**Data Structures and Algorithms**

**Exercise 1: Inventory Management System**

**Code:**

import java.util.\*;

class Product{

    int productId,quantity;

    double price;

    String productName;

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

        this.productId=productId;

        this.quantity=quantity;

        this.price=price;

        this.productName=productName;

    }

    int getProductId(){ return productId; }

    int getQuantity(){ return quantity;}

    double getPrice(){  return price; }

    String productName(){ return productName; }

    void setProductId(int id){

        productId=id;

    }

    void setQuantity(int quan){

        quantity=quan;

    }

    void setPrice(double p){

        price=p;

    }

    void setProductName(String name){

        productName=name;

    }

   public String toString(){

        return "productid: "+ productId +" , quantity: "+ quantity +" , productName: "+productName+" , price:"+ price;

    }

}

class Manage{

    Map<Integer,Product> mp= new HashMap<>();

    // Method to add

    void add(Product prod){

        mp.put(prod.getProductId(),prod);

        System.out.println("product included successfully "+ prod);

    }

    // Method to update

    void update(int productId,int quantity,double price,String name){

        Product product=mp.get(productId);

        if(product !=null){

            product.setProductName(name);

            product.setQuantity(quantity);

            product.setPrice(price);

            System.out.println("product updated successfully "+ product);

        }else System.out.println("product not found");

    }

    //Method to delete

    void delete(int id){

      if(mp.containsKey(id)){

        Product p=mp.remove(id);

        System.out.println("product removed successfully "+p);

      }else System.out.println("product not found");

    }

}

public class Inventory {

    public static void main(String[] args) {

        Product p1= new Product(1, 3, 120000, "samsung galaxy");

        Product p2= new Product(2, 10, 90000, "iphone");

        Manage ob= new Manage();

        ob.add(p1);

        ob.add(p2);

        ob.update(2,57,150000,"iphone pro");

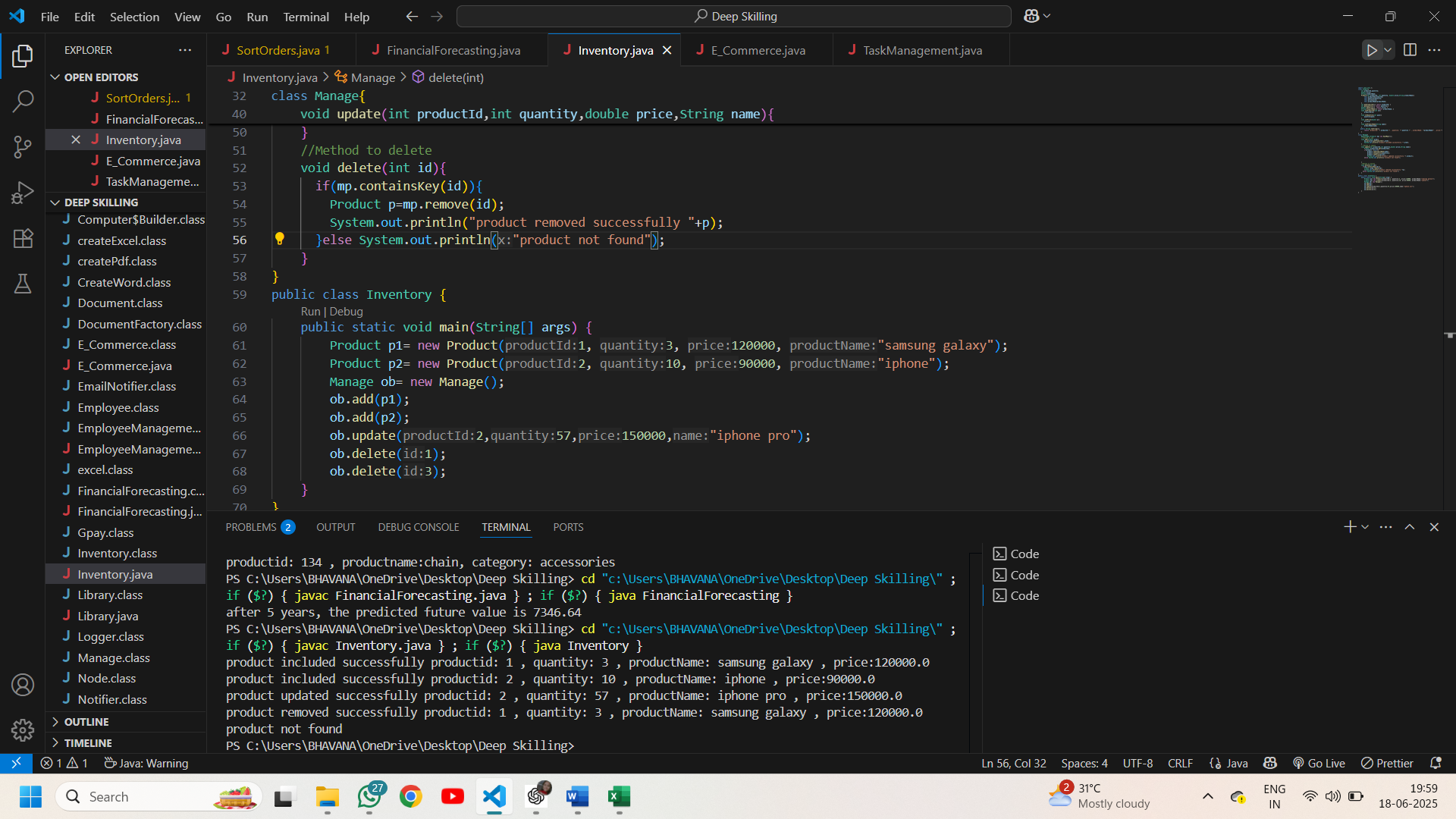
        ob.delete(1);

        ob.delete(3);

