**Exercise-1**

**Code:**

import java.util.HashMap;

import java.util.Scanner;

class Product {

String id;

String name;

int quantity;

double price;

Product(String id, String name, int quantity, double price) {

this.id = id;

this.name = name;

this.quantity = quantity;

this.price = price;

}

public String toString() {

return "ID: " + id + ", Name: " + name +

", Quantity: " + quantity + ", Price: $" + price;

}

}

public class InventorySystem {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

HashMap<String, Product> inventory = new HashMap<>();

while (true) {

System.out.println("\n1. Add Product");

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

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

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

System.out.println("5. View All Products");

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

System.out.print("Enter your choice: ");

int choice = input.nextInt();

input.nextLine(); // clear newline

String id, name;

int quantity;

double price;

switch (choice) {

case 1:

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

id = input.nextLine();

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

name = input.nextLine();

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

quantity = input.nextInt();

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

price = input.nextDouble();

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

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

break;

case 2:

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

id = input.nextLine();

if (inventory.containsKey(id)) {

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

quantity = input.nextInt();

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

price = input.nextDouble();

Product p = inventory.get(id);

p.quantity = quantity;

p.price = price;

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

} else {

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

}

break;

case 3:

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

id = input.nextLine();

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

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

} else {

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

}

break;

case 4:

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

id = input.nextLine();

if (inventory.containsKey(id)) {

System.out.println(inventory.get(id));

} else {

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

}

break;

case 5:

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

System.out.println(prod);

}

break;

case 6:

System.out.println("Goodbye!");

input.close();

return;

default:

