**Exercise 1: Inventory Management System**

**Scenario:**

You are developing an inventory management system for a warehouse. Efficient data storage and retrieval are crucial.

**Code:**

*Product.java:*

package inventory;

public 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;

}

@Override

public String toString() {

return "ID: " + productId + ", Name: " + productName +

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

}

}

*InventoryManager.java*

package inventory;

import java.util.HashMap;

public class InventoryManager {

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

public void addProduct(Product product) {

inventory.put(product.productId, product);

System.out.println("Product added: " + product);

}

public void updateProduct(int productId, int quantity, double price) {

Product product = inventory.get(productId);

if (product != null) {

product.quantity = quantity;

product.price = price;

System.out.println("Product updated: " + product);

} else {

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

}

}

public void deleteProduct(int productId) {

Product removed = inventory.remove(productId);

if (removed != null) {

System.out.println("Product deleted: " + removed);

} else {

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

}

}

public void displayAll() {

if (inventory.isEmpty()) {

System.out.println("No products in inventory.");

} else {

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

System.out.println(p);

}

}

}

}

*Main.java:*

package inventory;

public class Main {

public static void main(String[] args) {

InventoryManager manager = new InventoryManager();

Product p1 = new Product(101, "Keyboard", 10, 499.99);

Product p2 = new Product(102, "Mouse", 25, 299.99);

manager.addProduct(p1);

manager.addProduct(p2);

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

manager.displayAll();

System.out.println("\nUpdating Product 101...");

manager.updateProduct(101, 15, 459.99);

System.out.println("\nDeleting Product 102...");

manager.deleteProduct(102);

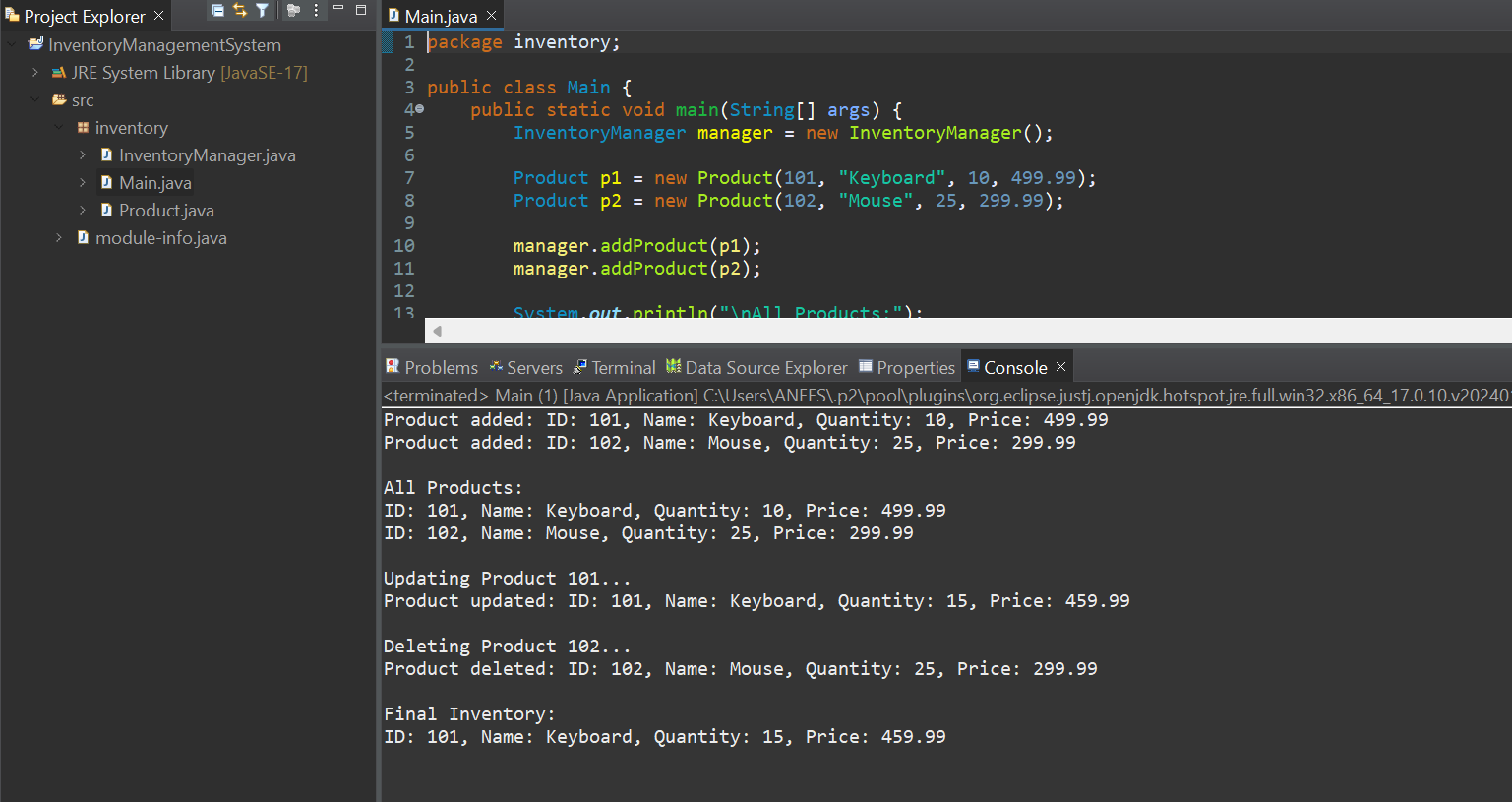
System.out.println("\nFinal Inventory:");