    }

}

Output:



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

**Code:**

import java.util.\*;

class Product{

    int productId;

    String productName,category;

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

         this.productId= productId;

         this.productName= productName;

         this.category= category;

    }

    String getproductName(){ return productName; }

    int getproductId(){ return productId; }

    public String toString(){

        return "productid: "+ productId + " , productname:" + productName + ", category: "+ category;

    }

}

class search{

    // linear search based on product id

    Product linearSearch(Product[] p, int id){

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

     for(Product pr:p){

       if(pr.getproductId()==id){

             return pr;

       }

     }

     return null;

    }

    // binary search based on product name

    Product binarySearch(Product[] p,int left, int right,String name){

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

        Arrays.sort(p,Comparator.comparing(Product::getproductName,String.CASE\_INSENSITIVE\_ORDER));

        while(left<=right){

            int mid=left+(right-left)/2;

            int  val= p[mid].getproductName().compareToIgnoreCase(name);

            if(val==0) return p[mid];

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

            else right=mid-1;

        }

        return null;

    }

}

public class E\_Commerce {

    public static void main(String args[]){

        Product[] p={

            new Product(111, "Sunscreen", "skincare"),

            new Product(134, "chain", "accessories"),

            new Product(451,"crop top","clothing"),

            new Product(123,"laptop","electronics"),

            new Product(642,"shoes","footwear")

        };

        search ob= new search();

        Product res= ob.linearSearch(p, 123);

        System.out.println(res==null?"product not found":res);

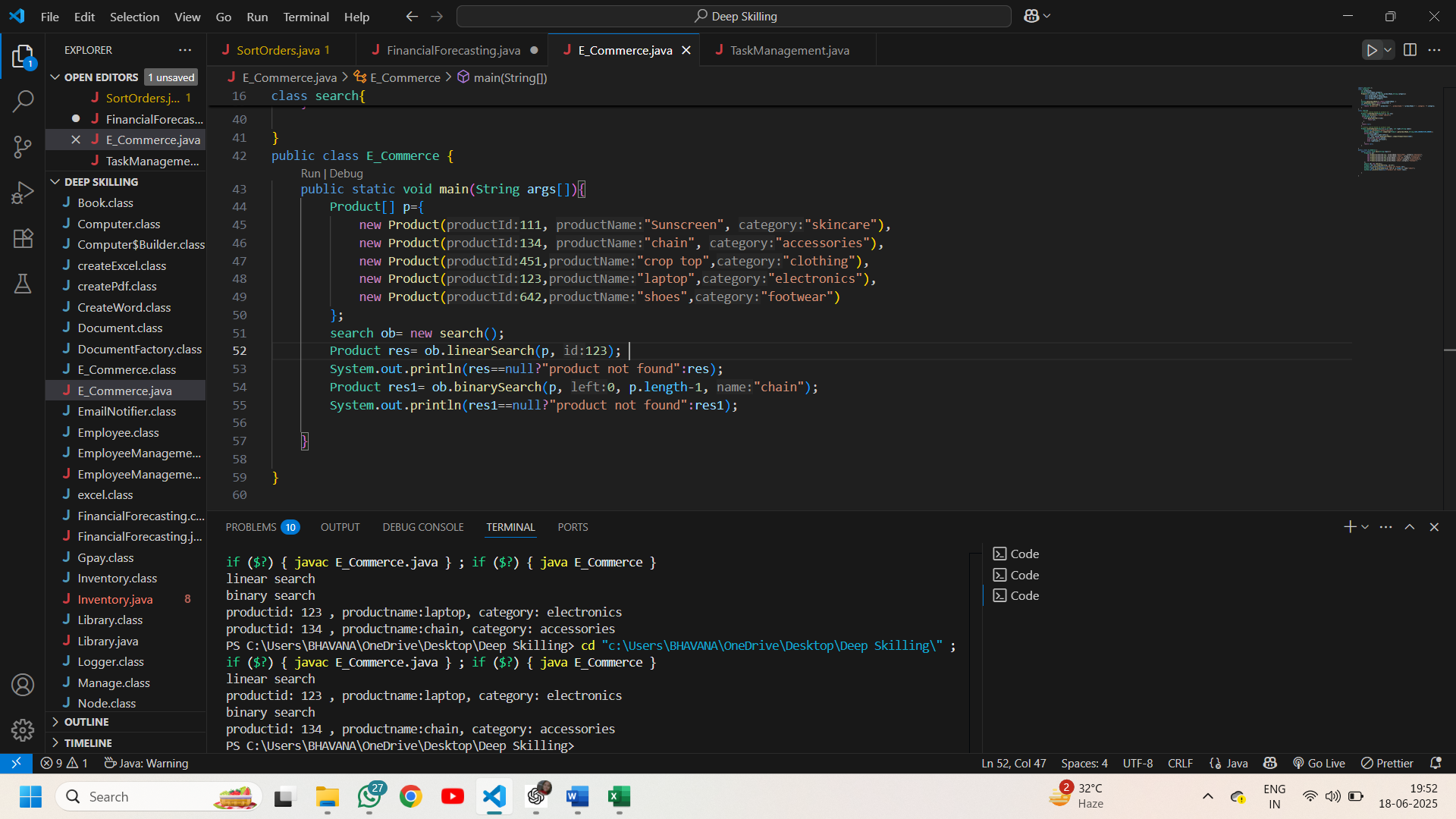
        Product res1= ob.binarySearch(p, 0, p.length-1, "chain");

