# Diploma Engineering

## Laboratory Manual

## (*Advanced Object-Oriented Programming*)
## (*4340701*)

[Computer Engineering, Semester IV]

| | |
|---|---|
| **Enrolment No** | |
| **Name** | |
| **Semester-Division** | |
| **Academic Term** | |
| **Institute** | |

**Directorate of Technical Education**

**Gandhinagar, Gujarat**

## DTE's Vision:

- To provide globally competitive technical education;
- Remove geographical imbalances and inconsistencies;
- Develop student friendly resources with a special focus on girls' education and support to weaker sections;
- Develop programs relevant to industry and create a vibrant pool of technical professionals.

## Institute's Vision:

## Institute's Mission:

## Department's Vision:

## Department's Mission:

**GOVERNMENT POLYTECHNIC PORBANDAR (627)**

*affiliated to*

*Gujarat Technological University, Ahmedabad*

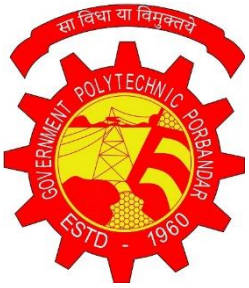
**COMPUTER ENGINEERING DEPARTMENT (07)**

# LAB MANUAL

## *Advanced Object Oriented Programming (4340701)*

**D.E. Second Year (Semester-IV)**

# GOVERNMENT POLYTECHNIC PORBANDAR




## COMPUTER ENGINEERING DEPARMENT

## ***Certificate***

*This is to certify that Mr. / Ms* _____

_____ *Enrollment No.* _____ *of* ***4th*** *Semester of* ***Diploma in Computer Engineering*** *of* ***Government Polytechnic Porbandar (627)*** *has satisfactorily completed the term work in course* ***Advanced Object Oriented Programming (4340701)*** *for the academic year:* ***2023-2024*** *Term:* ***Even*** *prescribed in the GTU curriculum.*

***Place:*** _____

***Date:*** _____

***Signature of Course Faculty*** ***Head of the Department***

# PREFACE

The primary aim of any laboratory/Practical/field work is enhancement of required skills as well as creative ability amongst students to solve real time problems by developing relevant competencies in psychomotor domain. Keeping in view, GTU has designed competency focused outcome-based curriculum -2021 (COGC-2021) for Diploma engineering programmes. In this more time is allotted to practical work than theory. It shows importance of enhancement of skills amongst students and it pays attention to utilize every second of time allotted for practical amongst Students, Instructors and Lecturers to achieve relevant outcomes by performing rather than writing practice in study type. It is essential for effective implementation of competency focused outcome- based Green curriculum-2021. Every practical has been keenly designed to serve as a tool to develop & enhance relevant industry needed competency in each and 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 has been 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 read procedure one day in advance to actual performance day of practical experiment which generates interest and also, they can have idea of judgement of magnitude prior to performance. This in turn enhances predetermined outcomes amongst students. Each and every Experiment /Practical in this manual begins by competency, industry relevant skills, course outcomes as well as practical outcomes which serve as a key role for doing the practical. The students will also have a clear idea of safety and necessary precautions to be taken while performing experiment.

This manual also provides guidelines to lecturers to facilitate student-centred lab activities for each practical/experiment by arranging and managing necessary resources in order that the students follow the procedures with required safety and necessary precautions to achieve outcomes. It also gives an idea that how students will be assessed by providing Rubrics.

Advanced Object-Oriented Programming course provides platform to learn object-oriented programming concepts, techniques, and applications using the Java programming language. Object-oriented programming emphasis on the fundamentals of the structured design with classes, including development, testing, implementation and documentation also includes object-oriented programming techniques, classes and objects. Java is a simple, portable, distributive, robust, secure, dynamic, architecture neutral, object-oriented programming language.

Although we try our level best to design this lab manual, but always there are chances of improvement. We welcome any suggestions for improvement.

## Programme Outcomes (POs) to be achieved through Practical of this Course

Following programme outcomes are expected to be achieved through the practical of the course:

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
2. **Problem analysis**: Identify and analyze well-defined engineering problems using codified standard methods
3. **Design/ development of solutions** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs
4. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements
5. **Engineering practices for society, sustainability and environment:** Apply appropriate technology in context of society, sustainability, environment and ethical practices.
6. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
7. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

## Practical Outcome - Course Outcome matrix

**Course Outcomes (COs):**

1. Write simple java programs for a given problem statement.
2. Use object oriented programming concepts to solve real world problems.
3. Develop an object-oriented program using inheritance and package concepts for a given problem statement.
4. Develop an object oriented program using multithreading and exception handling for a given problem statement.
5. Develop an object-oriented program by using the files and collection framework

| Sr. No. | Practical Outcome | CO1 | CO2 | CO3 | CO4 | CO5 |
|---|---|---|---|---|---|---|
| **1** | Install JDK, write a simple "Hello World" or similar java program, compilation, debugging, executing using java compiler and interpreter. | **√** | - | - | - | - |
| **2** | Write a program in Java to find maximum of three numbers using conditional operator. | **√** | - | - | - | - |
| **3** | Write a program in Java to reverse the digits of a number using while loop | **√** | - | - | - | - |
| **4** | Write a program in Java to add two 3*3 matrices. | **√** | - | - | - | - |
| **5** | Write a program in Java to generate first n prime numbers. | **√** | - | - | - | - |
| **6** | Write a program in Java which has a class Student having two instance variables enrollmentNo and name. Create 3 objects of Student class in main method and display student's name. | - | **√** | - | - | - |
| **7** | *Write a program in Java which has a class Rectangle having two instance variables height and weight. Initialize the class using constructor. | - | **√** | - | - | - |
| **8** | Write a program in Java demonstrate the use of "this" keyword. | - | **√** | - | - | - |
| **9** | Write a program in Java to demonstrate the use of "static" keyword. | - | **√** | - | - | - |
| **10** | Write a program in Java to demonstrate the use of "final" keyword. | - | **√** | - | - | - |
| **11** | Write a program in Java which has a class Shape having 2 overloaded methods area(float radius) and area(float length, float width). Display the area of circle and rectangle using overloaded methods. | - | **√** | - | - | - |
| **12** | Write a program in Java to demonstrate the constructor overloading. | - | **√** | - | - | - |
| **13** | Write a java program to demonstrate use of "String" class methods : chatAt(), contains(), format(), length(), split() | - | **√** | - | - | - |
| **14** | Write a program in Java to demonstrate single inheritance | - | - | **√** | - | - |
| **15** | Write a program in Java to demonstrate multilevel inheritance | - | - | **√** | - | - |
| **16** | Write a program in Java to demonstrate hierarchical inheritance. | - | - | **√** | - | - |
| **17** | Write a program in Java to demonstrate method overriding. | - | - | **√** | - | - |

| Sr. No. | Practical Outcome | CO1 | CO2 | CO3 | CO4 | CO5 |
|---|---|---|---|---|---|---|
| **18** | Write a program in Java which has a class Car having two instance variables topSpeed and name. Override toString() method in Car class. Create 5 instances of Car class and print the instances. | - | - | **√** | - | - |
| **19** | Write a program in Java to implement multiple inheritance using interfaces. | - | - | **√** | - | - |
| **20** | Write a program in Java which has an **abstract** class **Shape** having three subclasses: **Triangle**, **Rectangle**, **and Circle**. Define method **area()** in the abstract class Shape and override **area()** method to calculate the area. | - | - | **√** | - | - |
| **21** | Write a program in Java to demonstrate use of final class. | - | - | **√** | - | - |
| **22** | Write a program in Java to demonstrate use of package. | - | - | **√** | - | - |
| **23** | Write a program in Java to develop user defined exception for 'Divide by Zero' error. | - | - | - | **√** | - |
| **24** | Write a program in Java to develop Banking Application in which user deposits the amount Rs. 25000 and then start withdrawing of Rs. 20000, Rs. 4000 and it throws exception "Not Sufficient Fund" when user withdraws Rs. 2000 thereafter. | - | - | - | **√** | - |
| **25** | Write a program that executes two threads. One thread displays “Thread1” every 1000 milliseconds, and the other displays “Thread2” every 2000 milliseconds. Create the threads by extending the **Thread** class | - | - | - | **√** | - |
| **26** | Write a program that executes two threads. One thread will print the even numbers and another thread will print odd numbers from 1 to 200. | - | - | - | **√** | - |
| **27** | Write a program in Java to perform read and write operations on a Text file. | - | - | - | - | **√** |
| **28** | Write a program in Java to demonstrate use of List.<br>1) Create ArrayList and add weekdays (in string form)<br>2) Create LinkedList and add months (in string form)<br>Display both List. | - | - | - | - | **√** |
| **29** | Write a program in Java to create a new HashSet, add colors(in string form) and iterate through all elements using for-each loop to display the collection. | - | - | - | - | **√** |
| **30** | Write a Java program to create a new HashMap, add 5 students’ data (enrolment no and name). retrieve and display the student’s name from HashMap using enrolment no. | - | - | - | - | **√** |

## Industry Relevant Skills

The following industry relevant skills of the competency "**Develop java application using object-oriented approach.**" are expected to be developed in the student by undertaking the practical of this laboratory manual.