manager.displayAll();

}

}

**Output:**



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

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Code:**

*Product.java*

public 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;

}

@Override

public String toString() {

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

}

}

*SearchFunctionality.java*

public class SearchFunctionality {

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

for (Product p : products) {

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

return p;

}

}

return null;

}

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

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

while (left <= right) {

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

int compare = name.compareToIgnoreCase(products[mid].productName);

if (compare == 0)

return products[mid];

else if (compare < 0)

right = mid - 1;

else

left = mid + 1;

}

return null;

}

}

*Main.java*

import java.util.Arrays;

import java.util.Comparator;

public class Main {

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, "Phone", "Electronics"),

new Product(5, "Bag", "Fashion")

};

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

Product linearResult = SearchFunctionality.linearSearch(products, "Phone");

System.out.println(linearResult != null ? linearResult : "Product not found");

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

System.out.println("\n Binary Search for 'Phone':");

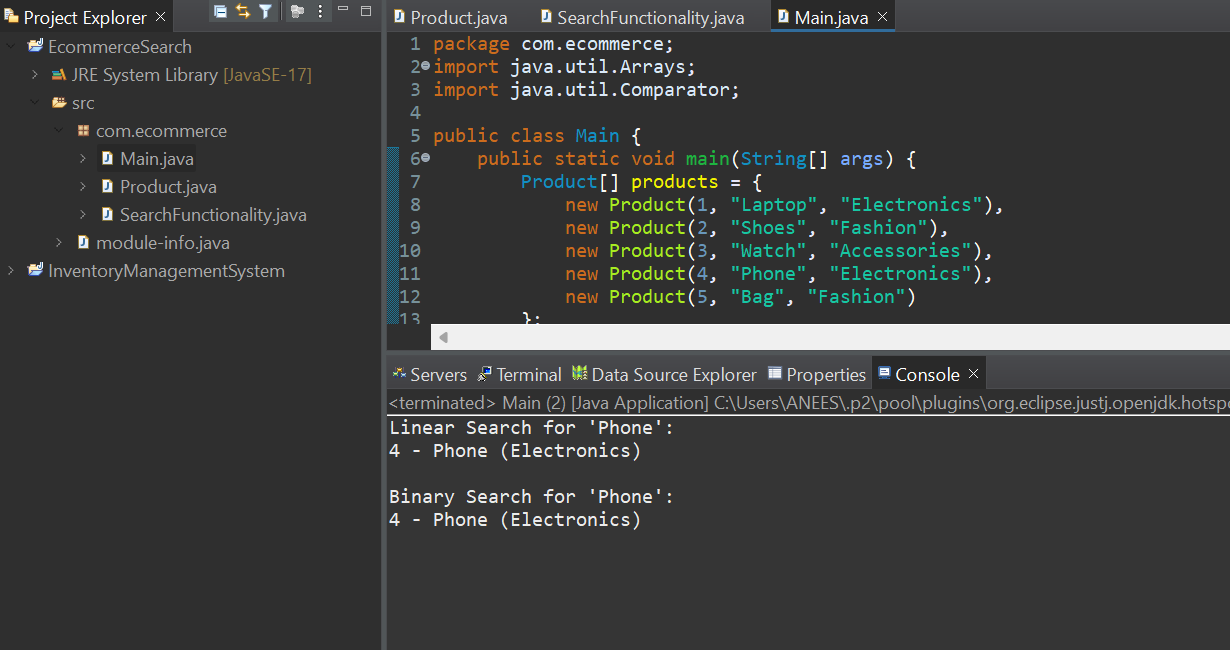
Product binaryResult = SearchFunctionality.binarySearch(products, "Phone");