        System.out.println(res1==null?"product not found":res1);

    }

}

**Output:**

****

**Exercise-3: Sorting customer orders**

**Code:**

import java.util.\*;

class Order{

    int orderId;

    String customerName;

    double totalPrice;

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

        this.orderId=orderId;

        this.customerName=customerName;

        this.totalPrice=totalPrice;

    }

    public String toString(){

        return "order id: "+orderId+" customer name: "+ customerName+" total price: "+ totalPrice;

    }

}

class Sort{

    // descending order

    void bubbleSort(Order a[]){

        for(int i=0;i<a.length-1;i++){

             for(int j=0;j<a.length-i-1;j++){

                if(a[j].totalPrice<a[j+1].totalPrice){

                    double temp=a[j].totalPrice;

                    a[j].totalPrice=a[j+1].totalPrice;

                    a[j+1].totalPrice=temp;

                }

             }

        }

    }

    // qucik sort- descending order(highest total price first)

    void quickSort(Order[] arr,int low,int high) {

        if (low<high) {

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

            quickSort(arr,low,pi-1);

            quickSort(arr,pi+1,high);

        }

    }

    int partition(Order[] arr,int low,int high) {

        double pivot = arr[high].totalPrice;

        int i =low-1;

        for (int j=low;j<high;j++) {

            if (arr[j].totalPrice>=pivot) {

                i++;

                double temp=arr[i].totalPrice;

                arr[i].totalPrice=arr[j].totalPrice;

                arr[j].totalPrice=temp;

            }

        }

        double temp= arr[i+1].totalPrice;

        arr[i+1].totalPrice=arr[high].totalPrice;

        arr[high].totalPrice=temp;

        return i+1;

    }

    }

public class SortOrders {

    public static void main(String[] args) {

        Order arr[]={

            new Order(3446, "bhavana", 4569),

            new Order(3467, "rahul", 3217),

            new Order(3486, "poorna", 5000),

            new Order(3491, "swarna", 3780),

            new Order(3495, "sameer", 6728),

        };

        Sort ob= new Sort();

        ob.bubbleSort(arr);

        System.out.println("after sorting using bubble sort:");

        for(Order i:arr){

            System.out.println(i);

        }

        ob.quickSort(arr, 0, arr.length-1);

         System.out.println("after sorting using quick sort:");

