**Algorithms\_Data Structures  
  
Exercise 2: E-commerce Platform Search Function**

CODE:

import java.util.\*;

   class Product {

     int proId;

     String proName;

     String proCat;

     Product( int proId,String proName, String proCat)

    {

      this.proName=proName;

      this.proId=proId;

      this.proCat=proCat;

    }

    void display()

    {

        System.out.println("ID:"+proId+" Name:"+proName+" Category:"+proCat);

    }

}

class LinearSearch

{

    static Product line(Product[] pro,String tar)

    {

        for(Product p:pro)

        {

            if(p.proName.equalsIgnoreCase(tar))

            {

                return p;

            }

        }

        return null;

    }

}

class BinarySearch

{

    static Product bisear(Product[] p,String tar)

    {

        int l=0;

        int r=p.length-1;

        while(l<=r)

        {

            int mid=(l+r)/2;

            int comp=p[mid].proName.compareToIgnoreCase(tar);

            if(comp==0)

            {

                return p[mid];

            }

            else if(comp<0)

            {

                l=mid+1;

            }

            else

            {

                r=mid-1;

            }

        }

       return null;

    }

   static void sortByProName(Product[] p)

   {

    Arrays.sort(p,Comparator.comparing(P->P.proName.toLowerCase()));

   }

}

class Main{

    public static void main(String[] args)

    {

       Product[] p={

        new Product(1,"Lap","elect"),

        new Product(2,"sha","health"),

        new Product(3,"shoe","foot"),

        new Product(4,"book","stat"),

        new Product(5,"tab","elect"),

       };

       Product f1=LinearSearch.line(p,"shoe");

       System.out.println("Linear search");

       if(f1!=null)

       {

        f1.display();

       }

        else

        {

           System.out.println("Product");

        }

        System.out.println("Binary Search");

BinarySearch.sortByProName(p);

        Product f2=BinarySearch.bisear(p,"lap");

        if(f2!=null)

        {

            f2.display();

        }

        else{

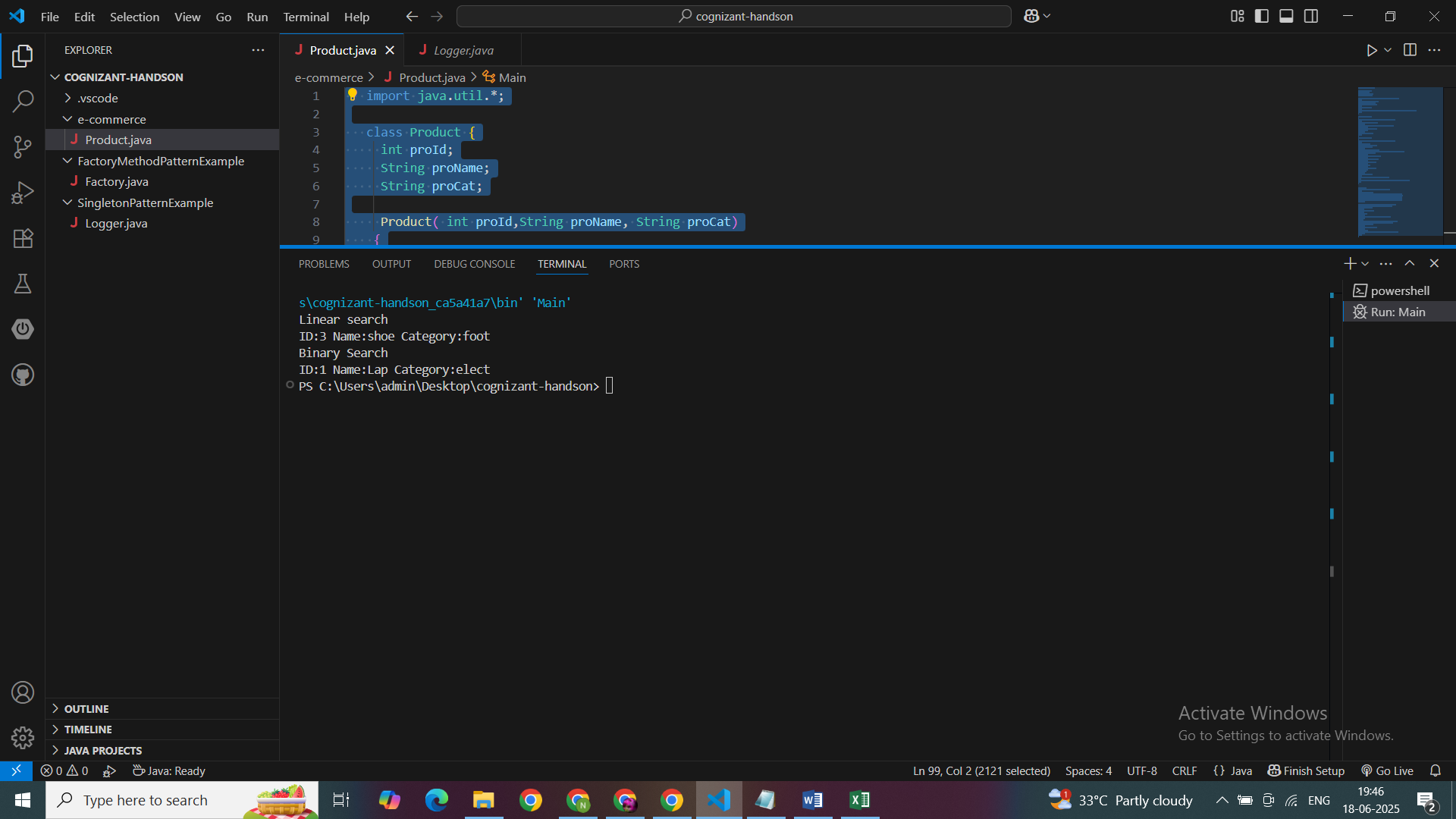
            System.out.println("Poduct not found");