System.out.println(binaryResult != null ? binaryResult : "Product not found");

}

}

**Output:**

****

**Exercise 3: Sorting Customer Orders**

**Scenario:**

You are tasked with sorting customer orders by their total price on an e-commerce platform. This helps in prioritizing high-value orders.

**Code:**

*Order.java*

package com.ecommerce;

public 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;

}

@Override

public String toString() {

return orderId + " - " + customerName + " - ₹" + totalPrice;

}}

*OrderSorting.java*

package com.ecommerce;

public class OrderSorting {

// Bubble Sort

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;

}}}}

// Quick Sort

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 title, Order[] orders) {

System.***out***.println("\n" + title);

for (Order order : orders) {

System.***out***.println(order);

}}}

*Main.java*

package com.ecommerce;

public class Main {

public static void main(String[] args) {

Order[] orders = {

new Order(101, "Anees", 950.0),

new Order(102, "Arshi", 1300.0),

new Order(103, "Aabi", 500.0),

new Order(104, "Shareefa", 1600.0),

new Order(105, "Madhu", 800.0)

};

//Bubble Sort

Order[] bubbleSorted = orders.clone();

OrderSorting.*bubbleSort*(bubbleSorted);

OrderSorting.*printOrders*("Bubble Sort by Price", bubbleSorted);

// Quick Sort

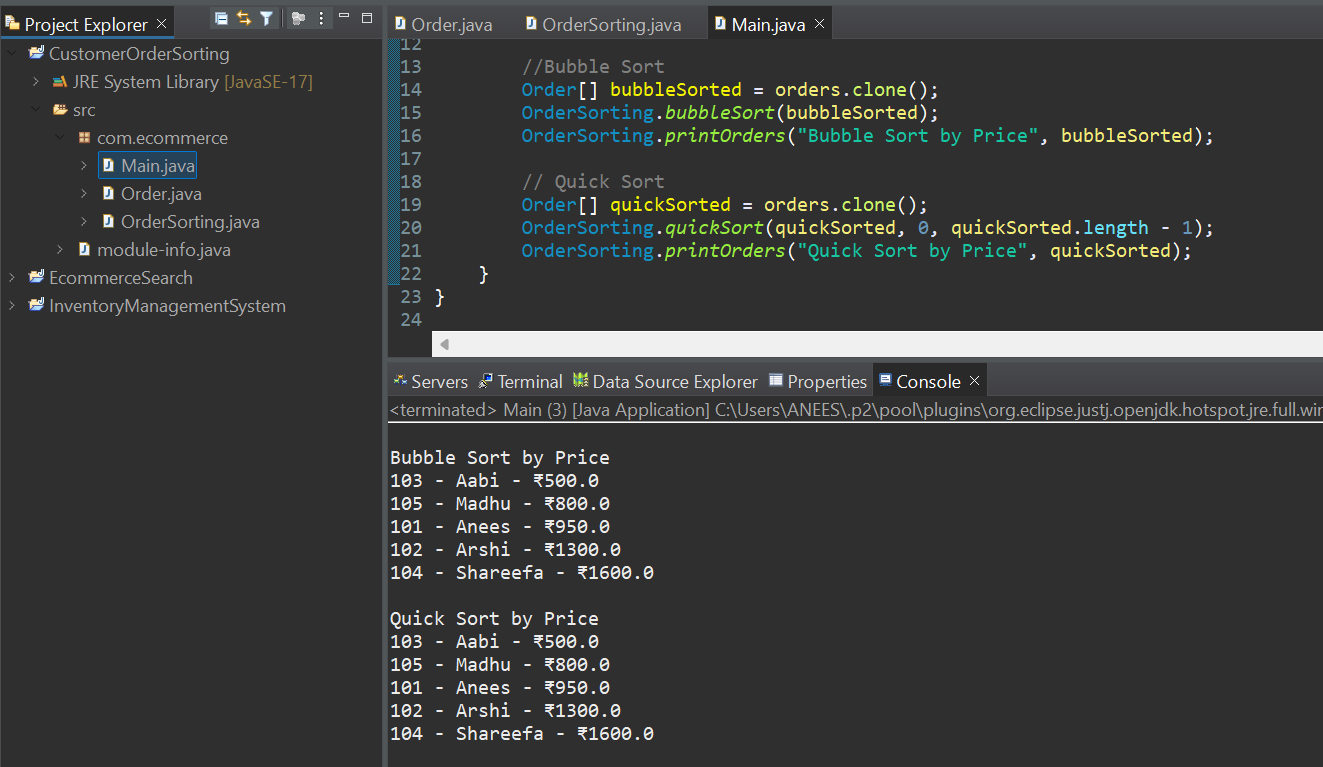
Order[] quickSorted = orders.clone();