        for(Order i:arr){

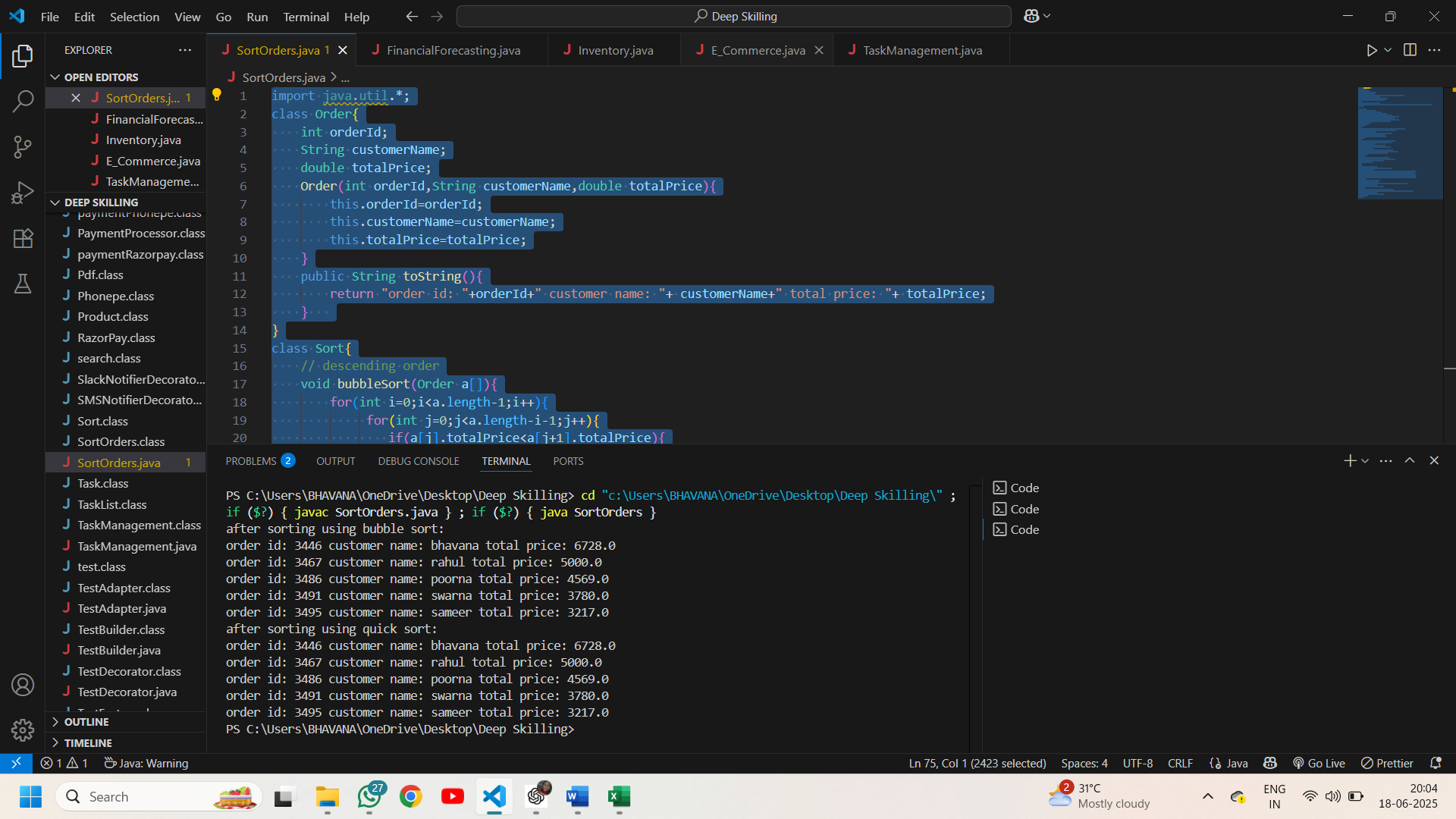
            System.out.println(i);

        }

    }

}

**Output:**



**Exercise-4 Employee Management system**

**Code:**

class Employee{

    int employeeId;

    String name,position;

    double salary;

    Employee(int employeeId,String name,String position,double salary){

        this.employeeId=employeeId;

        this.name=name;

        this.position=position;

        this.salary=salary;

    }

    int getEmployeeId(){ return employeeId; }

    public String toString(){

        return "employee id: "+ employeeId+" name: "+name+" position: "+position+" salary: "+salary;

    }

}

class Manage{

    // employee info

    Employee arr[];

    int size;

    Manage(int capacity){

       arr= new Employee[capacity];

       size=0;

    }

    // addition of employees

    void add(Employee emp){

        if(size<arr.length)

        arr[size++]=emp;

        else System.out.println("array's capacity is full");

    }

    // linear search based on employee's id

    Employee search(int id){

        for(int i=0;i<size-1;i++){

            if(arr[i].getEmployeeId()==id)

            return arr[i];

        }

        return null;

    }

    // to display

    void traverse(){

        if(size==0){System.out.println("array is empty");

        return;}

        for(int i=0;i<size;i++)

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

    }

    // delete based on id

    void delete(int id){

        int ind=-1;

        for(int i=0;i<size-1;i++){

            if(arr[i].getEmployeeId()==id){

                ind=i;

                break;

            }

        }

        if(ind==-1) System.out.println("no employee found");

        else{

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

                arr[i]=arr[i+1];

            arr[size-1]=null;

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

        }

    }

}

public class EmployeeManagement {

        public static void main(String[] args) {

        Manage ob= new Manage(4);

        ob.add( new Employee(2134,"Mamatha","Associate professor",45678));

        ob.add(new Employee(2156,"Suneetha","HOD",62054));

        ob.add(new Employee(2178,"Sekhar","principal",86822.98));

        ob.add(new Employee(2183,"prasanthi","lab technician",20987));

        System.out.println("employees:"); ob.traverse();

        ob.delete(2156);

        ob.delete(2157);

        System.out.println("employees after deleting");

        ob.traverse();

