**WEEK-1 HANDS ON**

1. Implementation of singleton design.

Code:public class Square{

    private static Square instance;

    private int side;

    private **Square**(int side) {

*this*.side = side;

    }

    public static Square **getInstance**(int side) {

        if (instance == null) {

            instance = new **Square(side)**;

        }

        return instance;

    }

    public int **getArea**() {

        return side \* side;

    }

    public int **getSide**() {

        return side;

    }

    public void **setSide**(int side) {

*this*.side = side;

    }

    public static void **main**(String[] args) {

        Square square1 = Square.**getInstance**(5);

        System.out.**println**("Square1 side: " + square1.**getSide**());

        System.out.**println**("Square1 area: " + square1.**getArea**());

        Square square2 = Square.**getInstance**(10);

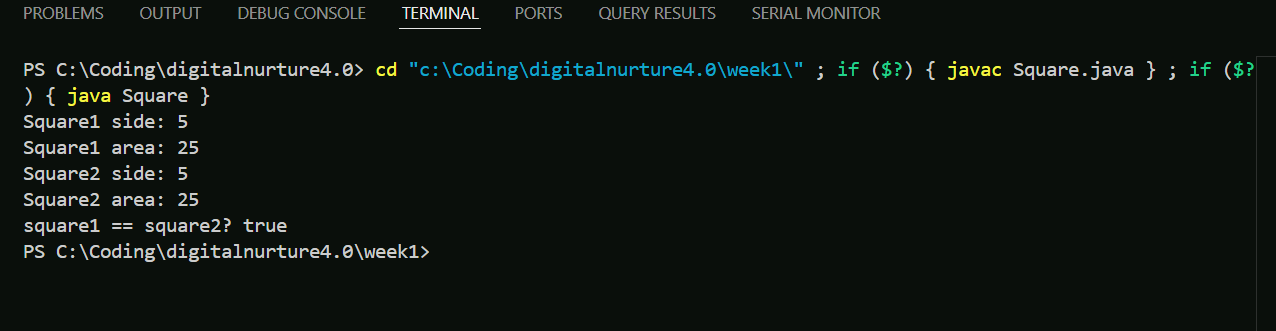
        System.out.**println**("Square2 side: " + square2.**getSide**());

        System.out.**println**("Square2 area: " + square2.**getArea**());

        System.out.**println**("square1 == square2? " + (square1 == square2));

    }

}

Output:

1. Implementation of factory pattern.

Code:interface Shape {

    void **draw**();

}

class Circle implements Shape {

    public void **draw**() {

        System.out.**println**("Drawing a Circle");

    }

}

class Square implements Shape {

    public void **draw**() {

        System.out.**println**("Drawing a Square");

    }

}

class Rectangle implements Shape {

    public void **draw**() {

        System.out.**println**("Drawing a Rectangle");

    }

}

class ShapeFactory {

    public Shape **createShape**(String type) {

        if (type == null) {

            return null;

        }

        switch (type.**toLowerCase**()) {

            case "circle":

                return new **Circle()**;

            case "square":

                return new **Square()**;

            case "rectangle":

                return new **Rectangle()**;

            default:

                throw new **IllegalArgumentException("Unknown shape type: " + type)**;

        }

    }

}

public class factorypatten {

    public static void **main**(String[] args) {

        ShapeFactory factory = new **ShapeFactory()**;

        Shape shape1 = factory.**createShape**("circle");

        shape1.**draw**();

        Shape shape2 = factory.**createShape**("square");

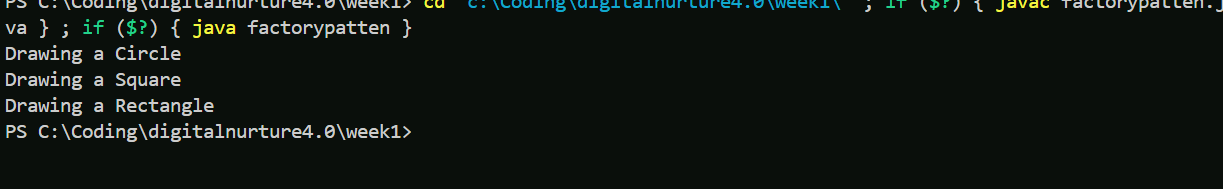
        shape2.**draw**();

        Shape shape3 = factory.**createShape**("rectangle");

        shape3.**draw**();

    }

}

Output:

3.E-commerce system using data structure.