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

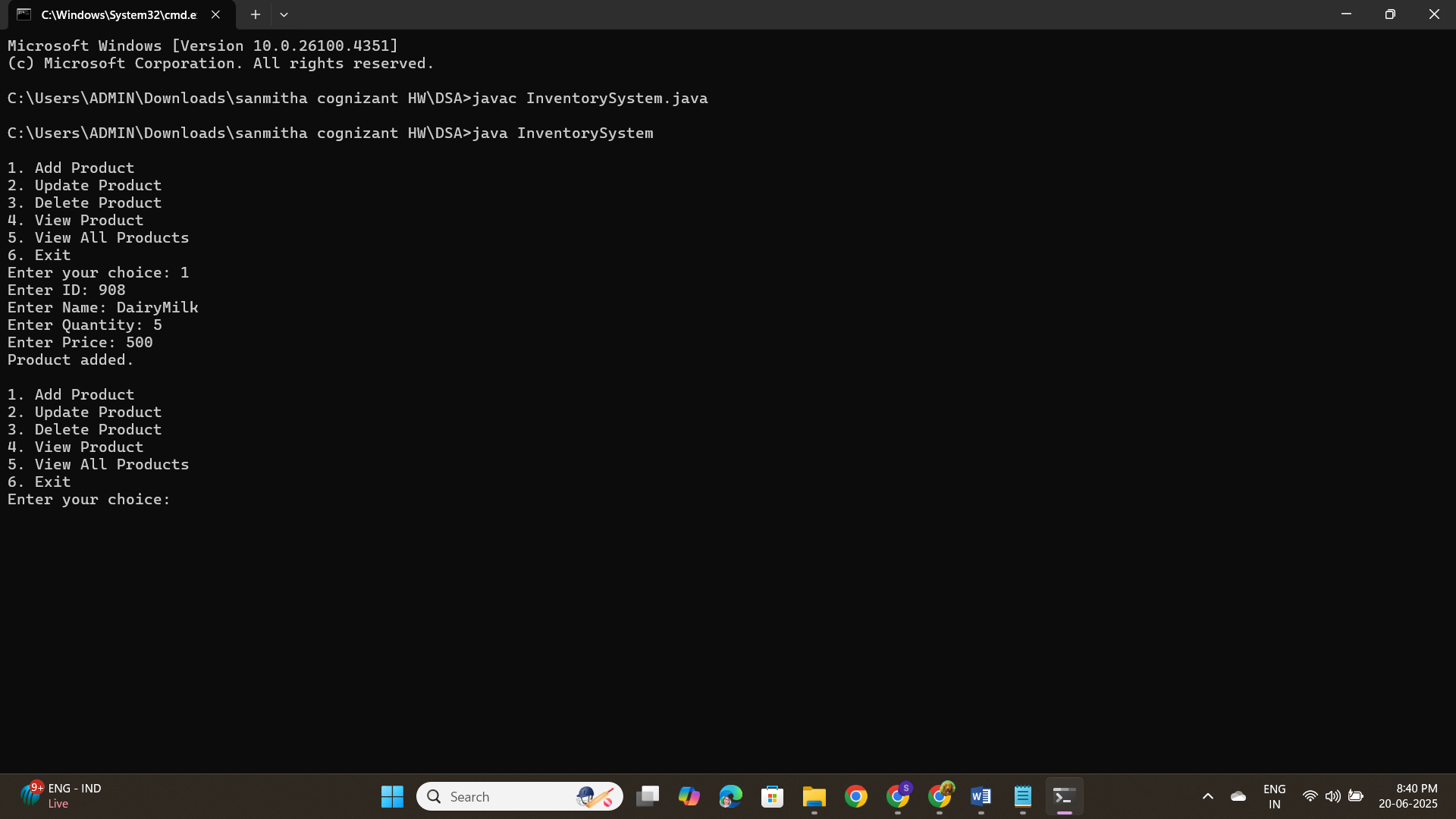
}

}

}

}

**Output:**

****

**Exercise-2**

**Code:**

import java.util.Arrays;

public class EcommerceSearchExample {

static class Product {

int productId;

String productName;

String category;

public Product(int id, String name, String category) {

this.productId = id;

this.productName = name;

this.category = category;

}

public String toString() {

return productId + " - " + productName + " (" + category + ")";

}

}

public static Product linearSearch(Product[] products, int targetId) {

for (Product product : products) {

if (product.productId == targetId) {

return product;

}

}

return null;

}

public static Product binarySearch(Product[] products, int targetId) {

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

while (left <= right) {

int mid = (left + right) / 2;

if (products[mid].productId == targetId)

return products[mid];

if (products[mid].productId < targetId)

left = mid + 1;

else

right = mid - 1;

}

return null;

}

public static void main(String[] args) {

Product[] productList = {

new Product(105, "Phone", "Electronics"),

new Product(101, "Shoes", "Fashion"),

new Product(104, "Laptop", "Electronics"),

new Product(102, "Watch", "Accessories"),

new Product(103, "Bag", "Fashion")

};

Product[] sortedList = productList.clone();

Arrays.sort(sortedList, (a, b) -> a.productId - b.productId);

int targetId = 104;

Product resultLinear = linearSearch(productList, targetId);

System.out.println("Linear Search Result: " + (resultLinear != null ? resultLinear : "Not Found"));

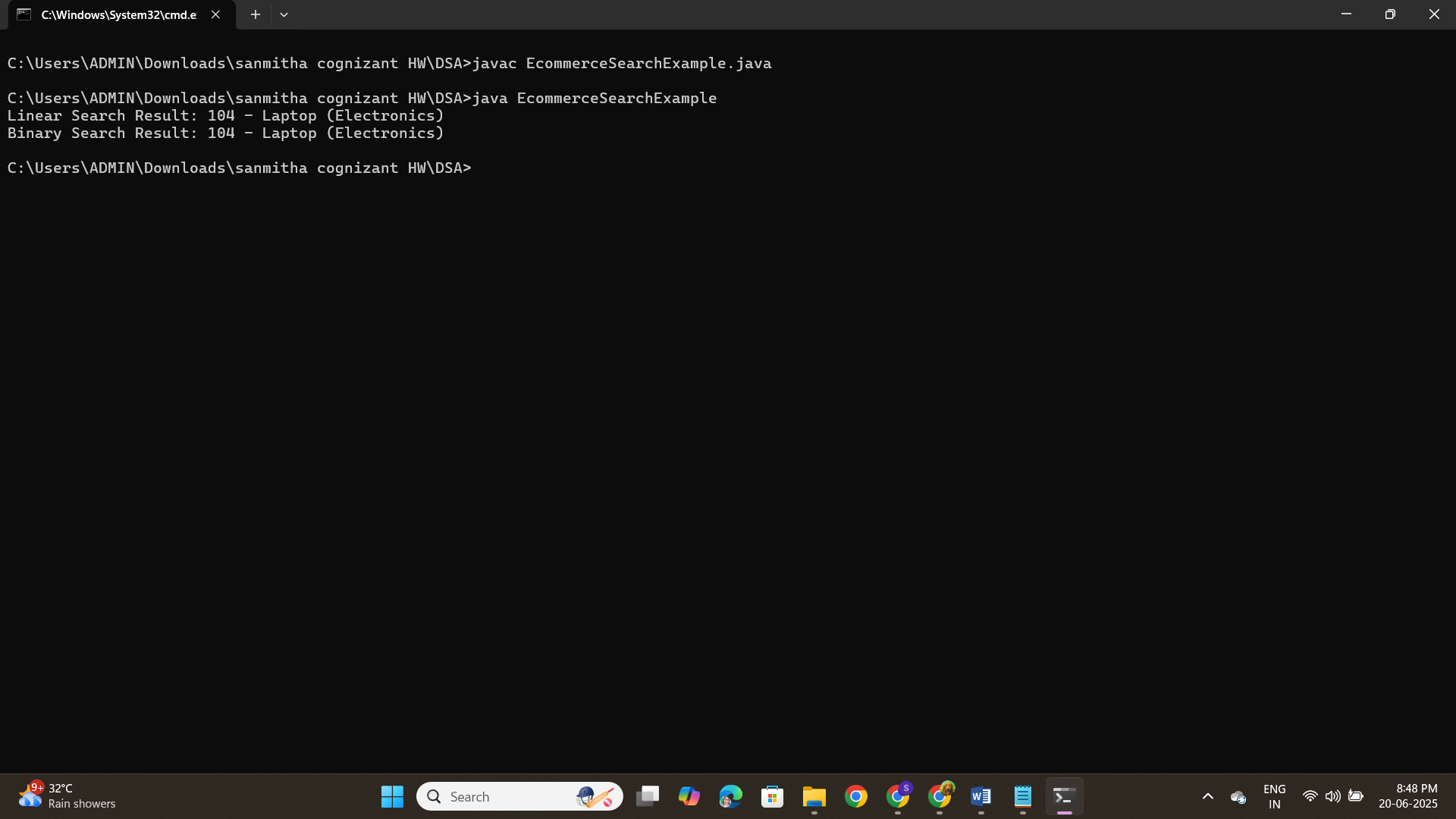
Product resultBinary = binarySearch(sortedList, targetId);

