**Exercise 1: Inventory Management System**

InventorySystem.java

import java.util.HashMap;

class Product {

int productId;

String productName;

int quantity;

double price;

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

this.productId = id;

this.productName = name;

this.quantity = qty;

this.price = price;

}

}

class Inventory {

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

void addProduct(Product p) {

products.put(p.productId, p);

}

void updateProduct(int id, int qty, double price) {

if (products.containsKey(id)) {

Product p = products.get(id);

p.quantity = qty;

p.price = price;

}

}

void deleteProduct(int id) {

products.remove(id);

}

void showAll() {

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

System.out.println(p.productId + " | " + p.productName + " | " + p.quantity + " | " + p.price);

}

}

}

public class Exercise1\_InventorySystem {

public static void main(String[] args) {

Inventory inv = new Inventory();

inv.addProduct(new Product(101, "Mouse", 50, 299.99));

inv.addProduct(new Product(102, "Keyboard", 30, 499.00));

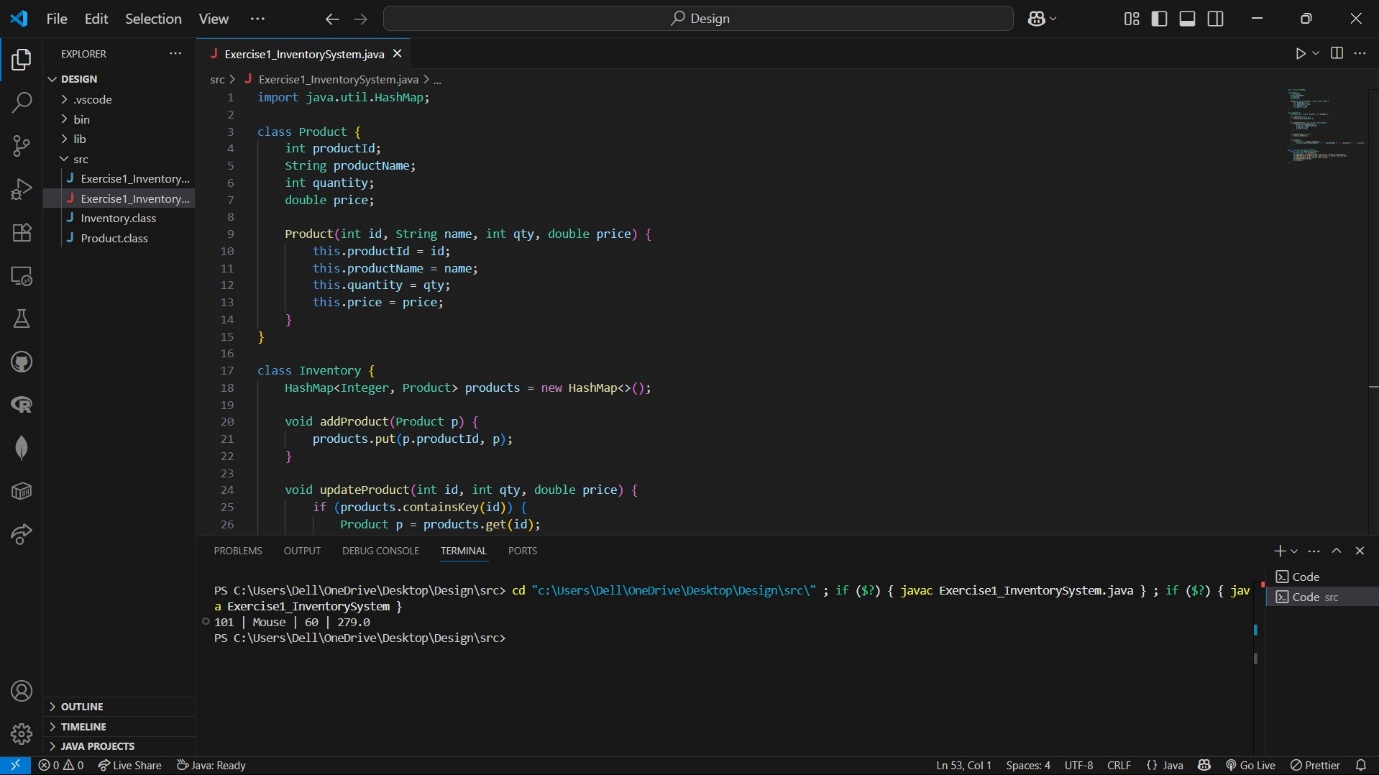
inv.updateProduct(101, 60, 279.00);