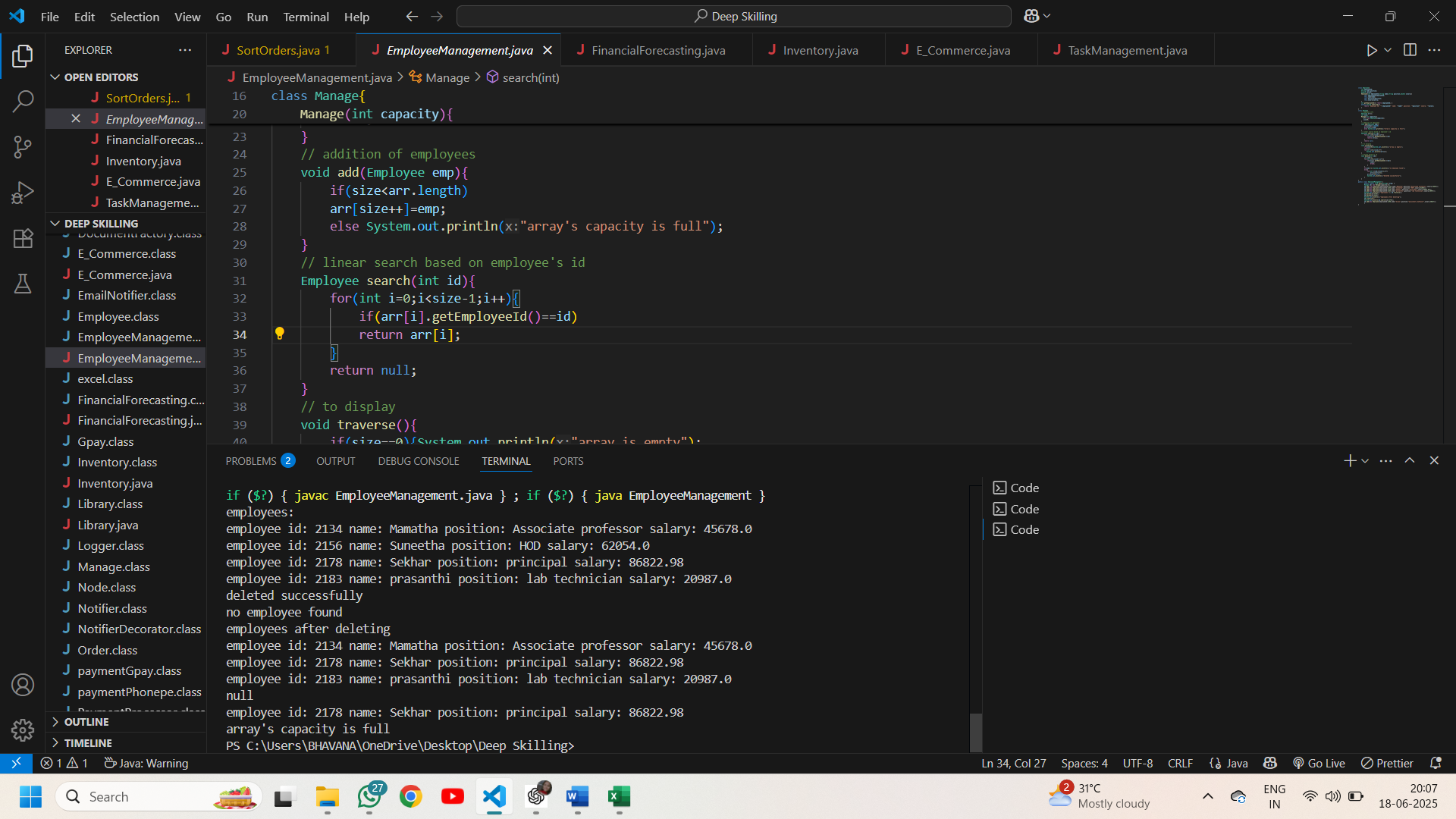
        System.out.println(ob.search(2178));

        ob.add(new Employee(2198,"kiran","assistant professor",38617));

        }

}

**Output:**



**Exercise -5 Task Management system**

**Code:**

class Task{

    int taskId;

    String taskName,status;

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

        this.taskId=taskId;

        this.taskName=taskName;

        this.status=status;

    }

    public String toString(){

        return "task id: "+taskId+" taskname: "+taskName+" status: "+ status;

    }

}

class Node{

    Task task;

    Node next;

    Node(Task ob){

        this.task=ob;

        this.next=null;

    }

}

class TaskList{

     Node head;

    // add

    void add(Task ob ){

        Node newnode= new Node(ob);

        if(head==null){

            head=newnode;

        }else{

            Node temp=head;

            while(temp.next!=null){

                temp=temp.next;

            }

            temp.next=newnode;

        }

    }

    // search based on id

    Task search(int task\_id){

        Node temp=head;

        while(temp!=null){

            if(temp.task.taskId==task\_id)

                return temp.task;

                temp=temp.next;

        }

        return null;

    }

    // traverse

    void traverse(Node head){

        if(head==null){

            System.out.println("linkedlist is empty! no tasks");

            return;

        }

        Node temp=head;

        while(temp!=null){

            System.out.println(temp.task);

            temp=temp.next;

        }

    }

    // delete based on id

    void delete(int task\_id){

        if(head==null) {

            System.out.println("No tasks to delete");

            return;

        }

        if(head.next==null){

            head=head.next; return;

        }

        Node temp=head;

        while(temp.next!=null && temp.next.task.taskId!=task\_id){

            temp=temp.next;

        }

        if(temp.next==null){System.out.println("no tasks to delete");

        return;}

        temp.next=temp.next.next;

    }

}

public class TaskManagement {

    public static void main(String[] args) {

         TaskList obj= new TaskList();

         obj.add(new Task(10,"user registration","completed"));

         obj.add(new Task(11,"filter products","pending"));

         obj.add(new Task(12,"cart management","completed"));

         obj.add(new Task(13,"order summary","pending"));

         obj.add(new Task(14,"wishlist","completed"));

         obj.traverse(obj.head);

         obj.delete(12);

         obj.traverse(obj.head);

         Task t1=obj.search(17);

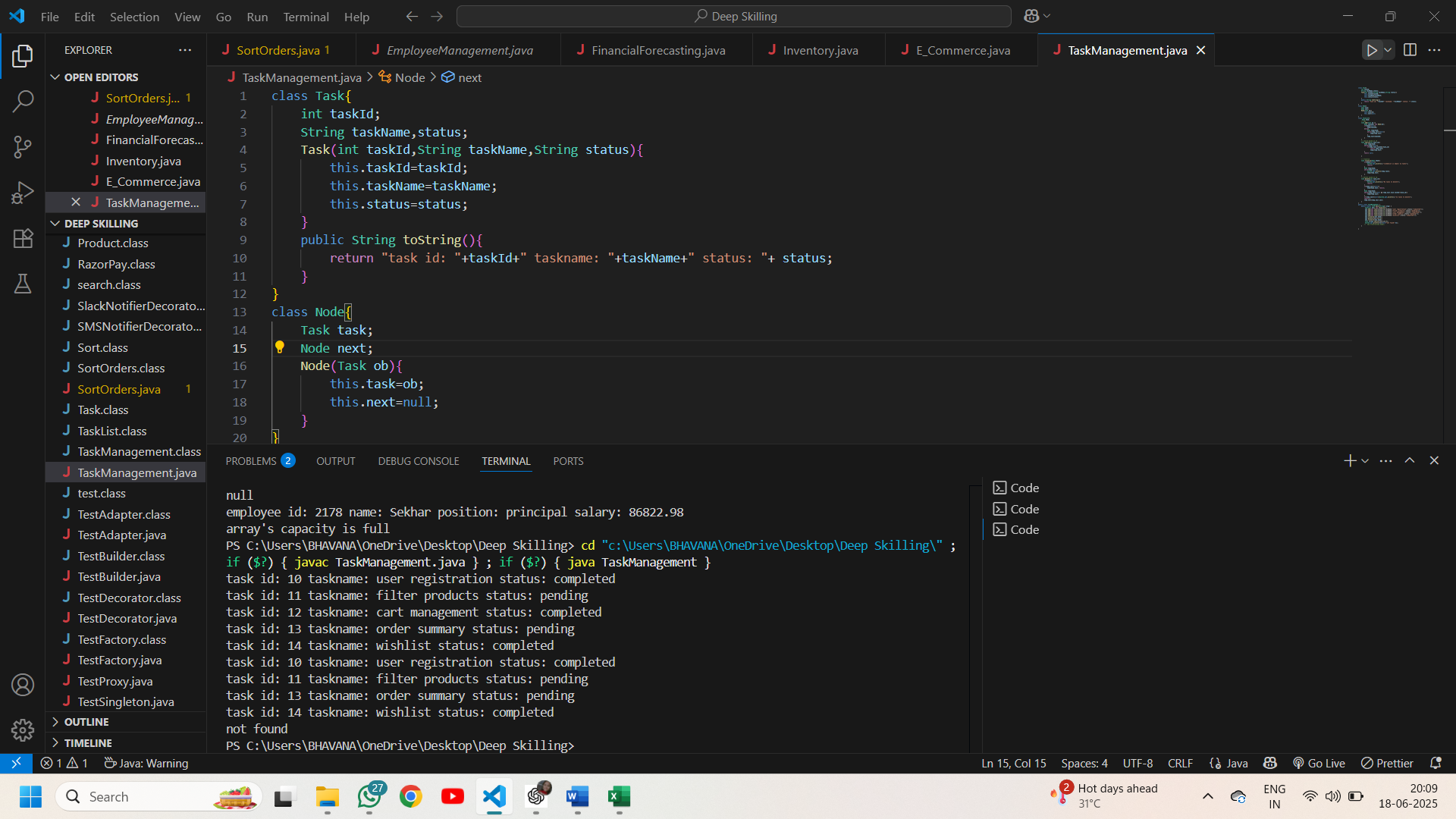
         System.out.println(t1==null?"not found":t1);

        //  obj.traverse(obj.head);

    }

}

**Output:**



**Exercise - 6 Library management system**

**Code:**

import java.util.\*;

class Book {

    int bookId;

    String title, author;

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

        this.bookId = bookId;

        this.title = title;

        this.author = author;

    }

    String getTitle() { return title; }

    String getAuthor() { return author; }

    public String toString() {

        return "book id: " + bookId + ", title: " + title + ", author: " + author;

    }

}

class Search {

    // Linear search based on title

    Book lin\_search\_title(String title, Book[] b) {

        for (Book book : b) {

            if (book.getTitle().equalsIgnoreCase(title)) return book;

        }

        return null;

    }

    // Linear search based on author

    Book lin\_search\_author(String author, Book[] b) {

        for (Book book : b) {

            if (book.getAuthor().equalsIgnoreCase(author)) return book;

        }

        return null;