        }

    }

}

**OUTPUT:**

****

**Time Complexity**

| **Algorithm** | **Time Complexity (Worst Case)** |
| --- | --- |
| Linear Search | - O(n) |
| Binary Search | - O(log n) |

**Linear Search**:

✅ Works on **unsorted data**  
❌ Slower for large datasets

**Binary Search**:  
✅ Much **faster** (log n), but  
❌ Requires the data to be **sorted**

**Efficiency:** Binary Search is ideal for performance — if data is sorted

 We use Linear Search for simplicity when data is small or unsorted

**Exercise 7: Financial Forecasting**

package FinancialForecasting;

import java.util.\*;

public class Finance {

    public static void main(String[] args)

    {

        Scanner sc=new Scanner(System.in);

        double present=sc.nextDouble();

        double rate=sc.nextDouble();

        int years=sc.nextInt();

        double futureValue=future(present,rate,years);

        System.out.printf("%.2f",futureValue);

    }

    public static double future(double p,double r,double y)

    {

        if(y==0)

        {

            return p;

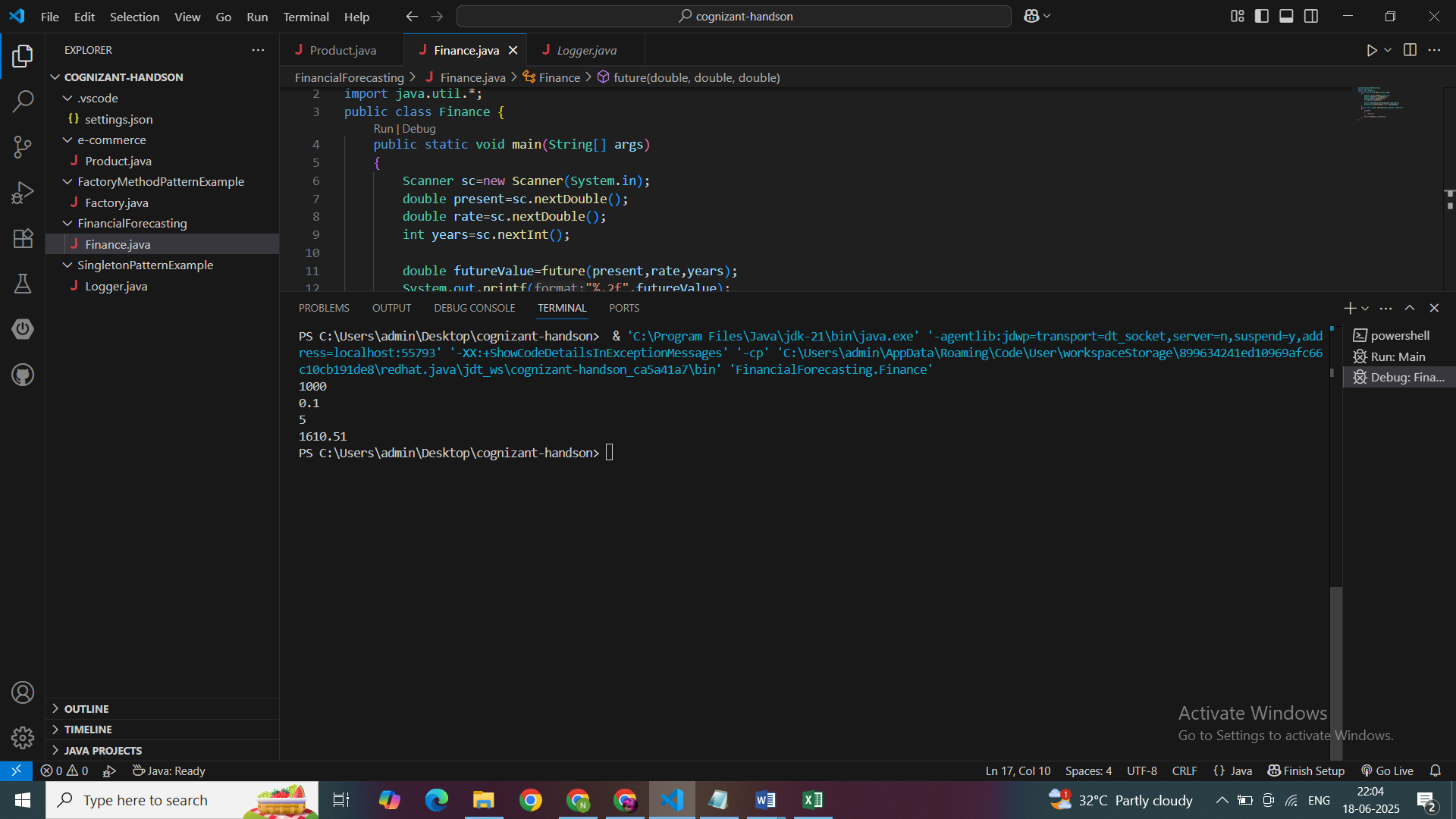
        }

        return future(p,r,y-1)\*(1+r);

    }

}

**OUTPUT:**

****

#### 🔸 Time Complexity:

Each recursive call reduces years by 1.