inv.deleteProduct(102);

inv.showAll();

}

}



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

Search.java

class ProductSearch {

int productId;

String productName;

String category;

ProductSearch(int id, String name, String cat) {

this.productId = id;

this.productName = name;

this.category = cat;

}

}

public class Exercise2\_Search {

static ProductSearch linearSearch(ProductSearch[] arr, String name) {

for (ProductSearch p : arr) {

if (p.productName.equals(name)) return p;

}

return null;

}

static ProductSearch binarySearch(ProductSearch[] arr, String name) {

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

while (left <= right) {

int mid = (left + right) / 2;

int cmp = arr[mid].productName.compareTo(name);

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

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

else right = mid - 1;

}

return null;

}

public static void main(String[] args) {

ProductSearch[] products = {

new ProductSearch(1, "Apple", "Fruits"),

new ProductSearch(2, "Banana", "Fruits"),

new ProductSearch(3, "Mango", "Fruits"),

new ProductSearch(4, "Orange", "Fruits")

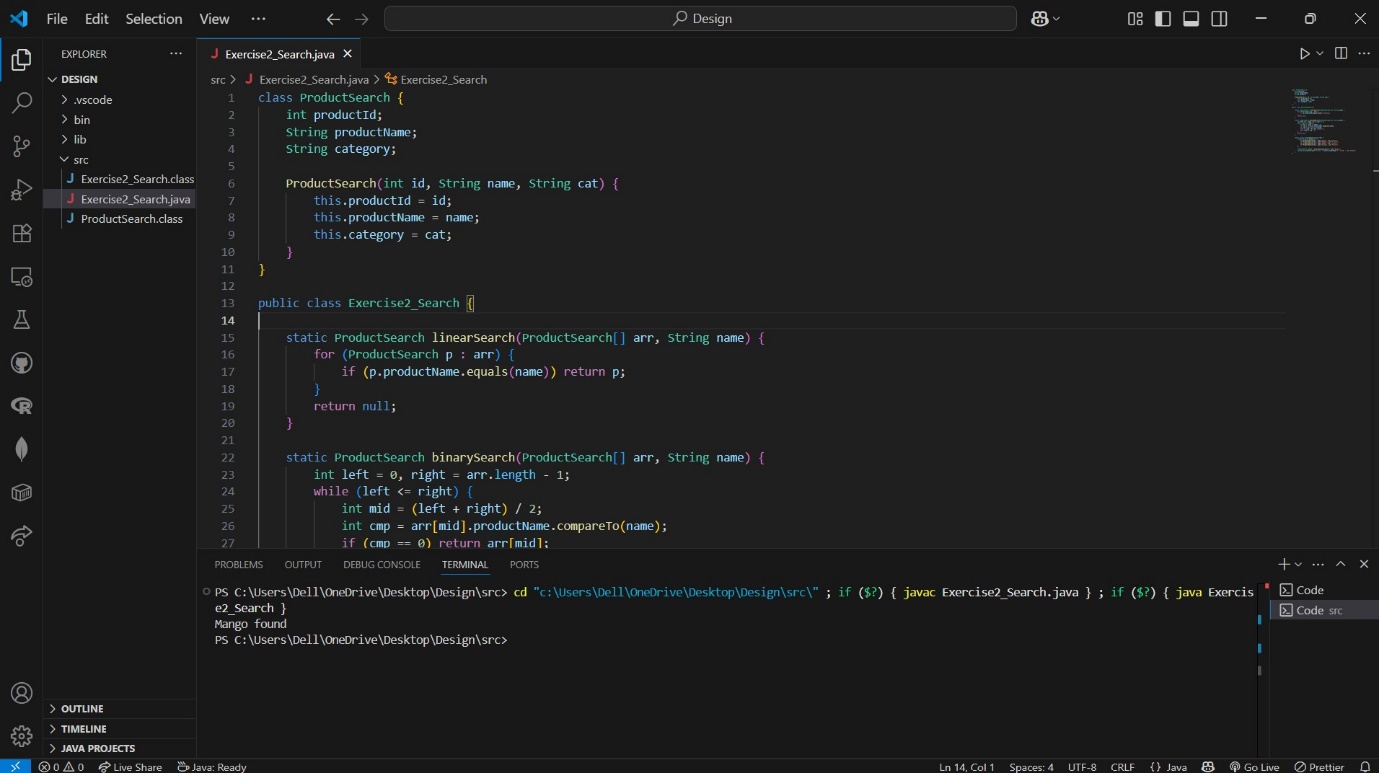
};