1. Develop java application using object oriented programming
2. Develop java application using exception handling, Multi-threading
3. Develop java application using java io and collections

## Guidelines to Course Faculty

1. Couse faculty should demonstrate experiment with all necessary implementation strategies described in curriculum.
2. Couse faculty should explain industrial relevance before starting of each experiment.
3. Course faculty should Involve & give opportunity to all students for hands on experience.
4. Course faculty should ensure mentioned skills are developed in the students by asking.
5. Utilise 2 hrs of lab hours effectively and ensure completion of write up with quiz also.
6. Encourage peer to peer learning by doing same experiment through fast learners.

## Instructions for Students

1. Organize the work in the group and make record of all observations.
2. Students shall develop coding standard skill as expected by industries.
3. Student shall attempt to develop related hand-on skills and build confidence.
4. Student shall develop the habits of evolving more ideas, innovations, skills etc.
5. Student shall refer technical magazines and data books.
6. Student should develop habit to submit the practical on date and time.
7. Student should well prepare while submitting write-up of exercise.

## ASSESSMENT RUBRICS

| Excellent (8-10 Marks) | Good (5-7 Marks) | Satisfactory (2-4 Marks) | Poor (0-1 Marks) |
|---|---|---|---|
| 1) Student executes the program with correct output.<br>2) Student submits the lab report in specified time limit. | 1) Student executes the program having correct output with external guidance.<br>2) Student submits the lab report in specified time limit. | 1) Student executes the program with incorrect output.<br>2) Student submits the lab report in specified time limit. | 1) Student is not able to execute the program.<br>2) Student does not submit the lab report in specified time limit. |

# ASSESSMENT SHEET

| SR. NO. | PRACTICAL OUTCOME | PAGE | DATE | MARKS (10) | SIGN |
|---|---|---|---|---|---|
| 1 | Install JDK, write a simple “Hello World” or similar java program, compilation, debugging, executing using java compiler and interpreter. | | | | |
| 2 | Write a program in Java to find maximum of three numbers using conditional operator. | | | | |
| 3 | Write a program in Java to reverse the digits of a number using while loop | | | | |
| 4 | Write a program in Java to add two 3*3 matrices. | | | | |
| 5 | Write a program in Java to generate first n prime numbers. | | | | |
| 6 | Write a program in Java which has a class Student having two instance variables enrollmentNo and name. Create 3 objects of Student class in main method and display student’s name. | | | | |
| 7 | *Write a program in Java which has a class Rectangle having two instance variables height and weight. Initialize the class using constructor. | | | | |
| 8 | Write a program in Java demonstrate the use of “this” keyword. | | | | |
| 9 | Write a program in Java to demonstrate the use of “static” keyword. | | | | |
| 10 | Write a program in Java to demonstrate the use of "final" keyword. | | | | |
| 11 | Write a program in Java which has a class Shape having 2 overloaded methods area(float radius) and area(float length, float width). Display the area of circle and rectangle using overloaded methods. | | | | |
| 12 | Write a program in Java to demonstrate the constructor overloading. | | | | |

| SR. NO. | PRACTICAL OUTCOME | PAGE | DATE | MARKS (10) | SIGN |
|---|---|---|---|---|---|
| **13** | Write a java program to demonstrate use of “String” class methods : chatAt(), contains(), format(), length(), split() | | | | |
| **14** | Write a program in Java to demonstrate single inheritance | | | | |
| **15** | Write a program in Java to demonstrate multilevel inheritance | | | | |
| **16** | Write a program in Java to demonstrate hierarchical inheritance. | | | | |
| **17** | Write a program in Java to demonstrate method overriding. | | | | |
| **18** | Write a program in Java which has a class Car having two instance variables topSpeed and name. Override toString() method in Car class. Create 5 instances of Car class and print the instances. | | | | |
| **19** | Write a program in Java to implement multiple inheritance using interfaces. | | | | |
| **20** | Write a program in Java which has an **abstract** class **Shape** having three subclasses: **Triangle**, **Rectangle**, **and Circle**. Define method **area()** in the abstract class Shape and override **area()** method to calculate the area. | | | | |
| **21** | Write a program in Java to demonstrate use of final class. | | | | |
| **22** | Write a program in Java to demonstrate use of package. | | | | |
| **23** | Write a program in Java to develop user defined exception for 'Divide by Zero' error. | | | | |
| **24** | Write a program in Java to develop Banking Application in which user deposits the amount Rs. 25000 and then start withdrawing of Rs. 20000, Rs. 4000 and it throws exception "Not Sufficient Fund" when user withdraws Rs. 2000 thereafter. | | | | |

| SR. NO. | PRACTICAL OUTCOME | PAGE | DATE | MARKS (10) | SIGN |
|---|---|---|---|---|---|
| **25** | Write a program that executes two threads. One thread displays “Thread1” every 1000 milliseconds, and the other displays “Thread2” every 2000 milliseconds. Create the threads by extending the **Thread** class | | | | |
| **26** | Write a program that executes two threads. One thread will print the even numbers and another thread will print odd numbers from 1 to 200. | | | | |
| **27** | Write a program in Java to perform read and write operations on a Text file. | | | | |
| **28** | Write a program in Java to demonstrate use of List.<br>Create ArrayList and add weekdays (in string form)<br>Create LinkedList and add months (in string form)<br>Display both List. | | | | |
| **29** | Write a program in Java to create a new HashSet, add colors(in string form) and iterate through all elements using for-each loop to display the collection. | | | | |
| **30** | Write a Java program to create a new HashMap, add 5 students’ data (enrolment no and name). retrieve and display the student’s name from HashMap using enrolment no. | | | | |

# PRACTICAL-1

**Aim:** Install JDK, write a simple "Hello World" or similar java program, compilation, debugging, executing using java compiler and interpreter.

A. **Objective:** Print "Hello World" on console

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
   - **PO7: Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes in field of engineering.

C. **Expected Skills to be developed based on competency:** The practical is expected to develop the following skills:
   a) **Develop Applications using Java**
      - Set up Java Environment for executing Java programs.
      - Execute simple program by setting path variable.
   b) **Setup a java programming development environment.**
      - Using Command prompt.
      - Using IDEs.

D. **Expected Course Outcomes(COs)**
   - **CO1:** Write simple java programs for a given problem statement.

E. **Practical Outcome(PrO):** Install JDK, write a simple "Hello World" or similar java program, compilation, debugging, executing using java compiler and interpreter.