OrderSorting.*quickSort*(quickSorted, 0, quickSorted.length - 1);

OrderSorting.*printOrders*("Quick Sort by Price", quickSorted);

}}

**Output:**

****

**Exercise 4: Employee Management System**

**Scenario:**

You are developing an employee management system for a company. Efficiently managing employee records is crucial.

**Code:**

*Employee.java:*

package com.company.ems;

public class Employee {

private int id;

private String name;

private String role;

private double salary;

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

this.id = id;

this.name = name;

this.role = role;

this.salary = salary;

}

public int getId() {

return id;

}

public void displayInfo() {

System.out.println("ID: " + id + ", Name: " + name + ", Role: " + role + ", Salary: ₹" + salary);

}}

*EmployeeManager.java*  
package com.company.ems;

public class EmployeeManager {

private Employee[] records;

private int current;

public EmployeeManager(int capacity) {

records = new Employee[capacity];

current = 0;

}

public void addEmployee(Employee emp) {

if (current < records.length) {

records[current++] = emp;

} else {

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

}}

public void listAll() {

if (current == 0) {

System.out.println("No employee records found.");

return;}}

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

records[i].displayInfo();

}}

public void findEmployee(int empId) {

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

if (records[i].getId() == empId) {

records[i].displayInfo();

return;

} }

System.out.println("Employee ID " + empId + " not located.");

}

public void deleteEmployee(int empId) {

boolean match = false;

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

if (records[i].getId() == empId) {

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

records[j] = records[j + 1];

}

records[--current] = null;

match = true;

System.out.println("Employee ID " + empId + " deleted.");

break;}}

if (!match) {

System.out.println("Employee ID " + empId + " not found.");

}}}

*Main.java*

package com.company.ems;

public class Main {

public static void main(String[] args) {

EmployeeManager manager = new EmployeeManager(10);

manager.addEmployee(new Employee(1, "Anees", "Engineer", 50000));

manager.addEmployee(new Employee(2, "Raji", "Designer", 42000));

manager.addEmployee(new Employee(3, "Leela", "Tester", 39000));

System.***out***.println("Employee List:");

manager.listAll();

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

manager.findEmployee(2);

System.***out***.println("\nRemoving employee ID 2:");