So:

* **Time Complexity = O(n)**
* **Space Complexity = O(n)** (due to call stack)

#### 🔸 Problem: Recursive overhead

For large years, too many recursive calls can lead to:

* Stack overflow
* Slow performance

**Optimization Techniques**

**1.Iteration  
2.Math formula  
  
1.Iteration**

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

double result = presentValue;

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

result \*= (1 + rate);

}

return result;

}  
  
 **Time Complexity**: O(n)

 **Space Complexity**: O(1)

**2.Math Formula(Exponentation)**static double futureValueFormula(double presentValue, double rate, int years) {

return presentValue \* Math.pow(1 + rate, years);

}  
  
**Time Complexity**: O(1)

**Space Complexity:** O(1)

**Exercise 1: Inventory Management System**

package InventoryManagement;

import java.util.\*;

public class Invent {

    public static void main(String[] args) {

        InventoryManager manager = new InventoryManager();

        Product p1 = new Product(101, "Laptop", 10, 75000.00);

        Product p2 = new Product(102, "Mouse", 50, 500.00);

        Product p3 = new Product(103, "Keyboard", 30, 1500.00);

        manager.addProduct(p1);

        manager.addProduct(p2);

        manager.addProduct(p3);

        manager.displayInventory();

        System.out.println("\nUpdating product 102...");

        manager.updateProduct(102, "Wireless Mouse", 60, 650.00);

        System.out.println("\nDeleting product 101...");

        manager.deleteProduct(101);

        System.out.println("\nFinal Inventory:");

        manager.displayInventory();

    }

}

// File: Product.java

class Product {

    int productId;

    String productName;

    int quantity;

    double price;

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

        this.productId = productId;

        this.productName = productName;

        this.quantity = quantity;

        this.price = price;

    }

    void display() {

        System.out.println("ID: " + productId + ", Name: " + productName +

                ", Quantity: " + quantity + ", Price: Rs." + price);

    }

}

class InventoryManager {

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

    void addProduct(Product p) {

        if (inventory.containsKey(p.productId)) {

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

        } else {

            inventory.put(p.productId, p);

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

        }

    }

    void updateProduct(int id, String name, int quantity, double price) {

        if (inventory.containsKey(id)) {

            Product p = inventory.get(id);

            p.productName = name;

            p.quantity = quantity;

            p.price = price;

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

        } else {

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

        }

    }

    void deleteProduct(int id) {

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

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

        } else {

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

        }

    }

    void displayInventory() {

        if (inventory.isEmpty()) {

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

        } else {

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

                p.display();