ProductSearch result = binarySearch(products, "Mango");

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

}

}



**Exercise 3: Sorting Customer Orders**

Sorting.java

class Order {

int orderId;

String customerName;

double totalPrice;

Order(int id, String name, double price) {

this.orderId = id;

this.customerName = name;

this.totalPrice = price;

}

}

public class Exercise3\_Sorting {

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;

}

}

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);

}

}

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 main(String[] args) {

Order[] orders = {

new Order(1, "Alice", 450),

new Order(2, "Bob", 1200),

new Order(3, "Charlie", 700)

};

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

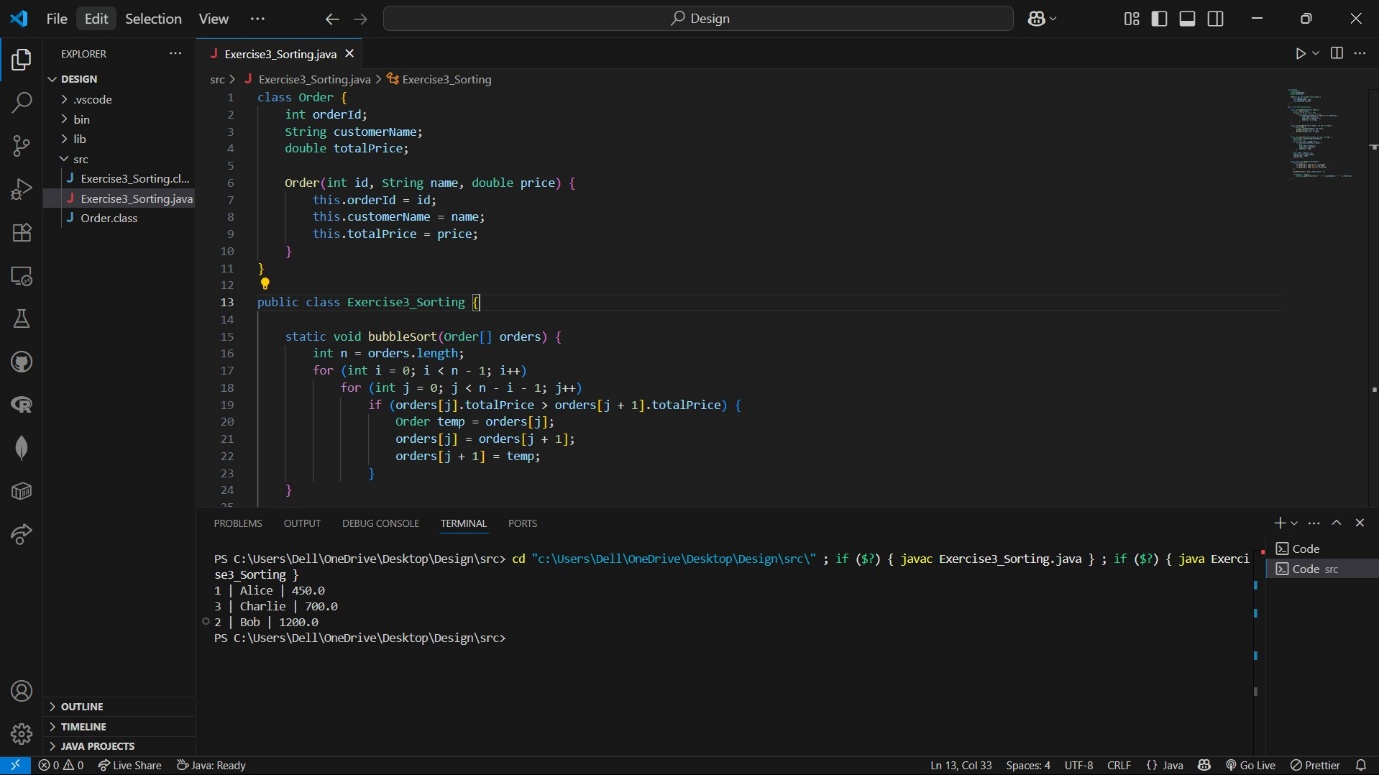
for (Order o : orders) {

System.out.println(o.orderId + " | " + o.customerName + " | " + o.totalPrice);

}

}

}



**Exercise 4: Employee Management System**

EmployeeManagement.java

class Employee {

int employeeId;

String name;

String position;

double salary;

Employee(int id, String name, String pos, double sal) {

this.employeeId = id;

this.name = name;

this.position = pos;

this.salary = sal;

}

}

public class Exercise4\_EmployeeManagement {

static final int MAX = 100;

static Employee[] employees = new Employee[MAX];

static int count = 0;

static void addEmployee(Employee e) {

if (count < MAX) {

employees[count++] = e;

} else {

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

}

}

static Employee searchEmployee(int id) {

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

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

}

return null;

}

static void traverseEmployees() {

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

Employee e = employees[i];

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

}

}

static 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;

break;

}

}

}

public static void main(String[] args) {

addEmployee(new Employee(1, "Alice", "Manager", 75000));

