**Exercise 1: Inventory Management System**

**Inventory.java**

package inventory;

import java.util.HashMap;

public class Inventory {

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

public void addProduct(Product product) {

products.put(product.productId, product);

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

}

public void updateProduct(int id, int newQty, double newPrice) {

Product p = products.get(id);

if (p != null) {

p.quantity = newQty;

p.price = newPrice;

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

} else {

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

}

}

public void deleteProduct(int id) {

if (products.containsKey(id)) {

products.remove(id);

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

} else {

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

}

}

public void displayAll() {

if (products.isEmpty()) {

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

} else {

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

System.out.println(p);

}

}

}

}

**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;

}

public String toString() {

return "Product[ID=" + productId + ", Name=" + productName + ", Qty=" + quantity + ", Price=" + price + "]";

}

}

**Main.java**

package inventory;

public class Main {

public static void main(String[] args) {

Inventory inv = new Inventory();

Product p1 = new Product(101, "Keyboard", 50, 499.99);

Product p2 = new Product(102, "Mouse", 70, 299.99);

Product p3 = new Product(103, "Monitor", 20, 9999.99);

inv.addProduct(p1);

inv.addProduct(p2);

inv.addProduct(p3);

System.out.println("Initial Inventory:");

inv.displayAll();

inv.updateProduct(102, 100, 249.99);

inv.deleteProduct(103);

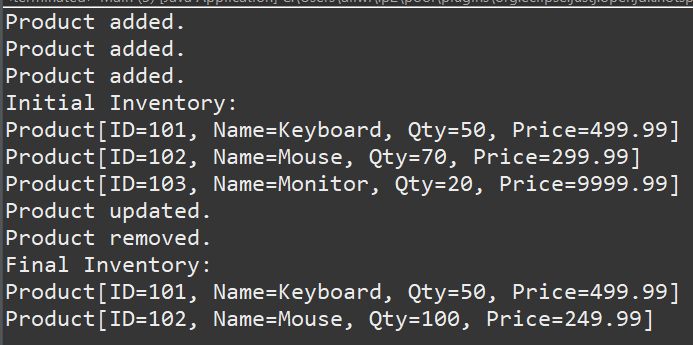
System.out.println("Final Inventory:");

inv.displayAll();

}

}

**Output**



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

**Product.java**

package ecommerce;

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;

}

public String toString() {

return "Product[ID=" + productId + ", Name=" + productName + ", Category=" + category + "]";

}

}

**SearchDemo.java**

package ecommerce;

import java.util.Arrays;

import java.util.Comparator;

public class SearchDemo {

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

for (Product product : products) {

if (product.productName.equalsIgnoreCase(targetName)) {

return product;

}

}

return null;

}

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

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

while (left <= right) {

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

int cmp = products[mid].productName.compareToIgnoreCase(targetName);

if (cmp == 0)

return products[mid];

else if (cmp < 0)

left = mid + 1;

else

right = mid - 1;

}

return null;

}

public static void main(String[] args) {

Product[] products = {

new Product(101, "Laptop", "Electronics"),

new Product(102, "Shoes", "Fashion"),

new Product(103, "Mobile", "Electronics"),

new Product(104, "Watch", "Accessories"),

new Product(105, "Bag", "Fashion")

};

Product foundLinear = linearSearch(products, "Watch");

System.out.println("Linear Search Result: " + (foundLinear != null ? foundLinear : "Not Found"));

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

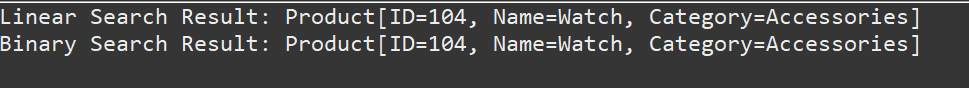
Product foundBinary = binarySearch(products, "Watch");

System.out.println("Binary Search Result: " + (foundBinary != null ? foundBinary : "Not Found"));

}

}

**Output**



**Exercise 3: Sorting Customer Orders**

**Main.java**

package ordersort;

public class Main {

public static void main(String[] args) {

Order[] orders = {

new Order(101, "Jona", 2500.00),

new Order(102, "Priya", 1500.00),

new Order(103, "samhita", 5500.00),

new Order(104, "shifa", 3000.00)

};

System.out.println("Original Orders:");

Sorter.printOrders(orders);

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

Sorter.bubbleSort(orders);

Sorter.printOrders(orders);

orders = new Order[]{

new Order(101, "Jona", 2500.00),

new Order(102, "Priya", 1500.00),

new Order(103, "samhita", 5500.00),

new Order(104, "shifa", 3000.00)

};

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

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

Sorter.printOrders(orders);

}

}

**Order.java**

package ordersort;

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;

}

public String toString() {

return "Order[ID=" + orderId + ", Name=" + customerName + ", Total=₹" + totalPrice + "]";

