**Superset ID: 6416829**

**Exercise 1: Inventory Management System**

**PROGRAM:**

import java.util.\*;

public class InventorySystem {

    private static Map<Integer, Product> inventory = new HashMap<>();

    private static Scanner sc = new Scanner(System.in);

    public static void addProduct() {

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

        int id = sc.nextInt();

        sc.nextLine(); // clear buffer

        System.out.print("Enter Product Name: ");

        String name = sc.nextLine();

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

        int qty = sc.nextInt();

        System.out.print("Enter Price: ");

        double price = sc.nextDouble();

        if (!inventory.containsKey(id)) {

            inventory.put(id, new Product(id, name, qty, price));

            System.out.println("Product added.");

        } else {

            System.out.println("Product ID already exists.");

        }

    }

    public static void updateProduct() {

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

        int id = sc.nextInt();

        if (inventory.containsKey(id)) {

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

            int qty = sc.nextInt();

            System.out.print("Enter new Price: ");

            double price = sc.nextDouble();

            Product p = inventory.get(id);

            p.quantity = qty;

            p.price = price;

            System.out.println("Product updated.");

        } else {

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

        }

    }

    public static void deleteProduct() {

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

        int id = sc.nextInt();

        if (inventory.remove(id) != null) {

            System.out.println("Product deleted.");

        } else {

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

        }

    }

    public static void displayInventory() {

        if (inventory.isEmpty()) {

            System.out.println("Inventory is empty.");

        } else {

            System.out.println("Inventory List:");

            for (Product p : inventory.values()) {

                System.out.println(p);

            }

        }

    }

    public static void main(String[] args) {

        while (true) {

            System.out.println("\n===== Inventory Menu =====");

            System.out.println("1. Add Product");

            System.out.println("2. Update Product");

            System.out.println("3. Delete Product");

            System.out.println("4. View Inventory");

            System.out.println("5. Exit");

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

            int choice = sc.nextInt();

            switch (choice) {

                case 1: addProduct(); break;

                case 2: updateProduct(); break;

                case 3: deleteProduct(); break;

                case 4: displayInventory(); break;

                case 5: System.out.println("Exiting..."); return;

                default: System.out.println(" Invalid choice.");

            }

        }

    }

}

class Product {

    int productId;

    String productName;

    int quantity;

    double price;

    public Product(int productId, String productName, int quantity, double price) {

        this.productId = productId;

        this.productName = productName;

        this.quantity = quantity;

        this.price = price;