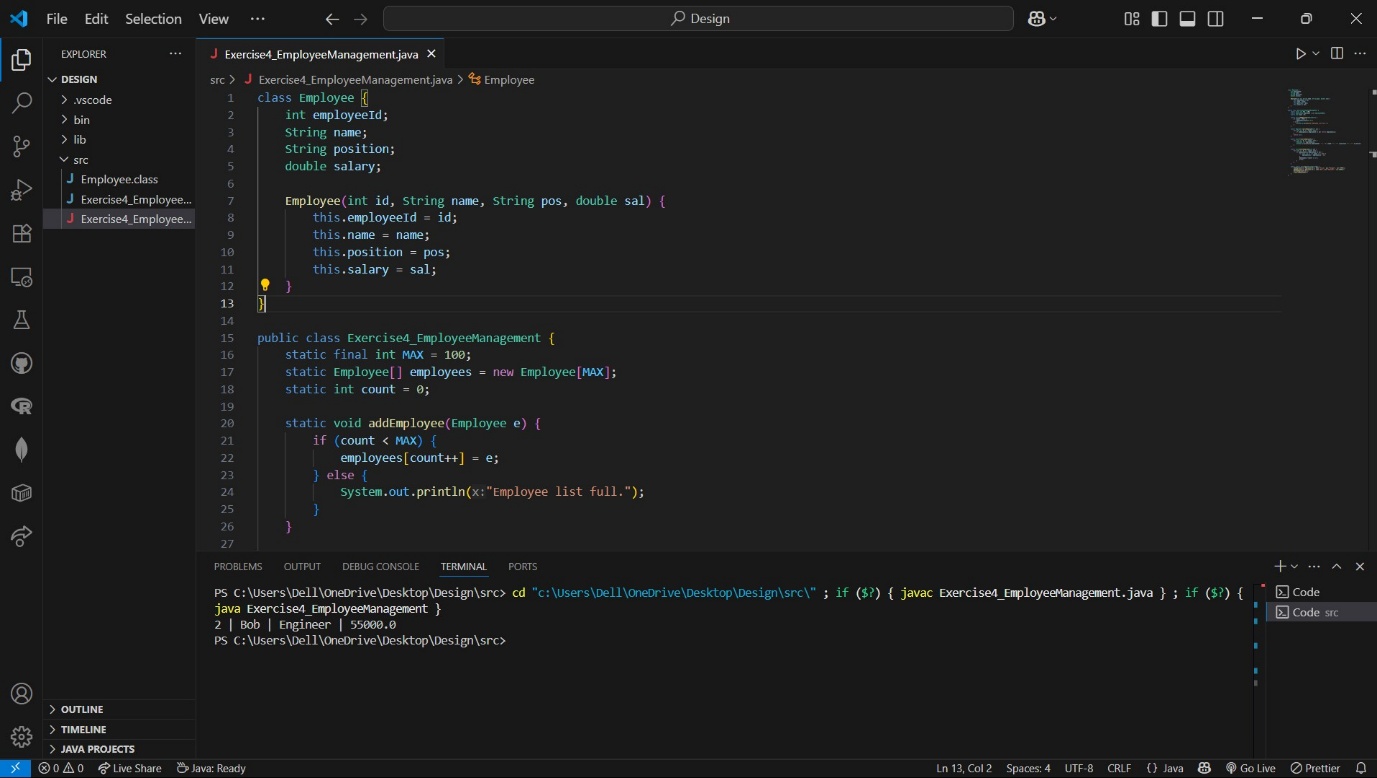
addEmployee(new Employee(2, "Bob", "Engineer", 55000));

deleteEmployee(1);

traverseEmployees();

}

}



**Exercise 5: Task Management System**

TaskManagement.java

class Task {

int taskId;

String taskName;

String status;

Task next;

Task(int id, String name, String status) {

this.taskId = id;

this.taskName = name;

this.status = status;

this.next = null;

}

}

class TaskList {

Task head = null;

void addTask(Task task) {

if (head == null) {

head = task;

} else {

Task temp = head;

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

temp.next = task;

}

}

Task searchTask(int id) {

Task temp = head;

while (temp != null) {

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

temp = temp.next;

}

return null;

}

void traverseTasks() {

Task temp = head;

while (temp != null) {

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

temp = temp.next;

}

}

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;

}

}

}

public class Exercise5\_TaskManagement {

public static void main(String[] args) {

TaskList list = new TaskList();

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

list.addTask(new Task(2, "Setup DB", "In Progress"));

