**Exercise 1: Inventory Management System**

**Code:  
=>** Product.java

**package** exno\_1;

**public** **class** Product {

**private** **int** productId;

**private** String productName;

**private** **int** quantity;

**private** **double** price; // price per unit

**public** Product(**int** productId, String productName, **int** quantity, **double** price) {

**this**.productId = productId;

**this**.productName = productName;

**this**.quantity = quantity;

**this**.price = price;

}

// Getters and setters

**public** **int** getProductId() {

**return** productId;

}

**public** String getProductName() {

**return** productName;

}

**public** **int** getQuantity() {

**return** quantity;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setQuantity(**int** quantity) {

**this**.quantity = quantity;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

**public** **double** getTotalCost() {

**return** **this**.quantity \* **this**.price;

}

@Override

**public** String toString() {

**return** "[" + productId + "] " + productName + " - Qty: " + quantity +

" - ₹" + price + " per unit - Total: ₹" + String.*format*("%.2f", getTotalCost());

}

}

**=>** Inventory.java

**package** exno\_1;

**import** java.util.HashMap;

**public** **class** Inventory {

**private** HashMap<Integer, Product> products;

**private** **static** Inventory *instance*;

**private** Inventory() {

products = **new** HashMap<>();

}

**public** **static** Inventory getInstance() {

**if** (*instance* == **null**) {

*instance* = **new** Inventory();

}

**return** *instance*;

}

**public** **void** addProduct(Product product) {

products.put(product.getProductId(), product);

}

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

Product product = products.get(id);

**if** (product != **null**) {

product.setQuantity(newQty);

product.setPrice(newPrice);

}

}

**public** **void** deleteProduct(**int** id) {

products.remove(id);

}

**public** **void** displayInventory() {

**if** (products.isEmpty()) {

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

**return**;

}

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

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

}

}

}

**=>** ProductFactory.java  
**package** exno\_1;

**public** **class** ProductFactory {

**public** **static** Product createProduct(**int** id, String name, **int** quantity, **double** price) {

**return** **new** Product(id, name, quantity, price);

}

}

**=>** Main.java  
**package** exno\_1;

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

Inventory inventory = Inventory.*getInstance*();

**int** choice;

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

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

System.***out***.println("2. Display Inventory");

System.***out***.println("3. Update Product");

System.***out***.println("4. Delete Product");

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

**do** {

System.***out***.print("\nEnter your choice: ");

choice = sc.nextInt();

**switch** (choice) {

**case** 1:

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

**int** id = sc.nextInt();

sc.nextLine(); // Consume newline

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

String name = sc.nextLine();

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

**int** quantity = sc.nextInt();

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

**double** price = sc.nextDouble();

Product p = ProductFactory.*createProduct*(id, name, quantity, price);

inventory.addProduct(p);

System.***out***.println("Product added successfully!");

**break**;

**case** 2:

inventory.displayInventory();

**break**;

**case** 3:

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

**int** updateId = sc.nextInt();

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

**int** newQty = sc.nextInt();

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

**double** newPrice = sc.nextDouble();

inventory.updateProduct(updateId, newQty, newPrice);

**break**;

**case** 4:

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

**int** deleteId = sc.nextInt();

inventory.deleteProduct(deleteId);

**break**;

**case** 5:

System.***out***.println("Exiting... Thank you!");

**break**;

**default**:

System.***out***.println("Invalid choice! Please try again.");