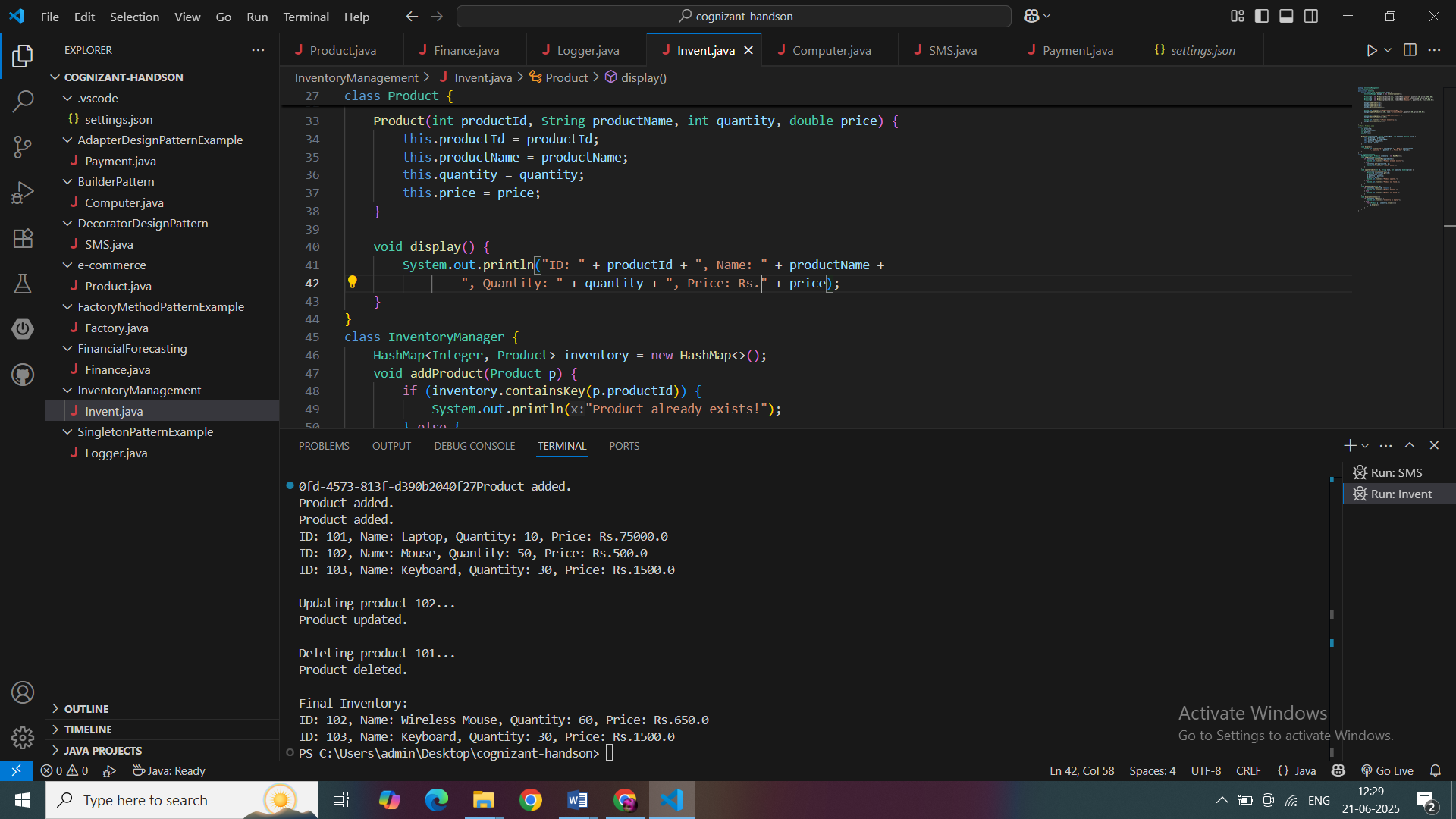
            }

        }

    }

}

OUTPUT:



**Exercise 4: Employee Management System**

package EmployeeManagement;

class Employee{

     int id;

     String name;

     int salary;

     String position;

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

    {

        this.id=id;

        this.name=name;

        this.salary=salary;

        this.position=position;

    }

    public void display()

    {

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

    }

}

class EmployeeManager

{

    int max\_size=100;

    Employee[] employee=new Employee[max\_size];

    int c=0;

    void add(Employee e)

    {

     if(c<max\_size)

     {

        employee[c]=e;

        c++;

        System.out.println("Employee is added");

     }

     else{

        System.out.println("Array is full");

     }

    }

    void search(int id1)

    {

        for(int i=0;i<c;i++)

        {

            if(employee[i].id==id1)

            {

               employee[i].display();

               return;

            }

            else{

            }

        }

    }

    void traverse()

    {

        if(c==0)

        {

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

            return;

        }

        else{

           for(int i=0;i<c;i++)

           {

              employee[i].display();

           }

        }

    }

    void delete(int id)

    {

        for(int i=0;i<c;i++)

        {

            if(employee[i].id==id)

            {

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

                {

                    employee[j]=employee[j+1];

                }

                employee[c-1]=null;

                c--;

                System.out.println("Deleted");

                return;

            }

            else{

                System.out.println(("Not found"));

            }

        }

    }

}

public class Manage {

    public static void main(String[] args)

    {

    EmployeeManager em=new EmployeeManager();

    Employee e1=new Employee(101,"Tom",45000,"SoftwareEngineer");

    Employee e2=new Employee(102,"Tim",55000,"SoftwareDesigner");

    Employee e3=new Employee(103,"Tem",65000,"SoftwareManager");

    em.add(e1);

    em.add(e2);

    em.add(e3);

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

        em.traverse();

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

        em.search(102);

        System.out.println("\nDeleting Employee ID 1:");

        em.delete(101);