    }

    // Binary search based on title

    Book bin\_search\_title(String title, Book[] b) {

        Arrays.sort(b, Comparator.comparing(Book::getTitle, String.CASE\_INSENSITIVE\_ORDER));

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

        while (left <= right) {

            int mid=left+(right-left)/2;

            int val=b[mid].getTitle().compareToIgnoreCase(title);

            if (val==0) return b[mid];

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

            else right = mid - 1;

        }

        return null;

    }

    // Binary search based on author

    Book bin\_search\_author(String author, Book[] b) {

        Arrays.sort(b,Comparator.comparing(Book::getAuthor, String.CASE\_INSENSITIVE\_ORDER));

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

        while (left <= right) {

            int mid=left+(right-left) / 2;

            int val=b[mid].getAuthor().compareToIgnoreCase(author);

            if (val==0) return b[mid];

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

            else right=mid-1;

        }

        return null;

    }

}

public class Library {

    public static void main(String[] args) {

        Book[] b = {

            new Book(101, "To Kill a Mockingbird", "Harper Lee"),

            new Book(102, "1984", "George Orwell"),

            new Book(103, "Pride and Prejudice", "Jane Austen"),

            new Book(104, "The Catcher in the Rye", "J.D. Salinger")

        };

        Search ob = new Search();

        Book l1 = ob.lin\_search\_author("Sudha Murthy", b);   // not found

        Book l2 = ob.lin\_search\_title("1984", b);             // found

        Book b1 = ob.bin\_search\_author("George Orwell", b);   // found

        Book b2 = ob.bin\_search\_title("Harry Potter", b);     // not found

        // Using ternary operator

        System.out.println(l1 == null ? "Linear Author Search: not found" : "Linear Author Search: " + l1);

        System.out.println(l2 == null ? "Linear Title Search: not found" : "Linear Title Search: " + l2);

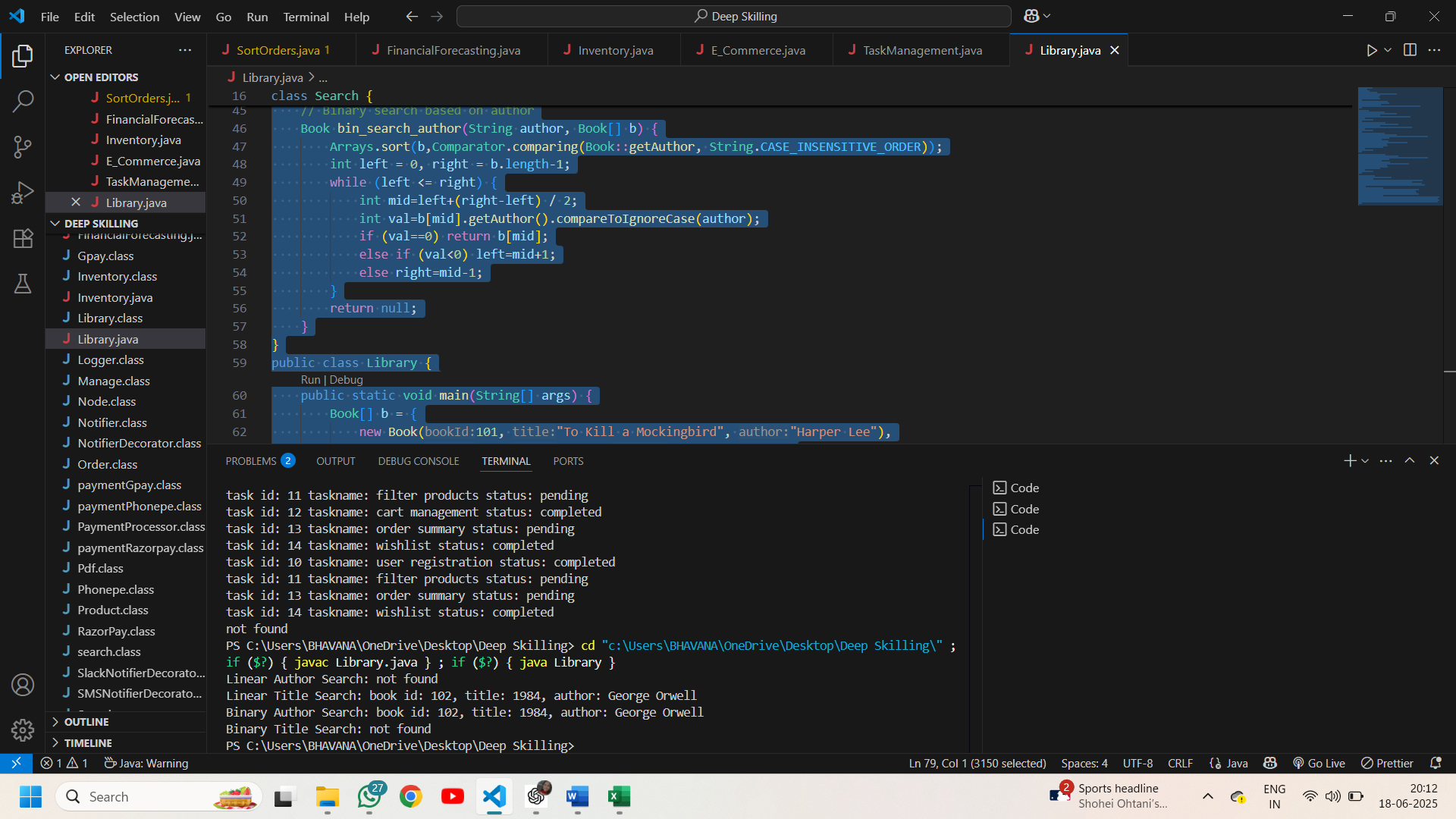
        System.out.println(b1 == null ? "Binary Author Search: not found" : "Binary Author Search: " + b1);

