**Exercise 1: Inventory Management System**

**Source Code:**

package com.AlgorithmsAndDataStructures;

import java.util.\*;

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;

}

}

public class Main {

List<Product> inventory = new ArrayList<>();

Scanner sc = new Scanner(System.in);

public void run() {

while (true) {

System.out.print("Menu:\n 1: Add Product\n 2: Delete Product\n 3: Update Product\n 4: Show Inventory\n");

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

int choice = sc.nextInt();

switch (choice) {

case 1 -> addProduct();

case 2 -> deleteProduct();

case 3 -> updateProduct();

case 4 -> showInventory();

}

if (choice > 4 || choice < 1) {

break;

}

}

}

public void addProduct() {

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

int id = sc.nextInt();

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

String name = sc.next();

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

int qty = sc.nextInt();

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

double price = sc.nextDouble();

Product p = new Product(id, name, qty, price);

inventory.add(p);

}

public void deleteProduct() {

System.out.print("Enter Product ID to delete: ");

int id = sc.nextInt();

boolean removed = inventory.removeIf(product -> product.productId == id);

if (removed) {

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

} else {

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

}

}

public void updateProduct() {

System.out.print("Enter Product ID to update: ");

int id = sc.nextInt();

for (Product p : inventory) {

if (p.productId == id) {

System.out.println("Current name: " + p.productName);

System.out.print("Enter new name (press Enter to keep current): ");

sc.nextLine(); // consume newline

String name = sc.nextLine();

if (!name.isEmpty()) {

p.productName = name;

}

System.out.println("Current quantity: " + p.quantity);

System.out.print("Enter new quantity (-1 to keep current): ");

int quantity = sc.nextInt();

if (quantity >= 0) {

p.quantity = quantity;

}

System.out.println("Current price: " + p.price);

System.out.print("Enter new price (-1 to keep current): ");

double price = sc.nextDouble();

if (price >= 0) {

p.price = price;

}

}

}

}

public void showInventory() {

for (Product p : inventory) {

System.out.println("ID: " + p.productId);

System.out.println("Name: " + p.productName);

System.out.println("Quantity: " + p.quantity);

System.out.println("Price: " + p.price);

}

}

public static void main(String[] args) {

new Main().run();

}

}

**Output:**

Menu:

1: Add Product

2: Delete Product

3: Update Product

4: Show Inventory

Enter choice: 1

Enter ID: 1

Enter Name: laptop

Enter Quantity: 12

Enter Price: 45000

Menu:

1: Add Product

2: Delete Product

3: Update Product

4: Show Inventory

Enter choice: 1

Enter ID: 2

Enter Name: mobile

Enter Quantity: 23

Enter Price: 20000

Menu:

1: Add Product

2: Delete Product

3: Update Product

4: Show Inventory

Enter choice: 1

Enter ID: 3

Enter Name: watch

Enter Quantity: 40

Enter Price: 3000

Menu:

1: Add Product

2: Delete Product

3: Update Product

4: Show Inventory

Enter choice: 4

ID: 1

Name: laptop

Quantity: 12

Price: 45000.0

ID: 2

Name: mobile

Quantity: 23

Price: 20000.0

ID: 3

Name: watch

Quantity: 40

Price: 3000.0

Menu:

1: Add Product

2: Delete Product

3: Update Product

4: Show Inventory

Enter choice: 2

Enter Product ID to delete: 3

Product removed successfully

Menu:

1: Add Product

2: Delete Product

3: Update Product

4: Show Inventory

Enter choice: 4

ID: 1

Name: laptop

Quantity: 12

Price: 45000.0

ID: 2

Name: mobile

Quantity: 23

Price: 20000.0

Menu:

1: Add Product

2: Delete Product

3: Update Product

4: Show Inventory

Enter choice: 3

Enter Product ID to update: 2

Current name: mobile

Enter new name (press Enter to keep current): mobile 5G

Current quantity: 23

Enter new quantity (-1 to keep current): -1

Current price: 20000.0

Enter new price (-1 to keep current): -1

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

**Source Code:**

import java.util.\*;

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;

}

public Product() {}

public void display(){

System.out.println("ID: " + productId + ", Name: " + productName + ", Category: " + category);

}

}

public class ECommerceSearching {

List<Product> products = new ArrayList<>();

public static Product binarySearch(List<Product> products, String name) {

Collections.sort(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName));

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

while (low <= high) {

int mid = (low + high) / 2;

