println("220130107094 - Megh Patel");
    }
}

```

```

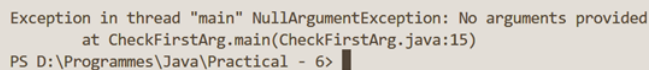
        System.out.println("Date: 27/03/24");
        System.out.println("Time: 04:00 PM");
        System.out.println();
        if (args.length == 0) {
            throw new NullPointerException("No arguments provided");
        }

        int firstArg = Integer.parseInt(args[0]);
        if (firstArg == 0) {
            throw new NullPointerException("First argument cannot be zero");
        }

        System.out.println("First argument: " + firstArg);
    }
}

```

Output:



```

Exception in thread "main" NullPointerException: No arguments provided
    at CheckFirstArg.main(CheckFirstArg.java:15)
PS D:\Programmes\Java\Practical - 6>

```

6. A marklist containing reg.no and marks for a subject is given. if the marks are <0, user-defined `IllegalMarkException` is thrown out and handled with the message "Illegal Mark". For all valid marks, the candidate will be declared as "PASS" if the marks are equal to or greater than 40, otherwise it will be declared as "FAIL". Write a class called `IllegalMarkException`.

Code:

```

class IllegalMarkException extends Exception {
    public IllegalMarkException(String message) {
        super(message);
    }
}

public class Marklist {

    public static void validMark(int marks) throws IllegalMarkException {
        if (marks < 0) {
            throw new IllegalMarkException("Illegal Mark");
        }
    }

    public static String getResult(int marks) {
        return marks >= 40 ? "PASS" : "FAIL";
    }

    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 27/03/24");
        System.out.println("Time: 04:18 PM");
        System.out.println();
    }
}

```



```

int[] regNo = { 101, 102, 103 };
int[] marks = { 39, -20, 40};

for (int i = 0; i < regNo.length; i++) {
    try {
        validMark(marks[i]);
        System.out.println("Reg No: " + regNo[i] + " Result: " + getResult(marks[i]));
    } catch (IllegalMarkException e) {
        System.out.println("Reg No: " + regNo[i] + " Error: " + e);
    }
}
}
}

```

Output:

```

Reg No: 101 Result: FAIL
Reg No: 102 Error: IllegalMarkException: Illegal Mark
Reg No: 103 Result: PASS
PS D:\Programmes\Java\Practical - 6> 

```

7. Assume that there are two packages, student and exam. A student package contains Student class and the exam package contains Result class. Write a program that generates mark sheet for students.

Code:

```

package exam;

import student.Student;

public class Result {
    private Student student;
    private int totalMarks;
    private double percentage;

    public Result(Student student) {
        this.student = student;
        calculateResult();
    }

    private void calculateResult() {
        totalMarks = getMarksFromExternalSource(student);
        percentage = ((double) totalMarks / getNumberOfSubjects());
    }

    private int getMarksFromExternalSource(Student student) {
        return 255;
    }

    private int getNumberOfSubjects() {
        return 3;
    }
}

```

```

public void generateMarkSheet() {
    System.out.println();
    System.out.println("Mark Sheet for Student: " + student.getName());
    System.out.println("Roll Number: " + student.getRollNo());
    System.out.println();
    System.out.println("Total Marks: " + totalMarks);
    System.out.println("Percentage: " + String.format("%.2f", percentage) + "%");
    System.out.println();
}

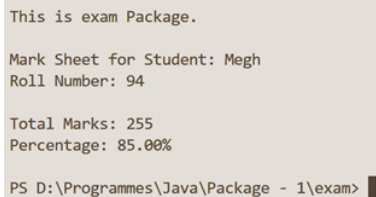
public static void main(String[] args) {
    System.out.println("220130107094 - Megh Patel");
    System.out.println("Date: 27/03/24");
    System.out.println("Time: 04:30 PM");
    System.out.println();

    System.out.println("This is exam Package.");

    Student student1 = new student.Student("Megh", 94);
    Result result = new Result(student1);
    result.generateMarkSheet();
}
}

```

Output:



```

This is exam Package.

Mark Sheet for Student: Megh
Roll Number: 94

Total Marks: 255
Percentage: 85.00%

PS D:\Programmes\Java\Package - 1\exam>

```

8. Define a class A in package a pack. In class A, three variables are defined of access modifiers protected, private and public. Define class B in package bpack which extends A and write display method which accesses variables of class A. Define class C in package cpack which has one method display() in that create one object of class A and display its variables. Define class ProtectedDemo in package dpack in which write main() method. Create objects of class B and C and class display method for both these objects.

Code - A:

```

package apack;

public class A {
    protected static int protectedVar = 10;
    private static int privateVar = 20;
    public static int publicVar = 30;

    // getters can be added to access private variables if needed

```

```

public static void main(String[] args) {
    System.out.println("220130107094 - Megh Patel");
    System.out.println("Date: 27/03/24");
    System.out.println("Time: 05:10 PM");
    System.out.println();
    System.out.println("This is Package apack");

    System.out.println("Protected Member is: " + protectedVar);
    System.out.println("Private Member is: " + privateVar);
    System.out.println("Public Member is: " + publicVar);
}
}

```

Output – A;

```

This is Package apack
Protected Member is: 10
Private Member is: 20
Public Member is: 30
PS D:\Programmes\Java\Package copy\apack>

```

Code – B:

```

package bpack;

import apack.A;

public class B extends A {

    public void display() {
        System.out.println("Protected variable: " + protectedVar);
        // Private variable cannot be accessed from outside the class A
        // System.out.println("Private variable: " + privateVar); // compile error
        System.out.println("Public variable: " + publicVar);
    }

    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 27/03/24");
        System.out.println("Time: 05:10 PM");
        System.out.println();
        System.out.println("This is Package bpack");

        B b = new B();
        b.display();
    }
}

```

Output – B:

```

This is Package bpack
Protected variable: 10
Public variable: 30
PS D:\Programmes\Java\Package copy\bpack>

```

Code – C:

```
package cpack;

import apack.A;

public class C {

    public void display() {
        A objA = new A();
        System.out.println("Public variable (from object): " + A.publicVar);
    }

    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 27/03/24");
        System.out.println("Time: 05:10 PM");
        System.out.println();
        System.out.println("This is Package apack");

        C c = new C();
        c.display();
    }
}
```

Output – C:

```
This is Package cpack
Public variable (from object): 30
PS D:\Programmes\Java\Package copy\cpack> █
```

Code – D:

```
package dpack;

import bpack.B;
import cpack.C;

public class ProtectedDemo {

    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 27/03/24");
        System.out.println("Time: 05:10 PM");
        System.out.println();
        System.out.println("This is Package dpack");

        B objB = new B();
        C objC = new C();
    }
}
```

```
        System.out.println("Class B display:");
        objB.display();

        System.out.println("\nClass C display:");
        objC.display();
    }
}
```

Output – D:

```
This is Package dpack
Class B display:
Protected variable: 10
Public variable: 30

Class C display:
Public variable (from object): 30
PS D:\Programmes\Java\Package Copy\dpack> █
```

Conclusion:

Quiz: (Sufficient space to be provided for the answers)

1. Explain the benefits of organizing classes into packages in a Java application.
2. How do you declare and define a package in Java?
3. What is an exception in Java, and why is exception handling important in software development?
4. Explain the try-catch-finally block and its role in handling exceptions.
5. What is difference between throw and throws?

Suggested Reference:

1. <https://www.tutorialspoint.com/java/>
2. <https://www.geeksforgeeks.org/>
3. <https://www.w3schools.com/java/>
4. <https://www.javatpoint.com/>

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Criteria	Need Improvement	Good	Excellent	Total
Marks	Design of logic (4) Correct output (4) Mock viva test (2)	Program has significant logic errors. (1) Output has multiple errors. (1) Delayed & only few correct answers (1)	Program has slight logic errors that do not significantly affect the results (2) Output has minor errors. (2) Partially correct response (1)	Program is logically well designed (3) Program displays correct output with no errors (3) All questions responded Correctly (2)	

Signature of Faculty:

Experiment No: 7

AIM: To demonstrate I/O from files.

Date:

CO mapped: CO-4

Objectives:

To showcase proficiency in reading data from and writing data to files in various formats using programming languages, demonstrating the ability to implement reliable and efficient file I/O operations, which are essential for tasks such as data storage, retrieval, and processing in software applications.

Background:

Java I/O (Input and Output) is used to process the input and produce the output. Java uses the concept of a stream to make I/O operations fast. The java.io package contains all the classes required for input and output operations.

Practical questions:

1. Write a program that removes all the occurrences of a specified string from a text file. For example, invoking java Practical7_1 John filename removes the string John from the specified file. Your program should read the string as an input.

Code:

```
import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.BufferedWriter;

import java.io.IOException;

import java.util.Scanner;

public class Practical7_1 {

    public static void main(String[] args) {

        System.out.println("220130107094-Megh Patel");

        System.out.println("Date: 11/04/24");

        System.out.println("Time: 03:29 PM");

        System.out.println();
```

```
    if (args.length < 2) {  
        System.out.println("Usage: java Practical7_1 <string_to_remove> <filename>");  
        System.exit(1);  
    }  
  
    String stringToRemove = args[0];  
    String filename = args[1];  
  
    String fileContent = "";  
    try (Scanner scanner = new Scanner(new FileReader(filename))) {  
        fileContent = scanner.useDelimiter("\\A").next(); // Read entire file content  
    } catch (FileNotFoundException e) {  
        System.err.println("Error: File not found: " + filename);  
        System.exit(1);  
    }  
  
    String modifiedContent = fileContent.replaceAll(stringToRemove, "");  
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename + ".modified"))) {  
        writer.write(modifiedContent);  
    } catch (IOException e) {  
        System.err.println("Error writing to file: " + filename + ".modified");  
    }  
  
    System.out.println("Successfully removed all occurrences of \"" + stringToRemove + "\" from "  
+ filename);  
}  
}
```

Output;


```
PS D:\Programmes\Java\Practicle - 7> javac Practical7_1.java
PS D:\Programmes\Java\Practicle - 7> javac Practical7_1.java
PS D:\Programmes\Java\Practicle - 7> java Practical7_1 "megh" First.txt
220130107094-Megh Patel
Date: 11/04/24
Time: 03:29 PM
```

- Write a program that will count the number of characters, words, and lines in a file. Words are separated by whitespace characters. The file name should be passed as a command-line argument.

Code:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class Practicle7_2 {

    public static void main(String[] args) {
        System.out.println("220130107094-Megh Patel");
        System.out.println("Date: 11/04/24");
        System.out.println("Time: 03:45 PM");
        System.out.println();

        if (args.length != 1) {
            System.out.println("Usage: java CountLinesWordsChars <filename>");
            System.exit(1);
        }

        String filename = args[0];
        int lineCount = 0;
        int wordCount = 0;
        int characterCount = 0;

        try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
            String line;
            while ((line = reader.readLine()) != null) {
                lineCount++;
                characterCount += line.length(); // Add line length to character count
                wordCount += line.split("\\s+").length; // Split on whitespace for word count
            }
        } catch (IOException e) {
            System.err.println("Error reading file: " + filename);
            System.exit(1);
        }

        System.out.println("Lines: " + lineCount);
        System.out.println("Words: " + wordCount);
        System.out.println("Characters: " + characterCount);
    }
}
```

Output:

```
Lines: 40  
Words: 152  
Characters: 1140
```

3. Write a program to create a file named Practical7.txt if it does not exist. Write 100 integers created randomly into the file. Integers are separated by spaces in the file. Read the data back from the file and display the data in increasing order.

Code:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;

public class Practical7_3 {

    public static void main(String[] args) {
        System.out.println("220130107094-Megh Patel");
        System.out.println("Date: 11/04/24");
        System.out.println("Time: 04:01 PM");
        System.out.println();
        String filename = "Practical7.txt";
        int numIntegers = 100;

        writeRandomIntsToFile(filename, numIntegers);

        int[] randomInts = readIntsFromFile(filename);

        sortAscending(randomInts);

        System.out.println("Sorted Integers:");
        for (int num : randomInts) {
            System.out.print(num + " ");
        }
    }

    private static void writeRandomIntsToFile(String filename, int numIntegers) {
        try (FileWriter writer = new FileWriter(filename, true)) {
            Random random = new Random();
            StringBuilder content = new StringBuilder();
            for (int i = 0; i < numIntegers; i++) {
                content.append(random.nextInt(1000) + 1).append(" ");
            }
            writer.write(content.toString());
        } catch (IOException e) {
            System.err.println("Error writing to file: " + filename);
        }
    }
```

```

private static int[] readIntsFromFile(String filename) {
    try (Scanner scanner = new Scanner(new File(filename))) {
        int[] numbers = new int[100];
        int index = 0;
        while (scanner.hasNextInt() && index < numbers.length) {
            numbers[index++] = scanner.nextInt();
        }
        return Arrays.copyOf(numbers, index);
    } catch (FileNotFoundException e) {
        System.err.println("Error: File not found: " + filename);
        return new int[0];
    }
}

private static void sortAscending(int[] numbers) {
    Arrays.sort(numbers);
}
}

```

Output:

```

Sorted Integers:
6 8 10 13 22 24 38 47 49 65 67 70 73 75 81 83 89 137 144 163 179 181 185 201 208 217 218 229 234 235 240 246 246 251 275 280 300 307 31
332 337 362 368 377 385 390 391 400 401 439 440 444 465 485 489 491 521 531 557 566 578 591 610 611 614 632 638 651 668 670 675 678 68
695 705 720 740 750 752 762 763 783 795 796 800 819 821 846 880 880 918 923 938 947 961 966 970 972 979 995
PS D:\Programmes\Java\Practice - 7>

```

Conclusion:

Quiz: (Sufficient space to be provided for the answers)

1. What is file input/output (I/O), and why is it important in software development?
2. What are the common modes for opening files, and how do they differ (e.g., read, write, append)?
3. Describe the concept of file streams and how they are used in file I/O operations.
4. Write short notes about I/O stream classes.

Suggested Reference:

1. <https://www.tutorialspoint.com/java/>
2. <https://www.geeksforgeeks.org/>
3. <https://www.w3schools.com/java/>
4. <https://www.javatpoint.com/>

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Criteria	Need Improvement	Good	Excellent	Total
Marks	Design of logic (4) Correct output (4) Mock viva test (2)	Program has significant logic errors. (1) Output has multiple errors. (1) Delayed & only few correct answers (1)	Program has slight logic errors that do not significantly affect the results (2) Output has minor errors. (2) Partially correct response (1)	Program is logically well designed (3) Program displays correct output with no errors (3) All questions responded Correctly (2)	

Signature of Faculty:

Experiment No: 8

AIM: To learn JAVA FX UI Controls.

Date:

CO mapped: CO-5

Objectives:

- a) To gain proficiency in JavaFX UI controls, including understanding their features and capabilities, and developing the ability to create interactive and visually appealing user interfaces for Java applications. This knowledge will enable the design and development of user-friendly, responsive, and feature-rich graphical user interfaces (GUIs) in Java applications.
- b) Learning JavaFX UI controls is essential for creating modern and engaging graphical user interfaces for Java applications. This objective emphasizes not only understanding the various UI controls but also the practical skills to design and implement user interfaces effectively.

Background:

Every user interface considers the following three main aspects –

UI elements – These are the core visual elements that the user eventually sees and interacts with. JavaFX provides a huge list of widely used and common elements varying from basic to complex, which we will cover in this practical.

Layouts – They define how UI elements should be organized on the screen and provide a final look and feel to the GUI (Graphical User Interface). This part will be covered in the Layout chapter.

Behavior – These are events that occur when the user interacts with UI elements.

JavaFX provides several classes in the package `javafx.scene.control`. To create various GUI components (controls), JavaFX supports several controls such as date picker, button text field, etc. Each control is represented by a class; you can create a control by instantiating its respective class.

Common elements in a JavaFX application

All JavaFX applications contain the following elements:

1. A main window, called a stage in JavaFX.
2. At least one Scene in the stage.
3. A system of panes and boxes to organize GUI elements in the scene.
4. One or more GUI elements, such as buttons and labels.

The usual procedure for setting up a scene is to build it from the bottom up. First, we make the GUI elements, then we make boxes and panes to organize the elements, and finally, we put everything in the scene.

All JavaFX elements such as boxes and panes that are meant to contain other elements have a child list that we can access via the `getChildren()` method. We put elements inside other elements by adding things to child lists. In the code above you can see the button and the label objects being added as children of a VBox, and the VBox, in turn, is set as the child of a StackPane.

In addition to setting the structure for the window, we also call methods designed to set the properties of various elements. For example, the code in this example uses the button's `setText()`

method to set the text the button will display.

Follow the procedure outlined in the section above to make a new JavaFX application. Replace the start() method in the App class with the following code:

```
public void start(Stage primaryStage) {  
    Button btn = new Button();  
    btn.setText("Say 'Hello World'");  
  
    StackPane root = new StackPane();  
    VBox box = new VBox();  
    box.getChildren().add(btn);  
    Label label = new Label();  
    box.getChildren().add(label);  
    root.getChildren().add(box);  
  
    btn.setOnAction(new ClickHandler(label));  
  
    Scene scene = new Scene(root, 300, 250);  
  
    primaryStage.setTitle("Hello World!");  
    primaryStage.setScene(scene);  
    primaryStage.show();  
}
```

Practical questions:

1. Write a program that displays five texts vertically, as shown in Figure. Set a random color and opacity for each text and set the font of each text to Times Roman, bold, italic, and 22 pixels.



Code:

```
package application;
```

```
import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.text.*;

import javafx.scene.layout.HBox;

import javafx.scene.paint.Color;

import javafx.stage.Stage;


public class Main extends Application {

    public void start(Stage primaryStage) {

        HBox hBox = new HBox(10);

        hBox.setPadding(new Insets(10, 10, 10, 10));

        hBox.setAlignment(Pos.CENTER);

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

            Text text = new Text("Java");

            text.setFont(Font.font("Times Roman", FontWeight.BOLD,
            FontPosture.ITALIC, 22));

            text.setRotate(90);

            text.setFill(new Color(Math.random(), Math.random(),
            Math.random(), Math.random()));

            hBox.getChildren().add(text);

        }

        Scene scene = new Scene(hBox, 300, 100);

        primaryStage.setTitle("Main");

        primaryStage.setScene(scene);
```

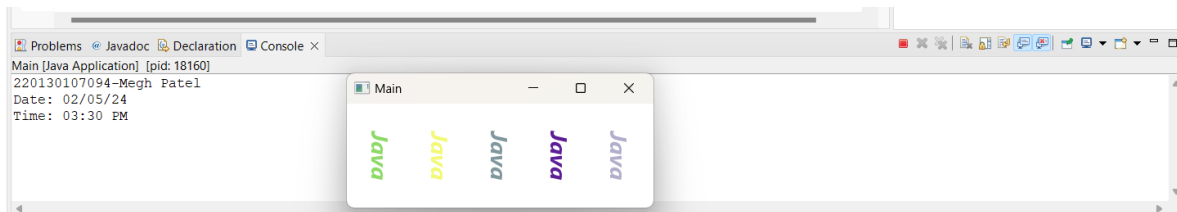
```

        primaryStage.show();
    }

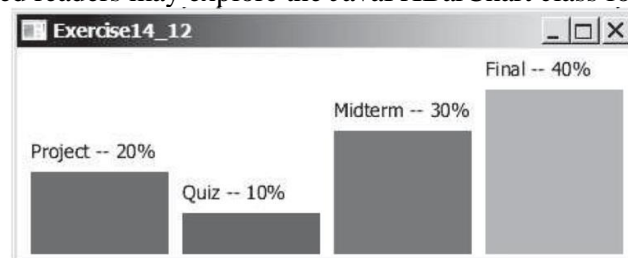
    public static void main(String[] args) {
        System.out.println("220130107094-Megh Patel");
        System.out.println("Date: 02/05/24");
        System.out.println("Time: 03:30 PM");
        System.out.println();
        launch(args);
    }
}

```

Output:



- Write a program that uses a bar chart to display the percentages of the overall grade represented by projects, quizzes, midterm exams, and the final exam, as shown in Figure b. Suppose that projects take 20 percent and are displayed in red, quizzes take 10 percent and are displayed in blue, midterm exams take 30 percent and are displayed in green, and the final exam takes 40 percent and is displayed in orange. Use the Rectangle class to display the bars. Interested readers may explore the JavaFXBarChart class for further study.



Code:

```

package application;

import javafx.application.Application;
import javafx.stage.Stage;

```



```
import javafx.scene.Scene;
import javafx.scene.shape.Rectangle;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.layout.StackPane;
import javafx.geometry.Pos;
import javafx.scene.text.Text;
import javafx.scene.paint.Color;
import javafx.geometry.Insets;

public class Main extends Application {

    public void start(Stage primaryStage) {
        HBox hBox = new HBox(15);
        hBox.setAlignment(Pos.BOTTOM_CENTER);

        String[] type = {"Project", "Quiz", "Midterm", "Final"};
        double[] grade = {20, 10, 30, 40};

        double max = getMax(grade);

        double height = 200;
        double width = 150;

        StackPane pane = new StackPane();
        pane.setPadding(new Insets(20, 15, 5, 15));

        Rectangle r1 = new Rectangle(0, 0, width, height * grade[0] / max);
        r1.setFill(Color.RED);
        Rectangle r2 = new Rectangle(0, 0, width, height * grade[1] / max);
        r2.setFill(Color.BLUE);
        Rectangle r3 = new Rectangle(0, 0, width, height * grade[2] / max);
        r3.setFill(Color.GREEN);
        Rectangle r4 = new Rectangle(0, 0, width, height * grade[3] / max);
        r4.setFill(Color.ORANGE);

        Text t1 = new Text(0, 0, type[0] + " -- " + grade[0] + "%");
        Text t2 = new Text(0, 0, type[1] + " -- " + grade[1] + "%");
        Text t3 = new Text(0, 0, type[2] + " -- " + grade[2] + "%");
        Text t4 = new Text(0, 0, type[3] + " -- " + grade[3] + "%");

        hBox.getChildren().addAll(getVBox(t1, r1), getVBox(t2, r2),
            getVBox(t3, r3), getVBox(t4, r4));
        pane.getChildren().add(hBox);

        Scene scene = new Scene(pane);
        primaryStage.setTitle("Bar Graph");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public double getMax(double[] l) {
        double max = l[0];
```

```

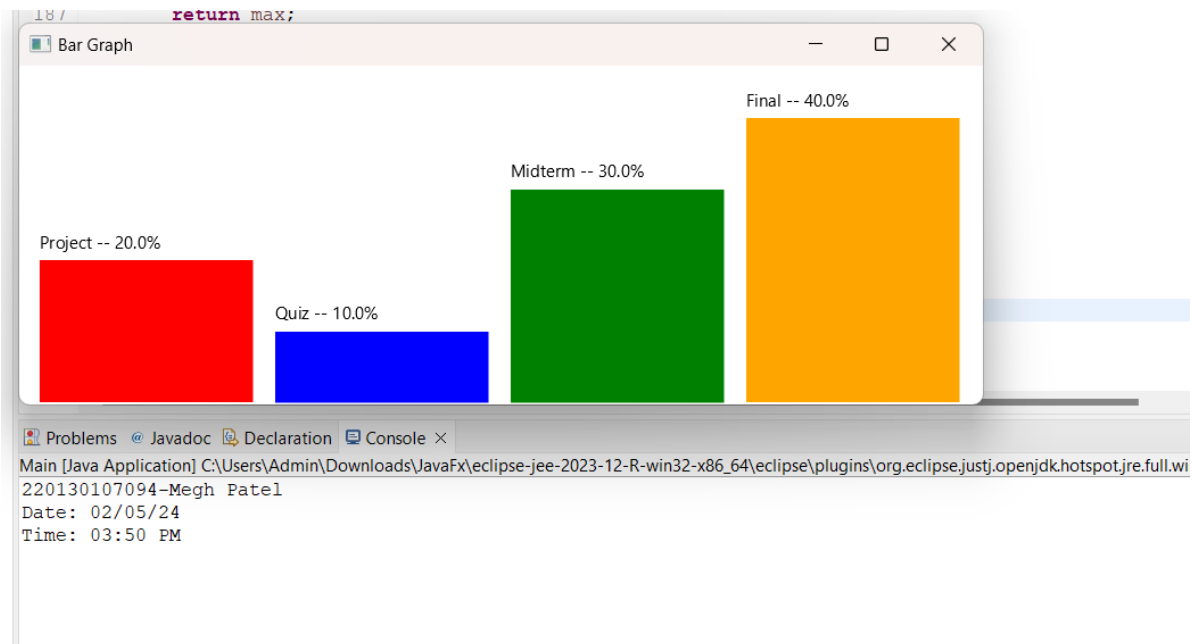
        for (int i = 0; i < l.length; i++) {
            if (l[i] > max)
                max = l[i];
        }
        return max;
    }

    public VBox getVBox(Text t, Rectangle r) {
        VBox vBox = new VBox(5);
        vBox.setAlignment(Pos.BOTTOM_LEFT);
        vBox.getChildren().addAll(t, r);
        return vBox;
    }