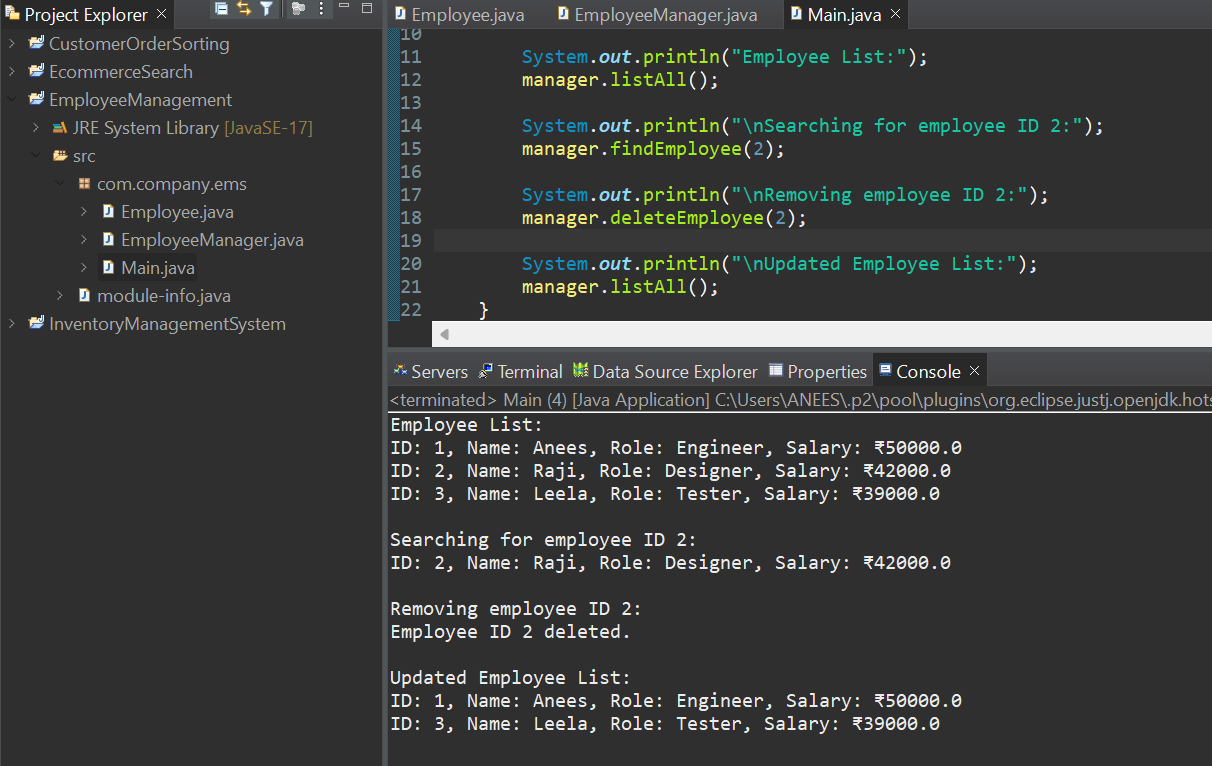
manager.deleteEmployee(2);

System.***out***.println("\nUpdated Employee List:");

manager.listAll();

}}

**Output:**



**Exercise 5: Task Management System**

**Scenario:**

You are developing a task management system where tasks need to be added, deleted, and traversed efficiently.

**Code:**

*Task.java*

package com.company.taskmanager;

public class Task {

private int id;

private String name;

private String state;

public Task(int id, String name, String state) {

this.id = id;

this.name = name;

this.state = state;

}

public int getId() {

return id;

}

public void show() {

System.out.println("Task ID: " + id + " | Name: " + name + " | Status: " + state);

}

}

*TaskNode.java*

package com.company.taskmanager;

public class TaskNode {

Task taskData;

TaskNode nextNode;

public TaskNode(Task task) {

this.taskData = task;

this.nextNode = null;

}

}

*TaskLinkedList.java*

package com.company.taskmanager;

public class TaskLinkedList {

private TaskNode start;

public TaskLinkedList() {

start = null;

}

public void insert(Task task) {

TaskNode fresh = new TaskNode(task);

if (start == null) {

start = fresh;

} else {

TaskNode walk = start;

while (walk.nextNode != null) {

walk = walk.nextNode;

}

walk.nextNode = fresh;

}

}

public void viewAll() {

if (start == null) {

System.out.println("No tasks in the list.");

return;

}

TaskNode node = start;

while (node != null) {

node.taskData.show();

node = node.nextNode;

}

}

public void searchById(int tid) {

TaskNode ref = start;

boolean found = false;

while (ref != null) {

if (ref.taskData.getId() == tid) {

ref.taskData.show();

found = true;

break;

}

ref = ref.nextNode;

}

if (!found) {

System.out.println("Task ID " + tid + " not found.");

}

}

public void removeById(int tid) {

if (start == null) {

System.out.println("Task list is empty.");

return;

}

if (start.taskData.getId() == tid) {

start = start.nextNode;

System.out.println("Task ID " + tid + " removed.");

return;

}

TaskNode prev = start;

TaskNode curr = start.nextNode;

while (curr != null) {

if (curr.taskData.getId() == tid) {

prev.nextNode = curr.nextNode;

System.out.println("Task ID " + tid + " removed.");

return;

}

prev = curr;

curr = curr.nextNode;

}

System.out.println("Task ID " + tid + " not found.");

}}

*Main.java*

package com.company.taskmanager;

public class Main {

public static void main(String[] args) {

TaskLinkedList manager = new TaskLinkedList();

manager.insert(new Task(11, "UI Mockup", "To Do"));

manager.insert(new Task(12, "Database Setup", "In Progress"));

manager.insert(new Task(13, "Deployment", "Done"));

System.out.println("== Task List ==");

manager.viewAll();