System.out.println("Binary Search Result: " + (resultBinary != null ? resultBinary : "Not Found"));

}

}

**Output:**

****

**Exercise-3**

**Code:**

public class CustomerOrderSorting {

static 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 orderId + " - " + customerName + " : ₹" + totalPrice;

}

}

public static void bubbleSort(Order[] orders) {

int n = orders.length;

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

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

}

}

private 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 printOrders(String label, Order[] orders) {

System.out.println(label);

for (Order order : orders) {

System.out.println(order);

}

}

public static void main(String[] args) {

Order[] orders1 = {

new Order(101, "Riya", 4500),

new Order(102, "Amit", 3000),

new Order(103, "Zara", 8000),

new Order(104, "Vikram", 2000),

new Order(105, "Nina", 7000)

};

Order[] orders2 = orders1.clone();

bubbleSort(orders1);

printOrders("Sorted by Bubble Sort:", orders1);

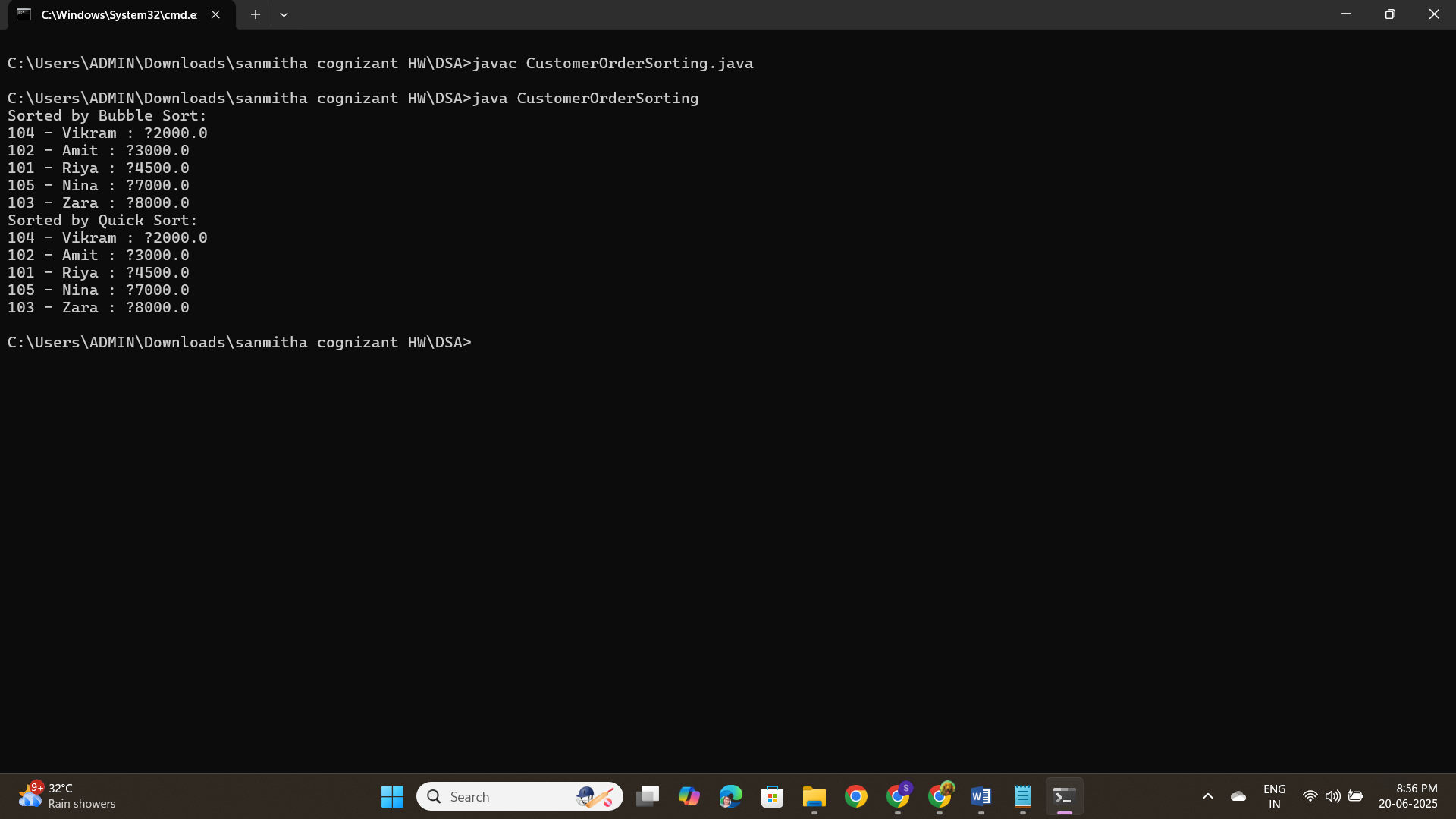
quickSort(orders2, 0, orders2.length - 1);