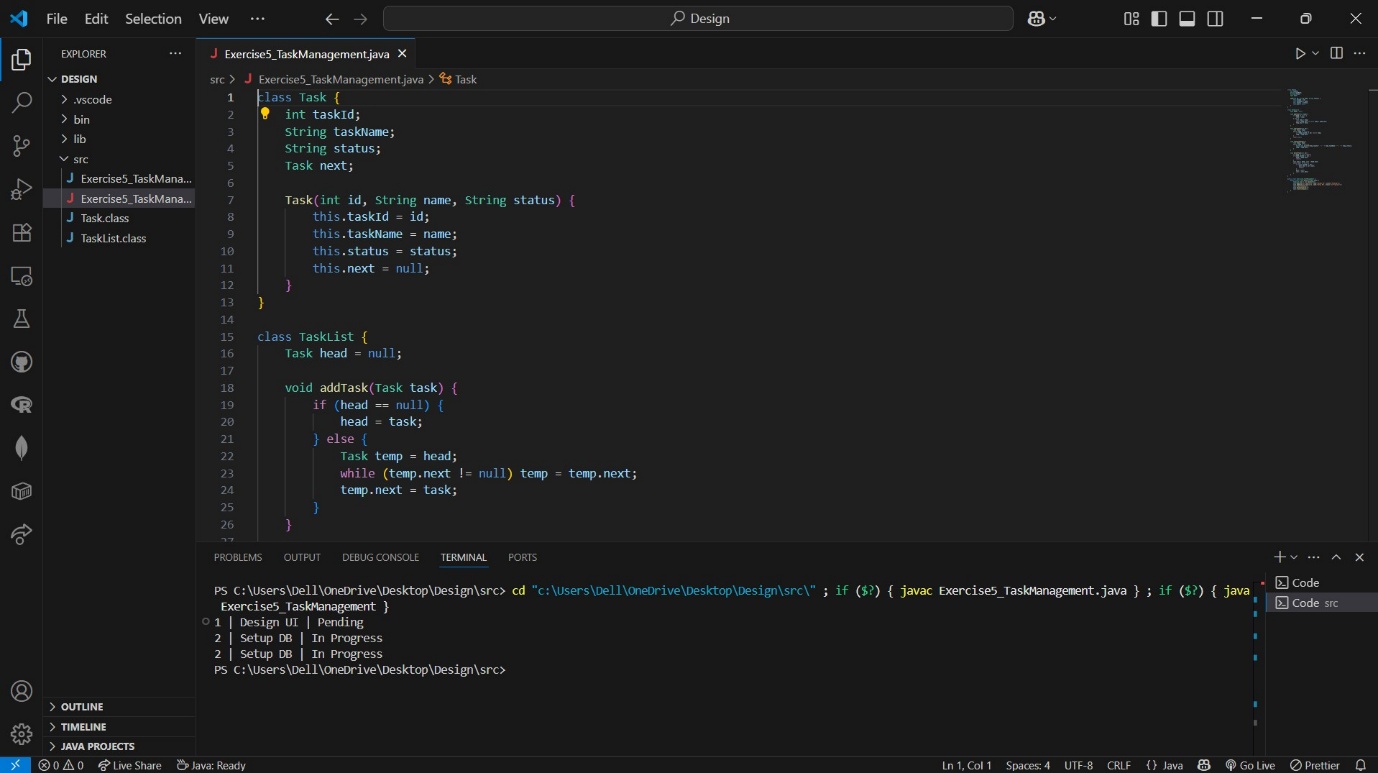
list.traverseTasks();

list.deleteTask(1);

list.traverseTasks();

}

}



**Exercise 6: Library Management System**

LibraryManagement.java

class Book {

int bookId;

String title;

String author;

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

this.bookId = id;

this.title = title;

this.author = author;

}

}

public class Exercise6\_LibraryManagement {

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

for (Book b : books) {

if (b.title.equalsIgnoreCase(title)) return b;

}

return null;

}

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

int low = 0, high = books.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

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

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

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

else high = mid - 1;

}

return null;

}

public static void main(String[] args) {

Book[] books = {

new Book(1, "A Tale of Two Cities", "Charles Dickens"),

new Book(2, "Brave New World", "Aldous Huxley"),

new Book(3, "Moby Dick", "Herman Melville"),

new Book(4, "War and Peace", "Leo Tolstoy")

};