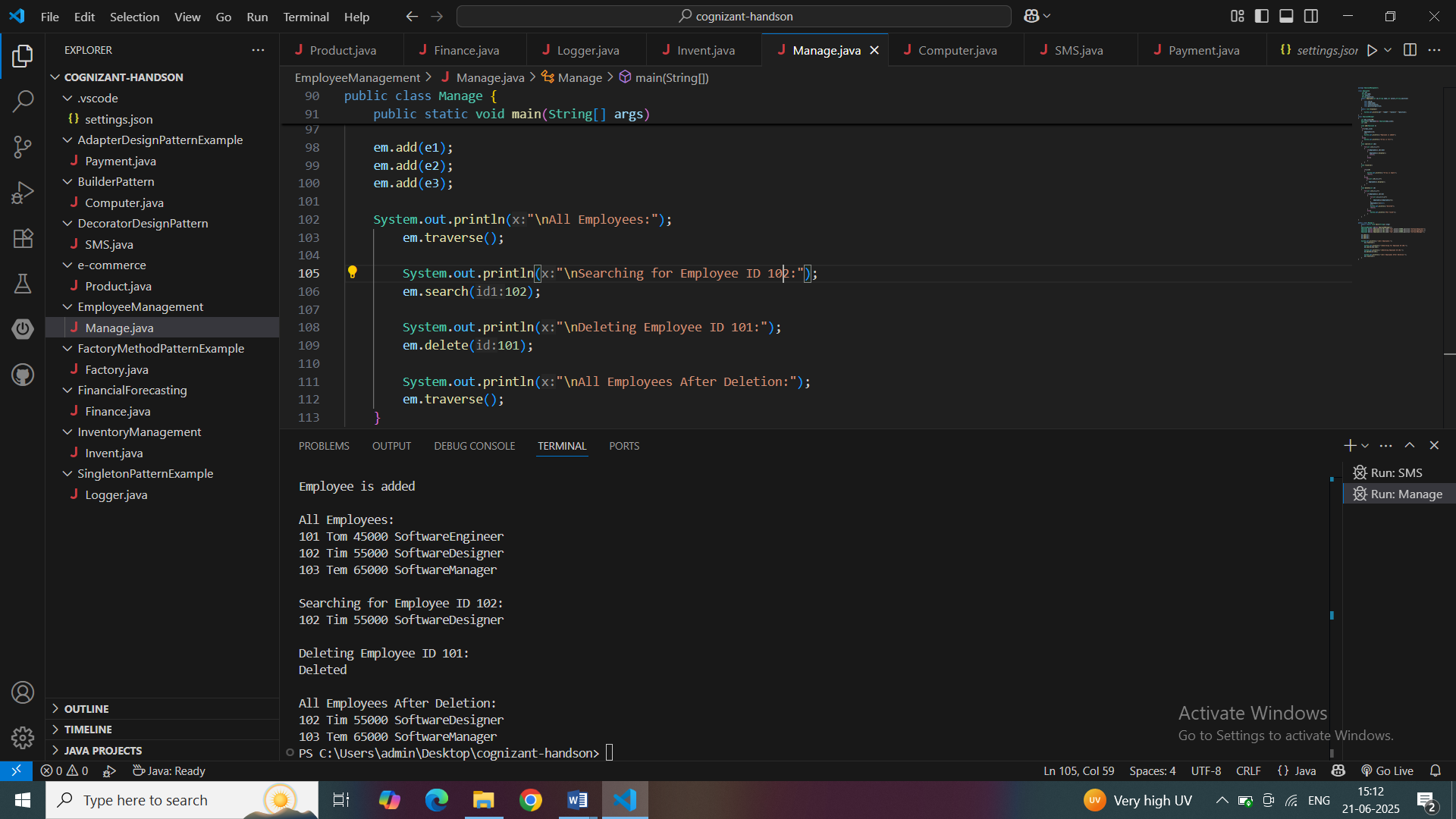
        System.out.println("\nAll Employees After Deletion:");

        em.traverse();

    }

}

OUTPUT:



**Exercise 5: Task Management System**

package TaskManagementSystem;

class Test{

   int id;

   String name;

   String status;

   Test(int id,String name,String status)

   {

    this.id=id;

    this.name=name;

    this.status=status;

   }

   public void display()

   {

    System.out.println(id+" "+name+" "+status);

   }

}

class Node

{

   Test task;

   Node next;

   Node(Test task)

   {

      this.task=task;

      this.next=null;

   }

}

class TaskManager

{

  Node head=null;

  void add(Test task)

  {

   Node newNode=new Node(task);

   if(head==null)

   {

      head=newNode;

   }

   else{

      Node temp=head;

      while(temp.next!=null)

      {

         temp=temp.next;

      }

      temp.next=newNode;

   }

   System.out.println("Task Added");

  }

  void search(int id)

  {

   Node temp=head;

   while(temp!=null)

   {

      if(temp.task.id==id)

      {

         temp.task.display();

         return;

      }

      temp=temp.next;

   }

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

  }

  void traverse()

  {

   if(head==null)

   {

      System.out.println("No task in list");

      return;

   }

   Node temp=head;

   while(temp!=null)

   {

      temp.task.display();

      temp=temp.next;

   }

  }

  void delete(int id)

  {

   if(head==null)

   {

      System.out.println("Empty");

      return;

   }

   if(head.task.id==id)

   {

      head=head.next;

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

      return;

   }

   Node prev=head;

   Node curr=head.next;

   while(curr!=null)

   {

      if(curr.task.id==id)

      {

         prev.next=curr.next;

         System.out.println("Task Deleted");

         return;

      }

      prev=curr;

      curr=curr.next;

   }

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

  }

}

public class Task {

   public static void main(String[] args)

   {

    TaskManager tm=new TaskManager();

    Test t1=new Test(101,"Tom","In progress");

    Test t2=new Test(102,"Tim","Completed");

    Test t3=new Test(103,"Tem","Not started");

    tm.add(t1);

    tm.add(t2);

    tm.add(t3);

    System.out.println("\nAll Tasks:");

        tm.traverse();

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

        tm.search(102);

        System.out.println("\nDeleting Task ID 101:");

        tm.delete(101);

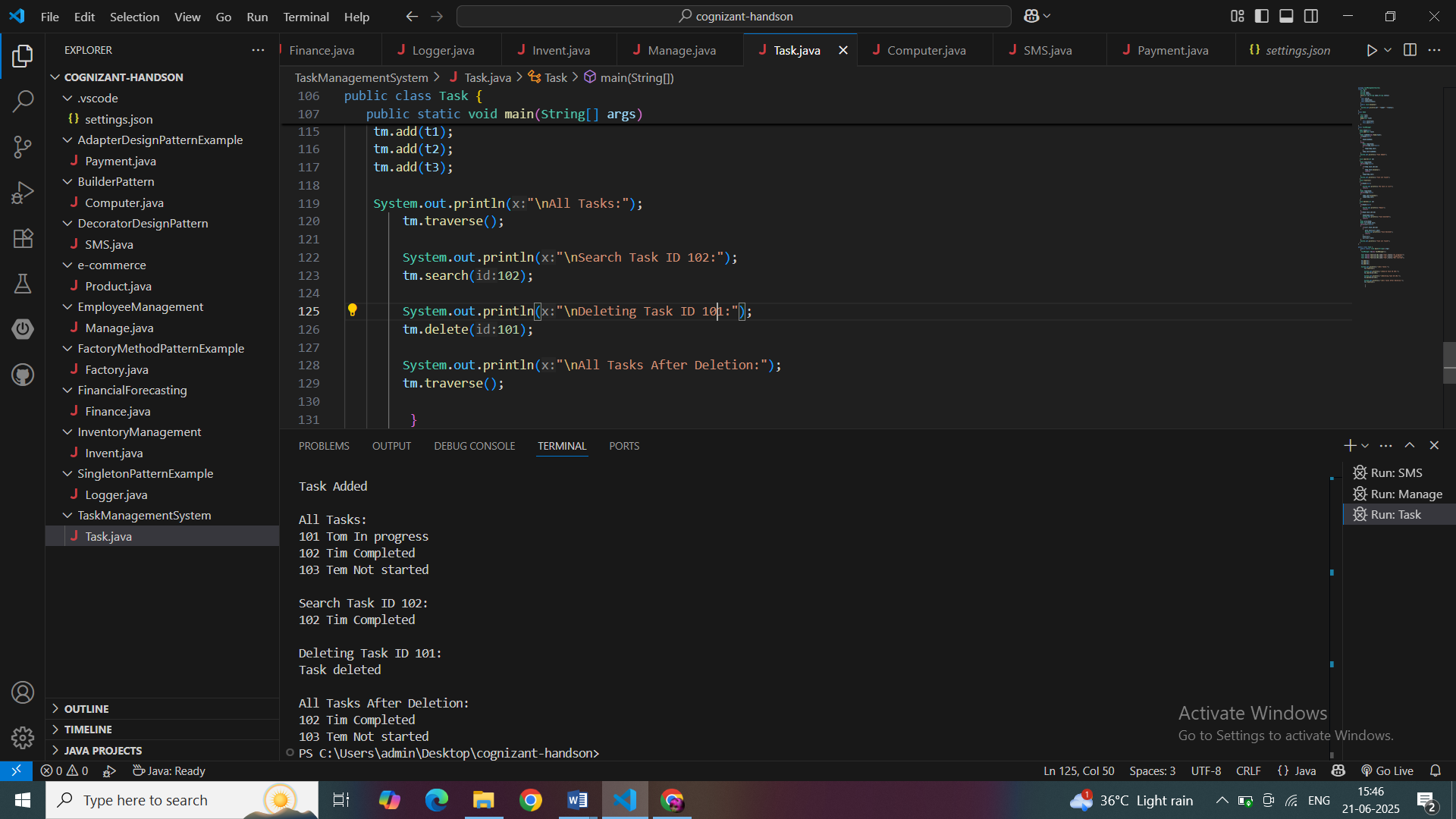
        System.out.println("\nAll Tasks After Deletion:");

        tm.traverse();

         }

         }

OUTPUT:



**Exercise 6: Library Management System**

package LibraryManagementSystem;

import java.util.Arrays;

import java.util.Comparator;

class Library{

   int id;

   String title;

   String author;

   Library(int id,String title,String author)

   {

    this.id=id;

    this.title=title;

    this.author=author;

   }

   public void display()

   {

    System.out.println(id+" "+title+" "+author);

   }

}

class LinearSearch

{

  static Library line(Library[] l,String target)

  {

    for(Library y:l)

    {

        if(y.title.equalsIgnoreCase(target))

        {

            return y;

        }

    }

    return null;

  }

}

class BinarySearch

{

    static Library bisea(Library[] l,String target)

    {

        int h=0;

        int r=l.length-1;

        while(h<=r)

        {

            int mid=(h+r)/2;

            int ry=l[mid].title.compareToIgnoreCase(target);

            if (ry==0) {

                return l[mid];

            }

            else if(ry<0)

            {

                h=mid+1;

            }

            else

            {

              r=mid+1;

            }

        }

        return null;

    }

    static void sortByTitle(Library[] l)

    {

        Arrays.sort(l,Comparator.comparing(L->L.title.toLowerCase()));

    }

}

public class Book {

    public static void main(String[] args)

