

ASSIGNMENT 4

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

CODE:

```
class Employee {  
    public void work() {  
        System.out.println("Employee is working.");  
    }  
  
    public double getSalary() {  
        return 50000.0; // Example salary  
    }  
}  
  
class HRManager extends Employee {  
    @Override  
    public void work() {  
        System.out.println("HR Manager is working.");  
    }  
  
    public void addEmployee() {  
        System.out.println("HR Manager is adding a new employee.");  
    }  
}  
  
public class Main {
```

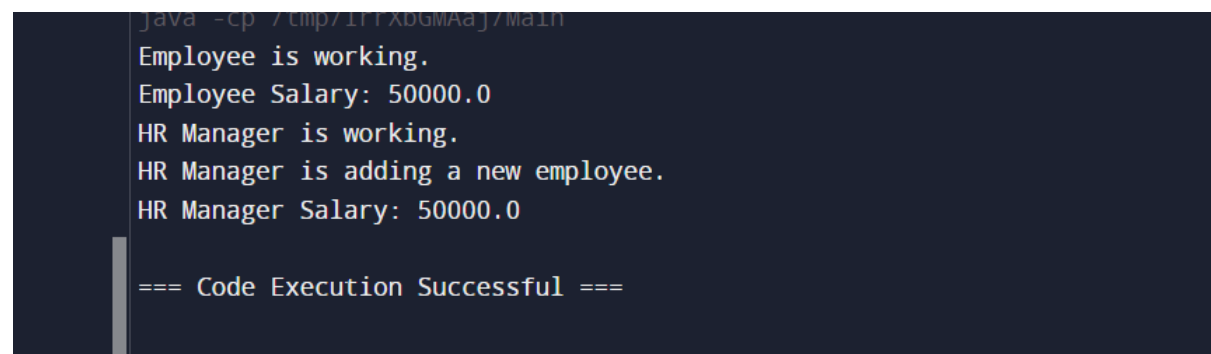
```

public static void main(String[] args) {
    Employee emp = new Employee();
    emp.work();
    System.out.println("Employee Salary: " + emp.getSalary());

    HRManager hr = new HRManager();
    hr.work();
    hr.addEmployee();
    System.out.println("HR Manager Salary: " + hr.getSalary());
}
}

```

OUTPUT:



```

java -cp /tmp/ITFXBGMAAJ/main
Employee is working.
Employee Salary: 50000.0
HR Manager is working.
HR Manager is adding a new employee.
HR Manager Salary: 50000.0

=== 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 travelled, and maximum speed.

CODE:

```

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());

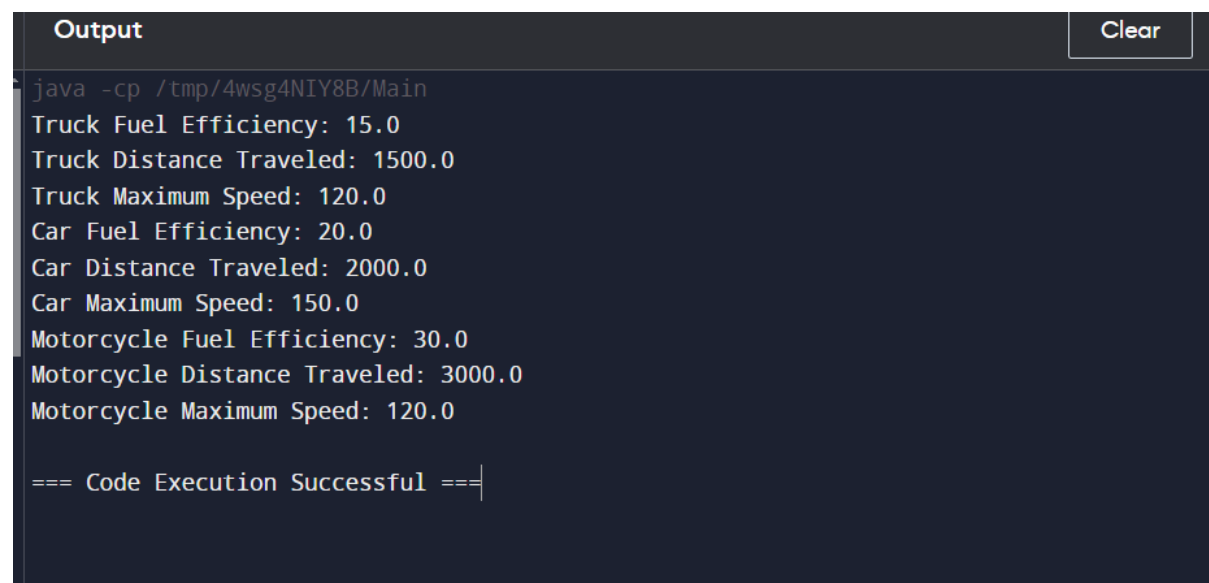

Motorcycle motorcycle = new Motorcycle("Harley-Davidson", "Electra
Glide", 2020, "Gasoline", 1200);

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());
}
}
```

OUTPUT:



```
Output Clear
java -cp /tmp/4wsg4NIY8B/Main
Truck Fuel Efficiency: 15.0
Truck Distance Traveled: 1500.0
Truck Maximum Speed: 120.0
Car Fuel Efficiency: 20.0
Car Distance Traveled: 2000.0
Car Maximum Speed: 150.0
Motorcycle Fuel Efficiency: 30.0
Motorcycle Distance Traveled: 3000.0
Motorcycle Maximum Speed: 120.0

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

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.

CODE:

```
abstract class Employee {
    String name, address, jobTitle;
    double salary;

    Employee(String name, String address, double salary, String jobTitle) {
        this.name = name;
        this.address = address;
        this.salary = salary;
        this.jobTitle = jobTitle;
    }

    abstract double calculateBonus();
    abstract String generatePerformanceReport();
    abstract void manageProject();
}

class Manager extends Employee {
    Manager(String name, String address, double salary) {
        super(name, address, salary, "Manager");
    }

    double calculateBonus() { return salary * 0.10; }
    String generatePerformanceReport() { return name + ": Excellent
```

```
manager.;" }

```

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

```

```
class Developer extends Employee {

```

```
    Developer(String name, String address, double salary) {
        super(name, address, salary, "Developer");
    }

```

```
    double calculateBonus() { return salary * 0.15; }

```

```
    String generatePerformanceReport() { return name + ": Outstanding
developer.;" }

```

```
    void manageProject() { System.out.println(name + " is developing a
project."); }
}

```

```
class Programmer extends Employee {

```

```
    Programmer(String name, String address, double salary) {
        super(name, address, salary, "Programmer");
    }

```

```
    double calculateBonus() { return salary * 0.12; }

```

```
    String generatePerformanceReport() { return name + ": Great
programmer.;" }

```

```
    void manageProject() { System.out.println(name + " is programming a
project."); }
}

```

```
public class Main {

```



```

public static void main(String[] args) {
    Employee[] employees = {
        new Manager("Alice", "123 Manager St", 80000),
        new Developer("Bob", "456 Developer Rd", 70000),
        new Programmer("Charlie", "789 Programmer Ln", 60000)
    };

    for (Employee e : employees) {
        System.out.println(e.generatePerformanceReport() + " Bonus: " +
            e.calculateBonus());
        e.manageProject();
    }
}

```

OUTPUT:

Output

Clear

```

java -cp /tmp/uTStnRViDA/Main
Alice: Excellent manager. Bonus: 8000.0
Alice is managing a project.
Bob: Outstanding developer. Bonus: 10500.0
Bob is developing a project.
Charlie: Great programmer. Bonus: 7200.0
Charlie is programming a project.

=== Code Execution Successful ===

```

4. Write a Java program to create an abstract class Shape with abstract methods calculate Area() and calculate Perimeter(). Create subclasses

Circle and Triangle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.

CODE:

```
abstract class Shape {  
    abstract double calculateArea();  
    abstract double calculatePerimeter();  
}
```

```
class Circle extends Shape {  
    double radius;  
  
    Circle(double radius) {  
        this.radius = radius;  
    }  
  
    double calculateArea() {  
        return Math.PI * radius * radius;  
    }  
  
    double calculatePerimeter() {  
        return 2 * Math.PI * radius;  
    }  
}
```

```
class Triangle extends Shape {  
    double base, height, side1, side2, side3;  
  
    Triangle(double base, double height, double side1, double side2, double
```

```
side3) {  
    this.base = base;  
    this.height = height;  
    this.side1 = side1;  
    this.side2 = side2;  
    this.side3 = side3;  
}
```

```
double calculateArea() {  
    return 0.5 * base * height;  
}
```

```
double calculatePerimeter() {  
    return side1 + side2 + side3;  
}  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Shape circle = new Circle(5);  
        Shape triangle = new Triangle(3, 4, 3, 4, 5);  
  
        System.out.println("Circle area: " + circle.calculateArea() + ", perimeter:  
" + circle.calculatePerimeter());  
  
        System.out.println("Triangle area: " + triangle.calculateArea() + ",  
perimeter: " + triangle.calculatePerimeter());  
    }  
}
```

OUTPUT:

```
Output Clear  
java -cp /tmp/7AlcdNnWmp/Main  
Circle area: 78.53981633974483, perimeter: 31.41592653589793  
Triangle area: 6.0, perimeter: 12.0  
  
=== Code Execution Successful ===
```