printOrders("Sorted by Quick Sort:", orders2);

}

}

**Output:**

****

**Exercise-4**

**Code:**

public class EmployeeManagementSystem {

static 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 String toString() {

return employeeId + " - " + name + " - " + position + " - ₹" + salary;

}

}

static class EmployeeManager {

Employee[] employees;

int size;

public EmployeeManager(int capacity) {

employees = new Employee[capacity];

size = 0;

}

public void addEmployee(Employee emp) {

if (size < employees.length) {

employees[size++] = emp;

} else {

System.out.println("Employee list full.");

}

}

public Employee searchEmployee(int id) {

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

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

return employees[i];

}

}

return null;

}

public void deleteEmployee(int id) {

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

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

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

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

}

employees[--size] = null;

return;

}

}

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

}

public void traverseEmployees() {

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

System.out.println(employees[i]);

}

}

}

public static void main(String[] args) {

EmployeeManager manager = new EmployeeManager(10);

manager.addEmployee(new Employee(101, "Anil", "Manager", 75000));

manager.addEmployee(new Employee(102, "Priya", "Developer", 55000));

manager.addEmployee(new Employee(103, "Ravi", "Tester", 40000));

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

manager.traverseEmployees();

System.out.println("\nSearch Employee with ID 102:");

Employee found = manager.searchEmployee(102);

System.out.println(found != null ? found : "Not Found");

System.out.println("\nDelete Employee with ID 101:");

manager.deleteEmployee(101);

