1. Write a Java program to create a class called Animal with a method called makeSound().Create a subclass called Cat that overrides the makeSound() method to bark.

A.

```
class Animal {
    public void makeSound() {
        System.out.println("Animal makes a sound");
    }
}
class Cat extends Animal {
    public void makeSound() {
        System.out.println("Cat meows"); // Cat makes a meowing sound
    }
}
public class AnimalTest {
    public static void main(String[] args) {
        Animal animal = new Animal();
        animal.makeSound();
        Cat cat = new Cat();
        cat.makeSound();
    }
}
```

2. Write a Java program to create a class called Vehicle with a method called drive(). Create a subclass called Car that overrides the drive() method to print "Repairing a car".

A.

```
class Vehicle {
    public void drive() {
        System.out.println("Driving a vehicle");
    }
}
class Car extends Vehicle {
    public void drive() {
```

```
        System.out.println("Repairing a car"); // Car is being repaired
    }
}
public class VehicleTest {
    public static void main(String[] args) {
        Vehicle vehicle = new Vehicle();
        vehicle.drive(); // Output: Driving a vehicle

        Car car = new Car();
        car.drive(); // Output: Repairing a car
    }
}
```

3. Write a Java program to create a class called Shape with a method called getArea(). Create a subclass called Rectangle that overrides the getArea() method to calculate the area of a rectangle.

A.

```
class Shape {
    public double getArea() {
        return 0.0;
    }
}
class Rectangle extends Shape {
    private double width;
    private double height;
    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }
    public double getArea() {
        return width * height; // Area of rectangle calculation
    }
}
```

```
public class ShapeTest {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(5.0, 3.0);
        double area = rectangle.getArea();
        System.out.println("Area of the rectangle: " + area);
    }
}
```

4. Write a Java program to create a class called Employee with methods called work() and getSalary(). Create a subclass called HRManager that overrides the work() method and adds a new method called addEmployee().

A.

```
class Employee {
    public void work() {
        System.out.println("Employee is working");
    }
    public double getSalary() {
        return 0.0; // Default implementation, to be overridden by subclasses
    }
}
class HRManager extends Employee {
    public void work() {
        System.out.println("HR Manager is managing HR tasks"); // Specific work for HR Manager
    }
    public void addEmployee() {
        System.out.println("HR Manager is adding a new employee"); // Adding new employee specific to HR Manager
    }
}
public class EmployeeTest {
    public static void main(String[] args) {
```

```
        Employee employee = new Employee();
        employee.work(); // Output: Employee is working
        System.out.println("Employee salary: " + employee.getSalary());
        System.out.println("---");
        HRManager hrManager = new HRManager();
        hrManager.work(); // Output: HR Manager is managing HR tasks
        System.out.println("HR Manager salary: " + hrManager.getSalary());
        hrManager.addEmployee();
    }
}
```

5. Write a Java program to create a class known as "BankAccount" with methods called deposit()and withdraw(). Create a subclass called SavingsAccount that overrides the withdraw() methodto prevent withdrawals if the account balance falls below one hundred.

A.

```
class BankAccount {
    protected double balance; // Account balance
    public BankAccount(double initialBalance) {
        this.balance = initialBalance;
    }
    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: " + amount);
        System.out.println("New Balance: " + balance);
    }
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
            System.out.println("New Balance: " + balance);
        } else {
            System.out.println("Insufficient funds. Withdrawal not processed.");
        }
```

```
    }
}
class SavingsAccount extends BankAccount {
    public SavingsAccount(double initialBalance) {
        super(initialBalance);
    }
    public void withdraw(double amount) {
        if (balance >= amount && balance - amount >= 100) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
            System.out.println("New Balance: " + balance);
        } else {
            System.out.println("Withdrawal not processed. Minimum balance requirement not met.");
        }
    }
}
public class BankAccountTest {
    public static void main(String[] args) {
        SavingsAccount savingsAccount = new SavingsAccount(500.0);
        savingsAccount.deposit(200.0);
        savingsAccount.withdraw(50.0);
        savingsAccount.withdraw(600.0);
    }
}
```

6. Write a Java program to create a class called Animal with a method named move(). Create a subclass called Cheetah that overrides the move() method to run.

A.

```
class Animal {
    public void move() {
        System.out.println("Animal moves");
    }
```

```
}
class Cheetah extends Animal {
    public void move() {
        System.out.println("Cheetah runs at top speed"); // Cheetah runs
    }
}
public class AnimalTest {
    public static void main(String[] args) {
        Animal animal = new Animal();
        animal.move(); // Output: Animal moves
        Cheetah cheetah = new Cheetah();
        cheetah.move(); // Output: Cheetah runs at top speed
    }
}
```

7. Write a Java program to create a class known as Person with methods called getFirstName() and getLastName(). Create a subclass called Employee that adds a new method named getEmployeeId() and overrides the getLastName() method to include the employee's job title.