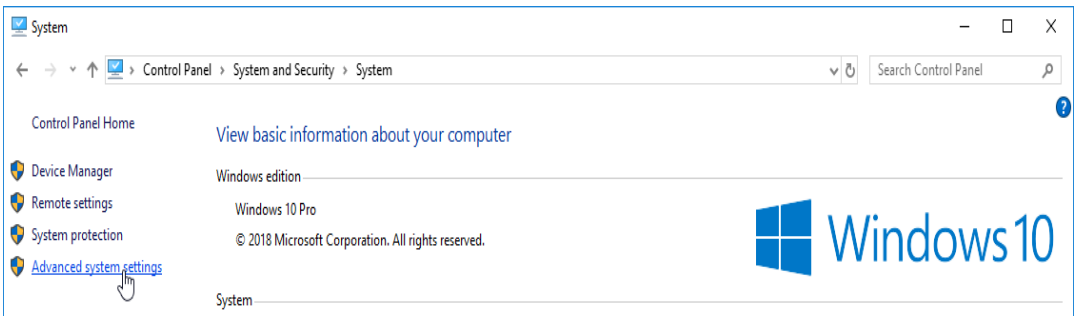
F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

G. **Prerequisite Theory:**
   - JDK Installation
   - Java compiler Path configuration
   - Install Text Editor

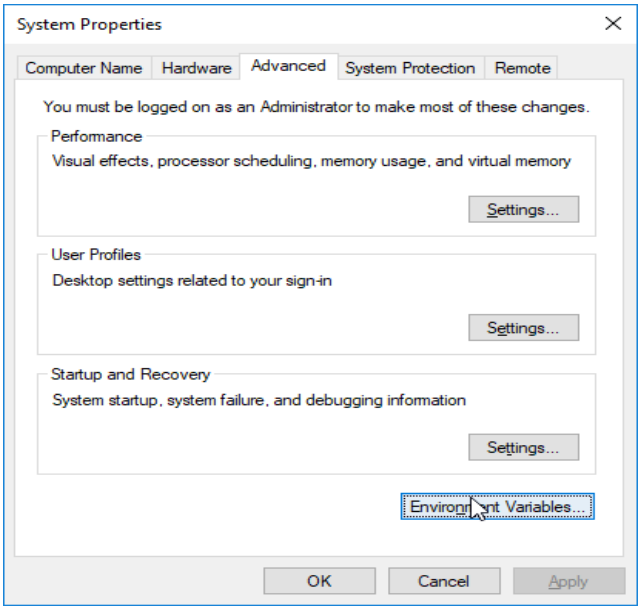
**Setup for Windows**

To install Java on Windows:

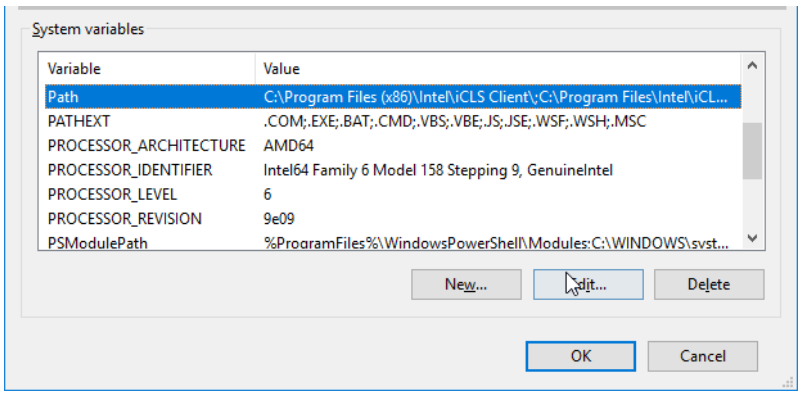
1. Go to "System Properties" (Can be found on Control Panel > System and Security > System > Advanced System Settings)



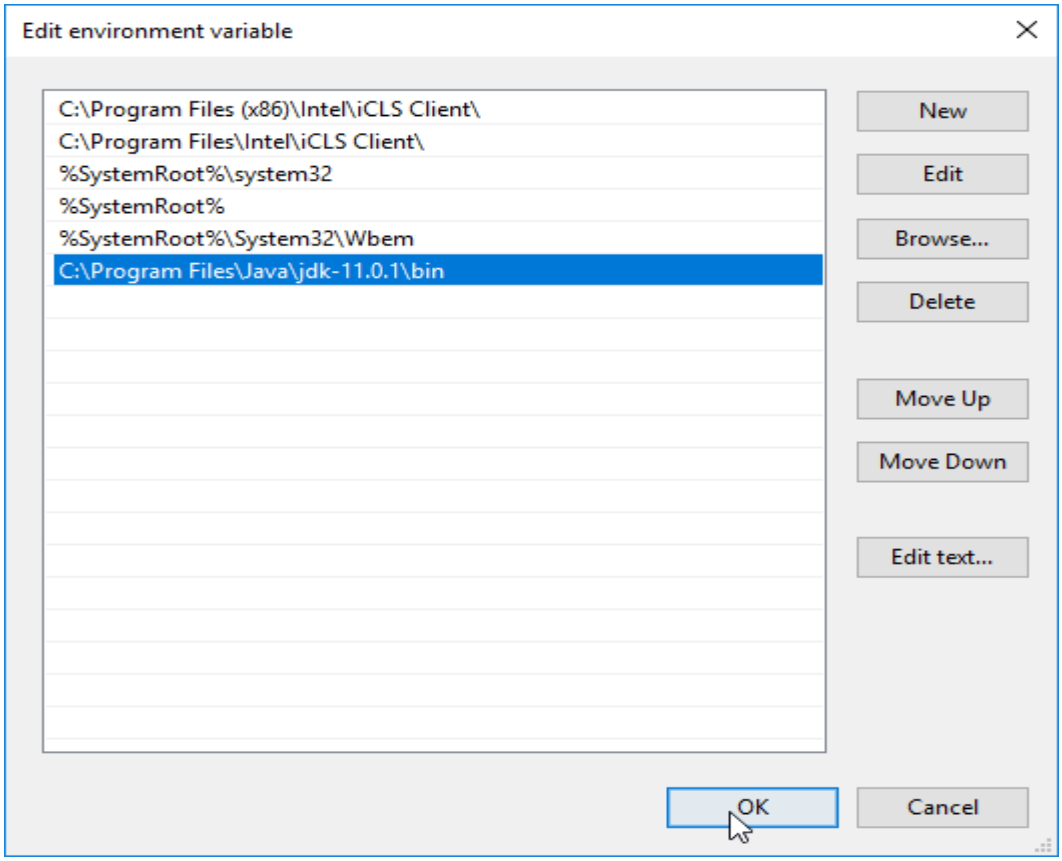

2. Click on the "Environment variables" button under the "Advanced" tab




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



4. Click on the "New" button and add the path where Java is installed, followed by \bin. By default, Java is installed in C:\Program Files\Java\jdk-11.0.1 (If nothing else was specified when you installed it). In that case, you will have to add a new path with: “C:\Program Files\Java\jdk-11.0.1\bin”. Then, click "OK", and save the settings




**5.** At last, open Command Prompt (cmd.exe) and type java -version to see if Java is running on your machine

Write the following in the command line (cmd.exe):

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

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

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

Main.java:-

```
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

Save the code in Notepad as "Main.java". Open Command Prompt (cmd.exe), navigate to the directory where you saved your file, and type

```
C:\Users\Your Name>javac Main.java
C:\Users\Your Name>java Main
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Make sure to install either 32bit or 64bit version of jdk based on your operating system.

**J. Procedure to be followed/Source code:**

K. **Input-Output:**

# PRACTICAL-2

**Aim:** Write a program in Java to find maximum of three numbers using conditional operator.

A. **Objective:** Find maximum of three numbers

**B. Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

**C. Expected Skills to be developed based on competency:** Develop Applications using Java Control structure

**D. Expected Course Outcomes(COs)**

- **CO1:** Write simple java programs for a given problem statement.

**E. Practical Outcome(PrO):** Write a program in Java to find maximum of three numbers using conditional operator.

**F. Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

**G. Prerequisite Theory:**

In Java, control structures are used to control the flow of execution in a program. There are three main types of control structures in Java: selection statements, iteration statements, and jump statements.

**Selection statements:** Selection statements are used to select one or more statements to execute based on the result of a boolean expression. In Java, the two main types of selection statements are if-else and switch statements.