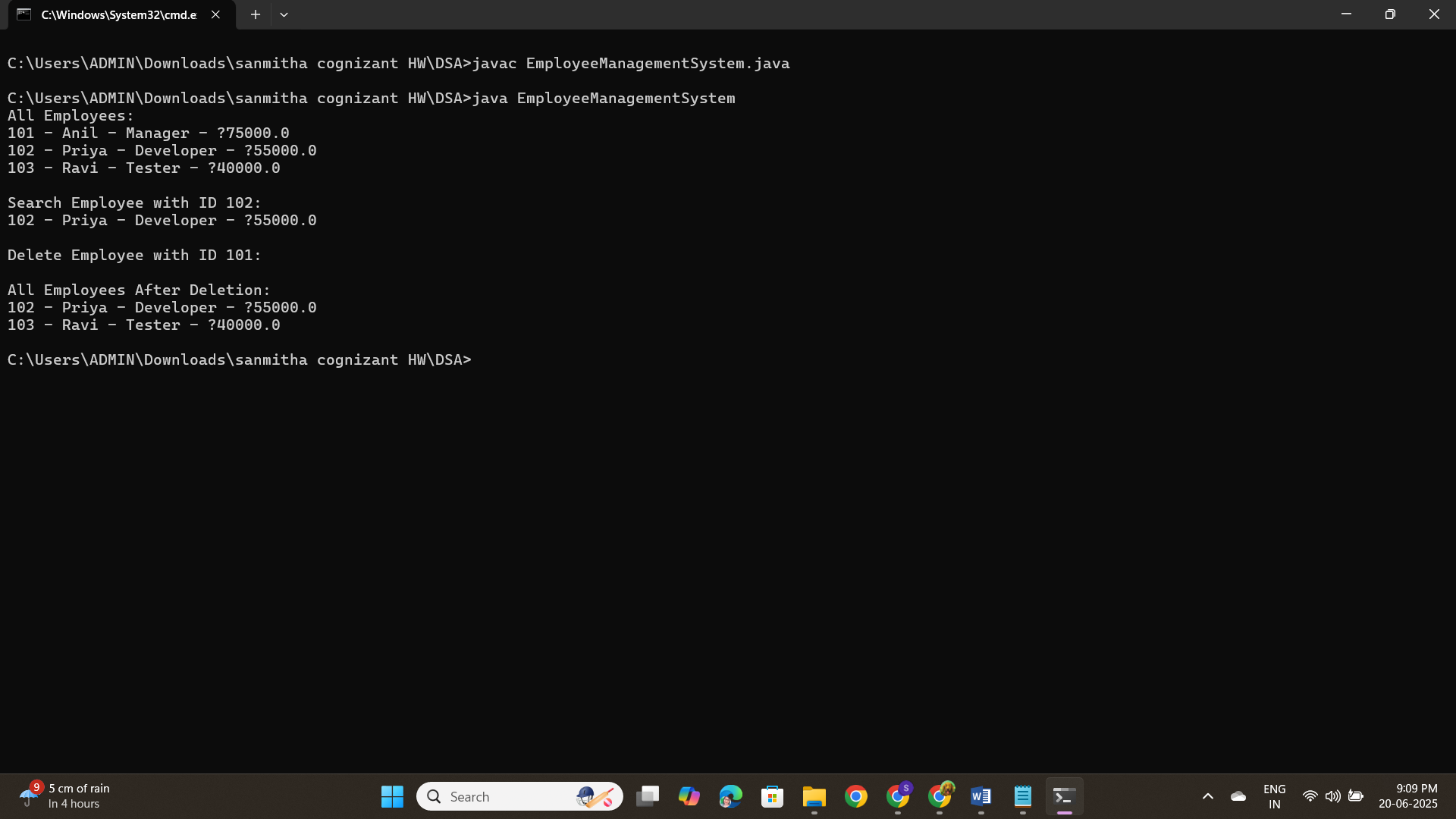
System.out.println("\nAll Employees After Deletion:");

manager.traverseEmployees();

}

}

**Output:**

****

**Exercise-5**

**Code:**

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;

}

}

class TaskManager {

private Task head;

public void addTask(int taskId, String taskName, String status) {

Task newTask = new Task(taskId, taskName, status);

if (head == null) {

head = newTask;

} else {

Task temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newTask;

}

}

public Task searchTask(int taskId) {

Task temp = head;

while (temp != null) {

if (temp.taskId == taskId) {

return temp;

}

temp = temp.next;

}

return null;

}

public void deleteTask(int taskId) {

if (head == null) return;

if (head.taskId == taskId) {

head = head.next;

return;

}

Task temp = head;

while (temp.next != null) {

if (temp.next.taskId == taskId) {

temp.next = temp.next.next;

return;

}

temp = temp.next;

}

}

public void traverseTasks() {

Task temp = head;

while (temp != null) {

System.out.println("Task ID: " + temp.taskId + ", Name: " + temp.taskName + ", Status: " + temp.status);

temp = temp.next;

}

}

}

public class Main {

public static void main(String[] args) {

TaskManager manager = new TaskManager();

manager.addTask(1, "Design UI", "Pending");

manager.addTask(2, "Develop Backend", "In Progress");

manager.addTask(3, "Testing", "Not Started");

manager.traverseTasks();

Task t = manager.searchTask(2);

if (t != null) {

System.out.println("Found Task: " + t.taskName);