System.out.println("\n== Searching Task ID 12 ==");

manager.searchById(12);

System.out.println("\n== Deleting Task ID 12 ==");

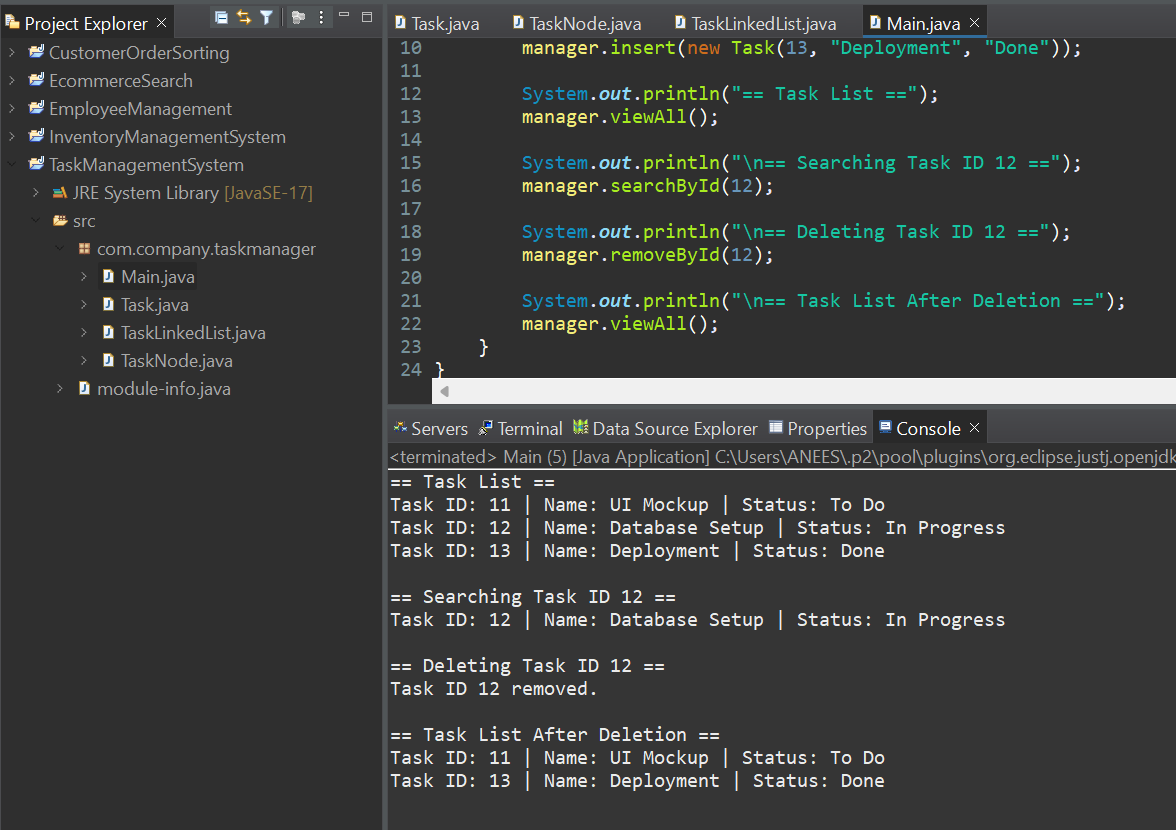
manager.removeById(12);

System.out.println("\n== Task List After Deletion ==");

manager.viewAll();

}}

**Output:**



**Exercise 6: Library Management System**

**Scenario:**

You are developing a library management system where users can search for books by title or author.

**Code:**

*Book.java*

package com.company.library;

public class Book {

private int bookId;

private String title;

private String author;

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

this.bookId = bookId;

this.title = title.toLowerCase();

this.author = author;

}

public String getTitle() {

return title;

}

public void showDetails() {

System.***out***.println("Book ID: " + bookId + " | Title: " + title + " | Author: " + author);

}}

*Library.java*

package com.company.library;

import java.util.Arrays;

import java.util.Comparator;

public class Library {

private Book[] collection;

private int size;

public Library(int capacity) {

collection = new Book[capacity];

size = 0;

}

public void addBook(Book b) {

if (size < collection.length) {

collection[size++] = b;

} else {

System.***out***.println("Library is full.");

}}

// Linear Search

public void searchByTitleLinear(String title) {

String query = title.toLowerCase();

boolean found = false;

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

if (collection[i].getTitle().equals(query)) {

collection[i].showDetails();

found = true;

}}

if (!found) {

System.***out***.println("Book titled \"" + title + "\" not found (Linear Search).");

}}

// Binary Search

public void searchByTitleBinary(String title) {

String query = title.toLowerCase();

Arrays.*sort*(collection, 0, size, Comparator.*comparing*(Book::getTitle));

int low = 0, high = size - 1;

boolean found = false;

while (low <= high) {

int mid = (low + high) / 2;

int compare = collection[mid].getTitle().compareTo(query);

if (compare == 0) {

collection[mid].showDetails();

found = true;

break;

} else if (compare < 0) {

low = mid + 1;

} else {

high = mid - 1;

}}

if (!found) {

System.***out***.println("Book titled \"" + title + "\" not found (Binary Search).");

}}

public void displayBooks() {

if (size == 0) {

System.***out***.println("No books in the library.");

return;

}

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

collection[i].showDetails();

}}}

*Main.java*

package com.company.library;

public class Main {

public static void main(String[] args) {

Library lib = new Library(10);

lib.addBook(new Book(1, "Harry Potter", "J.K. Rowling"));

lib.addBook(new Book(2, "Psychology of Money", "Morgan Housel"));

lib.addBook(new Book(3, "Atomic Habits", "James Clear"));

lib.addBook(new Book(4, "The Alchemist", "Paulo Coelho"));

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

lib.displayBooks();

System.***out***.println("\nSearch (Linear) for 'Atomic Habits':");

lib.searchByTitleLinear("Atomic Habits");

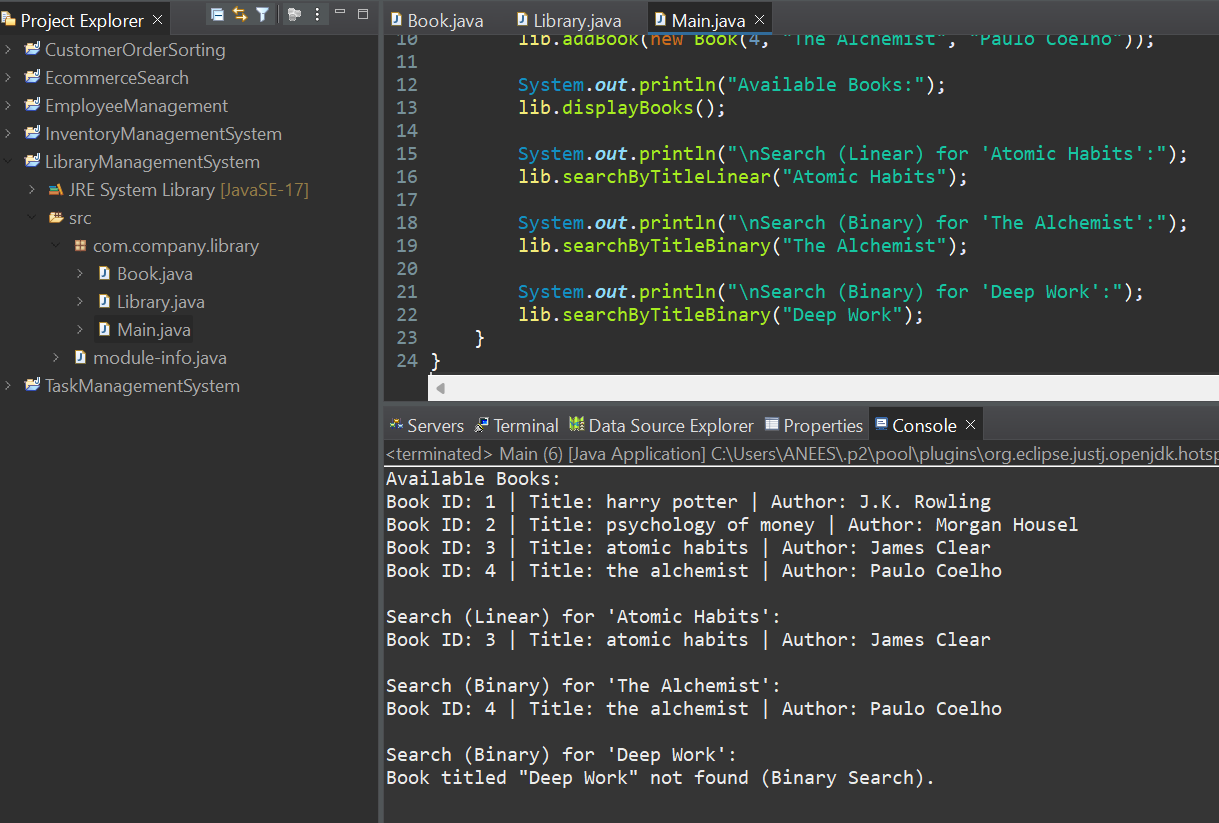
System.***out***.println("\nSearch (Binary) for 'The Alchemist':");

lib.searchByTitleBinary("The Alchemist");