**if-else statement:** It executes a block of code if a boolean expression is true, and executes an alternate block of code if the expression is false. Here's an example:

```
int x = 10;
if (x > 0) {
System.out.println("x is positive");
} else {
System.out.println("x is not positive");
}
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

K. **Input-Output:**

# PRACTICAL-3

**Aim:** Write a program in Java to reverse the digits of a number using while loop

A. **Objective:** Reverse the digits of a number

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Control structure

D. **Expected Course Outcomes(COs)**

- **CO1:** Write simple java programs for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java to reverse the digits of a number using while loop

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

In Java, loops are used to execute a block of code repeatedly until a certain condition is met. There are three types of loops in Java: for loop, while loop, and do-while loop. Here's a brief overview of each type of loop:

**for loop:** This loop is used to execute a block of code a fixed number of times. The syntax of a for loop is as follows:

```
for (initialization; condition; increment/decrement) {
   // block of code to be executed
}
```

In this syntax, initialization is a statement that is executed before the loop starts; condition is a boolean expression that is evaluated before each iteration of the loop; and increment/decrement is a statement that is executed at the end of each iteration of the loop.

**while loop:** This loop is used to execute a block of code repeatedly as long as a certain condition is true. The syntax of a while loop is as follows:

```
while (condition) {
   // block of code to be executed
}
```

In this syntax, condition is a boolean expression that is evaluated before each iteration of the loop.

**do-while loop:** This loop is similar to the while loop, except that the block of code is executed at least once, even if the condition is initially false. The syntax of a do-while loop is as follows:

```
do {
   // block of code to be executed
} while (condition);
```

In this syntax, condition is a boolean expression that is evaluated after each iteration of the loop.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-4

**Aim:** Write a program in Java to add two 3*3 matrices

A. **Objective:** Add two 3*3 matrices

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Control structure

D. **Expected Course Outcomes(COs)**

- **CO1:** Write simple java programs for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java to add two 3*3 matrices

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

In Java, a 2D array is an array of arrays. It is used to represent a matrix or a table of values. Here's an example of how to create a 2D array in Java:

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

In this example, arr is a 2D array with 3 rows and 4 columns. To access a specific element in the array, you need to specify the row and column index. Here's an example of how to assign a value to an element in the array:

```
arr[0][0] = 1;
```

This assigns the value 1 to the element in the first row and first column of the array.

You can also initialize a 2D array with values at the time of declaration. Here's an example:

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

In this example, arr is a 2D array with 3 rows and 3 columns. The values in the array are initialized to the values specified in the curly braces.

To iterate over a 2D array, you can use nested loops. Here's an example of how to iterate over the elements of a 2D array and print their values:

```
for (int i = 0; i < arr.length; i++) {
    for (int j = 0; j < arr[i].length; j++) {
        System.out.print(arr[i][j] + " ");
    }
    System.out.println();
}
```

This prints the elements of the array row by row.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-5

**Aim:** Write a program in Java to generate first n prime numbers

A. **Objective:** Generate first n prime numbers

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Control structure

D. **Expected Course Outcomes(COs)**

- **CO1:** Write simple java programs for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java to generate first n prime numbers

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

A prime number is a positive integer greater than 1 that has no positive integer divisors other than 1 and itself. In other words, a prime number is a number that is only divisible by 1 and itself.

For example, 2, 3, 5, 7, 11, 13, 17, 19, 23, and 29 are all prime numbers.

To check if a number is prime, we can use a simple algorithm that involves dividing the number by all integers from 2 to its square root. If the number is not divisible by any of these integers, then it is prime. Otherwise, it is composite.

H. **Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

I. **Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

J. **Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-6

**Aim:** Write a program in Java which has a class Student having two instance variables enrollmentNo and name. Create 3 objects of Student class in main method and display student's name

A. **Objective:** Create 3 objects of Student class in main method

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
   - **PO6: Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
   - **PO7: Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes in field of engineering.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java object-oriented concepts

D. **Expected Course Outcomes(COs)**
   - **CO2:** Use object oriented programming concepts to solve real world problems.

E. **Practical Outcome(PrO):** Write a program in Java which has a class Student having two instance variables enrollmentNo and name. Create 3 objects of Student class in main method and display student's name.

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

**G. Prerequisite Theory:**

In Java, a class is a blueprint or template for creating objects. It defines the attributes and behaviors of an object, and provides a way to create multiple objects of the same type.

Here is an example of a simple Java class:

```
public class Car {
    String make;
    String model;
    int year;
    public Car(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }
    public void start() {
        System.out.println("The " + make + " " + model + " is starting.");
    }
    public void stop() {
        System.out.println("The " + make + " " + model + " is stopping.");
    }
    public static void main(String[] args) {
        Car myCar = new Car("Toyota", "Camry", 2021);
        myCar.start();
        myCar.stop();
    }
}
```

This class represents a car and has three attributes (make, model, and year) and two methods (start and stop). The constructor is used to initialize the attributes when a new object is created. The start and stop methods simulate starting and stopping the car by printing messages to the console. The main method is used to create a new car object and call its start and stop methods.

To create an object of this class, you can use the new keyword followed by the class name and the constructor arguments:

```
Car myCar = new Car("Toyota", "Camry", 2021);
```

This creates a new Car object with the make "Toyota", model "Camry", and year 2021.

You can then call the object's methods using the dot notation:

```
myCar.start();
myCar.stop();
```

This would print the following output to the console:

```
The Toyota Camry is starting.
The Toyota Camry is stopping.
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-7

**Aim:** Write a program in Java which has a class Rectangle having two instance variables height and weight. Initialize the class using constructor.

A. **Objective:** Initialize the class using constructor

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java object-oriented concepts

D. **Expected Course Outcomes(COs)**
   - **CO2:** Use object oriented programming concepts to solve real world problems.

E. **Practical Outcome(PrO):** Write a program in Java which has a class Rectangle having two instance variables height and weight. Initialize the class using constructor.

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

G. **Prerequisite Theory:**

In Java, a constructor is a special method that is used to create an instance of a class. It is called automatically when an object is created using the "new" keyword and is used to initialize the object's instance variables.

Here is an example of a constructor in Java:

```
public class Person {
    String name;
```

```
    int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public static void main(String[] args) {
        Person person1 = new Person("John", 30);
        Person person2 = new Person("Jane", 25);
    }
}
```

In this example, the Person class has a constructor that takes two parameters: name and age. The constructor initializes the instance variables name and age with the values passed as arguments.

To create a new Person object using the constructor, you can use the new keyword followed by the class name and the constructor arguments.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-8

**Aim:** Write a program in Java demonstrate the use of “this” keyword.

A. **Objective:** Demonstrate the use of “this” keyword

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java object-oriented concepts

D. **Expected Course Outcomes(COs)**

- **CO2:** Use object oriented programming concepts to solve real world problems.

E. **Practical Outcome(PrO):** Write a program in Java demonstrate the use of “this” keyword.

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

In Java, "this" is a keyword that refers to the current object instance. It can be used inside a class to refer to the object on which a method or constructor is being called.

Here are some examples of how "this" can be used in Java:

**To access instance variables:** "this.variableName" can be used to refer to an instance variable of the current object.

**To call a constructor:** "this()" can be used to call another constructor in the same class. This is called constructor chaining. For example:

```
public class Rectangle {
    int width;
    int height;
    public Rectangle() {
        this(0, 0); // calls the Rectangle(int, int) constructor with arguments 0 and 0
    }
    public Rectangle(int width, int height) {
        this.width = width; // this refers to the width instance variable of the current object
        this.height = height; // this refers to the height instance variable of the current object
    }
}
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-9

**Aim:** Write a program in Java to demonstrate the use of “static” keyword.

A. **Objective:** Demonstrate the use of “static” keyword

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java object-oriented concepts

D. **Expected Course Outcomes(COs)**

- **CO2:** Use object oriented programming concepts to solve real world problems.

E. **Practical Outcome(PrO):** Write a program in Java to demonstrate the use of “static” keyword.

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

In Java, the "static" keyword is used to create variables and methods that belong to the class itself, rather than to an instance of the class. Here are key points related to the "static" keyword in Java:

**Static variables:** A static variable is a variable that is shared by all instances of a class. It is created only once when the class is loaded into memory, and all instances of the class share the same copy of the variable. To declare a static variable, use the "static" keyword before the variable type. For example:

```
public class MyClass {
    static int count = 0; // static variable
}
```

**Static methods:** A static method is a method that belongs to the class itself, rather than to an instance of the class. It can be called using the class name, without creating an instance of the class. To declare a static method, use the "static" keyword before the return type. For example:

```
public class MyClass {
    static void printMessage() { // static method
        System.out.println("Hello, world!");
    }
}
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-10

**Aim:** Write a program in Java to demonstrate the use of "final" keyword.

A. **Objective:** Demonstrate the use of “final” keyword

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java object-oriented concepts

D. **Expected Course Outcomes(COs)**

- **CO2:** Use object oriented programming concepts to solve real world problems.