}

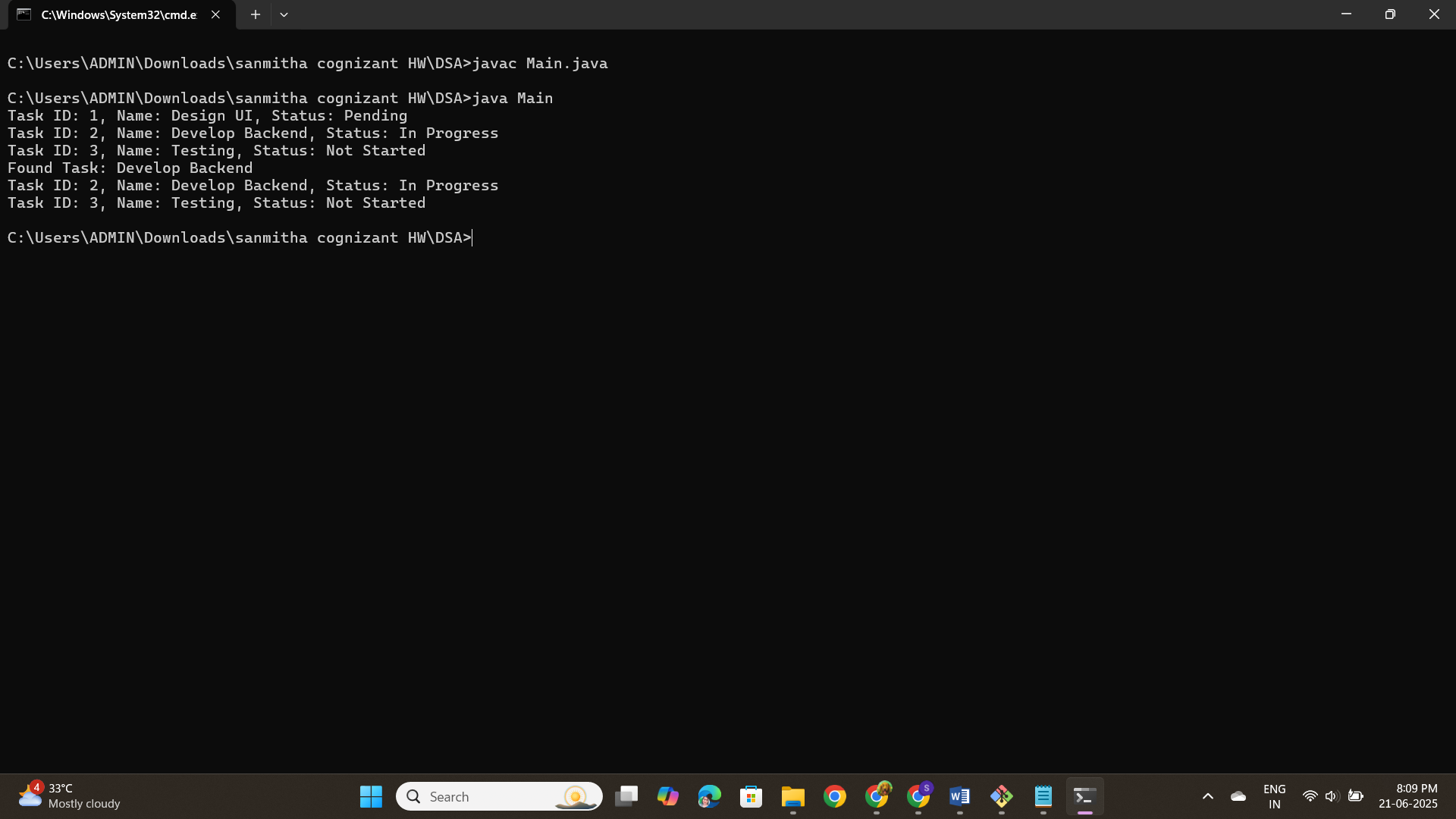
manager.deleteTask(1);

manager.traverseTasks();

}

}

**Output:**

****

**Exercise-6**

**Code:**

import java.util.Arrays;

import java.util.Comparator;

class LibraryBook {

int bookId;

String title;

String author;

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

this.bookId = bookId;

this.title = title;

this.author = author;

}

}

class LibrarySystem {

LibraryBook[] books;

int count;

public LibrarySystem(int size) {

books = new LibraryBook[size];

count = 0;

}

public void addBook(int bookId, String title, String author) {

if (count < books.length) {

books[count++] = new LibraryBook(bookId, title, author);

}

}

public LibraryBook linearSearchByTitle(String title) {

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

if (books[i].title.equalsIgnoreCase(title)) {

return books[i];

}

}

return null;

}

public LibraryBook binarySearchByTitle(String title) {

Arrays.sort(books, 0, count, Comparator.comparing(b -> b.title.toLowerCase()));

int left = 0, right = count - 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;

}

public void displayBooks() {

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

System.out.println("Book ID: " + books[i].bookId + ", Title: " + books[i].title + ", Author: " + books[i].author);

}

}

}

public class LibraryApp {

public static void main(String[] args) {

LibrarySystem system = new LibrarySystem(10);

system.addBook(101, "Java Programming", "James Gosling");

system.addBook(102, "Python Basics", "Guido van Rossum");

system.addBook(103, "C Programming", "Dennis Ritchie");

system.addBook(104, "Data Structures", "Mark Allen Weiss");

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

system.displayBooks();

LibraryBook foundLinear = system.linearSearchByTitle("Python Basics");

if (foundLinear != null)

System.out.println("Found by Linear Search: " + foundLinear.title + " by " + foundLinear.author);

LibraryBook foundBinary = system.binarySearchByTitle("Python Basics");

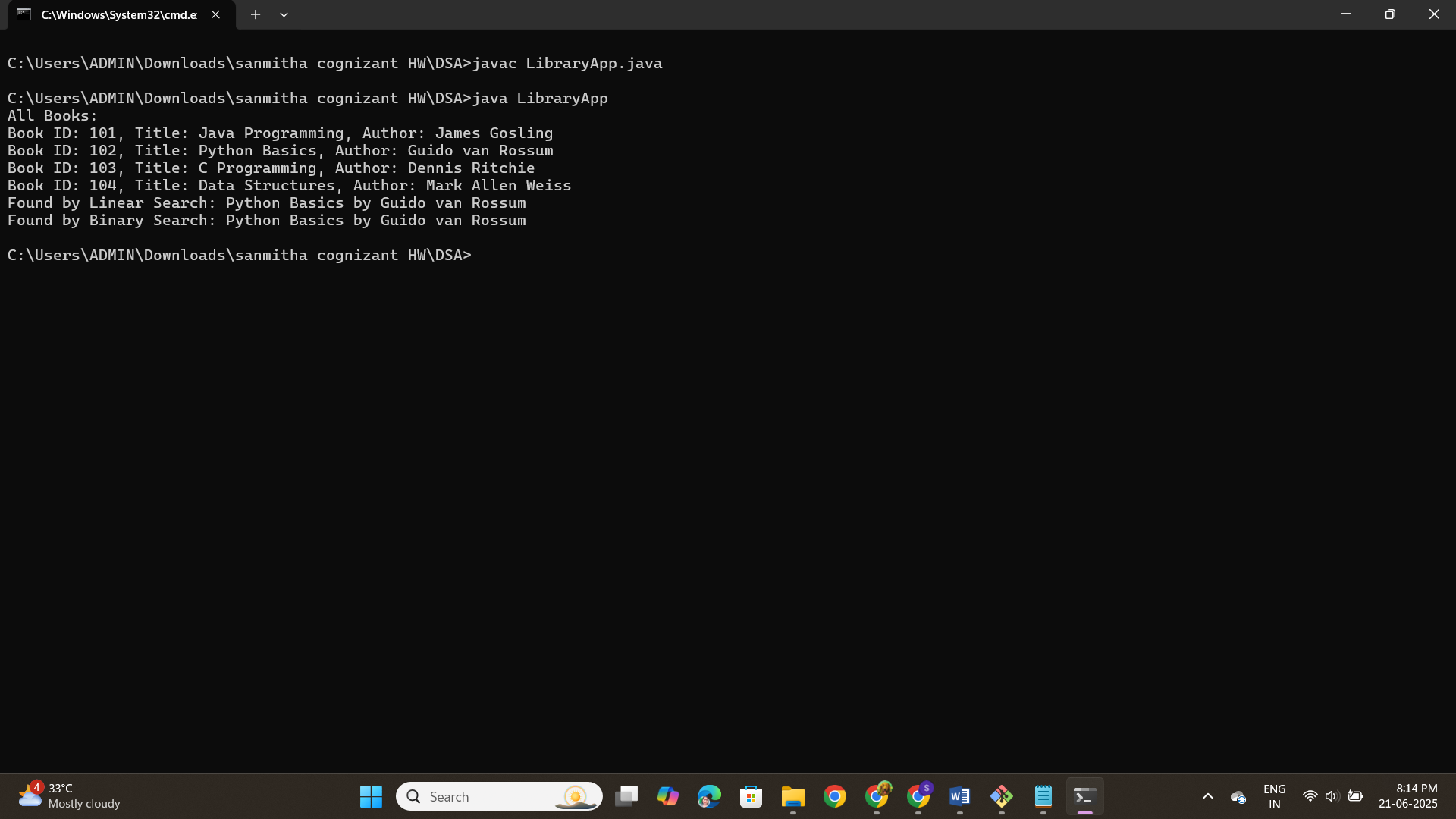
if (foundBinary != null)

System.out.println("Found by Binary Search: " + foundBinary.title + " by " + foundBinary.author);

}

}

**Output:**

****

**Exercise-7**

**Code:**

class FinancialForecast {

public double calculateFutureValue(double presentValue, double growthRate, int years) {

if (years == 0) return presentValue;

return calculateFutureValue(presentValue \* (1 + growthRate), growthRate, years - 1);

}

public double calculateFutureValueIterative(double presentValue, double growthRate, int years) {

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

presentValue \*= (1 + growthRate);

}

return presentValue;

}

}

public class ForecastApp {

public static void main(String[] args) {

FinancialForecast forecast = new FinancialForecast();

double presentValue = 10000;

double growthRate = 0.05;

int years = 5;

double futureValueRecursive = forecast.calculateFutureValue(presentValue, growthRate, years);

System.out.println("Future Value (Recursive) after " + years + " years: " + futureValueRecursive);

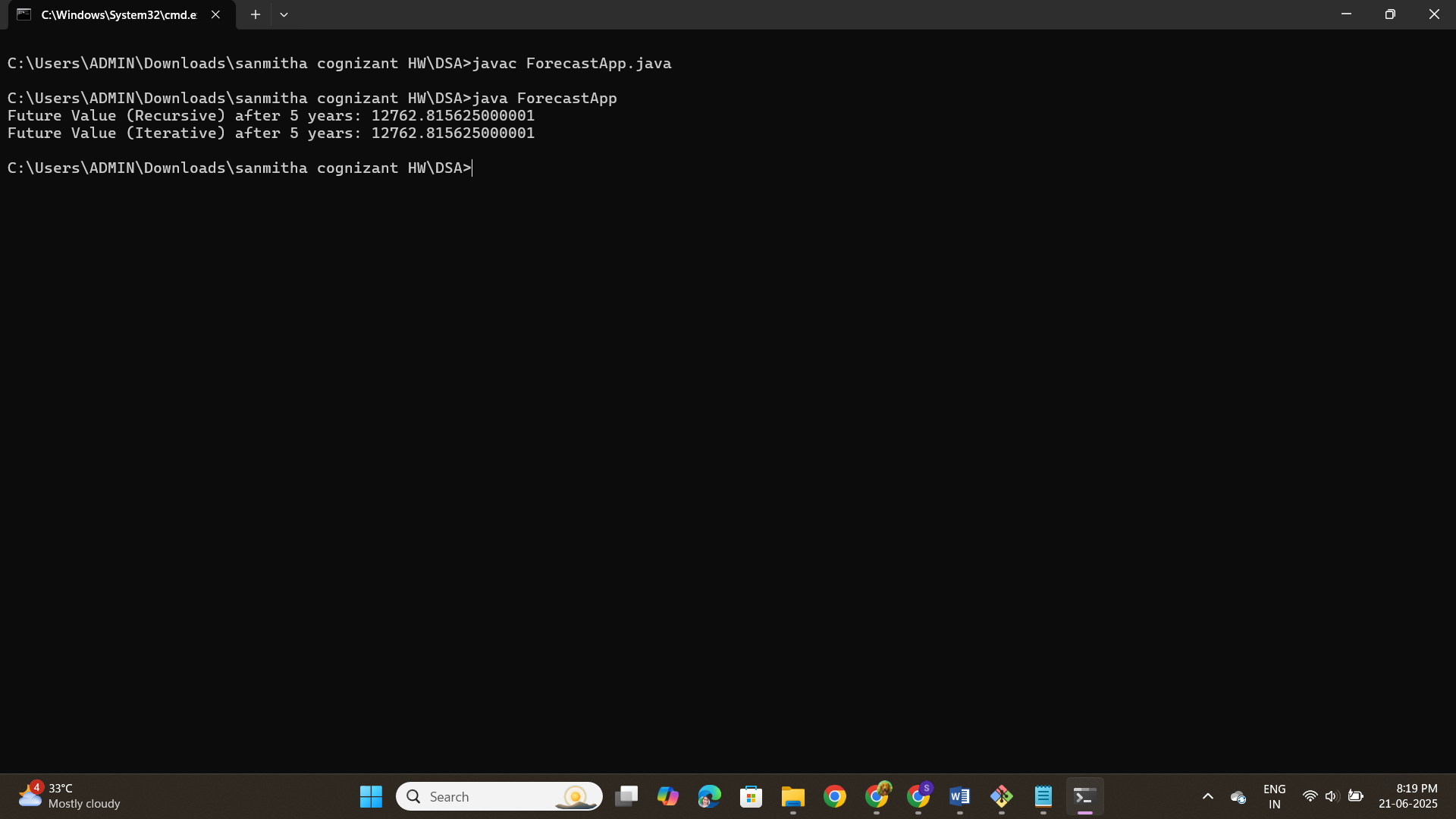
double futureValueIterative = forecast.calculateFutureValueIterative(presentValue, growthRate, years);

System.out.println("Future Value (Iterative) after " + years + " years: " + futureValueIterative);

}

}

**Output:**

****