Code:import java.util.*\**;

class Product {

    private int id;

    private String name;

    private double price;

    public **Product**(int id, String name, double price) {

*this*.id = id;

*this*.name = name;

*this*.price = price;

    }

    public int **getId**() { return id; }

    public String **getName**() { return name; }

    public double **getPrice**() { return price; }

    public void **display**() {

        System.out.**printf**("ID: %d | Name: %s | Price: ₹%.2f\n", id, name, price);

    }

}

class CartItem {

    Product product;

    int quantity;

    public **CartItem**(Product product, int quantity) {

*this*.product = product;

*this*.quantity = quantity;

    }

    public double **getTotalPrice**() {

        return product.**getPrice**() \* quantity;

    }

    public void **display**() {

        System.out.**printf**("%s x%d = ₹%.2f\n", product.**getName**(), quantity, **getTotalPrice()**);

    }

}

class Cart {

    private List<CartItem> items = new ArrayList<>();

    public void **addItem**(Product product, int quantity) {

        for (CartItem item : items) {

            if (item.product.**getId**() == product.**getId**()) {

                item.quantity += quantity;

                return;

            }

        }

        items.**add**(new **CartItem(product, quantity)**);

    }

    public void **viewCart**() {

        if (items.**isEmpty**()) {

            System.out.**println**("Cart is empty.");

            return;

        }

        System.out.**println**("Your Cart:");

        for (CartItem item : items) {

            item.**display**();

        }

        System.out.**printf**("Total: ₹%.2f\n", **getTotal()**);

    }

    public double **getTotal**() {

        double total = 0;

        for (CartItem item : items) {

            total += item.**getTotalPrice**();

        }

        return total;

    }

    public List<CartItem> **getItems**() {

        return items;

    }

    public void **clear**() {

        items.**clear**();

    }

}

class Order {

    private static int orderCount = 0;

    private int orderId;

    private List<CartItem> items;

    private double total;

    public **Order**(List<CartItem> items, double total) {

*this*.orderId = ++orderCount;

*this*.items = new ArrayList<>(items);

*this*.total = total;

    }

    public void **display**() {

        System.out.**println**("Order ID: " + orderId);

        for (CartItem item : items) {

            item.**display**();

        }

        System.out.**printf**("Total Amount: ₹%.2f\n", total);

    }

}

public class ECommerceSystem {

    public static void **main**(String[] args) {

        Scanner sc = new **Scanner(System.in)**;

        List<Product> catalog = Arrays.**asList**(

            new **Product(1, "Samgsung M33", 20000)**,

            new **Product(2, "HP Laptop", 50000)**,

            new **Product(3, "Reebok Shoes", 2000)**,

            new **Product(4, "LED TV", 30000)**

        );

        Cart cart = new **Cart()**;

        List<Order> orderHistory = new ArrayList<>();

        while (true) {

            System.out.**println**("\n--- E-Commerce System ---");

            System.out.**println**("1. View Products");

            System.out.**println**("2. Add to Cart");

            System.out.**println**("3. View Cart");

            System.out.**println**("4. Place Order");

            System.out.**println**("5. View Orders");

            System.out.**println**("6. Exit");

            System.out.**print**("Choose an option: ");

            int choice = sc.**nextInt**();

            switch (choice) {

                case 1:

                    System.out.**println**("Available Products:");

                    for (Product p : catalog) {

                        p.**display**();

                    }

                    break;

                case 2:

                    System.out.**print**("Enter Product ID: ");

                    int pid = sc.**nextInt**();

                    System.out.**print**("Enter Quantity: ");

                    int qty = sc.**nextInt**();

                    Product selected = null;

                    for (Product p : catalog) {

                        if (p.**getId**() == pid) {

                            selected = p;

                            break;

                        }

                    }

                    if (selected != null) {

                        cart.**addItem**(selected, qty);

                        System.out.**println**("Added to cart!");

                    } else {

                        System.out.**println**("Product not found.");

                    }

                    break;

                case 3:

                    cart.**viewCart**();

                    break;

                case 4:

                    if (cart.**getItems**().**isEmpty**()) {

                        System.out.**println**("Cart is empty.");

                    } else {

                        Order order = new **Order(cart.getItems(), cart.getTotal())**;

                        orderHistory.**add**(order);

                        System.out.**println**("Order placed successfully!");

                        order.**display**();

                        cart.**clear**();

                    }

                    break;

                case 5:

                    if (orderHistory.**isEmpty**()) {

                        System.out.**println**("No orders placed yet.");