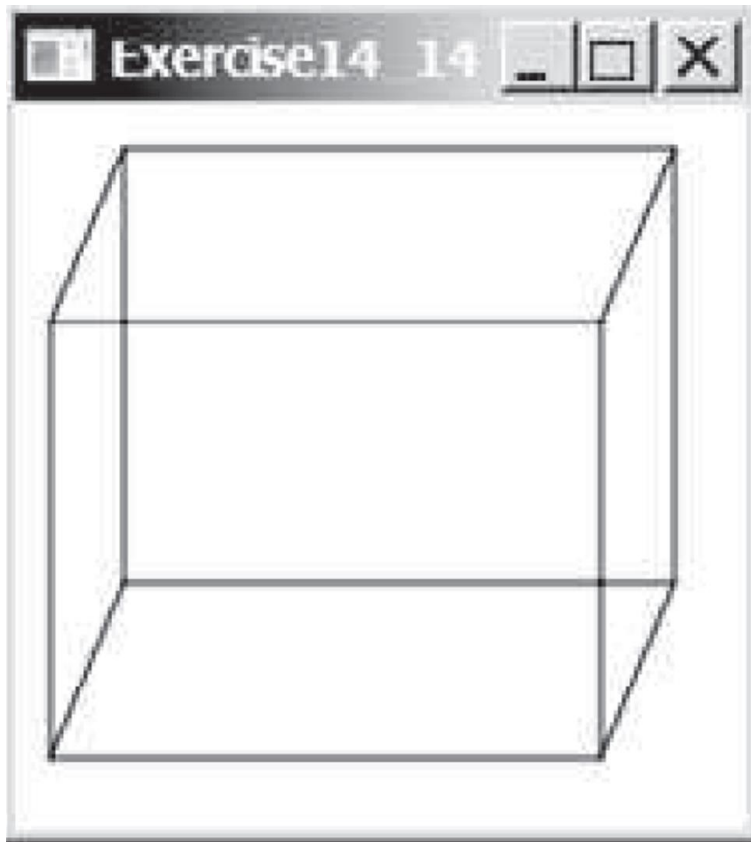
    public static void main(String[] args) {
        System.out.println("220130107094-Megh Patel");
        System.out.println("Date: 02/05/24");
        System.out.println("Time: 03:50 PM");
        System.out.println();
        launch(args);
    }
}

```

Output:



- Write a program that displays a rectanguloid, as shown in Figure a. The cube should grow and shrink as the window grows or shrinks.



Code:

```
package application;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Line;
import javafx.scene.shape.Rectangle;
import javafx.scene.shape.Shape;
import javafx.stage.Stage;

import java.util.ArrayList;

public class Main extends Application {

    public void start(Stage primaryStage) {

        Pane pane = new Pane();

        ArrayList<Shape> shapes = new ArrayList<>();

        Rectangle rec1 = new Rectangle(0,0, Color.TRANSPARENT);
        rec1.setStroke(Color.RED);
        rec1.xProperty().bind(pane.widthProperty().divide(4).subtract(25));
        rec1.yProperty().bind(pane.heightProperty().divide(4).add(25));
```

```
rec1.widthProperty().bind(pane.widthProperty().divide(2));
rec1.heightProperty().bind(pane.heightProperty().divide(2));
shapes.add(rec1);
```

```
Rectangle rec2 = new Rectangle(0,0, Color.TRANSPARENT);
rec2.setStroke(Color.BLACK);
rec2.xProperty().bind(pane.widthProperty().divide(4).add(25));
rec2.yProperty().bind(pane.heightProperty().divide(4).subtract(25));
rec2.widthProperty().bind(pane.widthProperty().divide(2));
rec2.heightProperty().bind(pane.heightProperty().divide(2));
shapes.add(rec2);
```

```
Line line1 = new Line();
line1.startXProperty().bind(rec1.xProperty());
line1.startYProperty().bind(rec1.yProperty());
line1.endXProperty().bind(rec2.xProperty());
line1.endYProperty().bind(rec2.yProperty());
shapes.add(line1);
```

```
Line line2 = new Line();
line2.startXProperty().bind(rec1.xProperty());
line2.startYProperty().bind(rec1.yProperty().add(rec1.heightProperty()));
line2.endXProperty().bind(rec2.xProperty());
line2.endYProperty().bind(rec2.yProperty().add(rec1.heightProperty()));
shapes.add(line2);
```

```
Line line3 = new Line();
line3.startXProperty().bind(rec1.xProperty().add(rec1.widthProperty()));
line3.startYProperty().bind(rec1.yProperty().add(rec1.heightProperty()));
line3.endXProperty().bind(rec2.xProperty().add(rec1.widthProperty()));
line3.endYProperty().bind(rec2.yProperty().add(rec1.heightProperty()));
shapes.add(line3);
```

```
Line line4 = new Line();
line4.startXProperty().bind(rec1.xProperty().add(rec1.widthProperty()));
line4.startYProperty().bind(rec1.yProperty());
line4.endXProperty().bind(rec2.xProperty().add(rec1.widthProperty()));
line4.endYProperty().bind(rec2.yProperty());
shapes.add(line4);
```

```
pane.getChildren().addAll(shapes);
Scene scene = new Scene(pane, 400, 400);
primaryStage.setTitle("Welcome to Java");
primaryStage.setScene(scene);
primaryStage.show();
```

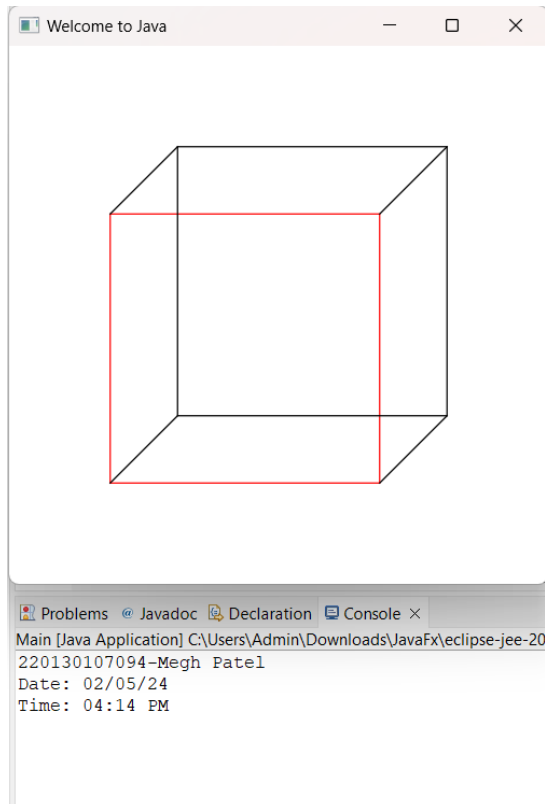
```
}
```

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

```
    System.out.println("220130107094-Megh Patel");
```

```
System.out.println("Date: 02/05/24");  
System.out.println("Time: 04:14 PM");  
System.out.println();  
Application.launch(args);  
  
}  
}
```

Output:



Conclusion:

Quiz: (Sufficient space to be provided for the answers)

1. Explain the evolution of Java GUI technologies since awt, swing and JavaFX.
2. What is the purpose of a TextField control, and how can it be used to collect user input?
3. How to create an ImageView from an Image, or directly from a file or a URL?
4. What are the primary layout controls in JavaFX, and how do they impact the arrangement of UI components?
5. What is CSS, and how is it used for styling JavaFX UI controls?

Suggested Reference:

1. <https://www.tutorialspoint.com/java/>
2. <https://www.geeksforgeeks.org/>
3. <https://www.w3schools.com/java/>
4. <https://www.javatpoint.com/>

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Criteria	Need Improvement	Good	Excellent	Total
Marks	Design of logic (4) Correct output (4) Mock viva test (2)	Program has significant logic errors. (1) Output has multiple errors. (1) Delayed & only few correct answers (1)	Program has slight logic errors that do no significantly affect the results (2) Output has minor errors. (2) Partially correct response (1)	Program is logically well designed (3) Program displays correct output with no errors (3) All questions responded Correctly (2)	

Signature of Faculty:

Experiment No: 9

AIM: To implement event handling and animation.

Date:

CO mapped: CO-5

Objectives:

To proficiently implement event handling and animation in software applications, fostering user interaction and engagement. Mastery of event-driven programming and animation techniques will empower the creation of dynamic, responsive, and visually captivating software experiences that cater to user needs and preferences.

Background:

Responding to user events

For a GUI application to be interactive, various elements such as buttons have to be able to respond to interactions from the user, such as clicks. In GUI applications user actions such as mouse clicks and key presses are called *events*. To set up an element such as a button to respond to user events, we arrange to connect special *event handling* code to the button.

Our first example demonstrates one way to do this in JavaFX. The first step is to connect an object to the button as the button's event handler via the button's `setOnAction()` method. The requirement here is that the object that we link to the button has to implement a particular interface, the `EventHandler<ActionEvent>` interface. That interface has one method in it, a `handle()` method that will get called when the user clicks on the button.

For the event handler code to do something useful, it will typically need to have access to one or more elements in the scene that will be affected by the button click. In this example, clicking the button will trigger a change in the text displayed in a label in the scene. To make this all work, the class we set up needs to have a member variable that is a reference to the label object. The code in `handle()` will use that reference to change the text shown in the label when the user clicks on the button.

Also, insert the code for the following class at the bottom of the `App.java` file:

```
class ClickHandler implements EventHandler<ActionEvent> {  
    public ClickHandler(Label label) {  
        this.label = label;  
    }  
  
    public void handle(ActionEvent evt) {  
        label.setText("Hello, World!");  
    }  
  
    private Label label;
```

```
}

```

A better way to handle events

Although the process for linking an event handler to a button is fairly straightforward, it is a little clunky. This process can get even more tedious when we start building applications with many buttons that need event handlers. As a fix for this, JavaFX allows us to use a simpler mechanism to set up event handlers.

To see how this mechanism works, remove the `ClickHandler` class completely and replace the line of code in `start()` that calls the button's `setOnAction()` method with this:

```
btn.setOnAction((e)->{label.setText("Hello, World!");});
```

The `ClickHandler` class that you just eliminated had a `handle()` method in it. That method took a single parameter, which was an `ActionEvent` object, `e`. The code we just put in place of the original code contains a *lambda expression* mapping that parameter `e` to a chunk of code that will run when the event takes place. This code is the code that used to live in the body of the `handle()` method. The new statement saves a lot of space over the original. The lambda expression replaces the `ClickHandler` class and its `handle()` method with a simpler alternative.

Practical questions:

1. Write a program that can dynamically change the font of a text in a label displayed on a stack pane. The text can be displayed in bold and italic at the same time. You can select the font name or font size from combo boxes, as shown in Figure. The available font names can be obtained using `Font.getFamilies()`. The combo box for the font size is initialized with numbers from 1 to 100.



Code:

```
package application;
```

```
import javafx.application.Application;
```

```
import javafx.geometry.Insets;
```

```
import javafx.geometry.Pos;
```

```
import javafx.scene.Scene;
```



```
import javafx.scene.control.CheckBox;

import javafx.scene.control.ComboBox;

import javafx.scene.control.ContentDisplay;

import javafx.scene.control.Label;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.StackPane;

import javafx.scene.text.Font;

import javafx.scene.text.FontPosture;

import javafx.scene.text.FontWeight;

import javafx.scene.text.Text;

import javafx.stage.Stage;


public class Main extends Application {

    Text text = new Text("Programming is fun");

    ComboBox<String> cbFontFamilies = new ComboBox<>();

    CheckBox chkBold = new CheckBox("Bold");

    ComboBox<Integer> cbFontSize = new ComboBox<>();

    CheckBox chkItalic = new CheckBox("Italic");


    public void start(Stage primaryStage) throws Exception {

        Integer[] sizes = new Integer[100];

        for (int i = 0; i < 100; i++)

            sizes[i] = i + 1;

        cbFontFamilies.getItems().addAll(Font.getFamilies());

        cbFontFamilies.setValue(text.getFont().getFamily());

        cbFontFamilies.setOnAction(e-> update());

        Label lblFontFamilies = new Label("Font Name", cbFontFamilies);

        lblFontFamilies.setContentDisplay(ContentDisplay.RIGHT);
```

```
cbFontSize.getItems().addAll(getSizes());

cbFontSize.setValue((int)text.getFont().getSize());

cbFontSize.setOnAction(e -> {
    update();
    primaryStage.sizeToScene();
});

Label lblFontSizes = new Label("Font Size",cbFontSize);
lblFontSizes.setContentDisplay(ContentDisplay.RIGHT);
HBox topPane = new HBox(lblFontFamilies, lblFontSizes);
topPane.setSpacing(5);
topPane.setPadding(new Insets(5));

chkBold.setOnAction(e-> update());
chkItalic.setOnAction(e-> update());

HBox bottomPane = new HBox(chkBold, chkItalic);
bottomPane.setAlignment(Pos.CENTER);

StackPane centerPane = new StackPane(text);
BorderPane borderPane = new BorderPane(centerPane);
borderPane.setTop(topPane);
borderPane.setBottom(bottomPane);
primaryStage.setTitle("Text Changer");
primaryStage.setScene(new Scene(borderPane));
primaryStage.show();
}

private void update(){
```

```

        FontWeight  fontWeight  = (chkBold.isSelected()) ?  FontWeight.BOLD    :
        FontWeight.NORMAL;

```

```

        FontPosture  fontPosture  = (chkItalic.isSelected()) ?  FontPosture.ITALIC    :
        FontPosture.REGULAR;

```

```

        String fontFamily = cbFontFamilies.getValue();

```

```

        double size = cbFontSize.getValue();

```

```

        text.setFont(Font.font(fontFamily, fontWeight, fontPosture, size));

```

```

    }

```

```

private Integer[] getSizes() {

```

```

    Integer[] sizes = new Integer[100];

```

```

    for (int i = 0; i < 100; i++)

```

```

        sizes[i] = i + 1;

```

```

    return sizes;

```

```

}

```

```

public static void main(String[] args) {

```

```

    System.out.println("220130107094-Megh Patel");

```

```

    System.out.println("Date: 02/05/24");

```

```

    System.out.println("Time: 04:30 PM");

```

```

    System.out.println();

```

```

    Application.launch(args);

```

```

}

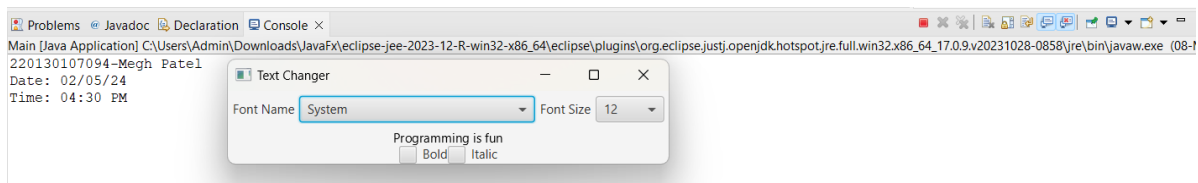
```

```

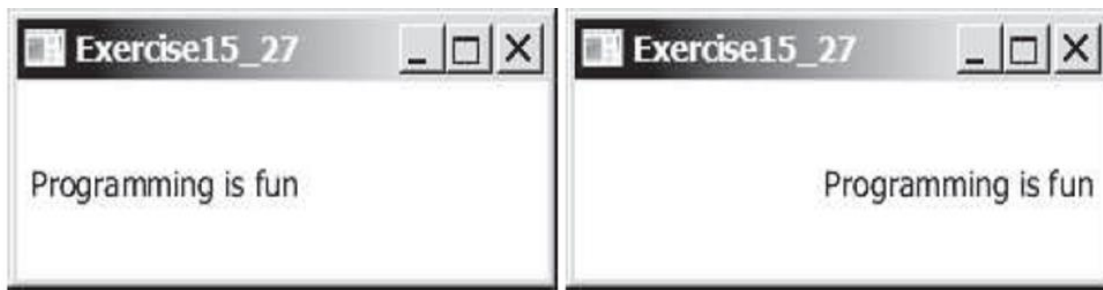
}

```

Output:



- Write a program that displays a moving text, as shown in Figure. The text moves from left to right circularly. When it disappears in the right, it reappears from the left. The text freezes when the mouse is pressed and moves again when the button is released.



Code:

```
package application;

import javafx.application.Application;
import javafx.beans.property.ReadOnlyDoubleProperty;
import javafx.beans.property.ReadOnlyIntegerProperty;
import javafx.animation.PathTransition;
import javafx.animation.Timeline;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.text.Text;
import javafx.scene.shape.Line;
import javafx.stage.Stage;
import javafx.util.Duration;

public class Main extends Application {

    public void start(Stage primaryStage) {
        Pane pane = new Pane();