E. **Practical Outcome(PrO):** Write a program in Java to demonstrate the use of "final" keyword.

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

In Java, the "final" keyword is used to create constants, prevent method or class overriding, and to create immutable objects. Here are some key points related to the "final" keyword in Java:

**Final variables:** A final variable is a variable that cannot be changed once it has been initialized. It is typically used to create constants, such as PI or VERSION_NUMBER. To declare a final variable, use the "final" keyword before the variable type. For example:

```
public class MyClass {
    final int MAX_COUNT = 100; // final variable
}
```

**Final methods:** A final method is a method that cannot be overridden by a subclass. It is typically used to ensure that the behavior of a method is consistent across all instances of a class. To declare a final method, use the "final" keyword before the method name. For example:

```
public class MyClass {
    final void doSomething() { // final method
        // ...
    }
}
```

**Final classes:** A final class is a class that cannot be subclassed. It is typically used to prevent others from extending a class and modifying its behavior. To declare a final class, use the "final" keyword before the class name. For example:

```
public final class MyClass { // final class
    // ...
}
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-11

**Aim:** Write a program in Java which has a class Shape having 2 overloaded methods area(float radius) and area(float length, float width). Display the area of circle and rectangle using overloaded methods.

A. **Objective:** Demonstrate the use of method overloading

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
   - **PO6: Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
   - **PO7: Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes in field of engineering.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java object-oriented concepts

D. **Expected Course Outcomes(COs)**
   - **CO2:** Use object oriented programming concepts to solve real world problems.

E. **Practical Outcome(PrO):** Write a program in Java which has a class Shape having 2 overloaded methods area(float radius) and area(float length, float width). Display the area of circle and rectangle using overloaded methods.

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

**G. Prerequisite Theory:**

In Java, method overloading is the process of defining multiple methods in the same class with the same name but different parameters. When a method is called, the Java compiler uses the number and type of arguments to determine which version of the method to call.

Here are some key points related to method overloading in Java:

- Method overloading is a way to provide multiple methods with the same name in a class, but with different argument lists.
- When you call an overloaded method, the Java compiler matches the method signature with the arguments passed in and decides which method to call.
- The signature of a method includes the method name and the types and order of the method's parameters. Return type and access modifier are not part of the method signature.
- Overloading can be used to simplify code and make it more readable by providing methods with the same name and different parameters that perform related tasks.
- You can overload constructors as well as regular methods.
- Java does not allow you to overload methods based solely on their return type. If you attempt to define two methods with the same name and parameter types but different return types, you will get a compile-time error.
- Java does not allow you to overload methods based solely on their access modifiers. If you attempt to define two methods with the same name and parameter types but different access modifiers, you will get a compile-time error.

Here's an example of method overloading in Java:

```
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
    public double add(double a, double b) {
        return a + b;
    }
    public int add(int a, int b, int c) {
        return a + b + c;
    }
}
```

When you call the add method on an instance of the Calculator class, the Java compiler uses the number and types of the arguments passed in to determine which version of the method to call. For example:

```
Calculator calc = new Calculator();
int result1 = calc.add(1, 2);           // Calls add(int a, int b)
double result2 = calc.add(3.14, 2.71);  // Calls add(double a, double b)
```

```
int result3 = calc.add(1, 2, 3);        // Calls add(int a, int b,
int c)
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-12

**Aim:** Write a program in Java to demonstrate the constructor overloading.

A. **Objective:** Demonstrate the use of constructor overloading

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java object-oriented concepts

D. **Expected Course Outcomes(COs)**

- **CO2:** Use object oriented programming concepts to solve real world problems.

E. **Practical Outcome(PrO):** Write a program in Java to demonstrate the constructor overloading.

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

Constructor overloading is the concept of having multiple constructors in a class, each with a different set of parameters. This allows you to create objects of the same class using different sets of values. Here's an example:

```
public class Person {
    private String name;
    private int age;
    // Constructor with no arguments
    public Person() {
```

```
        name = "Unknown";
        age = 0;
    }
    // Constructor with name argument
    public Person(String name) {
        this.name = name;
        age = 0;
    }
    // Constructor with name and age arguments
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-13

**Aim:** Write a java program to demonstrate use of "String" class methods: chatAt(), contains(), format(), length(), split()

A. **Objective:** Demonstrate the use of "String" class methods

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java object-oriented concepts

D. **Expected Course Outcomes(COs)**
   - **CO2:** Use object oriented programming concepts to solve real world problems.

E. **Practical Outcome(PrO):** Write a java program to demonstrate use of "String" class methods: chatAt(), contains(), format(), length(), split()

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

G. **Prerequisite Theory:**

Some important functions of the String class in Java:

- **length():** This method returns the length of the string.
- **charAt():** This method returns the character at a specified index in the string.
- **substring():** This method returns a substring of the string, starting from a specified index and ending at a specified index.
- **indexOf():** This method returns the index of the first occurrence of a specified character or substring in the string.

- **toUpperCase():** This method returns a new string with all the characters in uppercase.
- **toLowerCase():** This method returns a new string with all the characters in lowercase.
- **trim():** This method returns a new string with leading and trailing whitespace removed.
- **split():** This method splits the string into an array of substrings based on a specified delimiter.
- **equals():** This method compares two strings for equality.
- **compareTo():** This method compares two strings lexicographically and returns an integer value that indicates the result.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-14

**Aim:** Write a program in Java to demonstrate single inheritance.

A. **Objective:** Demonstrate single inheritance

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Inheritance

D. **Expected Course Outcomes(COs)**

- **CO3:** Develop an object-oriented program using inheritance and package concepts for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java to demonstrate single inheritance

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

**Single inheritance:** In single inheritance, a subclass inherits from a single superclass. For example:

```
class Animal {
  public void eat() {
    System.out.println("Animal is eating");
  }
}
class Dog extends Animal {
```

```
  public void bark() {
    System.out.println("Dog is barking");
  }
}
```

In this example, the Dog class inherits from the Animal class using single inheritance. The Dog class has access to the eat() method from the Animal class.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-15

**Aim:** Write a program in Java to demonstrate multilevel inheritance.

A. **Objective:** Demonstrate multilevel inheritance

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Inheritance

D. **Expected Course Outcomes(COs)**

- **CO3:** Develop an object-oriented program using inheritance and package concepts for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java to demonstrate multilevel inheritance

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

**Multilevel inheritance:** In multilevel inheritance, a subclass inherits from a superclass, which itself inherits from another superclass. For example:

```
class Animal {
  public void eat() {
    System.out.println("Animal is eating");
  }
}
```

```
class Mammal extends Animal {
  public void run() {
    System.out.println("Mammal is running");
  }
}
class Dog extends Mammal {
  public void bark() {
    System.out.println("Dog is barking");
  }
}
```

In this example, the Dog class inherits from the Mammal class, which itself inherits from the Animal class. The Dog class has access to the eat() method from the Animal class and the run() method from the Mammal class.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-16

**Aim:** Write a program in Java to demonstrate hierarchical inheritance.

A. **Objective:** Demonstrate hierarchical inheritance

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Inheritance

D. **Expected Course Outcomes(COs)**

- **CO3:** Develop an object-oriented program using inheritance and package concepts for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java to demonstrate hierarchical inheritance.

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

**Hierarchical inheritance:** In hierarchical inheritance, multiple subclasses inherit from a single superclass. For example:

```
class Animal {
  public void eat() {
    System.out.println("Animal is eating");
  }
}
```

```
class Dog extends Animal {
  public void bark() {
    System.out.println("Dog is barking");
  }
}
class Cat extends Animal {
  public void meow() {
    System.out.println("Cat is meowing");
  }
}
```

In this example, both the Dog and Cat classes inherit from the Animal class. They each have access to the eat() method from the Animal class.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-17

**Aim:** Write a program in Java to demonstrate method overriding.

A. **Objective:** Demonstrate method overriding

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Inheritance

D. **Expected Course Outcomes(COs)**
   - **CO3:** Develop an object-oriented program using inheritance and package concepts for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java to demonstrate method overriding.

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

G. **Prerequisite Theory:**

Method overriding is a technique in Java where a subclass provides its implementation of a method that is already provided by its parent class. To override a method in Java, the following conditions must be met:

- The method in the subclass must have the same method signature (method name and parameter list) as the method in the superclass.
- The access modifier of the method in the subclass cannot be more restrictive than the access modifier of the method in the superclass.
- The return type of the method in the subclass must be the same as, or a subclass of, the return type of the method in the superclass.

```
class Animal {
   public void makeSound() {
      System.out.println("Some sound");
   }
}
class Cat extends Animal {
   public void makeSound() {
      System.out.println("Meow");
   }
}
public class Main {
   public static void main(String args[]) {
      Animal animal = new Animal();
      animal.makeSound();   // prints "Some sound"
      Cat cat = new Cat();
      cat.makeSound();      // prints "Meow"
      Animal anotherAnimal = new Cat();
      anotherAnimal.makeSound(); // prints "Meow"
   }
}
```

In this example, we have a superclass Animal with a method makeSound() that prints "Some sound". We then have a subclass Cat that extends Animal and overrides the makeSound() method to print "Meow".

In the main method, we create an object of Animal and call its makeSound() method, which prints "Some sound". We then create an object of Cat and call its makeSound() method, which prints "Meow". Finally, we create an object of Cat and assign it to a variable of type Animal. When we call makeSound() on this variable, it still calls the makeSound() method of Cat and prints "Meow", even though the variable is of type Animal. This is because the makeSound() method is dynamically bound at runtime based on the type of the actual object being referred to.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |

| | | | |
|---|---|---|---|
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

I. **Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

J. **Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-18

**Aim:** Write a program in Java which has a class Car having two instance variables topSpeed and name. Override toString() method in Car class. Create 5 instances of Car class and print the instances.

A. **Objective:** Override toString() method in class

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Inheritance

D. **Expected Course Outcomes(COs)**
   - **CO3:** Develop an object-oriented program using inheritance and package concepts for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java which has a class Car having two instance variables topSpeed and name. Override toString() method in Car class. Create 5 instances of Car class and print the instances.

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

G. **Prerequisite Theory:**

The Object class provides many methods. They are as follows:

- **public final Class getClass()**: returns the Class class object of this object. The Class class can further be used to get the metadata of this class.

- **public int hashCode():** returns the hashcode number for this object.
- **public boolean equals(Object obj):** compares the given object to this object.
- **protected Object clone() throws CloneNotSupportedException:** creates and returns the exact copy (clone) of this object.
- **public String toString():** returns the string representation of this object.
- **public final void notify():** wakes up single thread, waiting on this object's monitor.
- **public final void notifyAll():** wakes up all the threads, waiting on this object's monitor.
- **public final void wait(long timeout) throws InterruptedException:** causes the current thread to wait for the specified milliseconds, until another thread notifies (invokes notify() or notifyAll() method).
- **public final void wait(long timeout,int nanos) throws InterruptedException:** causes the current thread to wait for the specified milliseconds and nanoseconds, until another thread notifies (invokes notify() or notifyAll() method).
- **public final void wait() throws InterruptedException:** causes the current thread to wait, until another thread notifies (invokes notify() or notifyAll() method).
- **protected void finalize() throws Throwable:** invoked by the garbage collector before object is being garbage collected.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-19

**Aim:** Write a program in Java to implement multiple inheritance using interfaces.

A. **Objective:** Demonstrate multiple inheritance using interfaces

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Inheritance

D. **Expected Course Outcomes(COs)**

- **CO3:** Develop an object-oriented program using inheritance and package concepts for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java to implement multiple inheritance using interfaces.

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

Before diving into multiple inheritance using interfaces in Java, it's essential to understand the following concepts:

- **Interfaces:** An interface is a collection of abstract methods and constants. An interface can be implemented by a class using the implements keyword. A class can implement multiple interfaces.
- **Diamond Problem:** The diamond problem is a scenario that arises when a class implements two interfaces that have a common method with the same signature, and the class does not provide an implementation for the method.

With these concepts understood, we can now discuss multiple inheritance using interfaces in Java. Multiple inheritance using interfaces allows a class to inherit from multiple interfaces. A class that implements multiple interfaces must provide an implementation for all the abstract methods defined by the interfaces.

To avoid the diamond problem, Java interfaces do not allow the declaration of instance variables, and all methods are by default public and abstract. Java interfaces can include default methods, which are methods with a default implementation that can be overridden by the implementing class.

By understanding these concepts, you will have a solid foundation for understanding multiple inheritance using interfaces in Java.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-20

**Aim:** Write a program in Java which has an abstract class Shape having three subclasses: Triangle, Rectangle, and Circle. Define method area() in the abstract class Shape and override area() method to calculate the area.

A. **Objective:** Demonstrate abstract method

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
   - **PO7: Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes in field of engineering.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Inheritance

D. **Expected Course Outcomes(COs)**
   - **CO3:** Develop an object-oriented program using inheritance and package concepts for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java which has an abstract class Shape having three subclasses: Triangle, Rectangle, and Circle. Define method area() in the abstract class Shape and override area() method to calculate the area.

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

G. **Prerequisite Theory:**

In Java, an abstract class is a class that cannot be instantiated, meaning that you cannot create an object of this class directly. Instead, you need to create a subclass that extends the abstract class and implements its abstract methods to create an instance.

An abstract class is declared using the abstract keyword. It may contain both abstract and non-abstract methods. An abstract method is a method that is declared but does not have an implementation in the abstract class. The implementation is left to the subclasses that extend the abstract class. Abstract methods are declared using the abstract keyword and end with a semicolon instead of a method body.

Here is an example of an abstract class in Java:

```
public abstract class Shape {
   public abstract double getArea();
   public abstract double getPerimeter();
   public void printShape() {
           System.out.println("This is a shape.");
   }
}
```

Abstract classes are useful when you want to define a common interface for a group of related classes. By defining the common interface in an abstract class, you can ensure that all the subclasses have a consistent set of methods and properties, while still allowing each subclass to provide its own implementation.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-21

**Aim:** Write a program in Java to demonstrate use of final class.

A. **Objective:** Demonstrate use of final class

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Inheritance

D. **Expected Course Outcomes(COs)**
   - **CO3:** Develop an object-oriented program using inheritance and package concepts for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java to demonstrate use of final class.

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

G. **Prerequisite Theory:**

In Java, a final class is a class that cannot be inherited or extended. This means that once a class is declared as final, it cannot be inherited by any other class, and no subclass can be created from it.

To declare a class as final, use the final keyword in the class declaration, like this:

```
public final class MyClass {
    // class body
}
```

Once you have declared a class as final, it cannot be inherited. Attempting to extend a final class will result in a compile-time error.