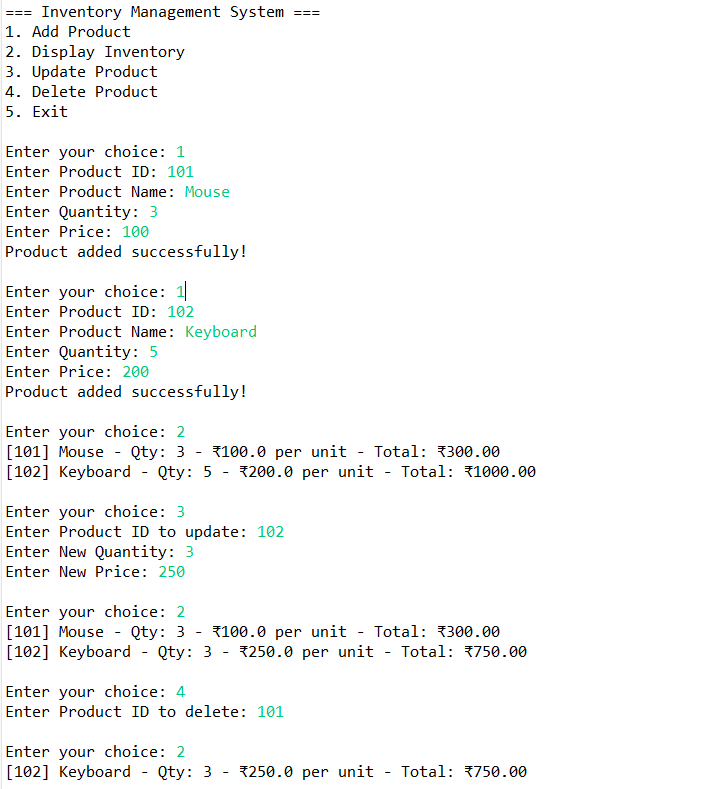
}

} **while** (choice != 5);

sc.close();

}

}

**Output:  
**

****

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

### **Code:**

=> Product.java

**package** exno\_2;

**public** **class** Product **implements** Comparable<Product> {

**private** **int** productId;

**private** String productName;

**private** String category;

**public** Product(**int** productId, String productName, String category) {

**this**.productId = productId;

**this**.productName = productName;

**this**.category = category;

}

**public** **int** getProductId() {

**return** productId;

}

**public** String getProductName() {

**return** productName;

}

**public** String getCategory() {

**return** category;

}

@Override

**public** **int** compareTo(Product other) {

**return** **this**.productId - other.productId;

}

@Override

**public** String toString() {

**return** "[" + productId + "] " + productName + " - " + category;

}

}

=> SearchEngine.java

**package** exno\_2;

**public** **class** SearchEngine {

// Linear Search

**public** **static** Product linearSearch(Product[] products, **int** id) {

**for** (Product product : products) {

**if** (product.getProductId() == id) {

**return** product;

}

}

**return** **null**;

}

// Binary Search (products must be sorted by productId)

**public** **static** Product binarySearch(Product[] products, **int** id) {

**int** left = 0;

**int** right = products.length - 1;

**while** (left <= right) {

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

**int** midId = products[mid].getProductId();

**if** (midId == id) **return** products[mid];

**else** **if** (id < midId) right = mid - 1;

**else** left = mid + 1;

}

**return** **null**;

}

}

=> Main.java

**package** exno\_2;

**import** java.util.Arrays;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Product[] products = {

**new** Product(102, "Phone", "Electronics"),

**new** Product(101, "Shirt", "Clothing"),

**new** Product(103, "Laptop", "Electronics"),

**new** Product(104, "Book", "Stationery")

};

System.***out***.println("Linear Search:");

Product result1 = SearchEngine.*linearSearch*(products, 103);

System.***out***.println(result1 != **null** ? result1 : "Product not found");

// Sort for binary search

Arrays.*sort*(products);

System.***out***.println("\nBinary Search:");

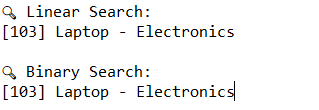
Product result2 = SearchEngine.*binarySearch*(products, 103);

System.***out***.println(result2 != **null** ? result2 : "Product not found");

}

}

**Output:**



## **Exercise 3: Sorting Customer Orders**

**Code:**

**=>** Order.java

**package** exno\_3;

**public** **class** Order {

**private** **int** orderId;

**private** String customerName;

**private** **double** totalPrice;

**public** Order(**int** orderId, String customerName, **double** totalPrice) {

**this**.orderId = orderId;

**this**.customerName = customerName;

**this**.totalPrice = totalPrice;

}

**public** **double** getTotalPrice() {

**return** totalPrice;

}

**public** **int** getOrderId() {

**return** orderId;

}

**public** String getCustomerName() {

**return** customerName;

}

@Override

**public** String toString() {

**return** "[" + orderId + "] " + customerName + " - ₹" + totalPrice;

}

}

=> OrderSorter.java

**package** exno\_3;

**public** **class** OrderSorter {

// Bubble Sort

**public** **static** **void** bubbleSort(Order[] orders) {

**int** n = orders.length;

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

**boolean** swapped = **false**;

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

**if** (orders[j].getTotalPrice() > orders[j + 1].getTotalPrice()) {

Order temp = orders[j];

orders[j] = orders[j + 1];

orders[j + 1] = temp;

swapped = **true**;

}

}

**if** (!swapped) **break**; // optimization

}

}

// Quick Sort

**public** **static** **void** quickSort(Order[] orders, **int** low, **int** high) {

**if** (low < high) {

**int** pivotIndex = *partition*(orders, low, high);

*quickSort*(orders, low, pivotIndex - 1);

*quickSort*(orders, pivotIndex + 1, high);

}

}

**private** **static** **int** partition(Order[] orders, **int** low, **int** high) {

**double** pivot = orders[high].getTotalPrice();

**int** i = low - 1;

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

**if** (orders[j].getTotalPrice() <= 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;

}

}

=> Main.java

**package** exno\_3;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Order[] orders = {

**new** Order(101, "Alice", 2499.99),

**new** Order(102, "Bob", 999.50),

**new** Order(103, "Charlie", 4999.00),

**new** Order(104, "Daisy", 1499.00)

};

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

*printOrders*(orders);

// Bubble Sort

Order[] bubbleSorted = orders.clone();

OrderSorter.*bubbleSort*(bubbleSorted);

System.***out***.println("\nBubble Sorted Orders:");

*printOrders*(bubbleSorted);

// Quick Sort

Order[] quickSorted = orders.clone();

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

System.***out***.println("\nQuick Sorted Orders:");

*printOrders*(quickSorted);

}

**private** **static** **void** printOrders(Order[] orders) {

**for** (Order o : orders) {

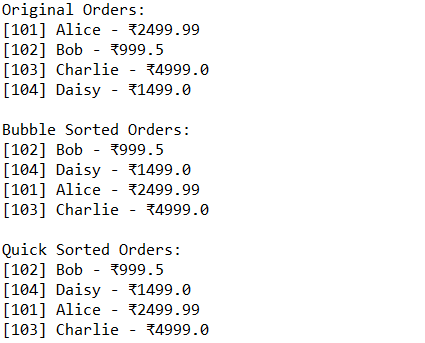
System.***out***.println(o);

}

}

}

**Output:**



**Exercise 4: Employee Management System**

**Code:**

**=>** Employee.java

**package** exno\_4;

**public** **class** Employee {

**private** **int** employeeId;

**private** String name;

**private** String position;

**private** **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** **int** getEmployeeId() {

**return** employeeId;

}

@Override

**public** String toString() {

**return** "[" + employeeId + "] " + name + " - " + position + " - ₹" + salary;

}

}

=> EmployeeManager.java

**package** exno\_4;

**public** **class** EmployeeManager {

**private** Employee[] employees;

**private** **int** count;

**public** EmployeeManager(**int** capacity) {

employees = **new** Employee[capacity];

count = 0;

}

// Add employee

**public** **void** addEmployee(Employee e) {

**if** (count < employees.length) {

employees[count++] = e;

} **else** {

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

}

}

// Search by employeeId

**public** Employee searchEmployee(**int** id) {

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

**if** (employees[i].getEmployeeId() == id) {

**return** employees[i];

}

}

**return** **null**;

}

// Traverse (list all employees)

**public** **void** displayEmployees() {

**if** (count == 0) {

System.***out***.println("No employees found.");

} **else** {

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

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

}

}

}

// Delete by employeeId

**public** **void** deleteEmployee(**int** id) {

**int** index = -1;

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

**if** (employees[i].getEmployeeId() == id) {

index = i;

**break**;

}

}

**if** (index != -1) {

**for** (**int** i = index; i < count - 1; i++) {

employees[i] = employees[i + 1];

}

employees[--count] = **null**;

System.***out***.println("Employee with ID " + id + " deleted.");

} **else** {

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

}

}

}

=> Main.java

**package** exno\_4;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

EmployeeManager manager = **new** EmployeeManager(5);

manager.addEmployee(**new** Employee(1, "Alice", "Manager", 70000));

manager.addEmployee(**new** Employee(2, "Bob", "Developer", 50000));

manager.addEmployee(**new** Employee(3, "Charlie", "Tester", 45000));