Product curr = products.get(mid);

int cmp = curr.productName.compareToIgnoreCase(name);

if (cmp == 0) return curr;

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

else high = mid - 1;

}

return null;

}

public Product linearSearch(String name) {

for (Product p : products) {

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

return p;

}

}

return null;

}

public void addSampleProducts() {

products.add(new Product(1, "Laptop", "Electronics"));

products.add(new Product(2, "Chair", "Furniture"));

products.add(new Product(3, "Pen", "Stationery"));

products.add(new Product(4, "Shampoo", "Personal Care"));

}

public void displayProducts() {

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

for (Product p : products) {

p.display();

}

}

public static void main(String[] args) {

ECommerceSearching e = new ECommerceSearching();

e.addSampleProducts();

e.displayProducts();

Scanner sc = new Scanner(System.in);

System.out.print("\nEnter product name to search: ");

String name = sc.nextLine();

Product result = e.linearSearch(name); // or use binarySearch(e.products, name)

if (result != null) {

System.out.println("Product found:");

result.display();

} else {

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

}

sc.close();

}

}

**Output:**

Product List:

ID: 1, Name: Laptop, Category: Electronics

ID: 2, Name: Chair, Category: Furniture

ID: 3, Name: Pen, Category: Stationery

ID: 4, Name: Shampoo, Category: Personal Care

Enter product name to search: laptop

Product found:

ID: 1, Name: Laptop, Category: Electronics

**Exercise 3: Sorting Customer Orders**

Source Code:

import java.util.\*;

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;

}

public Order() {}

public void display() {

System.out.println("Order ID: " + orderId + ", Customer: " + customerName + ", Total: ₹" + totalPrice);

}

}

public class ECommerceOrderSorting {

List<Order> orders = new ArrayList<>();

public void addSampleOrders() {

orders.add(new Order(101, "Rahul", 750.0));

orders.add(new Order(102, "Sneha", 1200.5));

orders.add(new Order(103, "Amit", 560.25));

orders.add(new Order(104, "Priya", 980.75));

}

public void displayOrders() {

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

for (Order o : orders) {

o.display();

}

}

public void bubbleSortOrders() {

int n = orders.size();

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

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

if (orders.get(j).totalPrice > orders.get(j + 1).totalPrice) {

Collections.swap(orders, j, j + 1);

}

}

}

}

public void quickSortOrders(int low, int high) {

if (low < high) {

int pi = partition(low, high);

quickSortOrders(low, pi - 1);

quickSortOrders(pi + 1, high);

}

}

private int partition(int low, int high) {

double pivot = orders.get(high).totalPrice;

int i = low - 1;

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

if (orders.get(j).totalPrice <= pivot) {

i++;

Collections.swap(orders, i, j);

}

}

Collections.swap(orders, i + 1, high);

return i + 1;

}

public static void main(String[] args) {

ECommerceOrderSorting ecom = new ECommerceOrderSorting();

ecom.addSampleOrders();

Scanner sc = new Scanner(System.in);

System.out.println("Choose sorting method:\n1. Bubble Sort\n2. Quick Sort");

int choice = sc.nextInt();

switch (choice) {

case 1:

ecom.bubbleSortOrders();

System.out.println("\nSorted using Bubble Sort:");

break;

case 2:

ecom.quickSortOrders(0, ecom.orders.size() - 1);

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

break;

default:

System.out.println("Invalid choice.");

return;

}

ecom.displayOrders();

sc.close();

}

}

Output:

Choose sorting method:

1. Bubble Sort

2. Quick Sort

1

Sorted using Bubble Sort:

Order List:

Order ID: 103, Customer: Amit, Total: ₹560.25

Order ID: 101, Customer: Rahul, Total: ₹750.0

Order ID: 104, Customer: Priya, Total: ₹980.75

Order ID: 102, Customer: Sneha, Total: ₹1200.5

Choose sorting method:

1. Bubble Sort

2. Quick Sort

2

Sorted using Quick Sort:

Order List:

Order ID: 103, Customer: Amit, Total: ₹560.25

Order ID: 101, Customer: Rahul, Total: ₹750.0

Order ID: 104, Customer: Priya, Total: ₹980.75

Order ID: 102, Customer: Sneha, Total: ₹1200.5

**Exercise 4: Employee Management System**

**Source code:**

import java.util.\*;

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;

}

public void display() {

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

}

}

public class EmployeeManager {

Employee[] employees = new Employee[100];

int count = 0;

Scanner sc = new Scanner(System.in);

public void addEmployee() {

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

int id = sc.nextInt();

sc.nextLine(); // consume leftover newline

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

String name = sc.nextLine();

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));

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

}

public void addEmployee(Employee emp) {

if (count < employees.length)

employees[count++] = emp;

}

public void searchEmployee() {

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

int id = sc.nextInt();

Employee e = searchEmployee(id);

if (e != null) {

e.display();

} else {

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

}

}

public Employee searchEmployee(int id) {

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

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

return employees[i];

return null;

}

public void deleteEmployee() {

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

int id = sc.nextInt();

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];

count--;

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

return;

}

}

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

}

public void displayEmployees() {

if (count == 0) {

System.out.println("No employees to show.");

} else {

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

employees[i].display();

}

}

public void run() {

while (true) {

System.out.println("\nMenu:");

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

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

System.out.println("3: Delete Employee");

System.out.println("4: Display All Employees");

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

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

int choice = sc.nextInt();

switch (choice) {

case 1 -> addEmployee();

case 2 -> searchEmployee();

case 3 -> deleteEmployee();

case 4 -> displayEmployees();

case 5 -> {

System.out.println("Exiting...");

return;

}

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

}

}

}

public static void main(String[] args) {

new EmployeeManager().run();

}

}

**Output:**

Menu:

1: Add Employee

2: Search Employee

3: Delete Employee

4: Display All Employees

5: Exit

Enter your choice:

1

Enter ID: 1

Enter Name: jaswanth

Enter Position: sde

Enter Salary: 4000000

Employee added.

Menu:

1: Add Employee

2: Search Employee

3: Delete Employee

4: Display All Employees

5: Exit

Enter your choice: 1

Enter ID: 2

Enter Name: siva sai

Enter Position: genc

Enter Salary: 400000

Employee added.

Menu:

1: Add Employee

2: Search Employee

3: Delete Employee

4: Display All Employees

5: Exit

Enter your choice: 2

Enter ID to search: 1

1 | jaswanth | sde | ₹4000000.0

Menu:

1: Add Employee

2: Search Employee

3: Delete Employee

4: Display All Employees

5: Exit

Enter your choice: 4

1 | jaswanth | sde | ₹4000000.0

2 | siva sai | genc | ₹400000.0

Menu:

1: Add Employee

2: Search Employee

3: Delete Employee

4: Display All Employees

5: Exit

Enter your choice: 3

Enter ID to delete: 2

Employee deleted.

Menu:

1: Add Employee

2: Search Employee

3: Delete Employee

4: Display All Employees

5: Exit

Enter your choice: 4

1 | jaswanth | sde | ₹4000000.0

**Exercise 5: Task Management System**

**Source Code:**

**import java.util.Scanner;**

class Task {

int taskId;

String taskName;

String status;

Task next;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

}

public void display() {

System.out.println(taskId + " - " + taskName + " [" + status + "]");

}

}

public class TaskManager {

Task head;

public void addTask(Task task) {

task.next = head;

head = task;

}

public Task searchTask(int id) {

Task curr = head;

while (curr != null) {

if (curr.taskId == id)

return curr;

curr = curr.next;

}

return null;

}

public void updateTask(int id, String newName, String newStatus) {

Task t = searchTask(id);

if (t != null) {

t.taskName = newName;

t.status = newStatus;

} else {

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

}

}

public void deleteTask(int id) {

if (head == null)

return;

if (head.taskId == id) {

head = head.next;

return;

}

Task curr = head;

while (curr.next != null) {

if (curr.next.taskId == id) {

curr.next = curr.next.next;

return;

}

curr = curr.next;

}

}

public void displayTasks() {

Task curr = head;

if (curr == null) {

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

return;

}

while (curr != null) {

curr.display();

curr = curr.next;

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

TaskManager manager = new TaskManager();

int choice;

do {

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

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

System.out.println("2. Delete Task");

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

System.out.println("4. Update Task");

System.out.println("5. Display All Tasks");

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

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

choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

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

int id = sc.nextInt();

sc.nextLine();

System.out.print("Enter Task Name: ");

String name = sc.nextLine();

System.out.print("Enter Task Status: ");

String status = sc.nextLine();

manager.addTask(new Task(id, name, status));

break;

case 2:

System.out.print("Enter Task ID to delete: ");

int delId = sc.nextInt();

manager.deleteTask(delId);

break;

case 3:

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

int searchId = sc.nextInt();