```
public class MySubClass extends MyClass { // compile-time error
    // subclass body
}
```

The final keyword can also be used to declare a method or a variable.

- When used with a method, the final keyword prevents the method from being overridden by any subclass.
- When used with a variable, the final keyword indicates that the variable's value cannot be changed after it has been initialized.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-22

**Aim:** Write a program in Java to demonstrate use of package.

A. **Objective:** Demonstrate use of package

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Packages

D. **Expected Course Outcomes(COs)**

- **CO3:** Develop an object-oriented program using inheritance and package concepts for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java to demonstrate use of package.

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

In Java, a package is a mechanism that allows you to group related classes, interfaces, and sub-packages together. It provides a way to organize and manage your Java code, making it easier to maintain and reuse.

A package is declared using the package keyword, followed by the package name, at the beginning of a Java file. For example, if you wanted to create a package called "myapp", you would declare it like this:

```
package myapp;

// Java code here
```

Packages are typically organized in a hierarchical structure, with sub-packages and classes within them. For example, the package name "myapp" might contain several sub-packages, such as "myapp.ui" and "myapp.model", each of which contains related classes and interfaces.

To use classes from a different package in your Java code, you need to import them using the import statement. For example, if you wanted to use the MyClass class from the "myapp.model" package in your code, you would import it like this:

```
import myapp.model.MyClass;
public class MyOtherClass {
    // code that uses MyClass
}
```

Using packages in your Java code can provide several benefits, including:

- Better code organization and maintainability.
- Reduced naming conflicts between classes and interfaces.
- Improved code reusability and modularity.
- Easier collaboration between developers, as different parts of the code can be separated into different packages.

Overall, packages are an important feature of Java that can help you write better-structured, more maintainable code.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-23

**Aim:** Write a program in Java to develop user defined exception for 'Divide by Zero' error.

A. **Objective:** Demonstrate user defined exception

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Exception Handling

D. **Expected Course Outcomes(COs)**

- **CO4:** Develop an object oriented program using multithreading and exception handling for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java to develop user defined exception for 'Divide by Zero' error.

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

In Java, a user-defined exception is an exception that is created by the programmer to handle a specific error condition in their code. It allows the programmer to define their own exception types, with custom error messages and behaviors, that can be thrown and caught like any other exception in Java.

To create a user-defined exception in Java, you need to create a new class that extends the Exception class (or one of its subclasses). For example, if you wanted to create a custom exception for handling invalid user input in your application, you might define a class like this:

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

This class extends the Exception class and adds a constructor that takes a String message parameter. This message can be used to provide more information about the error condition that caused the exception to be thrown.

Once you have defined your custom exception class, you can throw it like any other exception in your Java code. For example, if you wanted to throw an InvalidInputException when the user enters invalid data into a form, you might write code like this:

```
if (input == null || input.isEmpty()) {
    throw new InvalidInputException("Input cannot be empty");
}
```

This code checks if the input is empty or null, and throws an InvalidInputException if it is. This exception can then be caught and handled by other parts of the application, using standard Java exception handling techniques.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-24

**Aim:** Write a program in Java to develop Banking Application in which user deposits the amount Rs. 25000 and then start withdrawing of Rs. 20000, Rs. 4000 and it throws exception "Not Sufficient Fund" when user withdraws Rs. 2000 thereafter.

A. **Objective:** Demonstrate user defined exception using Banking Application

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
   - **PO6: Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
   - **PO7: Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes in field of engineering.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Exception Handling

D. **Expected Course Outcomes(COs)**
   - **CO4:** Develop an object oriented program using multithreading and exception handling for a given problem statement.

E. **Practical Outcome(PrO):** Write a program in Java to develop Banking Application in which user deposits the amount Rs. 25000 and then start withdrawing of Rs. 20000, Rs. 4000 and it throws exception "Not Sufficient Fund" when user withdraws Rs. 2000 thereafter.

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.

- Follow ethical practices.

**G. Prerequisite Theory:**

In Java, a runtime exception is a type of exception that can occur at any time during program execution, and does not need to be explicitly declared in the method signature or caught in a try-catch block. Some common examples of runtime exceptions in Java include NullPointerException, ArrayIndexOutOfBoundsException, and ArithmeticException.

You can use the throw keyword in Java to manually generate runtime exceptions, by throwing an instance of a runtime exception class. For example, if you wanted to throw an IllegalArgumentException when a method is called with an invalid argument, you might write code like this:

```
if (argument < 0) {
  throw new IllegalArgumentException("Argument cannot be negative");
}
```

This code checks if the argument is negative, and throws an IllegalArgumentException with a custom error message if it is. This exception can then be caught and handled by other parts of the application, using standard Java exception handling techniques.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-25

**Aim:** Write a program that executes two threads. One thread displays "Thread1" every 1000 milliseconds, and the other displays "Thread2" every 2000 milliseconds. Create the threads by extending the Thread class

A. **Objective:** Demonstrate threading

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Threading

D. **Expected Course Outcomes(COs)**
   - **CO4:** Develop an object oriented program using multithreading and exception handling for a given problem statement.

E. **Practical Outcome(PrO):** Write a program that executes two threads. One thread displays "Thread1" every 1000 milliseconds, and the other displays "Thread2" every 2000 milliseconds. Create the threads by extending the Thread class.

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

G. **Prerequisite Theory:**

Java threading is the process of creating and managing threads within a Java program. A thread is a lightweight sub-process that can run concurrently with other threads, sharing the same memory and resources of the main program.

In Java, you can create a thread by either extending the Thread class or implementing the Runnable interface. When you extend the Thread class, you override the run() method, which is the entry point for the thread. When you implement the Runnable interface, you need to define the run() method in a separate class, and then create a new Thread object passing it the Runnable instance.

Here's an example of creating a thread by extending the Thread class:

```
class MyThread extends Thread {
  public void run() {
    // code to be executed in this thread
  }
}
public class Main {
  public static void main(String[] args) {
    MyThread thread = new MyThread();
    thread.start(); // start the thread
  }
}
```

In this example, the MyThread class extends the Thread class and overrides the run() method, which is the entry point for the thread. When the start() method is called, a new thread is created and the run() method is executed in that thread.

When the start() method is called, the run() method of the MyRunnable instance is executed in a separate thread.

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-26

**Aim:** Write a program that executes two threads. One thread will print the even numbers and another thread will print odd numbers from 1 to 200.

A. **Objective:** Demonstrate threading

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Threading

D. **Expected Course Outcomes(COs)**
   - **CO4:** Develop an object oriented program using multithreading and exception handling for a given problem statement.

E. **Practical Outcome(PrO):** Write a program that executes two threads. One thread will print the even numbers and another thread will print odd numbers from 1 to 200.

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

G. **Prerequisite Theory:**

Creating a thread by implementing the Runnable interface:

```
class MyRunnable implements Runnable {
  public void run() {
      // code to be executed in this thread
  }
```

```
}
public class Main {
  public static void main(String[] args) {
    MyRunnable runnable = new MyRunnable();
    Thread thread = new Thread(runnable);
    thread.start(); // start the thread
  }
}
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-27

**Aim:** Write a program in Java to perform read and write operations on a Text file.

A. **Objective:** Demonstrate read and write operations on a Text file

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **PO6: Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
- **PO7: Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes in field of engineering.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java IO

D. **Expected Course Outcomes(COs)**

- **CO5:** Develop an object-oriented program by using the files and collection framework.

E. **Practical Outcome(PrO):** Write a program in Java to perform read and write operations on a Text file.

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

The java.io package in Java provides a set of classes and interfaces for handling input and output operations. It is used for reading and writing data to and from different types of sources such as files, streams, and network connections.

Some of the commonly used classes in the java.io package are:

**InputStream** - abstract class that represents an input stream of bytes.

**OutputStream** - abstract class that represents an output stream of bytes.

**File** - represents a file on the file system.

**FileReader** - reads text from a character file.

**FileWriter** - writes text to a character file.

**BufferedReader** - reads text from a character input stream, buffering characters to provide for the efficient reading of characters, arrays, and lines.

**PrintWriter** - prints formatted representations of objects to a text-output stream.

**DataInputStream** - reads primitive Java data types from an input stream.

**DataOutputStream** - writes primitive Java data types to an output stream.

**ObjectInputStream** - reads Java objects from an input stream.

**ObjectOutputStream** - writes Java objects to an output stream.

These classes and interfaces provide a wide range of functionalities for handling different types of input and output operations in Java.