    }

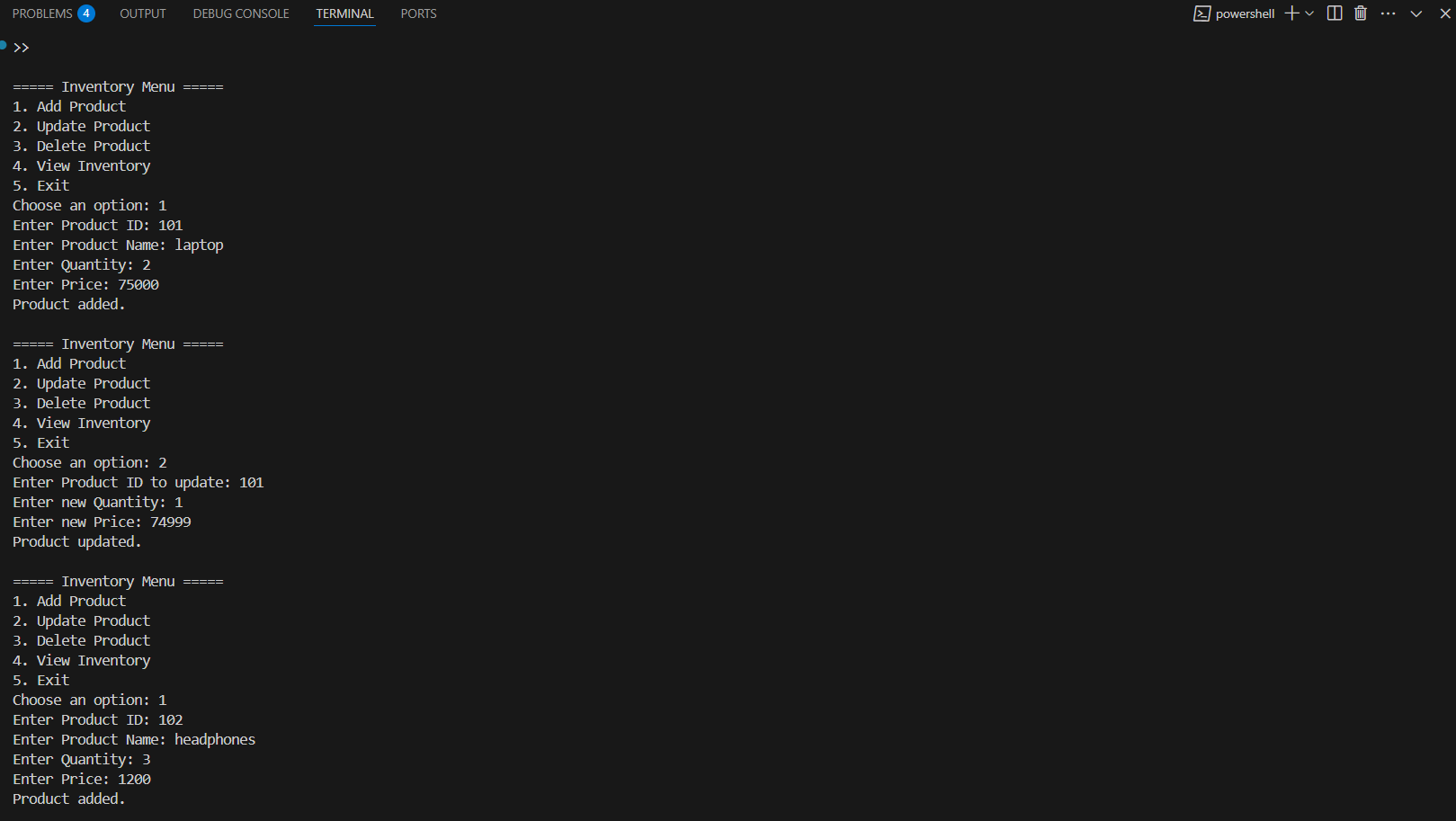
    public String toString() {

        return "ID: " + productId + ", Name: " + productName + ", Qty: " + quantity + ", Price: " + price;

    }

}

**OUTPUT:**

****

**Exercise 2: E-commerce Platform Search Function**

**PROGRAM:**

import java.util.\*;

class Product {

int productId;

String productName;

String category;

public Product(int productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String toString() {

return "ID: " + productId + ", Name: " + productName + ", Category: " + category;

}

}

public class ECommerceSearch {

public static Product linearSearch(Product[] products, String targetName) {

for (Product p : products) {

if (p.productName.equalsIgnoreCase(targetName)) {

return p;

}

}

return null;

}

public static Product binarySearch(Product[] products, String targetName) {

Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));

int left = 0, right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

int cmp = targetName.compareToIgnoreCase(products[mid].productName);

if (cmp == 0) {

return products[mid];

} else if (cmp < 0) {

right = mid - 1;

} else {

left = mid + 1;

}

}

return null;

}

public static void main(String[] args) {

Product[] products = {

new Product(1, "Laptop", "Electronics"),

new Product(2, "Shoes", "Fashion"),

new Product(3, "Watch", "Accessories"),

new Product(4, "Smartphone", "Electronics"),

new Product(5, "T-shirt", "Fashion")

};

Scanner sc = new Scanner(System.in);

System.out.print("Enter product name to search (Linear Search): ");

String search1 = sc.nextLine();

Product result1 = linearSearch(products, search1);

if (result1 != null) {

System.out.println("Found using Linear Search: " + result1);

} else {

System.out.println("Product not found using Linear Search.");