Task found = manager.searchTask(searchId);

if (found != null)

found.display();

else

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

break;

case 4:

System.out.print("Enter Task ID to update: ");

int updateId = sc.nextInt();

sc.nextLine();

Task existingTask = manager.searchTask(updateId);

if (existingTask != null) {

System.out.print("Enter new name (leave blank to keep current): ");

String newNameInput = sc.nextLine();

String newName = newNameInput.isEmpty() ? existingTask.taskName : newNameInput;

System.out.print("Enter new status (leave blank to keep current): ");

String newStatusInput = sc.nextLine();

String newStatus = newStatusInput.isEmpty() ? existingTask.status : newStatusInput;

manager.updateTask(updateId, newName, newStatus);

} else {

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

}

break;

case 5:

manager.displayTasks();

break;

case 0:

System.out.println("Exiting... Goodbye!");

break;

default:

System.out.println("Invalid choice.");

}

} while (choice != 0);

sc.close();

}

}

**Output:**

=== Task Manager ===

1. Add Task

2. Delete Task

3. Search Task

4. Update Task

5. Display All Tasks

0. Exit

Enter choice: 1

Enter Task ID: 1

Enter Task Name: code review

Enter Task Status: in progress

=== Task Manager ===

1. Add Task

2. Delete Task

3. Search Task

4. Update Task

5. Display All Tasks

0. Exit

Enter choice: 1

Enter Task ID: 11

Enter Task Name: debugging

Enter Task Status: yet to start

=== Task Manager ===

1. Add Task

2. Delete Task

3. Search Task

4. Update Task

5. Display All Tasks

0. Exit

Enter choice: 2

Enter Task ID to delete: 11

=== Task Manager ===

1. Add Task

2. Delete Task

3. Search Task

4. Update Task

5. Display All Tasks

0. Exit

Enter choice: 5

1 - code review [in progress]

=== Task Manager ===

1. Add Task

2. Delete Task

3. Search Task

4. Update Task

5. Display All Tasks

0. Exit

Enter choice: 4

Enter Task ID to update: 1

Enter new name (leave blank to keep current):

Enter new status (leave blank to keep current): completed

=== Task Manager ===

1. Add Task

2. Delete Task

3. Search Task

4. Update Task

5. Display All Tasks

0. Exit

Enter choice: 5

1 - code review [completed]

=== Task Manager ===

1. Add Task

2. Delete Task

3. Search Task

4. Update Task

5. Display All Tasks

0. Exit

Enter choice: 0

Exiting... Goodbye!

**Exercise 6: Library Management System**

**Source Code:**

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 void display() {

System.out.println(bookId + " | " + title + " | " + author);

}

}

public class LibraryManager {

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

public void addBooks() {

books.add(new Book(1, "Algorithms", "Cormen"));

books.add(new Book(2, "Design Patterns", "GoF"));

books.add(new Book(3, "Clean Code", "Robert Martin"));

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

}

public void displayBooks(){

for(Book book:books){

book.display();

}

}

public Book linearSearch(String title) {

for (Book b : books)

if (b.title.equalsIgnoreCase(title))

return b;

return null;

}

public Book binarySearch(String title) {

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

while (low <= high) {

int mid = (low + high) / 2;

int cmp = books.get(mid).title.compareToIgnoreCase(title);

if (cmp == 0)

return books.get(mid);

else if (cmp < 0)

low = mid + 1;

else

high = mid - 1;

}

return null;

}

public static void main(String[] args) {

LibraryManager l = new LibraryManager();

l.addBooks();

l.displayBooks();

Scanner sc=new Scanner(System.in);

System.out.println("Enter book title:");

Book b = l.binarySearch(sc.nextLine());

if (b != null)

b.display();

}

}

**Output:**

1 | Algorithms | Cormen

3 | Clean Code | Robert Martin

2 | Design Patterns | GoF

Enter book title:

Design Patterns

2 | Design Patterns | GoF

**Exercise 7: Financial Forecasting**

**Source Code:**

import java.util.\*;

public class Forecasting {

public static double predictFutureValue(int years, double amount, double rate) {

if (years == 0)

return amount;

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

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int tenure;

double amount;

double rate;

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

tenure=sc.nextInt();

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

amount=sc.nextDouble();

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

rate=sc.nextDouble();

double result = predictFutureValue(tenure, amount, rate);

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

}

}

**Output:**

Enter tenure:

5

Enter amount :

10000

Enter rate :

0.08

Forecasted Value: ₹14693.28