A.

```
class Person {
    private String firstName;
    private String lastName;
    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
    public String getFirstName() {
        return firstName;
    }
    public String getLastName() {
        return lastName;
    }
```

```
}
class Employee extends Person {
    private int employeeId;
    private String jobTitle;
    public Employee(String firstName, String lastName, int employeeId, String jobTitle) {
        super(firstName, lastName); // Call superclass constructor
        this.employeeId = employeeId;
        this.jobTitle = jobTitle;
    }
    public int getEmployeeId() {
        return employeeId;
    }
    public String getLastName() {
        return super.getLastName() + ", " + jobTitle; // Append job title to last name
    }
}
public class PersonTest {
    public static void main(String[] args) {
        Employee employee = new Employee("John", "Doe", 1001, "Software Engineer");
        System.out.println("First Name: " + employee.getFirstName());
        System.out.println("Last Name: " + employee.getLastName()); // Output: Doe, Software
Engineer
        System.out.println("Employee ID: " + employee.getEmployeeId());
    }
}
```

8. Write a Java program to create a class called Shape with methods called getPerimeter() and getArea(). Create a subclass called Circle that overrides the getPerimeter() and getArea() methods to calculate the area and perimeter of a circle.

A.

```
abstract class Shape {
    public abstract double getPerimeter();
```

```
    public abstract double getArea();
}
class Circle extends Shape {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    public double getPerimeter() {
        return 2 * Math.PI * radius; // Perimeter of a circle formula: 2 * π * radius
    }
    public double getArea() {
        return Math.PI * radius * radius; // Area of a circle formula: π * radius^2
    }
}
public class ShapeTest {
    public static void main(String[] args) {
        Circle circle = new Circle(5.0);
        System.out.println("Circle with radius " + circle.radius + ":");
        System.out.println("Perimeter: " + circle.getPerimeter());
        System.out.println("Area: " + circle.getArea());
    }
}
```

9. Write a Java program to create a vehicle class hierarchy. The base class should be Vehicle, with subclasses Truck, Car and Motorcycle. Each subclass should have properties such as make, model, year, and fuel type. Implement methods for calculating fuel efficiency, distance traveled, and maximum speed.**?**

A.

```
class Vehicle {
      protected String make;
    protected String model;
    protected int year;
```

```
    protected String fuelType;
    public Vehicle(String make, String model, int year, String fuelType) {
        this.make = make;
        this.model = model;
        this.year = year;
        this.fuelType = fuelType;
    }
    public double calculateFuelEfficiency() {
        return 0.0; // Default implementation, to be overridden by subclasses
    }
    public double calculateDistanceTraveled() {
        return 0.0; // Default implementation, to be overridden by subclasses
    }
    public int getMaximumSpeed() {
        return 0; // Default implementation, to be overridden by subclasses
    }
}
class Truck extends Vehicle {
    private double payloadCapacity; // Specific property for Truck
    public Truck(String make, String model, int year, String fuelType, double payloadCapacity) {
        super(make, model, year, fuelType); // Call superclass constructor
        this.payloadCapacity = payloadCapacity;
    }
    public double calculateFuelEfficiency() {
        return 15.0;
    }
    public double calculateDistanceTraveled() {
        return 500.0;
    }
    public int getMaximumSpeed() {
        return 80;
```

```
    }
}
class Car extends Vehicle {
    private int seatingCapacity;
    public Car(String make, String model, int year, String fuelType, int seatingCapacity) {
        super(make, model, year, fuelType);
        this.seatingCapacity = seatingCapacity;
    }
    public double calculateFuelEfficiency() {
        return 25.0;
    }
    public double calculateDistanceTraveled() {
        return 600.0;
    }
    public int getMaximumSpeed() {
        return 120;
    }
}
class Motorcycle extends Vehicle {
    private boolean hasFairing;
    public Motorcycle(String make, String model, int year, String fuelType, boolean hasFairing) {
        super(make, model, year, fuelType); // Call superclass constructor
        this.hasFairing = hasFairing;
    }
    public double calculateFuelEfficiency() {
        return 50.0;
    }

    @Override
    public double calculateDistanceTraveled() {
        return 300.0;
```

```
    }
    public int getMaximumSpeed() {
        return 150;
    }
}
public class VehicleTest {
    public static void main(String[] args) {
        Truck myTruck = new Truck("Ford", "F-150", 2022, "Diesel", 1500.0);
        Car myCar = new Car("Toyota", "Camry", 2023, "Gasoline", 5);
        Motorcycle myMotorcycle = new Motorcycle("Honda", "CBR600RR", 2021, "Gasoline", true);
        System.out.println("Truck: " + myTruck.make + " " + myTruck.model);
        System.out.println("Fuel Efficiency: " + myTruck.calculateFuelEfficiency() + " mpg");
        System.out.println("Distance Traveled: " + myTruck.calculateDistanceTraveled() + " miles");
        System.out.println("Maximum Speed: " + myTruck.getMaximumSpeed() + " mph");
        System.out.println("Car: " + myCar.make + " " + myCar.model);
        System.out.println("Fuel Efficiency: " + myCar.calculateFuelEfficiency() + " mpg");
        System.out.println("Distance Traveled: " + myCar.calculateDistanceTraveled() + " miles");
        System.out.println("Maximum Speed: " + myCar.getMaximumSpeed() + " mph");
        System.out.println();
        System.out.println("Motorcycle: " + myMotorcycle.make + " " + myMotorcycle.model);
        System.out.println("Fuel Efficiency: " + myMotorcycle.calculateFuelEfficiency() + " mpg");
        System.out.println("Distance Traveled: " + myMotorcycle.calculateDistanceTraveled() + " miles");
        System.out.println("Maximum Speed: " + myMotorcycle.getMaximumSpeed() + " mph");
    }
}
```

