Assignment 4

1.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().

```
class Employee {
    private double salary;

    public Employee(double salary) {
        this.salary = salary;
    }

    public void work() {
        System.out.println("Employee is working");
    }

    public double getSalary() {
        return salary;
    }
}
class HRManager extends Employee {
    public HRManager(double salary) {
        super(salary);
    }

    @Override
    public void work() {
        System.out.println("HR Manager is managing employees");
    }

    public void addEmployee() {
        System.out.println("HR Manager is adding an employee");
```

```
    }
}
public class Main {
    public static void main(String[] args) {
        Employee employee = new Employee(50000);
        employee.work();
        System.out.println("Salary: " + employee.getSalary());

        HRManager hrManager = new HRManager(70000);
        hrManager.work();
        System.out.println("Salary: " + hrManager.getSalary());
        hrManager.addEmployee();
    }
}
```

Output

```
java -cp /tmp/HijaTASbVE/Main
Employee is working
Salary: 50000.0
HR Manager is managing employees
Salary: 70000.0
HR Manager is adding an employee

=== Code Execution Successful ===
```

2.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.

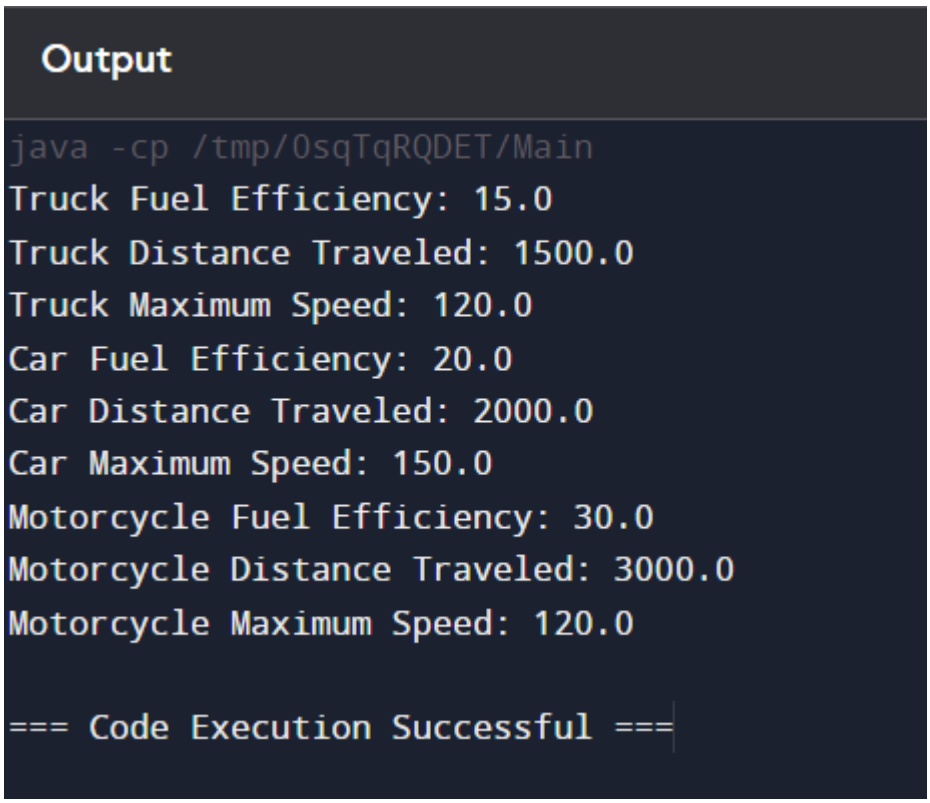
```
class Vehicle {
    protected String make;
    protected String model;
```

```
    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 20;
    }

    public double calculateDistanceTraveled(double fuelConsumed) {
        return fuelConsumed * calculateFuelEfficiency();
    }

    public double getMaximumSpeed() {
        return 120;
    }
}
class Truck extends Vehicle {
    private double cargoCapacity;

    public Truck(String make, String model, int year, String fuelType, double
cargoCapacity) {
        super(make, model, year, fuelType);
        this.cargoCapacity = cargoCapacity;
    }
```

```
    @Override
    public double calculateFuelEfficiency() {
        return 15;
    }

    public double getCargoCapacity() {
        return cargoCapacity;
    }
}
class Car extends Vehicle {
    private int numberOfDoors;

    public Car(String make, String model, int year, String fuelType, int
numberOfDoors) {
        super(make, model, year, fuelType);
        this.numberOfDoors = numberOfDoors;
    }

    @Override
    public double getMaximumSpeed() {
        return 150;
    }

    public int getNumberOfDoors() {
        return numberOfDoors;
    }
}
class Motorcycle extends Vehicle {
    private int engineSize;
```

```
    public Motorcycle(String make, String model, int year, String fuelType, int
engineSize) {
        super(make, model, year, fuelType);
        this.engineSize = engineSize;
    }

    @Override
    public double calculateFuelEfficiency() {
        return 30;
    }

    public int getEngineSize() {
        return engineSize;
    }
}
public class Main {
    public static void main(String[] args) {
        Truck truck = new Truck("Ford", "F-150", 2022, "Gasoline", 2000);
        System.out.println("Truck Fuel Efficiency: " + truck.calculateFuelEfficiency());
        System.out.println("Truck Distance Traveled: " +
truck.calculateDistanceTraveled(100));
        System.out.println("Truck Maximum Speed: " + truck.getMaximumSpeed());

        Car car = new Car("Toyota", "Camry", 2021, "Gasoline", 4);
        System.out.println("Car Fuel Efficiency: " + car.calculateFuelEfficiency());
        System.out.println("Car Distance Traveled: " +
car.calculateDistanceTraveled(100));
        System.out.println("Car Maximum Speed: " + car.getMaximumSpeed());

        Motorcycle motorcycle = new Motorcycle("Harley-Davidson", "Electra Glide",
2020, "Gasoline", 1200);
```

```
        System.out.println("Motorcycle Fuel Efficiency: " +
motorcycle.calculateFuelEfficiency());
        System.out.println("Motorcycle Distance Traveled: " +
motorcycle.calculateDistanceTraveled(100));
        System.out.println("Motorcycle Maximum Speed: " +
motorcycle.getMaximumSpeed());
    }
}
```



3.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.

```
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 salary * 0.1;
    }

    public void generatePerformanceReport() {
        System.out.println("Employee Name: " + name);
        System.out.println("Job Title: " + jobTitle);
        System.out.println("Salary: " + salary);
    }
}
class Manager extends Employee {
    private String department;

    public Manager(String name, String address, double salary, String jobTitle, String
department) {
        super(name, address, salary, jobTitle);
        this.department = department;
    }

    @Override
    public double calculateBonus() {
        return super.calculateBonus() * 1.5;
    }
```

```
    public void manageProject() {
        System.out.println("Manager is managing a project");
    }
}

class Developer extends Employee {
    private String programmingLanguage;

    public Developer(String name, String address, double salary, String jobTitle, String
programmingLanguage) {
        super(name, address, salary, jobTitle);
        this.programmingLanguage = programmingLanguage;
    }

    @Override
    public void generatePerformanceReport() {
        super.generatePerformanceReport();
        System.out.println("Programming Language: " + programmingLanguage);
    }

    public void developSoftware() {
        System.out.println("Developer is developing software");
    }
}
class Programmer extends Developer {
    public Programmer(String name, String address, double salary, String jobTitle,
String programmingLanguage) {
        super(name, address, salary, jobTitle, programmingLanguage);
    }

    @Override
```

```
    public double calculateBonus() {
        return super.calculateBonus() * 1.2;
    }

    public void writeCode() {
        System.out.println("Programmer is writing code");
    }
}
public class Main {
    public static void main(String[] args) {
        Manager manager = new Manager("John Doe", "123 Main St", 100000,
"Manager", "IT");
        manager.generatePerformanceReport();
        System.out.println("Bonus: " + manager.calculateBonus());
        manager.manageProject();

        Developer developer = new Developer("Jane Smith", "456 Elm St", 80000,
"Developer", "Java");
        developer.generatePerformanceReport();
        System.out.println("Bonus: " + developer.calculateBonus());
        developer.developSoftware();

        Programmer programmer = new Programmer("Bob Johnson", "789 Oak St",
70000, "Programmer", "Python");
        programmer.generatePerformanceReport();
        System.out.println("Bonus: " + programmer.calculateBonus());
        programmer.writeCode();
    }
}
```

4. Write a Java program to create an abstract class Shape with abstract methods calculateArea() and calculatePerimeter(). Create subclasses Circle and Triangle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape

```
abstract class Shape {
    public abstract double calculateArea();
    public abstract double calculatePerimeter();
}
class Circle extends Shape {
    private double radius;
```

```
    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }

    @Override
    public double calculatePerimeter() {
        return 2 * Math.PI * radius;
    }
}
class Triangle extends Shape {
    private double base;
    private double height;
    private double side1;
    private double side2;

    public Triangle(double base, double height, double side1, double side2) {
        this.base = base;
        this.height = height;
        this.side1 = side1;
        this.side2 = side2;
    }

    @Override
    public double calculateArea() {
        return 0.5 * base * height;
```

```
    }

    @Override
    public double calculatePerimeter() {
        return base + side1 + side2;
    }
}
public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle(5);
        System.out.println("Circle Area: " + circle.calculateArea());
        System.out.println("Circle Perimeter: " + circle.calculatePerimeter());

        Triangle triangle = new Triangle(3, 4, 5, 6);
        System.out.println("Triangle Area: " + triangle.calculateArea());
        System.out.println("Triangle Perimeter: " + triangle.calculatePerimeter());
    }
}
```

Output

```
java -cp /tmp/4Sj5HLS7bo/Main
Circle Area: 78.53981633974483
Circle Perimeter: 31.41592653589793
Triangle Area: 6.0
Triangle Perimeter: 14.0

=== Code Execution Successful ===
```