        Text text = new Text("Programing is fun");
        pane.getChildren().add(text);
        // ReadOnlyDoubleProperty width = pane.widthProperty();
        //
        // int width1 = (int)width.get();

        PathTransition pt = new PathTransition(Duration.millis(10000),
            new Line(-50, 50, 250, 50), text);
        pt.setCycleCount(Timeline.INDEFINITE);
        pt.play();

        pane.setOnMousePressed(e -> {
            pt.pause();
        });

        pane.setOnMouseReleased(e -> {
            pt.play();
        });

        Scene scene = new Scene(pane, 200, 100);
        primaryStage.setTitle("Exercise_15_27");
        primaryStage.setScene(scene);
    }
}
```

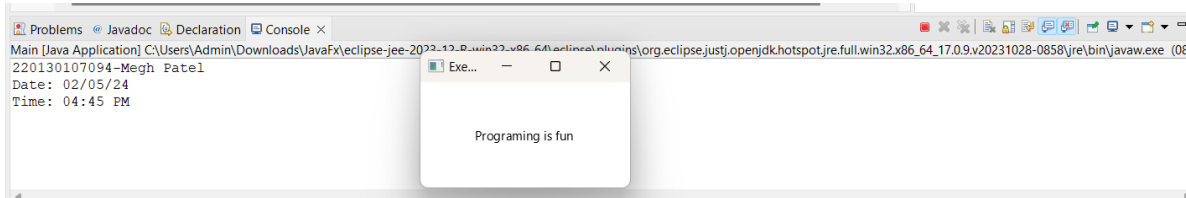
```

        primaryStage.show();
    }

    public static void main(String[] args) {
        System.out.println("220130107094-Megh Patel");
        System.out.println("Date: 02/05/24");
        System.out.println("Time: 04:45 PM");
        System.out.println();
        launch(args);
    }
}

```

Output:



3. Create animation in Figure to meet the following requirements:

- Allow the user to specify the animation speed in a text field.
- Get the number of iamges and image's file-name prefix from the user. For example, if the user enters n for the number of images and L for the image prefix, then the files are L1.gif, L2.gif, and so on, to Ln.gif. Assume that the images are stored in the image directory, a subdirectory of the program's class directory. The animation displays the images one after the other.
- Allow the user to specify an audio file URL. The audio is played while the animation runs.



Code:

```
package application;
```

```
import javafx.animation.KeyFrame;
```

```
import javafx.animation.Timeline;

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.StackPane;

import javafx.scene.media.Media;

import javafx.scene.media.MediaPlayer;

import javafx.stage.Stage;

import javafx.util.Duration;


public class Main extends Application {

    public void start(Stage primaryStage) throws Exception {

        TextAnimationPane pane = new TextAnimationPane();

        primaryStage.setScene(new Scene(pane, 575, 600));

        primaryStage.setTitle("Text Animation");

        primaryStage.show();

    }

    private class TextAnimationPane extends BorderPane {
```

```
Button btStartAnimation = new Button("Start Animation");

long animationSpeed;

int numOfImages;

int currentImgNum = 1;

String url;

String prefix;

String imgDir = "/Image/";

String imgExtension = ".png";


Timeline timeline = null;

StackPane centerPane = new StackPane();

TextAnimationPane() {

    setCenter(centerPane);

    HBox topPane = new HBox(btStartAnimation);

    topPane.setAlignment(Pos.CENTER_RIGHT);

    setTop(topPane);


    GridPane bottomPane = new GridPane();

    bottomPane.setPadding(new Insets(5));

    bottomPane.setHgap(5);

    Label lblInfo = new Label("Enter information for animation");

    bottomPane.add(lblInfo, 0, 0);


    TextField tfAnimationSpeed = new TextField();

    tfAnimationSpeed.setPrefColumnCount(30);

    Label lblAnimationSpeed = new Label("Animation Speed");

    bottomPane.add(lblAnimationSpeed, 0, 1);

    bottomPane.add(tfAnimationSpeed, 1, 1);
```

```
TextField tfImagePrefix = new TextField();

tfImagePrefix.setPrefColumnCount(30);

Label lblImagePrefix = new Label("Image file prefix");

bottomPane.add(lblImagePrefix, 0, 2);

bottomPane.add(tfImagePrefix, 1, 2);


TextField tfNumberOfImages = new TextField();

tfNumberOfImages.setPrefColumnCount(30);

Label lblNumberOfImages = new Label("Number of Images");

bottomPane.add(lblNumberOfImages, 0, 3);

bottomPane.add(tfNumberOfImages, 1, 3);


TextField tfAudioFileUrl = new TextField();

tfAudioFileUrl.setPrefColumnCount(30);

Label lblAudioFileUrl = new Label("Audio file URL");

bottomPane.add(lblAudioFileUrl, 0, 4);

bottomPane.add(tfAudioFileUrl, 1, 4);

setBottom(bottomPane);


btStartAnimation.setOnAction(e -> {

    animationSpeed = Long.parseLong(tfAnimationSpeed.getText().trim());

    prefix = tfImagePrefix.getText().trim();

    numOfImages = Integer.parseInt(tfNumberOfImages.getText());

    url = tfAudioFileUrl.getText();

    initTimeline();

});

}
```

```
private void initTimeline(){
```



```
        if (timeline != null) {

            timeline.stop();

            timeline = null;

            currentImgNum = 1;

        }

        timeline = new Timeline(

            new KeyFrame(Duration.millis(animationSpeed), e-> nextImage());

        MediaPlayer mp = new MediaPlayer(new Media(url));

        mp.play();

        mp.setCycleCount(MediaPlayer.INDEFINITE);

        timeline.setOnFinished(event -> mp.stop());

        timeline.setCycleCount(numOfImages);

        timeline.play();

    }

    private void nextImage(){

        centerPane.getChildren().clear();

        centerPane.getChildren().add(

            new ImageView(new Image(imgDir + prefix + currentImgNum++ +

imgExtension)));

    }

}

public static void main(String[] args) {

    System.out.println("220130107094-Megh Patel");

    System.out.println("Date: 02/05/24");

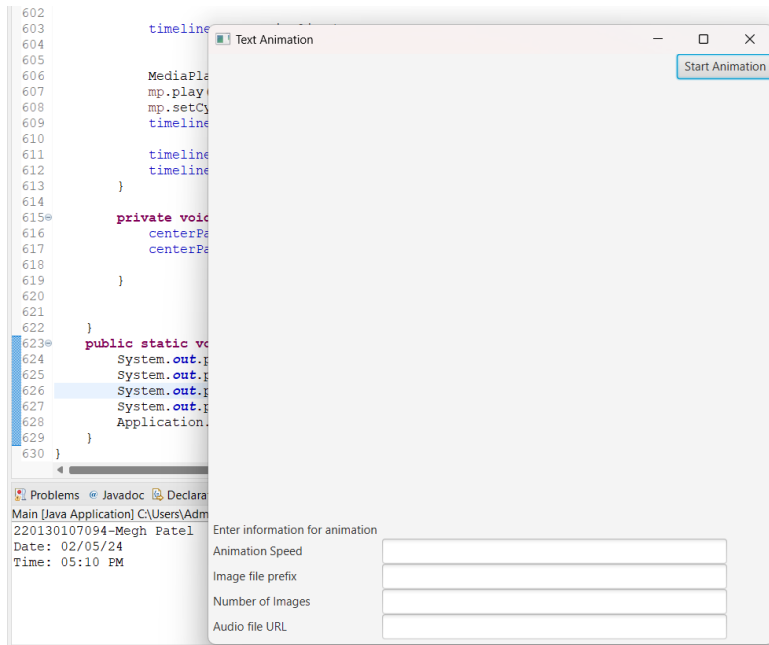
    System.out.println("Time: 05:10 PM");

    System.out.println();

    Application.launch(args);