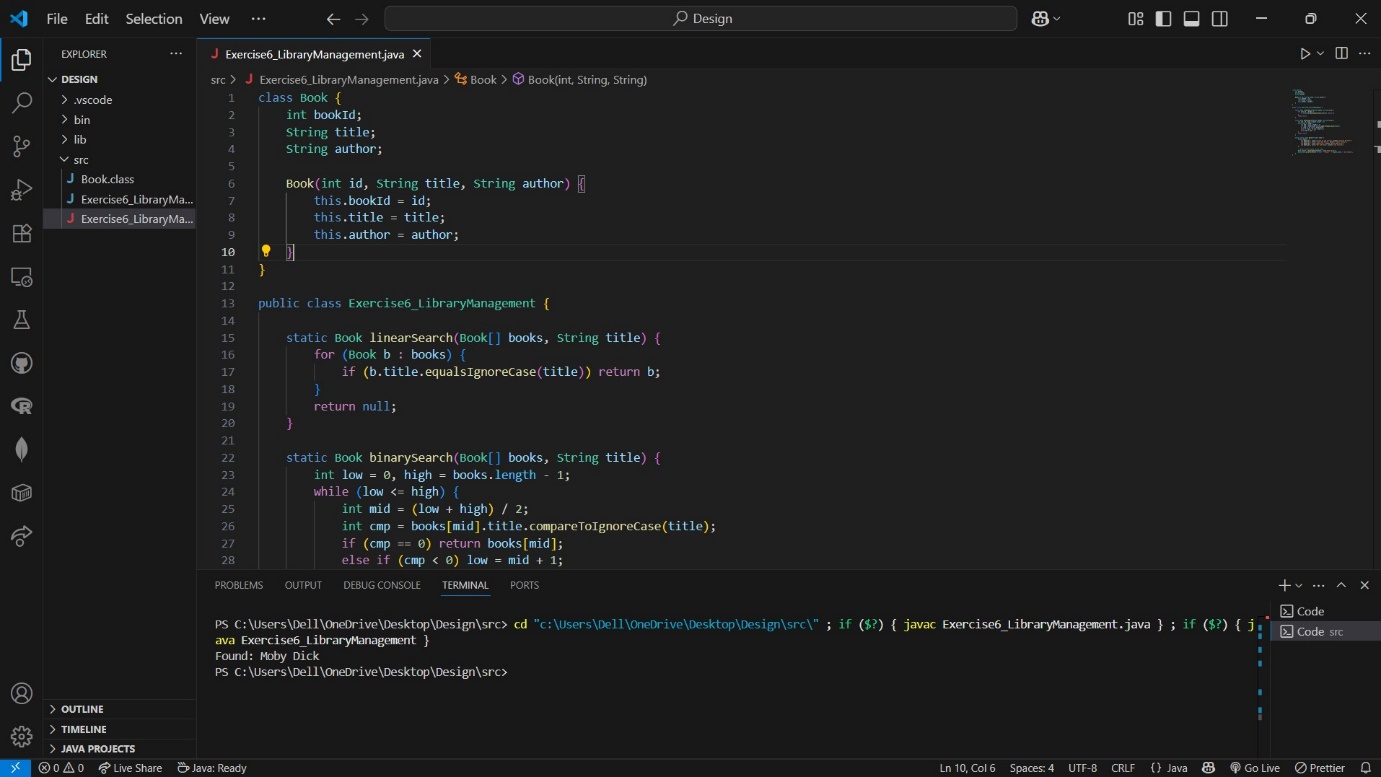
// Binary search needs sorted input

Book result = binarySearch(books, "Moby Dick");

System.out.println(result != null ? "Found: " + result.title : "Not Found");

}

}



**Exercise 7: Financial Forecasting**

FinancialForecast.java

public class Exercise7\_FinancialForecast {

static double predictFutureValue(double current, double rate, int years) {

if (years == 0) return current;

return predictFutureValue(current \* (1 + rate), rate, years - 1);

}

public static void main(String[] args) {

double currentValue = 10000;

double growthRate = 0.08; // 8%

int futureYears = 5;

double result = predictFutureValue(currentValue, growthRate, futureYears);

System.out.printf("Predicted Value after %d years: %.2f\n", futureYears, result);

}

}

