**Problem Set – 5**

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 Question:

1. 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.
2. 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.