}

System.out.print("\nEnter product name to search (Binary Search): ");

String search2 = sc.nextLine();

Product result2 = binarySearch(products, search2);

if (result2 != null) {

System.out.println("Found using Binary Search: " + result2);

} else {

System.out.println("Product not found using Binary Search.");

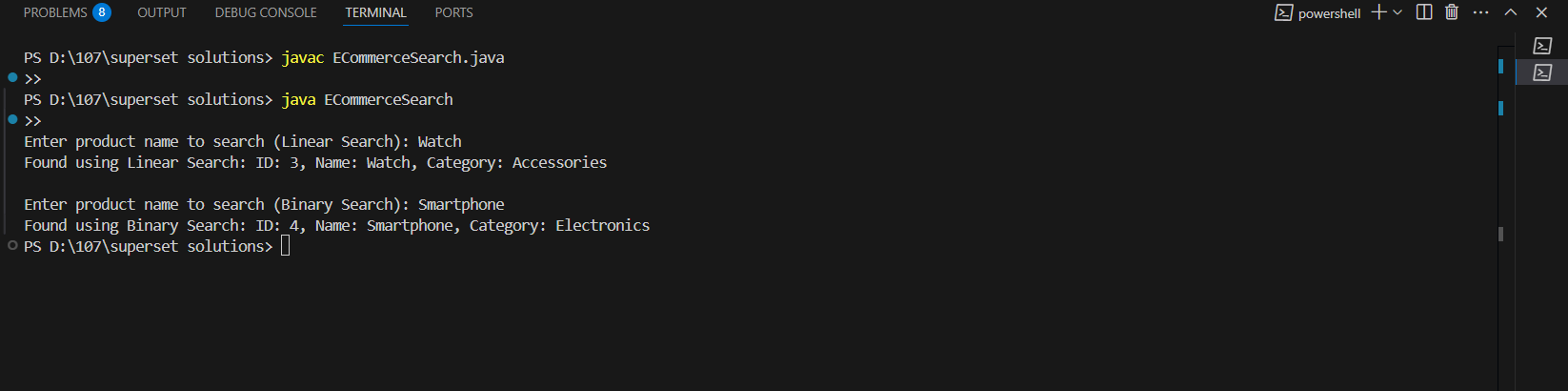
}

sc.close();

}

}

**OUTPUT:**

****

**Exercise 3: Sorting Customer Orders**

**PROGRAM:**

import java.util.\*;

class Order {

int orderId;

String customerName;

double totalPrice;

public Order(int orderId, String customerName, double totalPrice) {

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

public String toString() {

return "Order ID: " + orderId + ", Name: " + customerName + ", Total: " + totalPrice;

}

}

public class OrderSorting {

public static void bubbleSort(Order[] orders) {

int n = orders.length;

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - 1 - i; j++) {

if (orders[j].totalPrice > orders[j + 1].totalPrice) {

Order temp = orders[j];

orders[j] = orders[j + 1];

orders[j + 1] = temp;

}

}

}

}

public static void quickSort(Order[] orders, int low, int high) {

if (low < high) {

int pi = partition(orders, low, high);

quickSort(orders, low, pi - 1);

quickSort(orders, pi + 1, high);

}

}

public static int partition(Order[] orders, int low, int high) {

double pivot = orders[high].totalPrice;

int i = low - 1;

for (int j = low; j < high; j++) {

if (orders[j].totalPrice <= pivot) {

i++;

Order temp = orders[i];

orders[i] = orders[j];

orders[j] = temp;

}

}

Order temp = orders[i + 1];

orders[i + 1] = orders[high];

orders[high] = temp;

return i + 1;

}

public static void displayOrders(Order[] orders) {

for (Order o : orders) {

System.out.println(o);

}

}

public static void main(String[] args) {

Order[] orders = {

new Order(101, "Alice", 2500),

new Order(102, "Bob", 1200),

new Order(103, "Charlie", 4300),

new Order(104, "David", 3100)

};

System.out.println("Original Orders:");

displayOrders(orders);