                    } else {

                        for (Order o : orderHistory) {

                            o.**display**();

                            System.out.**println**("-----");

                        }

                    }

                    break;

                case 6:

                    System.out.**println**("Thank you for shopping!");

                    sc.**close**();

                    return;

                default:

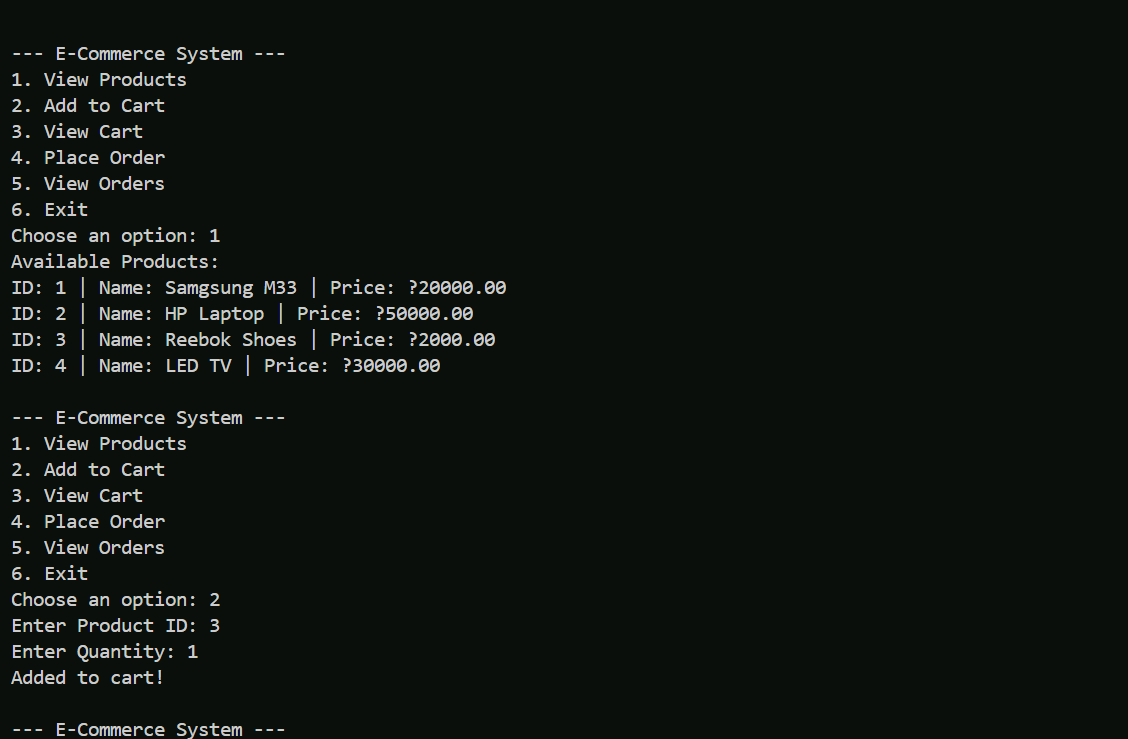
                    System.out.**println**("Invalid option. Try again.");

            }

        }

    }

}

Output:

4.Forecasting system using datastructure.

Code:import java.util.*\**;

public class FinancialForecast {

    private List<Double> historicalData;

    public **FinancialForecast**(List<Double> historicalData) {

*this*.historicalData = historicalData;

    }

    public double **predictNext**() {

        int n = historicalData.**size**();

        double sumX = 0, sumY = 0, sumXY = 0, sumXX = 0;

        for (int i = 0; i < n; i++) {

            double x = i + 1;

            double y = historicalData.**get**(i);

            sumX += x;

            sumY += y;

            sumXY += x \* y;

            sumXX += x \* x;

        }

        double slope = (n \* sumXY - sumX \* sumY) / (n \* sumXX - sumX \* sumX);

        double intercept = (sumY - slope \* sumX) / n;

        double nextX = n + 1;

        return intercept + slope \* nextX;

    }

    public static void **main**(String[] args) {

        List<Double> revenues = Arrays.**asList**(10000.0, 12000.0, 12500.0, 15000.0, 16000.0);

        FinancialForecast forecast = new **FinancialForecast(revenues)**;

        double nextMonthPrediction = forecast.**predictNext**();

        System.out.**printf**("Predicted revenue for next month: $%.2f\n", nextMonthPrediction);

    }

}

Output: