EXPERIMENT – 1

TITLE: Introduction to Java Environment

1. Explore and understand the role of JDK, JRE and JVM.
2. Install latest available JDK and verify the Java Environment.
3. Create a Sample Hello World Program using simple text editor (e.g. Notepad) and show the steps to compile and execute the program using command prompt.
4. Display your name and complete address in different lines.

Additional Question:

5. Design a visually appealing gradesheet that displays the Name, Roll Number, SAP ID, and Result. Use escape sequences and special characters like '*' to enhance its presentation. [No need to take any input from User].

EXPERIMENT – 2

TITLE: Basic Java Programming

1. Write a program to find area of triangle.
2. Write a program to find simple interest.
3. Write a program to implement a command line calculator. (Try for Add sub Mul in same program for 2 digits.)
   (Hint: Integer.parseInt will be used)
   For e.g. java calc 20 + 30
   Output should be Sum of 20 and 30 is 50

   java calc 50 * 30
   Output should be Product of 50 and 30 is 1500
4. Write a Java program to check whether a given number is positive, negative, or zero using an if-else statement.
5. Create a program that accepts three integers and determines the greatest among them using nested if-else statements.
6. Create a program that accepts a number (1–7) and displays the corresponding day of the week using a switch statement.

Additional Question:

1. Write a program to calculate the final grade of a student based on the marks entered in three subjects. Use the following grading scale:

Average >= 90: Grade A
Average >= 75: Grade B
Average >= 50: Grade C
Otherwise: Grade F

2. WAP to Take input as DD MM YYYY(04 08 2021) in command line and calculate number of days since 1 January 1970.

EXPERIMENT – 3

TITLE: Basic Java Programming (Loops & Arrays)

1. Write a program to calculate the sum of all integers between 10 and 950 that are divisible by both 6 and 9.
2. Write a Java program that takes an integer as input and calculates the sum of its digits using a while loop.
3. Write a Java program that prints the first N terms of the Fibonacci series using a loop.
4. Write a Java program to count and display the total number of prime numbers between 1 and 1000.
5. Write a Java program that counts how many times a given number appears in an array.
   Input: arr = [2, 3, 2, 5, 2, 6], target = 2
   Output: 3
6. Write a Java program to find the second largest element in an integer array without sorting the array.
7. WAP to print the following pattern using loop

```
?
# # #
? ? ? ? ?
# # # # # # #
? ? ? ? ? ? ? ? ?
```

Additional Questions:

1. Write a Java program that copies all elements from one array to another using a loop.
2. Given an array containing N-1 unique numbers from 1 to N, write a Java program to find the missing number.
   Input: [1, 5, 6, 2, 4]
   Output: 3
3. Write a Java program to rotate an array right by K positions.

   Input: arr = [1, 2, 3, 4, 5], K = 2

   Output: [4, 5, 1, 2, 3]

EXPERIMENT – 4

TITLE: Classes (Constructors, Access modifiers, Method Overloading, static & non static data members, this)

1. Create a Student class with attributes for name and age. Implement a default constructor to assign default values and a parameterized constructor to initialize the attributes with user-defined values. Create objects using both constructors and display their details.
2. Create a BankAccount class with a private variable balance to store the account balance. Implement a public method deposit(double amount) to add funds, a protected method withdraw(double amount) to deduct funds, and a default-access method checkBalance() to display the current balance. Create an object of the class and demonstrate which methods and variables can be accessed both inside and outside the class.
3. Create a Calculator class that contains a method add() to perform addition. Overload the add() method to handle different types and numbers of parameters, such as adding two integers, two doubles, and three integers. Create an object of the class and demonstrate all method variations.
4. Create a Student class that has a static variable universityName and a non-static variable studentName. Include a static method to display the university name. Then, create multiple student objects to demonstrate how the static variable is shared among all instances, while the non-static variable holds unique values for each object.
5. A student is developing a course registration system that allows students to enroll in courses. Each course has a course name and a course code. Implement a Course class with appropriate attributes and use the "this" keyword to differentiate between class attributes and constructor parameters during initialization. Create an object of the Course class and display the course details. [Keep the variable name of formal arguments in constructor and instance variables same.]

Additional Question:

1. Develop an Employee Management System in Java to track employee details such as ID, name, department, and salary, while calculating total salary expenditure and employee count. Create an Employee class with instance variables (employeeID, name, department, and private salary), a static variable totalEmployees to track the employee count, a default constructor for initializing default values, and a parameterized constructor for setting user-provided details. Include a static method to display totalEmployees, a calculateSalary() method to return the salary, and a displayEmployeeInfo() method to show employee details. Use the this keyword in constructors to differentiate class variables from parameters. In the main method, create Employee objects using both constructors, display the total number of employees, and show salary details for each employee.

EXPERIMENT – 5

TITLE Inheritance

1. Write a Java program to demonstrate that a private member of a superclass cannot be accessed directly from a derived class.
2. Create a Java program with a Player class and derive three subclasses: Cricket_Player, Football_Player, and Hockey_Player. Implement attributes such as name, age, and position, and methods like play() and train() to represent these players.
3. Define a Worker class with DailyWorker and SalariedWorker as its subclasses. Each worker has a name and salary rate. Implement a method computePay(int hours) to compute weekly pay. DailyWorker is paid based on the number of days worked (assuming 8 hours per day), whereas SalariedWorker receives a fixed wage for 40 hours per week, regardless of actual hours worked. Use polymorphism to implement this program and test worker salary calculations.
4. Implement a Java program to calculate trunk call charges based on duration and type (Ordinary, Urgent, or Lightning). Use polymorphism to manage different charge rates for each type. Implement a Java program to calculate trunk call charges based on duration (in minutes) and type (Ordinary, Urgent, or Lightning). Use polymorphism to manage different charge rates for each type. The program should take user input for duration and type and display the total charge.
5. Design a Java class Employee with attributes name, empid, and salary. Implement a default constructor, a parameterized constructor, and methods to return the employee's name and salary. Add a method increaseSalary(double percentage) to raise the salary by a user-specified percentage. Create a subclass Manager with an additional instance variable department. Develop a test program to validate these functionalities.

Additional Questions:

6. A vehicle manufacturing company produces different types of vehicles, such as cars and motorcycles. The base class Vehicle contains common properties like brand, model, and price. The class Car extends Vehicle by adding attributes like seatingCapacity and fuelType. Further, a subclass ElectricCar extends Car, introducing additional attributes like batteryCapacity and chargingTime. The Motorcycle class extends Vehicle and adds engineCapacity and type (e.g., "Sport", "Cruiser"). Implement this vehicle hierarchy system using multilevel inheritance in Java. Use constructor chaining to initialize attributes efficiently and demonstrate polymorphism by overriding a method displayDetails() in each subclass.
7. A university has different types of people associated with it, including staff members and students. The base class Person contains common attributes such as name, age, and address.The class Staff extends Person and adds attributes like staffId and department. Further, a subclass Professor extends Staff by introducing an additional attribute, specialization, and a method conductLecture(). Similarly, the Student class extends Person

and adds studentId and course. Finally, the subclass GraduateStudent extends Student, adding researchTopic and a method submitThesis(). Implement this university management system in Java using multilevel inheritance and method overriding. Demonstrate polymorphism by creating an array of Person objects containing instances of Professor and GraduateStudent, and call their respective methods.

EXPERIMENT – 6

TITLE Packages & use of final

1. Create a package named Balance containing a class Account with a method Display_Balance that displays the account balance. Write another program to import the Balance package and call the Display_Balance method from the Account class.

2. Create a package p containing a class A with four methods, each having different access modifiers (e.g., public, protected, default, and private). Write another class B in a different package Q with a main() method. Demonstrate how to access the methods of class A from class B, considering the access modifiers.

3. Write a Java program to demonstrate the use of the final keyword with a variable and a method. Create a class MathConstants with a final variable PI (value = 3.14159) and a final method displayPI() that prints the value of PI. Create another class Circle that extends MathConstants and includes a method calculateArea(double radius) to calculate and print the area of a circle using the formula: area = PI * radius * radius. In the main() method (in a separate class or Circle), test the calculateArea() method and observe the behavior when attempting to modify the PI variable or override the displayPI() method.