// Bubble Sort

System.out.println("\nOrders Sorted by Bubble Sort (by totalPrice):");

bubbleSort(orders);

displayOrders(orders);

// Reset original order

orders = new Order[]{

new Order(101, "Alice", 2500),

new Order(102, "Bob", 1200),

new Order(103, "Charlie", 4300),

new Order(104, "David", 3100)

};

// Quick Sort

System.out.println("\nOrders Sorted by Quick Sort (by totalPrice):");

quickSort(orders, 0, orders.length - 1);

displayOrders(orders);

}

}

**OUTPUT:**

****

**Exercise 4: Employee Management System**

**PROGRAM:**

public class EmployeeManagementSystem {

public static void main(String[] args) {

EmployeeManagement manager = new EmployeeManagement();

manager.addEmployee(new Employee(101, "Alice", "Manager", 60000));

manager.addEmployee(new Employee(102, "Bob", "Developer", 50000));

manager.addEmployee(new Employee(103, "Charlie", "Analyst", 45000));

System.out.println("All Employees:");

manager.traverseEmployees();

System.out.println("\nSearching for Employee ID 102:");

Employee found = manager.searchEmployee(102);

if (found != null) found.display();

else System.out.println("Employee not found.");

System.out.println("\nDeleting Employee ID 102...");

manager.deleteEmployee(102);

System.out.println("\nEmployees after deletion:");

manager.traverseEmployees();

}

}

class Employee {

int employeeId;

String name;

String position;

double salary;

public Employee(int id, String name, String position, double salary) {

this.employeeId = id;

this.name = name;

this.position = position;

this.salary = salary;

}

public void display() {

System.out.println(employeeId + " " + name + " " + position + " " + salary);

}

}

class EmployeeManagement {

Employee[] employees = new Employee[100];

int count = 0;

public void addEmployee(Employee e) {

employees[count++] = e;

}

public Employee searchEmployee(int id) {

for (int i = 0; i < count; i++) {

if (employees[i].employeeId == id) return employees[i];

}

return null;

}

public void traverseEmployees() {

for (int i = 0; i < count; i++) {

employees[i].display();

}

}

public void deleteEmployee(int id) {

for (int i = 0; i < count; i++) {

if (employees[i].employeeId == id) {

for (int j = i; j < count - 1; j++) {

employees[j] = employees[j + 1];

}

employees[--count] = null;

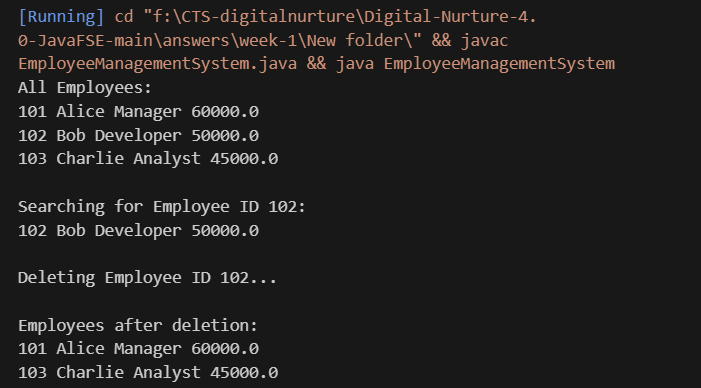
return;

}

}

}

}

**OUTPUT: **

**Exercise 5: Task Management System**

**PROGRAM:**

public class TaskManagementSystem {

public static void main(String[] args) {

TaskManagement tm = new TaskManagement();

tm.addTask(new Task(1, "Design UI", "Pending"));

tm.addTask(new Task(2, "Write Code", "In Progress"));

tm.addTask(new Task(3, "Test App", "Pending"));

System.out.println("All Tasks:");

tm.traverseTasks();

System.out.println("\nSearching for Task ID 2:");

Task found = tm.searchTask(2);

if (found != null) found.display();

else System.out.println("Task not found.");

System.out.println("\nDeleting Task ID 2...");

tm.deleteTask(2);