System.***out***.println("\nSearch (Binary) for 'Deep Work':");

lib.searchByTitleBinary("Deep Work"); }}

**Output:**

****

**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**Code:**

*Forecast.java*

package com.company.finance;

public class Forecast {

public double predictValue(double currentValue, double growthRate, int years) {

if (years == 0) {

return currentValue;

}

return predictValue(currentValue \* (1 + growthRate), growthRate, years - 1); }

public double predictValueTail(double currentValue, double growthRate, int years) {

return predictHelper(currentValue, growthRate, years);}

private double predictHelper(double value, double rate, int n) {

if (n == 0) {

return value;}

return predictHelper(value \* (1 + rate), rate, n - 1);

}}

*Main.java*

package com.company.finance;

public class Main {

public static void main(String[] args) {

Forecast forecaster = new Forecast();

double initialAmount = 10000;

double growthRate = 0.10;

int years = 5;

double predicted = forecaster.predictValue(initialAmount, growthRate, years);

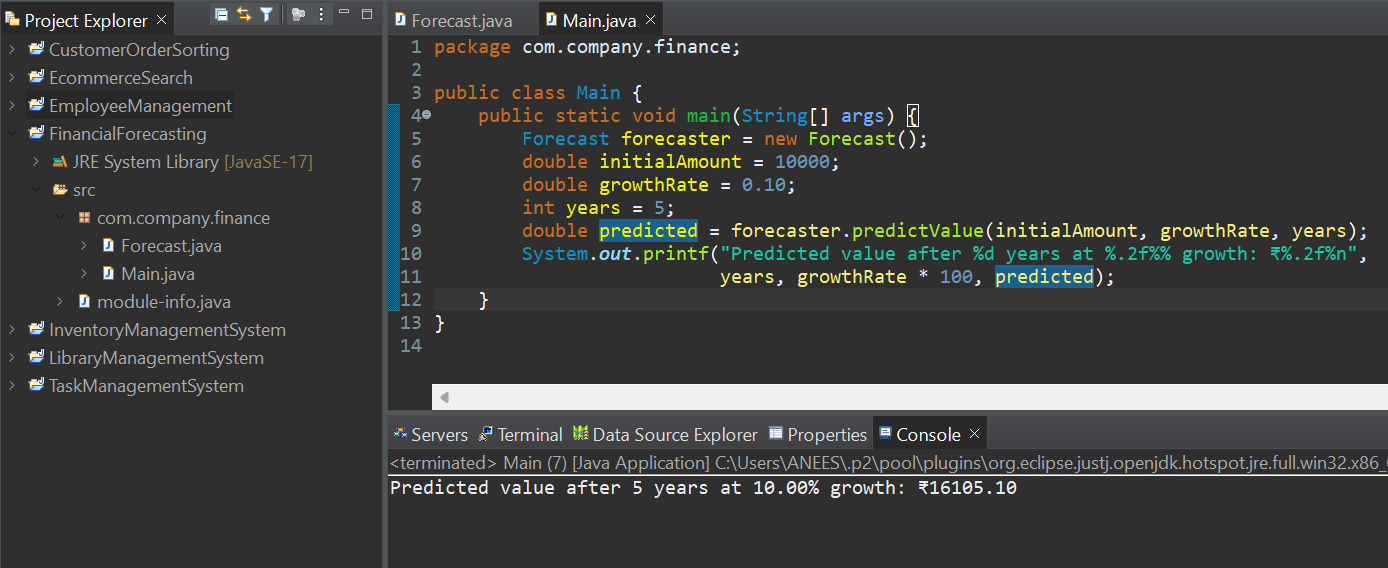
System.***out***.printf("Predicted value after %d years at %.2f%% growth: ₹%.2f%n",

years, growthRate \* 100, predicted);

}

}

**Output:**

****