10. Write a Java program that creates a class hierarchy for employees of a company. The base class should be Employee, with subclasses Manager, Developer, and Programmer. Each subclass should have properties such as name, address, salary, and job title. Implement methods for calculating bonuses, generating performance reports, and managing projects.

A.

```
class Employee {
```

```
    protected String name;
    protected String address;
    protected double salary;
    protected String jobTitle;
    public Employee(String name, String address, double salary, String jobTitle) {
        this.name = name;
        this.address = address;
        this.salary = salary;
        this.jobTitle = jobTitle;
    }
    public double calculateBonus() {
        return 0.0; // Default implementation, to be overridden by subclasses
    }
    public void generatePerformanceReport() {
        System.out.println("Performance report for " + name + ":");
    }
    public void manageProjects() {
        System.out.println(name + " is managing projects.");
    }
}
class Manager extends Employee {
    private int teamSize;
    public Manager(String name, String address, double salary, String jobTitle, int teamSize) {
        super(name, address, salary, jobTitle);
        this.teamSize = teamSize;
    }
    public double calculateBonus() {
        return salary * 0.2;
    }
    public void generatePerformanceReport() {
        super.generatePerformanceReport();
```

```
        System.out.println("Manager's performance report details.");
    }
    public void manageProjects() {
        System.out.println(name + " is managing projects for a team of " + teamSize + " members.");
    }
}
class Developer extends Employee {
    private String programmingLanguage;
    public Developer(String name, String address, double salary, String jobTitle, String
programmingLanguage) {
        super(name, address, salary, jobTitle); // Call superclass constructor
        this.programmingLanguage = programmingLanguage;
    }

    public double calculateBonus() {
        return salary * 0.1;
    }
    public void generatePerformanceReport() {
        super.generatePerformanceReport();
        System.out.println("Developer's performance report details.");
    }
    public void manageProjects() {
        System.out.println(name + " is developing software in " + programmingLanguage + "."); //
Example
    }
}
class Programmer extends Developer {
    private String specializedArea;
    public Programmer(String name, String address, double salary, String jobTitle, String
programmingLanguage, String specializedArea) {
        super(name, address, salary, jobTitle, programmingLanguage);
        this.specializedArea = specializedArea;
```

```
    }
}
public class EmployeeTest {
    public static void main(String[] args) {
        Manager manager = new Manager("John Doe", "123 Main St, Anytown", 80000.0, "Manager",
10);
        Developer developer = new Developer("Jane Smith", "456 Elm St, Othertown", 60000.0,
"Developer", "Java");
        Programmer programmer = new Programmer("Mike Johnson", "789 Oak St, Thistown", 70000.0,
"Programmer", "Python", "Machine Learning");
        System.out.println("Manager: " + manager.name);
        System.out.println("Salary: $" + manager.salary);
        System.out.println("Bonus: $" + manager.calculateBonus());
        manager.manageProjects();
        manager.generatePerformanceReport();
        System.out.println();
        System.out.println("Developer: " + developer.name);
        System.out.println("Salary: $" + developer.salary);
        System.out.println("Bonus: $" + developer.calculateBonus());
        developer.manageProjects();
        developer.generatePerformanceReport();
        System.out.println();
        System.out.println("Programmer: " + programmer.name);
        System.out.println("Salary: $" + programmer.salary);
        System.out.println("Bonus: $" + programmer.calculateBonus());
        programmer.manageProjects();
        programmer.generatePerformanceReport();
        System.out.println("Specialized Area: " + programmer.specializedArea);
    }
}
```