System.out.println("\nTasks After Deletion:");

tm.traverseTasks();

}

}

class Task {

int taskId;

String taskName;

String status;

Task next;

public Task(int taskId, String taskName, String status) {

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

this.next = null;

}

public void display() {

System.out.println(taskId + " " + taskName + " " + status);

}

}

class TaskManagement {

Task head = null;

public void addTask(Task newTask) {

if (head == null) head = newTask;

else {

Task temp = head;

while (temp.next != null) temp = temp.next;

temp.next = newTask;

}

}

public Task searchTask(int id) {

Task temp = head;

while (temp != null) {

if (temp.taskId == id) return temp;

temp = temp.next;

}

return null;

}

public void traverseTasks() {

Task temp = head;

while (temp != null) {

temp.display();

temp = temp.next;

}

}

public void deleteTask(int id) {

if (head == null) return;

if (head.taskId == id) {

head = head.next;

return;

}

Task prev = head, curr = head.next;

while (curr != null) {

if (curr.taskId == id) {

prev.next = curr.next;

return;

}

prev = curr;

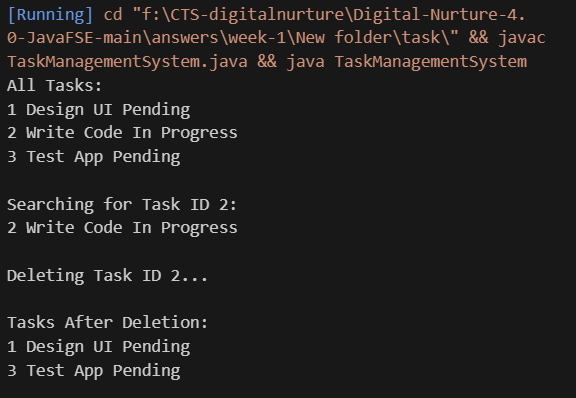
curr = curr.next;

}

}

}

**OUTPUT:**



**Exercise 6: Library Management System**

**PROGRAM**

import java.util.Arrays;

public class LibraryManagementSystem {

public static void main(String[] args) {

Book[] books = {

new Book(1, "Java Programming", "James Gosling"),

new Book(2, "Algorithms", "Robert Sedgewick"),

new Book(3, "Data Structures", "Mark Allen Weiss")

};

System.out.println("Linear Search for 'Algorithms':");

Book found = linearSearch(books, "Algorithms");

if (found != null) found.display();

else System.out.println("Book not found.");

Arrays.sort(books, (b1, b2) -> b1.title.compareToIgnoreCase(b2.title));

System.out.println("\nBinary Search for 'Java Programming':");

found = binarySearch(books, "Java Programming");

if (found != null) found.display();

else System.out.println("Book not found.");

}

public static Book linearSearch(Book[] books, String title) {

for (Book b : books) {

if (b != null && b.title.equalsIgnoreCase(title)) return b;

}

return null;

}

public static Book binarySearch(Book[] books, String title) {

int left = 0, right = books.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

int cmp = books[mid].title.compareToIgnoreCase(title);

if (cmp == 0) return books[mid];

else if (cmp < 0) left = mid + 1;

else right = mid - 1;

}

return null;

}

}

class Book {

int bookId;

String title;

String author;

public Book(int bookId, String title, String author) {

this.bookId = bookId;

this.title = title;

this.author = author;

}

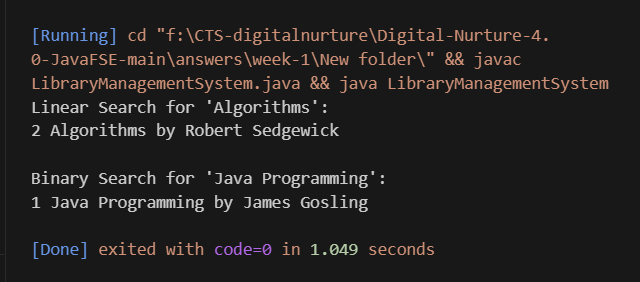
public void display() {

System.out.println(bookId + " " + title + " by " + author);