    {

        Library[] l={

            new Library(1,"To Kill a Mockingbird" ,"Harper Lee"),

            new Library(2,"1984", "George Orwell"),

            new Library(3,"Pride and Prejudice","Jane Austen"),

            new Library(4,"The Alchemist","Paulo Coelho"),

            new Library(5,"The Great Gatsby","F. Scott Fitzgerald")

        };

        Library l1=LinearSearch.line(l,"1984");

        System.out.println("Linear Search");

        if(l1!=null)

        {

            l1.display();

        }

        else{

            System.out.println("Not found");

        }

        System.out.println("Binary Search");

        BinarySearch.sortByTitle(l);

        Library l2=BinarySearch.bisea(l,"The Great Gatsby");

        if(l2!=null)

        {

            l2.display();

        }

        else{

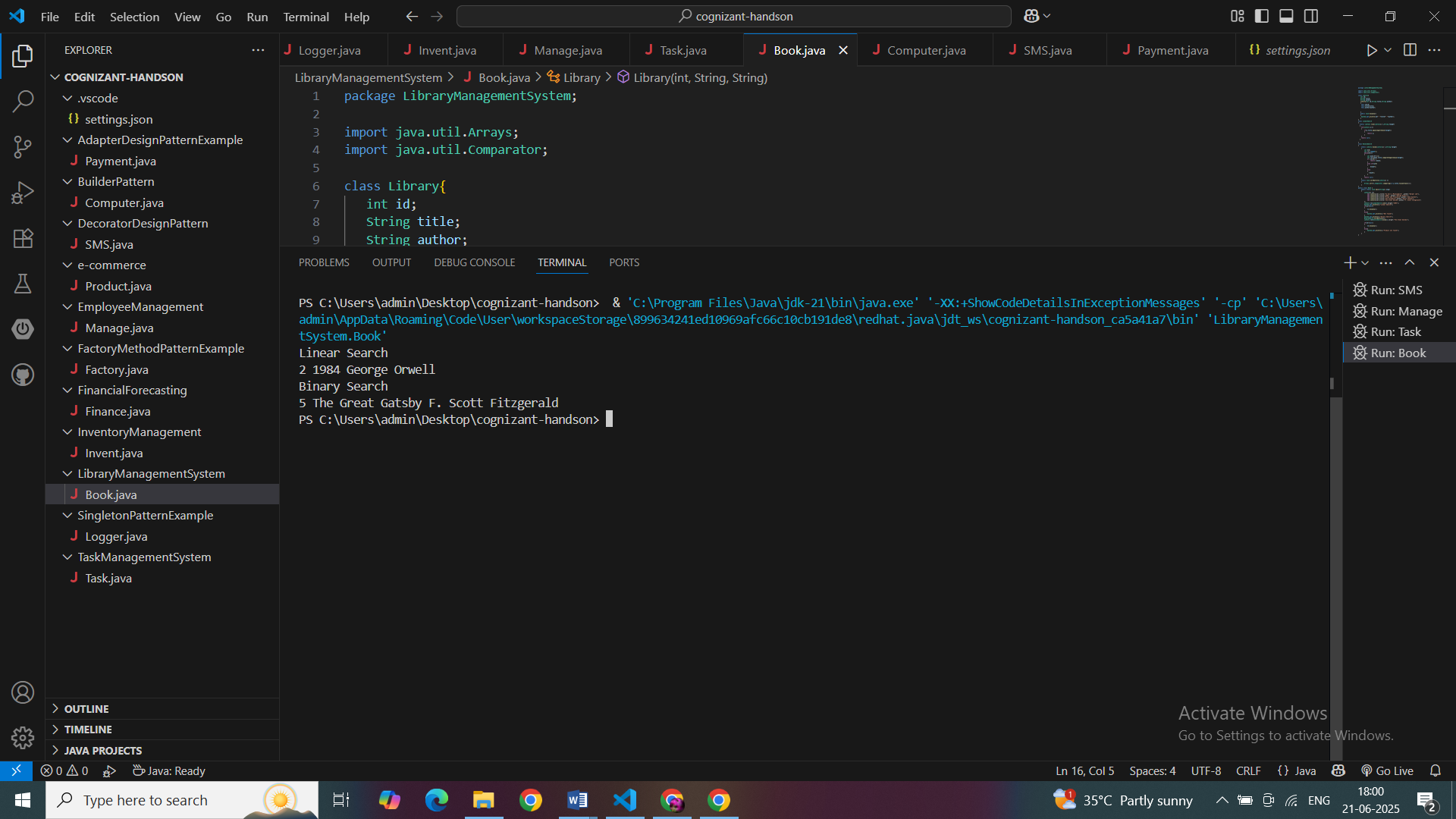
            System.out.println("Product not found");

        }

    }

}

**OUTPUT:**

****

**Exercise 3: Sorting Customer Orders**

package SortingCustomer;

class Order {

    int orderId;

    String customerName;

    double totalPrice;

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

        this.orderId = orderId;

        this.customerName = customerName;

        this.totalPrice = totalPrice;

    }

    void display() {

        System.out.println("ID: " + orderId + ", Name: " + customerName + ", Total Price: Rs." + totalPrice);

    }

}

class BubbleSort {

    static void sort(Order[] orders) {

        int n = orders.length;

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

            boolean swapped = false;

            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;

                    swapped = true;

                }

            }

            if (!swapped) break;        }

    }

}

class QuickSort {

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

        if (low < high) {

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

            sort(orders, low, pi - 1);

            sort(orders, pi + 1, high);

        }

    }

    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 class Sorting {

     public static void main(String[] args) {

        Order[] orders = {

            new Order(101, "Alice", 1999.99),

            new Order(102, "Bob", 499.50),

            new Order(103, "Charlie", 850.75),

            new Order(104, "Diana", 2500.00),

            new Order(105, "Eve", 750.25)

        };

        System.out.println("=== Bubble Sort ===");

        BubbleSort.sort(orders);

        for (Order o : orders) o.display();

        orders = new Order[]{

            new Order(101, "Alice", 1999.99),

            new Order(102, "Bob", 499.50),

            new Order(103, "Charlie", 850.75),

            new Order(104, "Diana", 2500.00),

            new Order(105, "Eve", 750.25)

        };

        System.out.println("\n=== Quick Sort ===");

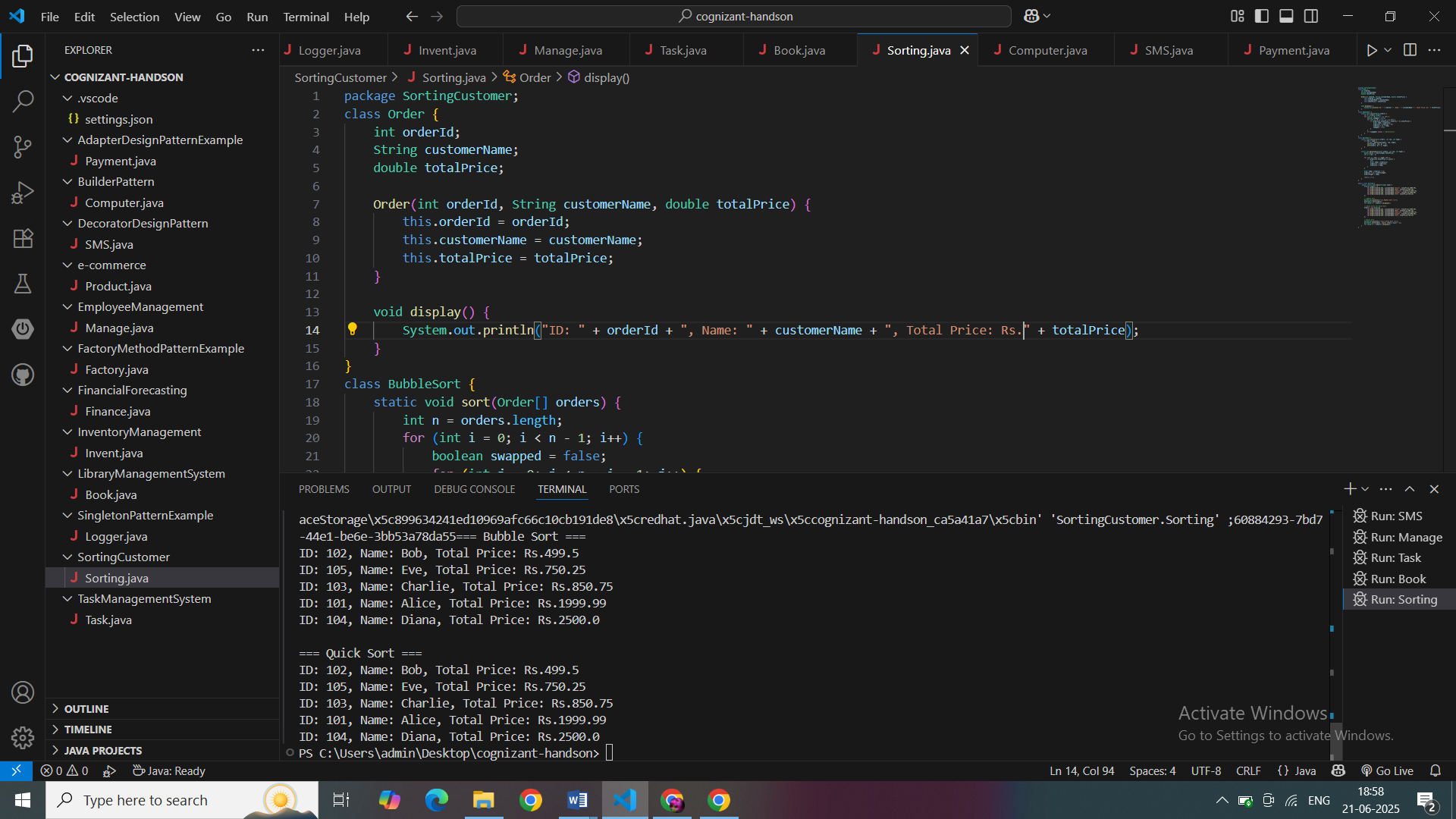
        QuickSort.sort(orders, 0, orders.length - 1);

        for (Order o : orders) o.display();

    }

}

**OUTPUT:**

****