4. Write a Java program to demonstrate the use of the final keyword with a class. Create a final class Logger with a method logMessage(String message) that prints the message to the console. Attempt to create another class ExtendedLogger that extends the Logger class, and observe and explain the result. In the main() method (in a separate class), create an object of the Logger class and call the logMessage() method to print a sample message.

EXPERIMENT – 7

TITLE Abstract Classes and Interfaces

1. Write a Java program to create an abstract class Shape with an abstract method calculateArea(). Derive two classes Rectangle and Circle from Shape and override the calculateArea() method to calculate and print the area of a rectangle and a circle, respectively. Use the main() method to create objects of Rectangle and Circle and test their calculateArea() methods.
2. Write a Java program to create an abstract class Employee with abstract methods calculateSalary() and displayDetails(). Derive two classes Manager and Developer from Employee and implement the methods to calculate the salary (e.g., based on fixed salary or hourly wage) and display employee details (e.g., name, role, salary). In the main() method, create objects of Manager and Developer and test their functionality.
3. Write a Java program to create an interface Bank with methods deposit(double amount) and withdraw(double amount). Implement this interface in a class Account that overrides these methods to perform deposit and withdrawal operations on a balance variable. Create another class BankDemo with a main() method to test the functionality by depositing and withdrawing amounts and displaying the updated balance.
4. Write a Java program to create an interface Playable with methods play(), pause(), and stop(). Implement this interface in a class MusicPlayer that overrides these methods to print appropriate messages (e.g., "Music is playing," "Music is paused," "Music is stopped"). Create another class TestPlayer with a main() method to test the functionality by calling the play(), pause(), and stop() methods.

Additional Question:

5. Write a Java program that defines an interface StackInterface with three methods: push(), pop(), and display(). Implement this interface in a class called StackClass. The StackClass should also contain the main method, where a switch-case structure is used to allow users to select and perform stack operations.

Experiment 8

TITLE Inner/Nested Classes, Exception Handling & File Handling

1. Create an outer class Library with a static nested class Book. The Book class should have attributes like title, author, and ISBN, and a method displayDetails() to print these details. In the main method, create an instance of the Book class and call displayDetails() to show the book information.
2. Create an outer class Car with an inner class Engine. The Engine class should have a method start() that prints "Engine started" and a method stop() that prints "Engine stopped". The Car class should have a method drive() that creates an instance of the Engine class and calls its start() and stop() methods.
3. Create an interface EventHandler with a method handleEvent(). In the main method, demonstrate the use of:
    a. A local inner class inside a method registerEvent() that implements EventHandler and prints "Event handled by local inner class".
    b. An anonymous inner class that implements EventHandler and prints "Event handled by anonymous inner class".
4. Write a Java program that takes two integers as input from the user and performs division. Handle the ArithmeticException that occurs if the denominator is zero. Use a try-catch block to catch the exception and display an appropriate error message. Additionally, use a finally block to print "Operation completed" regardless of whether an exception occurs or not.
5. Write a Java program that creates an array of 5 integers and asks the user to enter an index to access the array element. Handle the ArrayIndexOutOfBoundsException if the user enters an invalid index. Use a try-catch block to catch the exception and display an appropriate error message. Use the finally block to print "Array access attempted."
6. Write a Java program that reads a file name from the user and attempts to open and read the file. Define a method readFile() that throws a FileNotFoundException using the throws keyword. In the main method, call this method and handle the exception using a try-catch block. Display an appropriate message if the file is not found. Use a finally block to ensure a message like "File operation attempted" is printed.
7. Write a Java program that takes user input for a student's name, roll number, and grade, and writes this information to a file named student.txt using **FileWriter**. Ensure the program appends the data to the file if it already exists. Handle any exceptions using try-catch and display an appropriate message if an error occurs.
   Sample File Content:
   Name: Aman, Roll Number: 120112, Grade: A
   Name: Parul, Roll Number: 120131, Grade: B
8. Write a Java program that reads the contents of a file named student.txt using **FileReader** and displays the data on the console. Handle FileNotFoundException if the file does not exist and display an appropriate error message. Use a try-catch block for exception handling.