}

}

**OUTPUT:**



**Exercise 7: Financial Forecasting**

**PROGRAM:**

import java.util.Scanner;

public class FinancialForecasting {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Taking user input

System.out.print("Enter Present Value (e.g., 10000): ");

double presentValue = scanner.nextDouble();

System.out.print("Enter Annual Growth Rate (e.g., 0.10 for 10%): ");

double rate = scanner.nextDouble();

System.out.print("Enter Number of Years: ");

int years = scanner.nextInt();

System.out.println("\nForecast Using Recursive Method:");

double futureRecursive = futureValueRecursive(presentValue, rate, years);

System.out.println("Future Value after " + years + " years = " + futureRecursive);

System.out.println("\nForecast Using Iterative Method (Year-wise):");

futureValueIterative(presentValue, rate, years);

}

// Recursive approach

public static double futureValueRecursive(double presentValue, double rate, int years) {

if (years == 0) return presentValue;

return futureValueRecursive(presentValue \* (1 + rate), rate, years - 1);

}

// Iterative approach with year-wise display

public static void futureValueIterative(double presentValue, double rate, int years) {

double value = presentValue;

for (int i = 1; i <= years; i++) {

value = value \* (1 + rate);

System.out.printf("Year %d: %.2f\n", i, value);

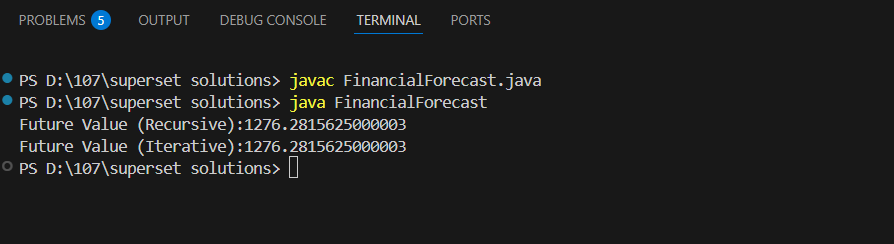
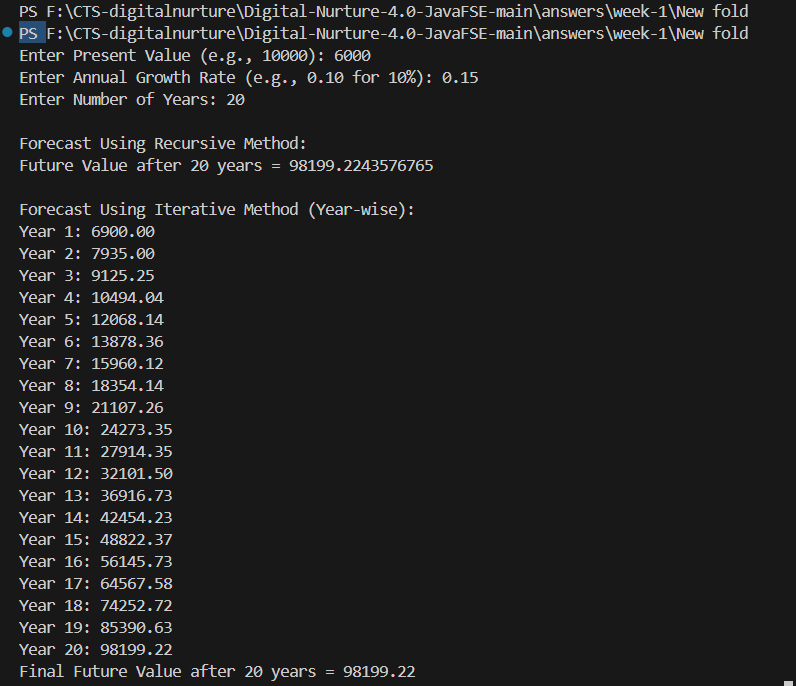
}

System.out.printf("Final Future Value after %d years = %.2f\n", years, value);

}

}

**OUTPUT:**

****