Here's an example of how to read from a text file using FileInputStream:

```
import java.io.FileInputStream;
import java.io.IOException;

public class Main {
  public static void main(String[] args) {
    try (FileInputStream fis = new FileInputStream("filename.txt")) {
      byte[] buffer = new byte[1024];
      int bytesRead;
      while ((bytesRead = fis.read(buffer)) != -1) {
        String line = new String(buffer, 0, bytesRead);
        System.out.println(line);
      }
    } catch (IOException e) {
      System.out.println("Error reading file");
      e.printStackTrace();
    }
  }
}
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-28

**Aim:** Write a program in Java to demonstrate use of List. 1) Create ArrayList and add weekdays (in string form) & 2) Create LinkedList and add months (in string form). Display both List.

A. **Objective:** Demonstrate use of List

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Collections

D. **Expected Course Outcomes(COs)**
   - **CO5:** Develop an object-oriented program by using the files and collection framework.

E. **Practical Outcome(PrO):** Write a program in Java to demonstrate use of List. 1) Create ArrayList and add weekdays (in string form) & 2) Create LinkedList and add months (in string form). Display both List.

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

G. **Prerequisite Theory:**

**ArrayList:** ArrayList is a resizable array implementation of the List interface in Java. It allows you to add and remove elements dynamically, making it useful when you need to add or remove elements from a collection frequently. It uses an array to store elements, and when the array is full, it creates a new array with a larger size to accommodate more elements. This resizing operation can be costly in terms of performance, especially for large collections. However, accessing elements in an ArrayList is very fast because it uses an index-based approach.

```
import java.util.ArrayList;
public class ArrayListExample {
   public static void main(String[] args) {
      ArrayList<String> list = new ArrayList<String>();
      // Add elements to the ArrayList
      list.add("apple");
      list.add("banana");
      list.add("cherry");
      // Print the ArrayList
      System.out.println("ArrayList: " + list);
      // Access elements in the ArrayList using index
      System.out.println("First element: " + list.get(0));
      System.out.println("Second element: " + list.get(1));
      // Remove an element from the ArrayList
      list.remove(2);
      // Print the updated ArrayList
      System.out.println("Updated ArrayList: " + list);
   }
}
```

**Output:**

```
ArrayList: [apple, banana, cherry]
First element: apple
Second element: banana
Updated ArrayList: [apple, banana]
```

**LinkedList:** LinkedList is a doubly-linked list implementation of the List interface in Java. Unlike ArrayList, it does not use an array to store elements. Instead, it uses a series of nodes, where each node contains a reference to the previous and next nodes. This makes it more efficient than ArrayList for adding or removing elements, especially in the middle of the list. However, accessing elements in a LinkedList can be slower than ArrayList because it does not use an index-based approach. Instead, it needs to traverse the list to find the element.

```
import java.util.LinkedList;
public class LinkedListExample {
   public static void main(String[] args) {
      LinkedList<String> list = new LinkedList<String>();
      // Add elements to the LinkedList
      list.add("apple");
```

```
        list.add("banana");
        list.add("cherry");
        // Print the LinkedList
        System.out.println("LinkedList: " + list);
        // Access elements in the LinkedList using index
        System.out.println("First element: " + list.get(0));
        System.out.println("Second element: " + list.get(1));
        // Remove an element from the LinkedList
        list.remove(2);
        // Print the updated LinkedList
        System.out.println("Updated LinkedList: " + list);
    }
}
```

**Output:**

```
LinkedList: [apple, banana, cherry]
First element: apple
Second element: banana
Updated LinkedList: [apple, banana]
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-29

**Aim:** Write a program in Java to create a new HashSet, add colors(in string form) and iterate through all elements using for-each loop to display the collection.

A. **Objective:** Demonstrate use of HashSet

B. **Expected Program Outcomes (POs):**

- **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
- **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
- **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Collections

D. **Expected Course Outcomes(COs)**

- **CO5:** Develop an object-oriented program by using the files and collection framework.

E. **Practical Outcome(PrO):** Write a program in Java to create a new HashSet, add colors(in string form) and iterate through all elements using for-each loop to display the collection.

F. **Expected Affective Domain Outcome (ADOs)**

- Follow coding standards.
- Demonstrate working as a leader/ a team member.
- Follow ethical practices.

G. **Prerequisite Theory:**

Java Collection HashSet is a class that implements the Set interface and represents an unordered collection of unique elements, where duplicates are not allowed. The elements are stored using a hash table, which provides constant time performance for basic operations such as add, remove, and contains. Here's an example of how to use HashSet in Java:

```
import java.util.HashSet;
public class HashSetExample {
```

```
    public static void main(String[] args) {
        // Create a new HashSet
        HashSet<String> set = new HashSet<String>();
        // Add elements to the HashSet
        set.add("apple");
        set.add("banana");
        set.add("cherry");
        // Display the elements of the HashSet
        System.out.println("HashSet: " + set);
        // Check if an element is in the HashSet
        System.out.println("contain 'banana'? " + set.contains("banana"));
        // Remove an element from the HashSet
        set.remove("cherry");
        // Display the elements of the HashSet after removing an element
        System.out.println("HashSet after removing 'cherry': " + set);
        // Clear the HashSet
        set.clear();
        // Display the elements of the HashSet after clearing it
        System.out.println("HashSet after clearing it: " + set);
    }
}
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**

# PRACTICAL-30

**Aim:** Write a Java program to create a new HashMap, add 5 students' data (enrolment no and name). retrieve and display the student's name from HashMap using enrolment no.

A. **Objective:** Demonstrate use of HashMap

B. **Expected Program Outcomes (POs):**
   - **PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
   - **PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.
   - **PO3: Design/ development of solutions:** Design solutions for engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
   - **PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
   - **PO6: Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
   - **PO7: Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes in field of engineering.

C. **Expected Skills to be developed based on competency:** Develop Applications using Java Collections

D. **Expected Course Outcomes(COs)**
   - **CO5:** Develop an object-oriented program by using the files and collection framework.

E. **Practical Outcome(PrO):** Write a Java program to create a new HashMap, add 5 students' data (enrolment no and name). retrieve and display the student's name from HashMap using enrolment no.

F. **Expected Affective Domain Outcome (ADOs)**
   - Follow coding standards.
   - Demonstrate working as a leader/ a team member.
   - Follow ethical practices.

G. **Prerequisite Theory:**

Java Collection HashMap is a class that implements the Map interface and represents a collection of key-value pairs, where each key is unique and maps to a corresponding value. The elements are stored using a hash table, which provides constant time performance for basic operations such as put, get, and remove. Here's an example of how to use HashMap in Java:

```
import java.util.HashMap;
public class HashMapExample {
    public static void main(String[] args) {
        // Create a new HashMap
        HashMap<String, Integer> map = new HashMap<String, Integer>();
        // Add elements to the HashMap
        map.put("apple", 1);
        map.put("banana", 2);
        map.put("cherry", 3);
        // Display the elements of the HashMap
        System.out.println("HashMap: " + map);
        // Get the value associated with a key
        int value = map.get("banana");
        System.out.println("The value of 'banana' is: " + value);
        // Check if a key is in the HashMap
        boolean containsKey = map.containsKey("cherry");
        System.out.println("contain 'cherry'? " + containsKey);
        // Remove an element from the HashMap
        map.remove("apple");
        // Display the elements of the HashMap after removing an element
        System.out.println("HashMap after removing 'apple': " + map);
        // Clear the HashMap
        map.clear();
        // Display the elements of the HashMap after clearing it
        System.out.println("HashMap after clearing it: " + map);
    }
}
```

**H. Resources/Equipment Required**

| Sr. No. | Instrument/Equipment /Components/Trainer kit | Specification | Quantity |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 1 | Hardware: Computer System | Computer (i3-i5 preferable), RAM 2 GB and onwards | As per batch size |
| 2 | Software: Operating System | Windows/ Linux | |
| 3 | Software | Jdk 1.8.0 or above, Notepad++ or other text editor | |

**I. Safety and necessary Precautions followed:** Shutdown computer system properly once the Lab hours are finished.

**J. Procedure to be followed/Source code:**

**K. Input-Output:**