        System.out.println(b2 == null ? "Binary Title Search: not found" : "Binary Title Search: " + b2);

    }

}

Output:



**Exercise 7: Financial Forecasting**

**Code:**

public class FinancialForecasting {

    // recursion

    static double futureValue(double pastValue, double rate, int years){

        if(years==0) return pastValue;

        return (1+rate)\*futureValue(pastValue, rate, years-1);

    }

    public static void main(String[] args) {

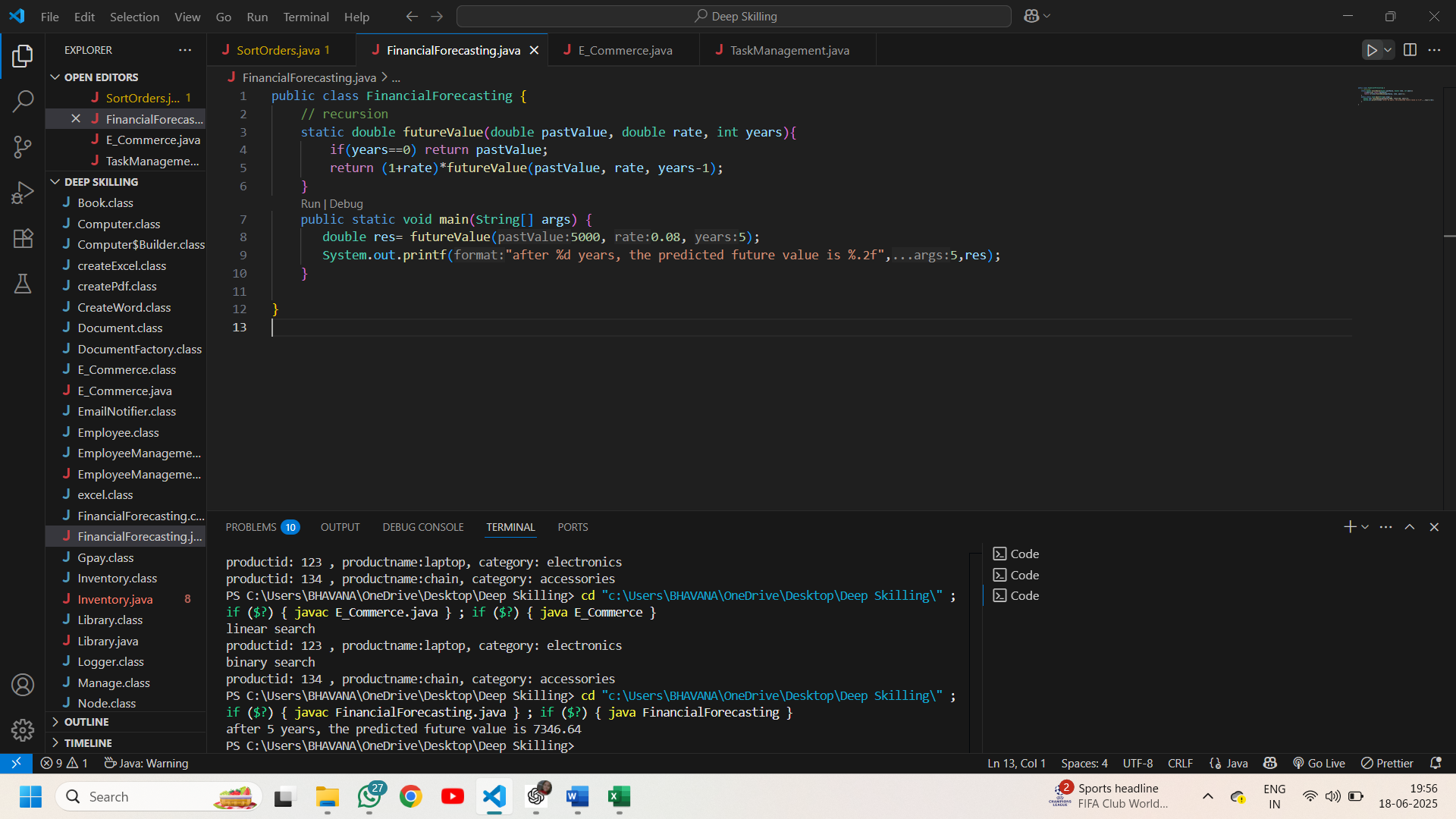
       double res= futureValue(5000, 0.08, 5);

       System.out.printf("after %d years, the predicted future value is %.2f",5,res);

    }

}

**Output:**

****