**Exercise 1: Inventory Management System**

**Week 1**

**Data Structures and Algorithm – Hands on (All Problems Finished)**

**Scenario:**

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

**Program:**

import java.util.\*;

class Products {

int pid;

String pname;

int qty;

double price;

public Products(int pid, String pname, int qty, double price) {

this.pid = pid;

this.pname = pname;

this.qty = qty;

this.price = price;

}

@Override

public String toString() {

return "[" + pid + "] " + pname + " | Qty: " + qty + " | Price: $" + price;

}

}

public class Main {

static HashMap<Integer, Products> ww = new HashMap<>();

public static void addProduct(Products p) {

if (ww.containsKey(p.pid)) {

System.out.println("Item already exists.");

} else {

ww.put(p.pid, p);

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

}

}

public static void updateProduct(int pid, String pname, int qty, double price) {

if (ww.containsKey(pid)) {

Products p = ww.get(pid);

p.pname = pname;

p.qty = qty;

p.price = price;

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

} else {

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

}

}

public static void deleteProduct(int pid) {

if (ww.containsKey(pid)) {

ww.remove(pid);

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

} else {

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

}

}

public static void display() {

if (ww.isEmpty()) {

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

} else {

System.out.println("The Product Warehouse contains:");

for (Products p : ww.values()) {

System.out.println(p);

}

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- Inventory Menu ---");

System.out.println("1. Add Product");

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

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

System.out.println("4. Display Products");

System.out.println("0. Exit");

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

choice = sc.nextInt();

switch (choice) {

case 1 -> {

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

int id = sc.nextInt();

sc.nextLine();

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

String name = sc.nextLine();

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

int qty = sc.nextInt();

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

double price = sc.nextDouble();

addProduct(new Products(id, name, qty, price));

}

case 2 -> {

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

int id = sc.nextInt();

sc.nextLine();

System.out.print("Enter new name: ");

String name = sc.nextLine();

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

int qty = sc.nextInt();

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

double price = sc.nextDouble();

updateProduct(id, name, qty, price);

}

case 3 -> {

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

int id = sc.nextInt();

deleteProduct(id);

}

case 4 -> display();

case 0 -> System.out.println("Exiting...");

default -> System.out.println("Invalid choice.");

}

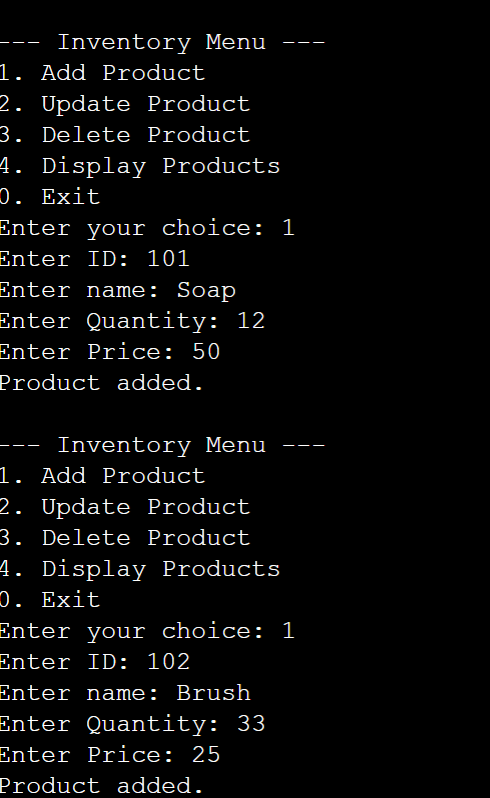
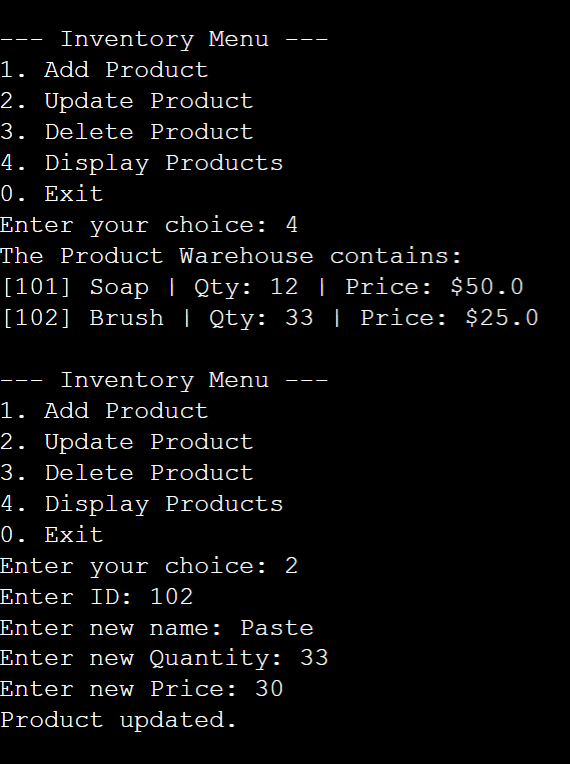
} while (choice != 0);

sc.close();

}

}

**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.

Program:

import java.util.\*;

class Product{

int pid;

String pname;

int qty;

double price;

public Product(int pid,String pname,int qty,double price){

this.pid=pid;

this.pname=pname;

this.qty=qty;

this.price=price;

}

@Override

public String toString(){

return "pid"+" Pname: "+pname+" Qty: "+qty+" Price "+ price;

}

}

public class Main

{

static HashMap<Integer, Product> ww= new HashMap<>();

public static void addProduct(Product p){

if(ww.containsKey(p.pid)){

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

}

else{

ww.put(p.pid,p);

System.out.print("Product Added");

}}

public static void LinearSearch(int id){

if(ww.containsKey(id)){

int count=0;

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

count++;

if(p.pid==id)

System.out.println("Element Found "+count);

break;

}}}}

public static void BinarySearch(int id){

if(ww.containsKey(id)){

int count=0;

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

count++;

if(p.pid==id){

System.out.println("Element Found "+count);

break;

} } } }

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

int choice;

System.out.println("1. Add the Elements");

System.out.println("2. Linear Serach");

System.out.println("3. Binary Search");

System.out.println("4. Exit");

do{

choice= sc.nextInt();

switch(choice){

case 1 ->{

System.out.println("Enter ID: "); int pid = sc.nextInt();

System.out.println("Enter name: "); String pname = sc.next();

System.out.println("Enter qty: "); int qty = sc.nextInt();

System.out.println("Enter Price: "); double price=sc.nextDouble();

addProduct(new Product(pid,pname,qty,price)); }

case 2->{

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

int pid=sc.nextInt();

LinearSearch(pid);}

case 3->{

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

int pid=sc.nextInt();

BinarySearch(pid);}

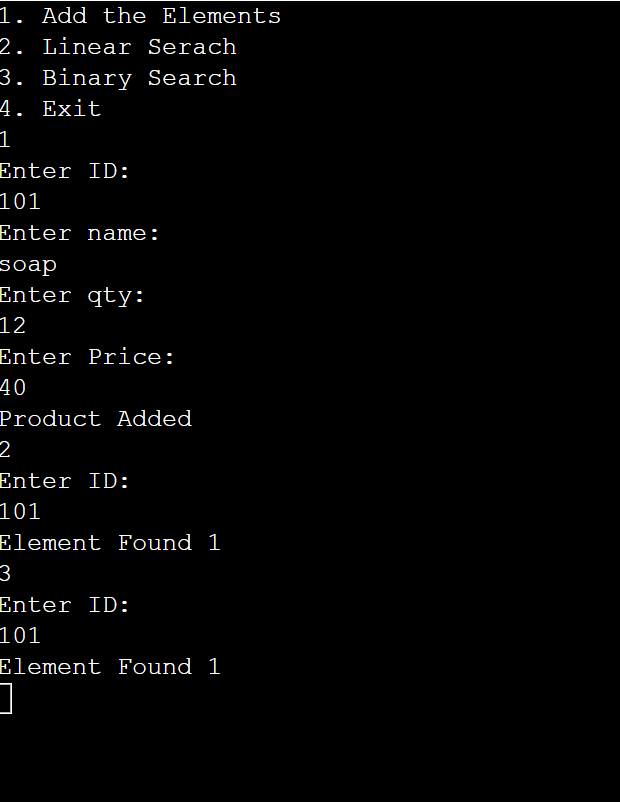
case 4->{

System.out.println("Existing");}}

}while(choice!=0);

sc.close();} }

**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**.**

**Program:**

import java.util.\*;

class Order {

int orderid;

String cname;

double price;

public Order(int orderid, String cname, double price) {

this.orderid = orderid;

this.cname = cname;

this.price = price;

}

@Override

public String toString() {

return orderid + " " + cname + " " + price;

}

}

public class Main {

public static void bubblesort(Order[] order) {

int n = order.length;

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

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

if (order[j].price < order[j + 1].price) {

Order temp = order[j];

order[j] = order[j + 1];

order[j + 1] = temp;

}

}

}

}

public static void quicksort(Order[] order, int low, int high) {

if (low < high) {

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

quicksort(order, low, pi - 1);

quicksort(order, pi + 1, high);

}

public static int partition(Order[] order, int low, int high) {

double pivot = order[high].price;

int i = low - 1;

for (int j = low; j < high; j++) {

if (order[j].price > pivot) {

i++;

Order temp = order[i];

order[i] = order[j];

order[j] = temp;} }

Order temp = order[i + 1];

order[i + 1] = order[high];

order[high] = temp;

return i + 1;

}

public static void display(Order[] order) {

for (Order o : order) {

System.out.println(o); } }

public static void main(String[] args) {

Order[] order = {

new Order(101, "AAA", 500.50),

new Order(102, "BBB", 300.00),

new Order(103, "CCC", 700.25),

new Order(104, "DDD", 150.75)

};

Order[] bubbleSorted = Arrays.copyOf(order, order.length);

bubblesort(bubbleSorted);

System.out.println("After Bubble Sort:");

display(bubbleSorted);

Order[] quickSorted = Arrays.copyOf(order, order.length);

quicksort(quickSorted, 0, quickSorted.length - 1);

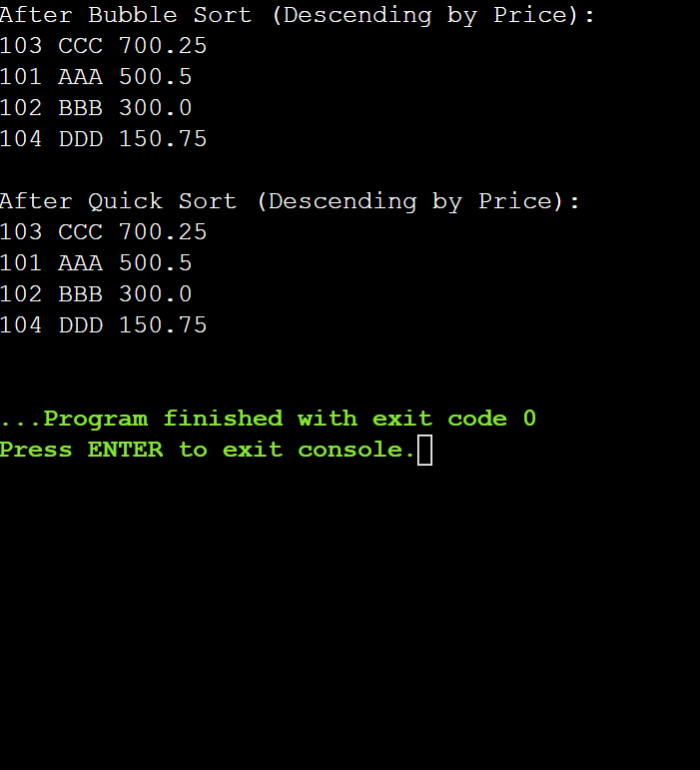
System.out.println("\nAfter Quick Sort:");

display(quickSorted);

}

}

**Output:**

****

**Exercise 4: Employee Management System**

**Scenario:**

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

**Program:**

import java.util.Scanner;

class Employee {

int employeeId;

String name;

String position;

double salary;

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

this.employeeId = employeeId;

this.name = name;

this.position = position;

this.salary = salary;

}

@Override

public String toString() {

return "ID: "+employeeId+", Name:"+name+",Position:" +position + ", Salary: " + salary;

}

}

public class Main {

static Employee[] employees = new Employee[100];

static int count = 0;

public static void addEmployee(Employee emp) {

if (count < employees.length) {

employees[count++] = emp;

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

} else {

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

}

}

public static void searchEmployee(int id) {

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

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

System.out.println("Employee Found: " + employees[i]);

return; } }

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

public static void traverseEmployees() {

if (count == 0) {

System.out.println("No employees in the system.");

} else {

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

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

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

public 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;

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

return;} }

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

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- Employee Management Menu ---");

System.out.println("1. Add Employee");

System.out.println("2. Search Employee");

System.out.println("3. Display All Employees");

System.out.println("4. Delete Employee");

System.out.println("0. Exit");

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

choice = sc.nextInt();

switch (choice) {

case 1 -> {

System.out.print("Enter Employee ID: "); int id = sc.nextInt();

System.out.print("Enter Name: "); String name = sc.next()’

System.out.print("Enter Position: "); String position = sc.nextLine();

System.out.print("Enter Salary: "); double salary = sc.nextDouble();

addEmployee(new Employee(id, name, position, salary)); }

case 2 -> {

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

int id = sc.nextInt();

searchEmployee(id); }

case 3 -> traverseEmployees();

case 4 -> {

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

int id = sc.nextInt();

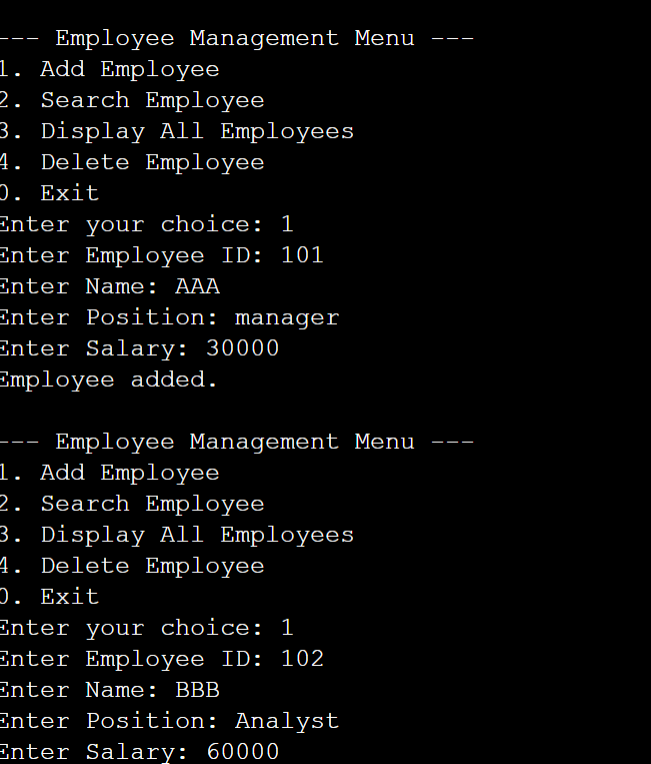
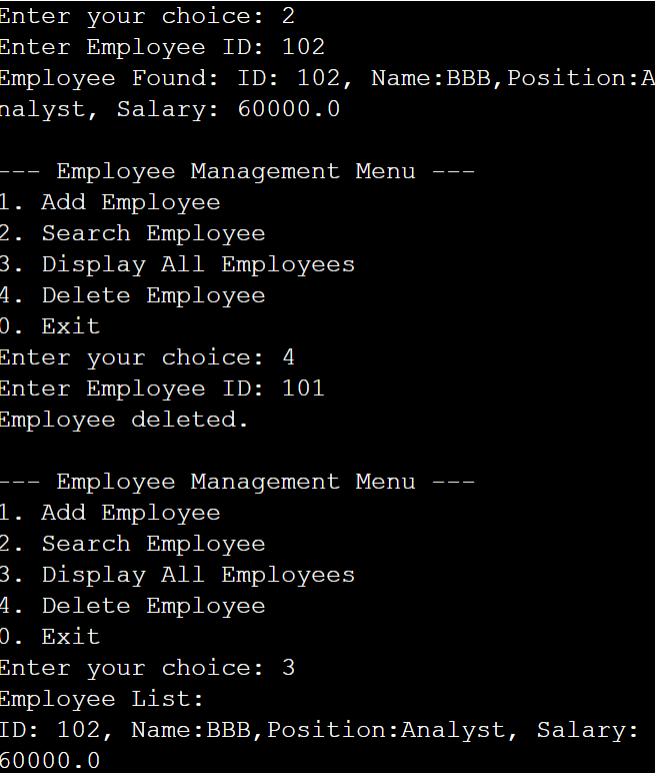
deleteEmployee(id);}

case 0 -> System.out.println("Exit");

default -> System.out.println("Invalid"); }

} while (choice != 0);} }

**output:**

**** ****

**Exercise 5: Task Management System**

**Scenario:**

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

**Program:**

import java.util.\*;

class Task {

int id;

String taskname;

String status;

public Task(int id,String taskname,String status) {

this.id =id;

this.status= status;

this.taskname =taskname; }

public String toString() {

return id+" "+taskname+" "+status;

} }

public class Main {

static LinkedList<Task> ww = new LinkedList<>();

public static boolean contains(int id) {

for(Task t:ww) {

if(t.id==id) return true;}

return false;}

public static void addd(Task t){

if(contains(t.id)){

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

}else{

ww.add(t);

System.out.println("Task Added"); } }

public static void removee(int id) {

Iterator<Task> it =ww.iterator();

while(it.hasNext()){

Task t = it.next();

if(t.id == id){

it.remove();

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

return; } }

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

public static void found(int id){

for(Task t : ww){

if(t.id == id){

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

return; } }

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

}

public static void display(){

if(ww.isEmpty()){

System.out.println("Linked List is Empty");

} else {

for(Task i:ww){

System.out.println(i); } } }

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

System.out.println("1. Add Elements");

System.out.println("2. Remove Elements");

System.out.println("3. Search Element");

System.out.println("4. Display");

System.out.println("5. Exit");

int choice;

do {

choice = sc.nextInt();

switch (choice) {

case 1 -> {

System.out.println("Enter Task ID: ");

int id = sc.nextInt();

System.out.println("Enter Task Name: "); String taskname = sc.nextLine();

System.out.println("Enter Status: "); String status = sc.nextLine();

addd(new Task(id, taskname, status)); ?

case 2 -> {

System.out.println("Enter Task ID: ");

int id =sc.nextInt();

removee(id);}

case 3 -> {

System.out.println("Enter Task ID: ");

int id =sc.nextInt();

found(id);}

case 4 -> { display();}

case 5 -> {

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

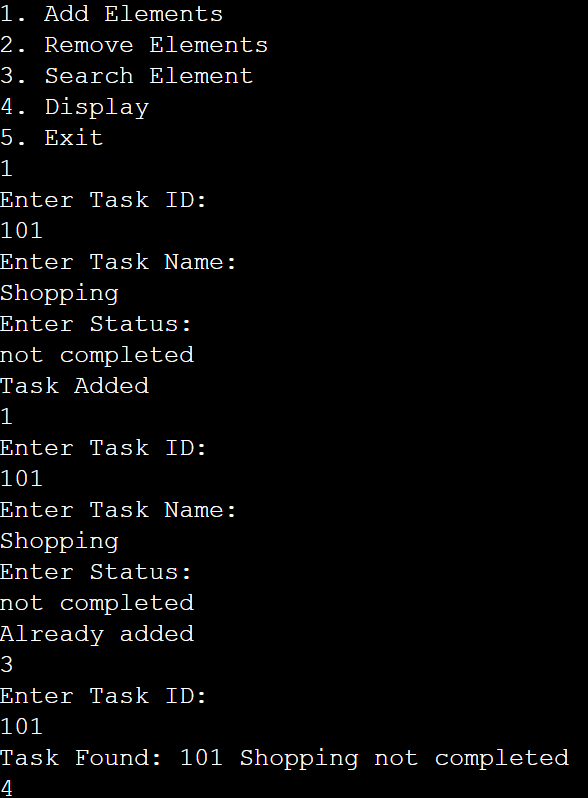
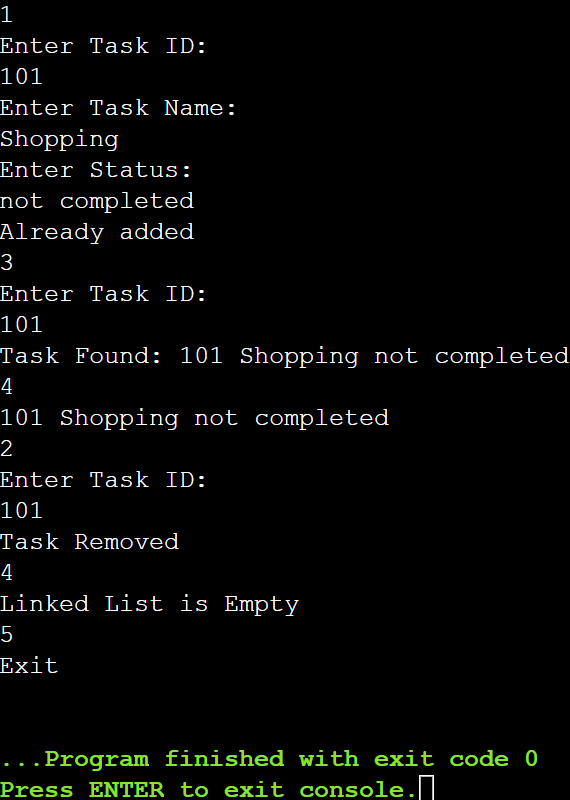
default -> {

System.out.println("Invalid choice. Try again."); } }

} while (choice != 5);

sc.close(); } }

**output:**

**** ****

**Exercise 6: Library Management System**

**Scenario:**

You are developing a library management system that allows users to search for books by title or author.

**Program:**

import java.util.\*;

class Book{

int bookId;

String title;

String author;

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

this.bookId = bookId;

this.title = title;

this.author = author;}

public String toString(){

return "ID: "+bookId+", Title: "+title+", Author: "+author; } }

public class Main{

static ArrayList<Book>books=new ArrayList<>();

public static void linearSearch(String title){

boolean found=false;

for (Book b :books){

if (b.title.equalsIgnoreCase(title)) {

System.out.println("Linear Search: " +b);

found = true; } }

if (!found){

System.out.println("Book not found"); } }

public static void binarySearch(String title) {

books.sort(Comparator.comparing(b->b.title.toLowerCase()));

int low = 0,high =books.size()-1;

while (low <=high){

int mid =(low+high)/2;

Book midBook=books.get(mid);

int compare=title.compareToIgnoreCase(midBook.title);

if (compare==0)

System.out.println("Binary Search: " + midBook);

return;

} else if(compare<0){

high=mid-1;

} else{

low =mid+1;} }

System.out.println("Book not found"); }

public static void displayBooks(){

if (books.isEmpty()){

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

return;}

System.out.println("Books in Library:");

for (Book b:books){

System.out.println(b);} }

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n=== Library Management System ===");

System.out.println("1. Add Book");

System.out.println("2. Linear Search");

System.out.println("3. Binary Search");

System.out.println("4. Display All Books");

System.out.println("5. Exit");

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

choice = sc.nextInt();

switch (choice) {

case 1 -> {

System.out.print("Enter Book ID: ");int id = sc.nextInt();

System.out.print("Enter Title: ");String title = sc.nextLine();

System.out.print("Enter Author: ");String author = sc.nextLine();

books.add(new Book(id, title, author));

System.out.println("Book added successfully."); }

case 2 -> {

System.out.print("Enter title to search: ");

String title = sc.nextLine();

linearSearch(title); }

case 3 -> {

System.out.print("Enter title to search: ");

String title = sc.nextLine();

binarySearch(title);}

case 4 -> displayBooks();

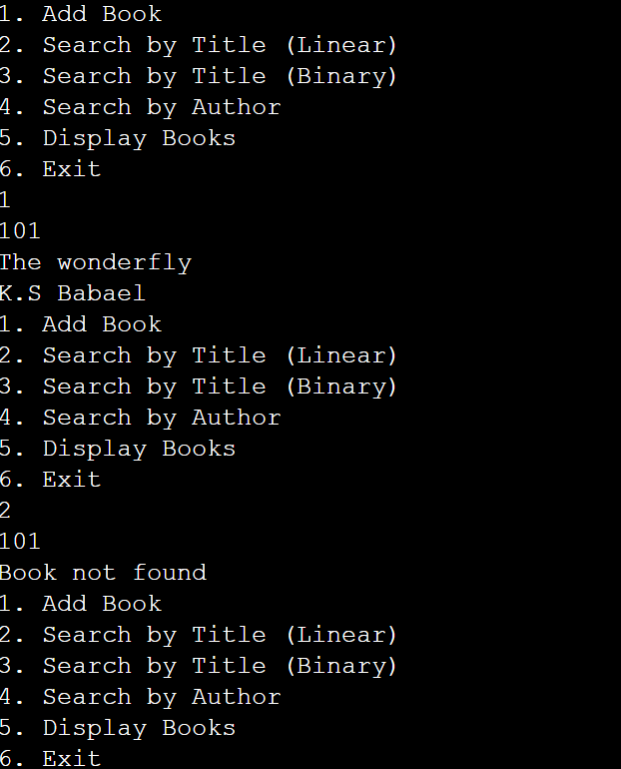
case 5 -> System.out.println("Exit");

default -> System.out.println("Invalid choice");}

} while (choice != 5);

sc.close(); } }

**output:**

****

**Exercise 7: Financial Forecasting**

**Scenario:**

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

**Program:**

public class Main{

public static double futureValue(double value,double rate,int years) {

if(years==0) return value;

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

}

public static void main(String[] args){

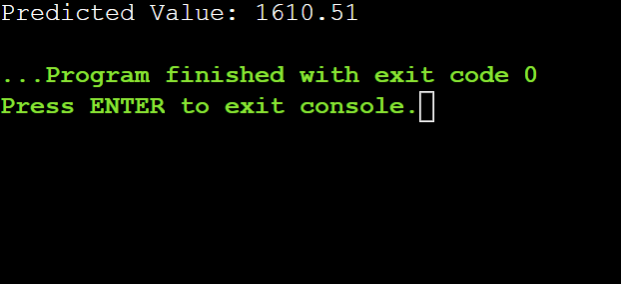
double result=futureValue(1000,0.1,5);

System.out.printf("Predicted Value: %.2f",result);

}

}

**Output:**

****