}

}

**Sorter.java**

package ordersort;

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;

}

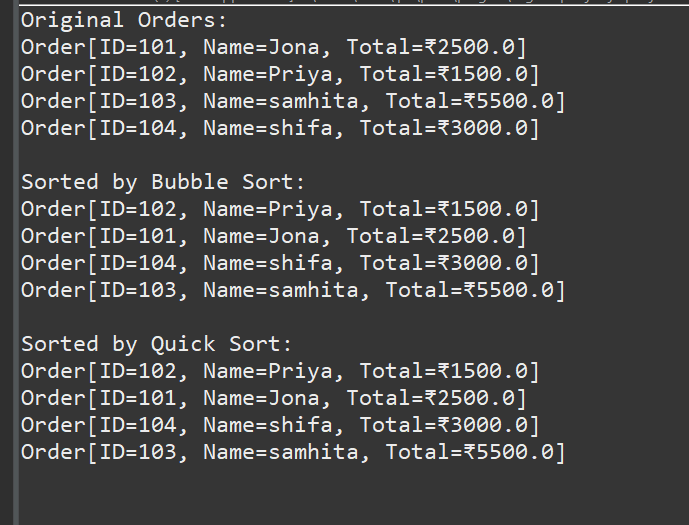
public String toString() {

return "Order[ID=" + orderId + ", Name=" + customerName + ", Total=₹" + totalPrice + "]";

}

}

**Output**



**Exercise 4: Employee Management System**

**Employee.java**

package employee;

public 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 String toString() {

return "Employee[ID=" + employeeId + ", Name=" + name + ", Position=" + position + ", Salary=₹" + salary + "]";

}

}

**EmployeeManager.java**

package employee;

public class EmployeeManager {

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

private int count = 0;

public 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 Employee searchEmployee(int id) {

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

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

return employees[i];

}

}

return null;

}

public 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 void displayAllEmployees() {

if (count == 0) {

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

return;

}

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

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

}

}

}

**Main.java**

package employee;

public class Main {

public static void main(String[] args) {

EmployeeManager manager = new EmployeeManager();

Employee e1 = new Employee(101, "Jona", "Manager", 75000);

Employee e2 = new Employee(102, "Priya", "Developer", 55000);

Employee e3 = new Employee(103, "Suba", "HR", 48000);

manager.addEmployee(e1);

manager.addEmployee(e2);

manager.addEmployee(e3);

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

manager.displayAllEmployees();

System.out.println("\nSearching for employee with ID 102:");

Employee found = manager.searchEmployee(102);

System.out.println(found != null ? found : "Not found");

System.out.println("\nDeleting employee with ID 101:");

manager.deleteEmployee(101);

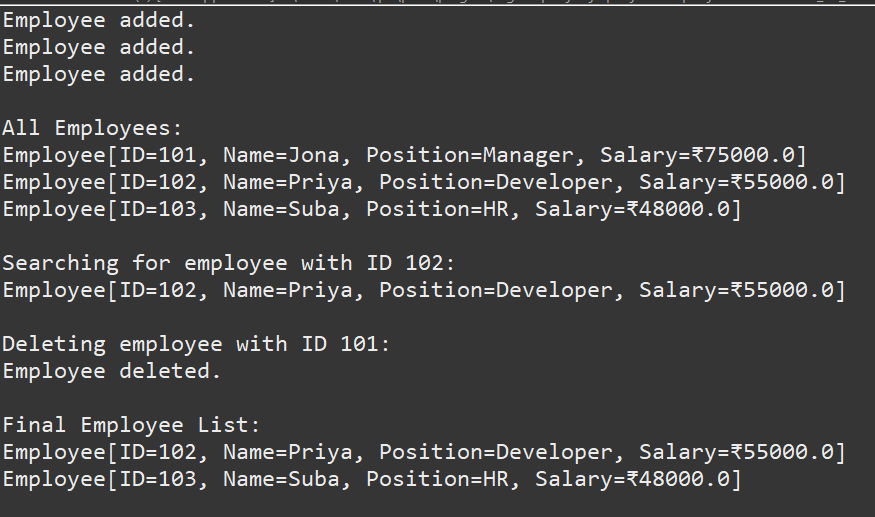
System.out.println("\nFinal Employee List:");

manager.displayAllEmployees();

}

}

**Output**



**Exercise 5: Task Management System**

**Task.java**

package taskmanager;

public class Task {

int taskId;

String taskName;

String status;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

}

public String toString() {

return "Task[ID=" + taskId + ", Name=" + taskName + ", Status=" + status + "]";

}

}

**TaskManager.java**

package taskmanager;

public class TaskManager {

private TaskNode head = null;

public void addTask(Task task) {

TaskNode newNode = new TaskNode(task);

if (head == null) {

head = newNode;

} else {

TaskNode current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

System.out.println("Task added: " + task.taskName);