System.***out***.println("All Employees:");

manager.displayEmployees();

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

Employee e = manager.searchEmployee(2);

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

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

manager.deleteEmployee(1);

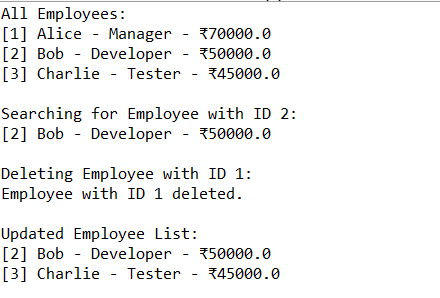
System.***out***.println("\nUpdated Employee List:");

manager.displayEmployees();

}

}

**Output:**

****

**Exercise 5: Task Management System using Linked List**

**Code:**

**=>** Task.java

**package** exno\_5;

**public** **class** Task {

**private** **int** taskId;

**private** String taskName;

**private** String status; // e.g., Pending, Completed

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

**this**.taskId = taskId;

**this**.taskName = taskName;

**this**.status = status;

}

**public** **int** getTaskId() {

**return** taskId;

}

@Override

**public** String toString() {

**return** "[" + taskId + "] " + taskName + " - " + status;

}

}

=> TaskNode.java

**package** exno\_5;

**public** **class** TaskNode {

Task task;

TaskNode next;

**public** TaskNode(Task task) {

**this**.task = task;

**this**.next = **null**;

}

}

=> TaskManager.java

**package** exno\_5;

**public** **class** TaskManager {

**private** TaskNode head;

// Add task to end

**public** **void** addTask(Task task) {

TaskNode newNode = **new** TaskNode(task);

**if** (head == **null**) {

head = newNode;

} **else** {

TaskNode temp = head;

**while** (temp.next != **null**) {

temp = temp.next;

}

temp.next = newNode;

}

}

// Search task by taskId

**public** Task searchTask(**int** id) {

TaskNode temp = head;

**while** (temp != **null**) {

**if** (temp.task.getTaskId() == id) {

**return** temp.task;

}

temp = temp.next;

}

**return** **null**;

}

// Traverse tasks

**public** **void** displayTasks() {

**if** (head == **null**) {

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

**return**;

}

TaskNode temp = head;

**while** (temp != **null**) {

System.***out***.println(temp.task);

temp = temp.next;

}

}

// Delete task by taskId

**public** **void** deleteTask(**int** id) {

**if** (head == **null**) **return**;

**if** (head.task.getTaskId() == id) {

head = head.next;

System.***out***.println("Task with ID " + id + " deleted.");

**return**;

}

TaskNode prev = **null**, curr = head;

**while** (curr != **null** && curr.task.getTaskId() != id) {

prev = curr;

curr = curr.next;

}

**if** (curr == **null**) {

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

**return**;

}

prev.next = curr.next;

System.***out***.println("Task with ID " + id + " deleted.");

}

}

=> Main.java

**package** exno\_5;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

TaskManager manager = **new** TaskManager();

manager.addTask(**new** Task(1, "Write proposal", "Pending"));

manager.addTask(**new** Task(2, "Develop module", "In Progress"));

manager.addTask(**new** Task(3, "Test features", "Pending"));

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

manager.displayTasks();

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

Task found = manager.searchTask(2);

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

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

manager.deleteTask(1);

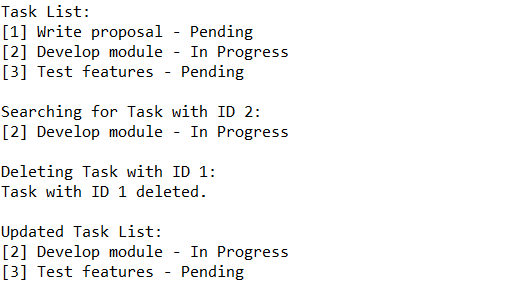
System.***out***.println("\nUpdated Task List:");

manager.displayTasks();