}

}
```

Output:**Conclusion:****Quiz:** (Sufficient space to be provided for the answers)

1. How does event handling work in Java, and what is the event-driven programming model?
2. What is the role of the java.awt.event and javafx.event packages in Java event handling?
3. Explain: MouseEvent, KeyEvent, ActionEvent
4. What are the primary libraries or frameworks for creating animations in Java, and which one do you prefer?
5. How to set the cycle count of an animation to infinite?

Suggested Reference:

1. <https://www.tutorialspoint.com/java/>
2. <https://www.geeksforgeeks.org/>
3. <https://www.w3schools.com/java/>
4. <https://www.javatpoint.com/>

References used by the students: (Sufficient space to be provided)**Rubric wise marks obtained:**

Rubrics	Criteria	Need Improvement	Good	Excellent	Total
Marks	Design of logic (4) Correct output (4) Mock viva test (2)	Program has significant logic errors. (1) Output has multiple errors. (1) Delayed & only few correct answers (1)	Program has slight logic errors that do no significantly affect the results (2) Output has minor errors. (2) Partially correct response (1)	Program is logically well designed (3) Program displays correct output with no errors (3) All questions responded Correctly (2)	

Signature of Faculty:

Experiment No: 10

AIM: To learn recursion and generics.

Date:

CO mapped: CO-4

Objectives:

- a) To develop a deep understanding of recursion and generics in programming. Mastery of recursion will enable the development of elegant and efficient algorithms for solving complex problems. Understanding generics will facilitate the creation of flexible, reusable, and type-safe code in various programming languages.
- b) Learning recursion and generics is crucial for building efficient algorithms and writing more versatile and type-safe code in software development. Achieving this objective will help you become a more proficient and well-rounded programmer.

Background:

Recursion in java is a process in which a method calls itself continuously. A method in java that calls itself is called the recursive method.

Java Generics programming is introduced in J2SE 5 to deal with type-safe objects. It makes the code stable by detecting the bugs at compile time. Before generics, we can store any type of object in the collection, i.e., non-generic. Now generics force the java programmer to store a specific type of object.

Practical questions:

1. Write a recursive method that converts a decimal number into a binary number as a string. The method header is: `public static String dec2Bin(int value)`

Write a test program that prompts the user to enter a decimal number and displays its binary equivalent.

Code:

```
import java.util.Scanner;

public class DecimalToBinary {

    public static String dec2Bin(int value) {

        if (value == 0) {

            return "0";

        } else {

            return dec2Bin(value / 2) + String.valueOf(value % 2);

        }

    }

}
```

```

    }

    }

    public static void main(String[] args) {

        System.out.println("220130107094 - Megh Patel");

        System.out.println("Date: 18/04/24");

        System.out.println("Time: 03:25 PM");

        System.out.println();

        System.out.print("Enter a decimal number: ");

        Scanner scanner = new Scanner(System.in);

        int decimal = scanner.nextInt();

        String binary = dec2Bin(decimal);

        System.out.println(decimal + " in binary is " + binary);

    }

}

```

Output:

```

Enter a decimal number: 94
94 in binary is 01011110
PS D:\Programmes\Java\Practicle - 10>

```

2. Write the following method that returns a new ArrayList. The new list contains the non-duplicate elements from the original list.

```
public static <E>ArrayList<E>removeDuplicates(ArrayList<E> list)
```

Code:

```

import java.util.ArrayList;
import java.util.HashSet;

public class RemoveDuplicates {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 18/04/24");
        System.out.println("Time: 03:41 PM");
        System.out.println();

        ArrayList<Integer> orgList = new ArrayList<>();
        orgList.add(1);
        orgList.add(2);
        orgList.add(3);
        orgList.add(1);
    }
}

```

```

        orgList.add(4);
        orgList.add(2);
        orgList.add(5);

        ArrayList<Integer> newList = removeDuplicates(orgList);

        System.out.println("Original List: " + orgList);
        System.out.println("List with Duplicates Removed: " + newList);
    }

    public static <E> ArrayList<E> removeDuplicates(ArrayList<E> list) {
        HashSet<E> set = new HashSet<>(list);
        return new ArrayList<>(set);
    }
}

```

Output:

```

Original List: [1, 2, 3, 1, 4, 2, 5]
List with Duplicates Removed: [1, 2, 3, 4, 5]
PS D:\Programmes\Java\Practicle - 10>

```

3. Implement the following method using binary search.

```
public static <E extends Comparable<E>>
```

```
int binarySearch(E list, E key)
```

Code:

```

import java.util.List;
import java.util.ArrayList;

public class BinarySearch {

    public static <E extends Comparable<E>> int binarySearch(List<E> list, E key) {
        int low = 0;
        int high = list.size() - 1;

        while (low <= high) {
            int mid = (low + high) / 2;
            int comparison = key.compareTo(list.get(mid));

            if (comparison == 0) {
                return mid;
            } else if (comparison < 0) {
                high = mid - 1;
            } else {
                low = mid + 1;
            }
        }

        return -1;
    }
}

```

```
}

public static void main(String[] args) {
    System.out.println("220130107094 - Megh Patel");
    System.out.println("Date: 18/04/24");
    System.out.println("Time: 03:55 PM");
    System.out.println();

    ArrayList<Integer> nums = new ArrayList<>();
    nums.add(1);
    nums.add(2);
    nums.add(3);
    nums.add(4);
    nums.add(5);

    System.out.println("Original list: " + nums);

    int key = 4;
    int index = binarySearch(nums, key);

    if (index != -1) {
        System.out.println("Key " + key + " found at index: " + index);
    } else {
        System.out.println("Key " + key + " not found in the list.");
    }
}
}
```

Output:

```
Original list: [1, 2, 3, 4, 5]
Key 4 found at index: 3
PS D:\Programmes\Java\Practicle - 10>
```

Conclusion:

Quiz: (Sufficient space to be provided for the answers)

1. What is recursion in Java, and how does it differ from iteration in solving problems?
2. What are the advantages and disadvantages of using recursion in Java?
3. What are generics in Java, and why are they used for creating parameterized types?
4. How to define Generic class? What are restrictions of generic programming?
5. Can you provide an example of a generic class in Java, such as a generic ArrayList?

Suggested Reference:

1. <https://www.tutorialspoint.com/java/>
2. <https://www.geeksforgeeks.org/>
3. <https://www.w3schools.com/java/>
4. <https://www.javatpoint.com/>

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Criteria	Need Improvement	Good	Excellent	Total
Marks	Design of logic (4) Correct output (4) Mock viva test (2)	Program has significant logic errors. (1) Output has multiple errors. (1) Delayed & only few correct answers (1)	Program has slight logic errors that do not significantly affect the results (2) Output has minor errors. (2) Partially correct response (1)	Program is logically well designed (3) Program displays correct output with no errors (3) All questions responded Correctly (2)	

Signature of Faculty:

Experiment No: 11

AIM: To demonstrate the use of Collection framework.

Date:

CO mapped: CO-4

Objectives:

- a) To proficiently demonstrate the use of Java's Collection framework, including understanding its core interfaces (List, Set, Map), implementing and manipulating data structures like lists, sets, and maps, and effectively applying collections for data storage, retrieval, and manipulation in Java applications.
- b) Mastery of the Java Collection framework is essential for managing and organizing data efficiently in Java applications. This objective focuses on understanding the core collection interfaces and using them to build versatile data structures to meet various application needs.

Background:

The Collection in Java is a framework that provides architecture to store and manipulate a group of objects. Java Collections can achieve all the operations that you perform on 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, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

Practical questions:

1. Write a program that lets the user enter numbers from a graphical user interface and displays them in a text area, as shown in Figure. Use a linked list to store the numbers. Do not store duplicate numbers. Add the buttons Sort, Shuffle, and Reverse to sort, shuffle, and reverse the list.



2. Create two priority queues, {"George", "Jim", "John", "Blake", "Kevin", "Michael"} and {"George", "Katie", "Kevin", "Michelle", "Ryan"}, and find their union, difference, and intersection.

Code:

```
import java.util.PriorityQueue;
import java.util.Set;
import java.util.HashSet;

public class PriorityQueueOperations {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 18/04/24");
        System.out.println("Time: 04:20 PM");
        System.out.println();
    }
}
```

```

PriorityQueue<String> queue1 = new PriorityQueue<>();
queue1.add("George");
queue1.add("Jim");
queue1.add("John");
queue1.add("X-Men");
queue1.add("Kevin");
queue1.add("Michael");

PriorityQueue<String> queue2 = new PriorityQueue<>();
queue2.add("George");
queue2.add("Jim");
queue2.add("Kevin");
queue2.add("Michelle");
queue2.add("Deadpool");

PriorityQueue<String> union = new PriorityQueue<>(queue1);
union.addAll(queue2);

PriorityQueue<String> difference = new PriorityQueue<>(queue1);
difference.removeAll(queue2);

PriorityQueue<String> intersection = new PriorityQueue<>(queue1);
intersection.retainAll(queue2);

System.out.println("Union: " + union);
System.out.println("Difference: " + difference);
System.out.println("Intersection: " + intersection);
}
}

```

Output:

```

Union: [Deadpool, George, George, Jim, Jim, Michael, John, X-Men, Kevin, Michelle, Kevin]
Difference: [John, X-Men, Michael]
Intersection: [George, Jim, Kevin]
PS D:\Programmes\Java\Practicle - 11>

```

3. Store pairs of 10 states and its capital in a map. Your program should prompt the user to enter a state and should display the capital for the state.

Code:

```

import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class StorePairs {
    public static void main(String[] args) {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 18/04/24");
        System.out.println("Time: 04:40 PM");
        System.out.println();
    }
}

```

```
Map<String, String> stateCapitalMap = new HashMap<>();
stateCapitalMap.put("DELHI", "Delhi");
stateCapitalMap.put("HIMACHAL PRADESH", "Manali");
stateCapitalMap.put("RAJASTHAN", "Jaipur");
stateCapitalMap.put("GUJARAT", "Gandhinagar");
stateCapitalMap.put("MADHYA PRADESH", "Indore");
stateCapitalMap.put("MAHARASTRA", "Mumbai");
stateCapitalMap.put("TELANGANA", "Hyderabad");
stateCapitalMap.put("TAMIL NADU", "Chennai");
stateCapitalMap.put("KARNATAKA", "Bengaluru");
stateCapitalMap.put("ANDHRA PRADESH", "Vizag");

Scanner scanner = new Scanner(System.in);
System.out.print("Enter a state: ");
String state = scanner.nextLine().toUpperCase();

String capital = stateCapitalMap.get(state.toUpperCase());
if (capital != null) {
    System.out.println("Capital of " + state + " is " + capital);
} else {
    System.out.println("Capital for the entered state is not found.");
}
scanner.close();
}
}
```

Output:

```
Enter a state: DELHI
Capital of DELHI is Delhi
PS D:\Programmes\Java\Practicle - 11>
```

Conclusion:

Quiz: (Sufficient space to be provided for the answers)

1. Write a note on 'Collection in JAVA'. Also discuss List and Enumeration Interface.
2. Differentiate between Enumeration and Iterator.
3. Compare List, Set and Map interfaces. Also compare ArrayList, TreeSet and HashMap classes in java.
4. Explain the unique features of Map interface.
5. How do you perform common operations like sorting, searching, or filtering on Collections?

Suggested Reference:

1. <https://www.tutorialspoint.com/java/>
2. <https://www.geeksforgeeks.org/>
3. <https://www.w3schools.com/java/>
4. <https://www.javatpoint.com/>

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Criteria	Need Improvement	Good	Excellent	Total
Marks	Design of logic (4) Correct output (4) Mock viva test (2)	Program has significant logic errors. (1) Output has multiple errors. (1) Delayed & only few correct answers (1)	Program has slight logic errors that do not significantly affect the results (2) Output has minor errors. (2) Partially correct response (1)	Program is logically well designed (3) Program displays correct output with no errors (3) All questions responded Correctly (2)	

Signature of Faculty:

Experiment No: 12

AIM: To demonstrate the use of multithreading.

Date:

CO mapped: CO-3

Objectives:

- a) To effectively demonstrate the use of multithreading in software applications, including creating and managing multiple threads, synchronizing their execution, and leveraging the power of concurrent programming to improve performance, responsiveness, and resource utilization.
- b) Demonstrating the use of multithreading is crucial for building responsive and efficient software applications, and this objective emphasizes understanding the concepts and practical implementation of multithreading to achieve these goals.

Background:

Multithreading in Java is a process of executing multiple threads simultaneously. A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking. However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory areas so saves memory, and context-switching between the threads takes less time than the process. Java Multithreading is mostly used in games, animation, etc.

Practical questions:

1. Write a program to create a thread extending Thread class and demonstrate the use of sleep() method.

Code:

```
public class Sleep extends Thread {

    private long sleepTime;

    public Sleep(long sleepTime) {
        this.sleepTime = sleepTime;
    }

    public void run() {
        System.out.println("Thread started: " + getName());
        try {
            System.out.println("Thread going to sleep for " + sleepTime + " milliseconds.");
            sleep(sleepTime); // Pause execution for sleepTime milliseconds
            System.out.println("Thread woke up after sleeping for " + sleepTime + " milliseconds.");
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted " + e);
        }
    }
}
```

```

public static void main(String[] args) {
    System.out.println("220130107094 - Megh Patel");
    System.out.println("Date: 25/04/24");
    System.out.println("Time: 03:40 PM");
    System.out.println();

    Sleep thread1 = new Sleep(1000);
    Sleep thread2 = new Sleep(2000);

    thread1.start();
    thread2.start();
}
}

```

Output:

```

Thread started: Thread-1
Thread started: Thread-0
Thread going to sleep for 2000 milliseconds.
Thread going to sleep for 1000 milliseconds.
Thread woke up after sleeping for 1000 milliseconds.
Thread woke up after sleeping for 2000 milliseconds.
PS D:\Programmes\Java\Practicle - 12>

```

2. Write a program to create a thread implementing Runnable interface and demonstrate the use of join() method.

Code:

```

public class Join implements Runnable {

    private String name;

    public Join(String name) {
        this.name = name;
    }

    public void run() {
        System.out.println("Thread " + name + " started.");
        try {
            for (int i = 1; i <= 5; i++) {
                System.out.println("Thread " + name + ": " + i);
                Thread.sleep(1000); // Sleep for 1 second
            }
        } catch (InterruptedException e) {
            System.out.println("Thread " + name + " interrupted.");
        }
        System.out.println("Thread " + name + " finished.");
    }

    public static void main(String[] args) throws InterruptedException {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 25/04/24");
        System.out.println("Time: 03:52 PM");
        System.out.println();
    }
}

```

```

        Join thread1 = new Join("Thread 1");
        Join thread2 = new Join("Thread 2");

        Thread t1 = new Thread(thread1);
        Thread t2 = new Thread(thread2);
        t1.start();
        t2.start();

        t1.join();

        System.out.println("Thread 1 finished, continuing with main thread.");
    }
}

```

Output:

```

Thread Thread 2 started.
Thread Thread 1 started.
Thread Thread 1: 1
Thread Thread 2: 1
Thread Thread 1: 2
Thread Thread 2: 2
Thread Thread 1: 3
Thread Thread 2: 3
Thread Thread 1: 4
Thread Thread 2: 4
Thread Thread 2: 5
Thread Thread 1: 5
Thread Thread 2 finished.
Thread Thread 1 finished.
Thread 1 finished, continuing with main thread.
PS D:\Programmes\Java\Practice - 12>

```

- Write a program that launches 10 threads. Each thread adds 1 to a variable sum that initially is 0. Define an Integer wrapper object to hold sum. Run the program with and without synchronization to see its effect.

Code:

```

class Counter {

    private Integer sum = 0;

    public synchronized void increment() {
        sum++;
    }

    public int getSum() {
        return sum;
    }
}

public class Threads {

    public static void main(String[] args) throws InterruptedException {
        System.out.println("220130107094 - Megh Patel");
        System.out.println("Date: 25/04/24");
    }
}

```

```
System.out.println("Time: 04:30 PM");
System.out.println();
Counter counter = new Counter();

Thread[] threads = new Thread[10];
for (int i = 0; i < 10; i++) {
    threads[i] = new Thread(() -> counter.increment());
    threads[i].start();
}

for (Thread thread : threads) {
    thread.join();
}

System.out.println("Synchronized Sum: " + counter.getSum());
}
```

Output:

```
Synchronized Sum: 10
PS D:\Programmes\Java\Practicle - 12>
```

Conclusion:

Quiz: (Sufficient space to be provided for the answers)

1. Can you explain the difference between a process and a thread in the context of multithreading?
2. What are the different states in the lifecycle of a Java thread, and how does a thread transition between them?
3. What is runnable interface? How can you use this interface in creating thread?
4. Explain the concept of thread synchronization and the role of the synchronized keyword.
5. Explain: wait, sleep, notify and notify all.

Suggested Reference:

1. <https://www.tutorialspoint.com/java/>
2. <https://www.geeksforgeeks.org/>
3. <https://www.w3schools.com/java/>
4. <https://www.javatpoint.com/>

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Criteria	Need Improvement	Good	Excellent	Total
Marks	Design of logic (4) Correct output (4) Mock viva test (2)	Program has significant logic errors. (1) Output has multiple errors. (1) Delayed & only few correct answers (1)	Program has slight logic errors that do no significantly affect the results (2) Output has minor errors. (2) Partially correct response (1)	Program is logically well designed (3) Program displays correct output with no errors (3) All questions responded Correctly (2)	

Signature of Faculty: