Week-3 Lab Assignment

ANSH PANDEY

2300290130036

IT – A -36

Date: April 25, 2025

Problem Statements:

3.1 Write a Java program to create a class called "Person" with a name and age attribute. Create two instances of the "Person" class, set their attributes using the constructor, and print their name and age.

```
package week_3;

public class Person {
    String name;
    int age;
    Person(String name,int age){
        this.name = name;
        this.age = age;
        System.out.println("Name is: "+name);
        System.out.println("Age is: "+age);
    }

    Person(){
    }

    public static void main(String[] args) {

        new Person("adarsh" , 20);
        Person p1 = new Person();
        p1.name = "ansh";
        p1.age = 20;
        System.out.println("name is: "+p1.name);
        System.out.println("age is: "+p1.age);
    }
}
```

Output:

```
javac Person.java

java Person.java

Name is: adarsh

Age is: 20

name is: ansh

age is: 20
```

3.2 Write a Java program to create a class called "Dog" with a name and breed attribute. Create two instances of the "Dog" class, set their attributes using the constructor and modify the attributes using the setter methods and print the updated values.

```java
package week_3;

public class Dog {
    private String name;
    private String breed;

    Dog(){

    }

    Dog(String name,String breed){
        this.name = name;
        this.breed = breed;
    }

    String getname(){
        return this.name;
    }

    void setname(String name){
        this.name  = name;
    }
    String getbreed(){
        return this.breed;
    }

    void setbreed(String breed){
        this.breed = breed;
    }


    public static void main(String[] args) {
        Dog p1 = new Dog();
        p1.setname("Bravo");
        p1.setbreed("German Shepherd");

        System.out.println("name: "+ p1.getname());
        System.out.println("breed is: "+p1.getbreed());

        Dog p2 = new Dog("Tommy" , "Husky");
        System.out.println("name: "+ p2.getname());
        System.out.println("breed is: "+p2.getbreed());
    }
}
```

Output :

```
javac Dog.java

Java dog.java

name: Bravo

breed is: German Shepherd

name: Tommy

breed is: Husky
```

3.3 Write a java program to illustrate the concept of method overloading without constructor.

```
package week_3;
```

```java
public class methodoverloading {
    void add(int a,int b){
        System.out.println("sum: "+(a+b));
    }

    void add(int a,int b,int c){
        System.out.println("sum: "+(a+b+c));
    }

    void add(double a,double b,double c){
        System.out.println("sum :"+(a+b+c));
    }

    public static void main(String[] args) {
        methodoverloading n1 = new methodoverloading();
        n1.add(1, 2);
        n1.add(1, 3, 4);
        n1.add(3.5, 4.6, 9.3);
    }
}
```

Output : javac methodoverloading.java

```
Java methodoverloading.java

sum: 3

sum: 8

sum :17.4
```

3.4 Write a Java program to demonstrate constructor overloading using both default constructor and parameterized constructor

```java
package week_3.constructor_overloading;
public class box {
    int length;
    int width;
    int height;

    box(){
        this.length=this.width=this.height=0;
    }

    box(int l){
        this.length=l;
        this.width=this.height=0;
    }

    box(int l,int b,int h){
        this.length = l;
        this.width = b;
        this.height = h;
    }

    void display(){
        System.out.println("length is: "+length+" width is: "+width+" height is: 
"+height);
    }

    public static void main(String[] args) {
        box b1 = new box();
        box b2 = new box(2);
        box b3 = new box(1,2,3);
        b1.display();
        b2.display();
        b3.display();
    }
}
```

Output: javac box.java

```
java box.java

length is: 0 width is: 0 height is: 0

length is: 2 width is: 0 height is: 0
length is: 1 width is: 2 height is: 3
```

3.5 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() method to prevent withdrawals if the account balance falls below one hundred.

```java
package week_3.bank_account;

class bankaccount {
    public int balance;

    public bankaccount(int amount){
        balance = amount;
    }

    public void deposit(int amount){
        balance = balance+amount;
        System.out.println("Deposit: "+amount);
    }

    public void withdraw(int amount){
        balance = balance - amount;
        System.out.println("withdrew: "+amount);
    }

    public void displaybalance(){
        System.out.println("Balance is :"+balance);
    }
}

class savingaccount extends bankaccount{

    public savingaccount(int amount){
        super(amount);
    }

    @Override
    public void withdraw(int amount){
        if(balance - amount >=100){
            balance= balance - amount;
            System.out.println("withdrew: "+amount);
        }
    }
}

public class Main{
    public static void main(String[] args) {
        savingaccount m1= new savingaccount(500);

        m1.displaybalance();
        m1.withdraw(450);
        m1.withdraw(300);
        m1.displaybalance();
        m1.deposit(100);
        m1.displaybalance();
    }

}
```

```
Output: javac Main.java

          java Main.java

          Balance is :500

          withdrew: 300

          Balance is :200

          Deposit: 100

          Balance is :300
```

3.6 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.

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

```java
}
public class Cheetah extends Animal {
    @Override
    public void move() {
        System.out.println("The cheetah runs swiftly!");
    }
}
public class Main {
    public static void main(String[] args) {
        Animal a1 = new Animal();
        a1.move();

        Cheetah c1 = new Cheetah();
        c1.move();
    }
}
```

Output: javac *.java

```
java Main

The animal moves.

The cheetah runs swiftly!
```

3.7 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.

```java
public 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;
    }
}

public class Employee extends Person {
    private String employeeId;
    private String jobTitle;

    public Employee(String firstName, String lastName, String employeeId, String
jobTitle) {
        super(firstName, lastName);
        this.employeeId = employeeId;
        this.jobTitle = jobTitle;
    }

    public String getEmployeeId() {
        return employeeId;
    }

    @Override
    public String getLastName() {
        return super.getLastName() + " (" + jobTitle + ")";
    }
}

public class Main {
    public static void main(String[] args) {
        Employee emp = new Employee("Alice", "Johnson", "E123", "Software
Engineer");

        System.out.println("First Name: " + emp.getFirstName());
        System.out.println("Last Name: " + emp.getLastName()); // includes job title
        System.out.println("Employee ID: " + emp.getEmployeeId());
    }
}
```

Output: javac *.java

```
java Main
First Name: Alice
Last Name: Johnson (Software Engineer)
Employee ID: E123
```

3.8 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.

```java
public class Shape {
    public double getPerimeter() {
        return 0.0;
    }

    public double getArea() {
        return 0.0;
    }
}

public class Circle extends Shape {
    private double radius;

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

    @Override
    public double getPerimeter() {
        return 2 * Math.PI * radius;
    }

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

```java
public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle(5.0);

        System.out.println("Circle Radius: 5.0");
        System.out.println("Perimeter: " + circle.getPerimeter());
        System.out.println("Area: " + circle.getArea());
    }
}
```

Output: javac *.java

```
java Main

Circle Radius: 5.0

Perimeter: 31.41592653589793

Area: 78.53981633974483
```

3.9 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.

```java
public 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;
    }
    public double calculateDistanceTraveled(double fuelUsed) {
        return fuelUsed * calculateFuelEfficiency();
    }

    public int getMaxSpeed() {
        return 0;
    }

    public void displayInfo() {
        System.out.println("Make: " + make);
        System.out.println("Model: " + model);
        System.out.println("Year: " + year);
        System.out.println("Fuel Type: " + fuelType);
    }
}

public class Car extends Vehicle {
    public Car(String make, String model, int year, String fuelType) {
        super(make, model, year, fuelType);
    }

    @Override
    public double calculateFuelEfficiency() {
        return 15.0;
    }

    @Override
    public int getMaxSpeed() {
        return 180;
    }
}

public class Truck extends Vehicle {
    public Truck(String make, String model, int year, String fuelType) {
        super(make, model, year, fuelType);
    }

    @Override
    public double calculateFuelEfficiency() {
        return 6.0; // km per liter (example)
    }

    @Override
    public int getMaxSpeed() {
        return 120; // km/h
    }
}

public class Motorcycle extends Vehicle {
    public Motorcycle(String make, String model, int year, String fuelType) {
        super(make, model, year, fuelType);
    }

    @Override
    public double calculateFuelEfficiency() {
        return 35.0;
    }

    @Override
    public int getMaxSpeed() {
        return 160;
    }
}

public class Main {
    public static void main(String[] args) {
        Truck truck = new Truck("Volvo", "FH16", 2022, "Diesel");
        Car car = new Car("Toyota", "Camry", 2023, "Petrol");
        Motorcycle motorcycle = new Motorcycle("Yamaha", "R15", 2021, "Petrol");

        System.out.println("Truck Info:");
        truck.displayInfo();
        System.out.println("Fuel Efficiency: " + truck.calculateFuelEfficiency() + "
km/l");
        System.out.println("Distance with 10L: " +
truck.calculateDistanceTraveled(10) + " km");
        System.out.println("Max Speed: " + truck.getMaxSpeed() + " km/h\n");

        System.out.println("Car Info:");
        car.displayInfo();
        System.out.println("Fuel Efficiency: " + car.calculateFuelEfficiency() + "
km/l");
        System.out.println("Distance with 10L: " + car.calculateDistanceTraveled(10)
+ " km");
        System.out.println("Max Speed: " + car.getMaxSpeed() + " km/h\n");

        System.out.println("Motorcycle Info:");
        motorcycle.displayInfo();
        System.out.println("Fuel Efficiency: " + motorcycle.calculateFuelEfficiency()
+ " km/l");
        System.out.println("Distance with 10L: " +
motorcycle.calculateDistanceTraveled(10) + " km");
        System.out.println("Max Speed: " + motorcycle.getMaxSpeed() + " km/h");
    }
}
```

Output: javac *.java

```
java Main

Truck Info:

Make: Volvo

Model: FH16

Year: 2022

Fuel Type: Diesel

Fuel Efficiency: 6.0 km/l

Distance with 10L: 60.0 km

Max Speed: 120 km/h


Car Info:
```

```
Make: Toyota
Model: Camry
Year: 2023
Fuel Type: Petrol
Fuel Efficiency: 15.0 km/l
Distance with 10L: 150.0 km
Max Speed: 180 km/h

Motorcycle Info:
Make: Yamaha
Model: R15
Year: 2021
Fuel Type: Petrol
Fuel Efficiency: 35.0 km/l
Distance with 10L: 350.0 km
Max Speed: 160 km/h
```

3.10 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.

```java
public 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 * 0.05; // base bonus
    }

    public void generatePerformanceReport() {
        System.out.println(name + " (" + jobTitle + ") performance: Satisfactory");
    }

    public void manageProjects() {
        System.out.println(name + " is assigned to general tasks.");
    }

    public void displayInfo() {
        System.out.println("Name: " + name);
        System.out.println("Address: " + address);
        System.out.println("Salary: " + salary);
        System.out.println("Job Title: " + jobTitle);
    }
}

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

    @Override
    public double calculateBonus() {
        return salary * 0.15;
    }

    @Override
    public void generatePerformanceReport() {
        System.out.println(name + " (" + jobTitle + ") performance: Excellent");
    }

    @Override
    public void manageProjects() {
        System.out.println(name + " is managing multiple departments and strategic 
projects.");
    }
}

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

    @Override
    public double calculateBonus() {
        return salary * 0.08;
    }

    @Override
    public void generatePerformanceReport() {
        System.out.println(name + " (" + jobTitle + ") performance: Good");
    }

    @Override
    public void manageProjects() {
        System.out.println(name + " is writing code and debugging software.");
    }
}

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

    @Override
    public double calculateBonus() {
        return salary * 0.10;
    }
    @Override
    public void generatePerformanceReport() {
        System.out.println(name + " (" + jobTitle + ") performance: Very Good");
    }

    @Override
    public void manageProjects() {
        System.out.println(name + " is developing software modules and managing
coding tasks.");
    }
}

public class Main {
    public static void main(String[] args) {
        Manager mgr = new Manager("Alice Smith", "123 Main St", 90000);
        Developer dev = new Developer("Bob Johnson", "456 Elm St", 70000);
        Programmer prog = new Programmer("Charlie Brown", "789 Oak St",
60000);

        System.out.println("Manager Info:");
        mgr.displayInfo();
        System.out.println("Bonus: $" + mgr.calculateBonus());
        mgr.generatePerformanceReport();
        mgr.manageProjects();

        System.out.println("\nDeveloper Info:");
        dev.displayInfo();
        System.out.println("Bonus: $" + dev.calculateBonus());
        dev.generatePerformanceReport();
        dev.manageProjects();

        System.out.println("\nProgrammer Info:");
        prog.displayInfo();
        System.out.println("Bonus: $" + prog.calculateBonus());
        prog.generatePerformanceReport();
        prog.manageProjects();
    }
}
```

```
Output: javac *.java
java Main
Programmer Info:
Name: Charlie Brown
Address: 789 Oak St
Salary: 60000.0
Job Title: Programmer
Bonus: $4800.0
Charlie Brown (Programmer) performance: Good
Charlie Brown is writing code and debugging software.
```