}

public Task searchTask(int id) {

TaskNode current = head;

while (current != null) {

if (current.task.taskId == id) {

return current.task;

}

current = current.next;

}

return null;

}

public void deleteTask(int id) {

if (head == null) return;

if (head.task.taskId == id) {

head = head.next;

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

return;

}

TaskNode current = head;

while (current.next != null && current.next.task.taskId != id) {

current = current.next;

}

if (current.next != null) {

current.next = current.next.next;

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

} else {

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

}

}

public void displayTasks() {

if (head == null) {

System.out.println("No tasks available.");

return;

}

TaskNode current = head;

while (current != null) {

System.out.println(current.task);

current = current.next;

}

}

}

**TaskNode.java**

package taskmanager;

class TaskNode {

Task task;

TaskNode next;

public TaskNode(Task task) {

this.task = task;

this.next = null;

}

}

**Main.java**

package taskmanager;

public class Main {

public static void main(String[] args) {

TaskManager manager = new TaskManager();

Task t1 = new Task(1, "Design Module", "Pending");

Task t2 = new Task(2, "Write Tests", "Pending");

Task t3 = new Task(3, "Deploy to Production", "Completed");

manager.addTask(t1);

manager.addTask(t2);

manager.addTask(t3);

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

manager.displayTasks();

System.out.println("\nSearching for Task with ID 2:");

Task found = manager.searchTask(2);

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

System.out.println("\nDeleting Task with ID 1:");

manager.deleteTask(1);

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

manager.displayTasks();

}

}

**Exercise 6: Library Management System**

**Book.java**

package library;

public 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 "Book[ID=" + bookId + ", Title=" + title + ", Author=" + author + "]";

}

}

**LibrarySearch.java**

package library;

import java.util.Arrays;

import java.util.Comparator;

public class LibrarySearch {

public static Book linearSearch(Book[] books, String targetTitle) {

for (Book book : books) {

if (book.title.equalsIgnoreCase(targetTitle)) {

return book;

}

}

return null;

}

public static Book binarySearch(Book[] books, String targetTitle) {

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

while (left <= right) {

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

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

if (cmp == 0)

return books[mid];

else if (cmp < 0)

left = mid + 1;

else

right = mid - 1;

}

return null;

}

public static void sortBooksByTitle(Book[] books) {

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

}

public static void printBooks(Book[] books) {

for (Book b : books) {

System.out.println(b);

}

}

}

**Main.java**

package library;

public class Main {

public static void main(String[] args) {

Book[] books = {

new Book(101, "Java Programming", "James Gosling"),

new Book(102, "Python Basics", "Guido van Rossum"),

new Book(103, "Data Structures", "Narasimha Karumanchi"),

new Book(104, "C++ Primer", "Stanley Lippman")

};

System.out.println("All Books:");

LibrarySearch.printBooks(books);

System.out.println("\nLinear Search for 'Python Basics':");

Book linear = LibrarySearch.linearSearch(books, "Python Basics");

System.out.println(linear != null ? linear : "Not found");

System.out.println("\n Sorting books by title");

LibrarySearch.sortBooksByTitle(books);

System.out.println("\nBinary Search for 'Python Basics':");

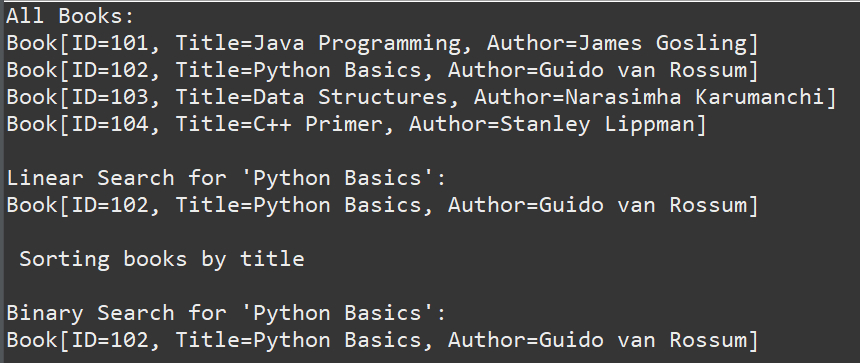
Book binary = LibrarySearch.binarySearch(books, "Python Basics");

System.out.println(binary != null ? binary : "Not found");

}

}

**Output**



**Exercise 7: Financial Forecasting**

package forecast;

public class FinancialForecast {

public static double forecast(double currentValue, double rate, int years) {

if (years == 0) {

return currentValue;

}

return forecast(currentValue, rate, years - 1) \* (1 + rate);

}

public static void main(String[] args) {

double currentValue = 10000;

double growthRate = 0.10;

int forecastYears = 5;

double futureValue = forecast(currentValue, growthRate, forecastYears);

System.out.printf("Predicted value after %d years: ₹%.2f\n", forecastYears, futureValue);

}

}

**Output**