}

}

**Output:**

****

**Exercise 6: Library Management System**

**Code:**

**=>** Book.java

**package** exno\_6;

**public** **class** Book **implements** Comparable<Book> {

**private** **int** bookId;

**private** String title;

**private** String author;

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

**this**.bookId = bookId;

**this**.title = title;

**this**.author = author;

}

**public** String getTitle() {

**return** title.toLowerCase(); // for case-insensitive search

}

**public** String getOriginalTitle() {

**return** title;

}

@Override

**public** **int** compareTo(Book other) {

**return** **this**.getTitle().compareTo(other.getTitle());

}

@Override

**public** String toString() {

**return** "[" + bookId + "] " + title + " by " + author;

}

}

=> LibrarySearch.java

**package** exno\_6;

**public** **class** LibrarySearch {

// Linear Search by title

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

**for** (Book book : books) {

**if** (book.getTitle().equals(title.toLowerCase())) {

**return** book;

}

}

**return** **null**;

}

// Binary Search by title (requires sorted array)

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

**int** left = 0;

**int** right = books.length - 1;

String searchTitle = title.toLowerCase();

**while** (left <= right) {

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

**int** cmp = books[mid].getTitle().compareTo(searchTitle);

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

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

**else** right = mid - 1;

}

**return** **null**;

}

}

=> Main.java

**package** exno\_6;

**import** java.util.Arrays;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Book[] books = {

**new** Book(1, "The Alchemist", "Paulo Coelho"),

**new** Book(2, "Atomic Habits", "James Clear"),

**new** Book(3, "Rich Dad Poor Dad", "Robert Kiyosaki"),

**new** Book(4, "The 5 AM Club", "Robin Sharma")

};

// Linear Search

System.***out***.println("Linear Search: Find 'Atomic Habits'");

Book linearResult = LibrarySearch.*linearSearch*(books, "Atomic Habits");

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

// Sort books for binary search

Arrays.*sort*(books);

// Binary Search

System.***out***.println("\nBinary Search: Find 'Atomic Habits'");

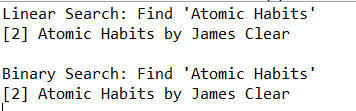
Book binaryResult = LibrarySearch.*binarySearch*(books, "Atomic Habits");

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

}

}

**Output:**

****

**Exercise 7: Financial Forecasting**

**Code:**

**=>**FinancialForecast.java

**package** exno\_7;

**public** **class** FinancialForecast {

// Recursive method to calculate future value

**public** **static** **double** futureValueRecursive(**double** initialValue, **double** growthRate, **int** years) {

**if** (years == 0) {

**return** initialValue;

}

**return** (1 + growthRate) \* *futureValueRecursive*(initialValue, growthRate, years - 1);

}

// Optimized method using memoization

**public** **static** **double** futureValueMemoized(**double** initialValue, **double** growthRate, **int** years, **double**[] memo) {

**if** (years == 0) **return** initialValue;

**if** (memo[years] != 0) **return** memo[years];

memo[years] = (1 + growthRate) \* *futureValueMemoized*(initialValue, growthRate, years - 1, memo);

**return** memo[years];

}

}

=>Main.java

**package** exno\_7;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

**double** initialValue = 10000; // ₹10,000

**double** growthRate = 0.10; // 10% annual growth

**int** years = 5;

System.***out***.println("Initial Value: "+initialValue);

System.***out***.println("Growth Rate: "+growthRate);

System.***out***.println("Years: "+years);

System.***out***.println("\nRecursive Forecast:");

**double** result = FinancialForecast.*futureValueRecursive*(initialValue, growthRate, years);

System.***out***.printf("Future value after %d years: ₹%.2f\n", years, result);

System.***out***.println("\nOptimized Forecast using Memoization:");

**double**[] memo = **new** **double**[years + 1];

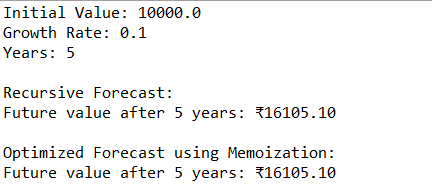
**double** optimizedResult = FinancialForecast.*futureValueMemoized*(initialValue, growthRate, years, memo);

System.***out***.printf("Future value after %d years: ₹%.2f\n", years, optimizedResult);

}

}

**Output:**

****