**Exercise 1: Inventory Management System**

**Inventory.java**

import java.util.HashMap;

public class Inventory {

HashMap<Integer, Product> inventoryMap;

public Inventory() {

inventoryMap = new HashMap<>();

}

public void addProduct(Product product) {

inventoryMap.put(product.productId, product);

System.*out*.println("Added:");

product.display();

}

public void updateProduct(int productId, String name, int qty, double price) {

Product product = inventoryMap.get(productId);

if (product != null) {

product.productName = name;

product.quantity = qty;

product.price = price;

System.*out*.println("Updated:");

product.display();

} else {

System.*out*.println("Product ID " + productId + " is not available");

}

}

public void deleteProduct(int productId) {

Product removed = inventoryMap.remove(productId);

if (removed != null) {

System.*out*.println("Deleted:");

removed.display();

} else {

System.*out*.println("Product ID " + productId + " is not available");

}

}

public void displayAll() {

System.*out*.println("\nInventory");

for (Product product : inventoryMap.values()) {

product.display();

}

}

}

**Product.java**

public class Product {

public int productId;

public String productName;

public int quantity;

public 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 void display() {

System.*out*.println("Product ID: " + productId +

", Name: " + productName +

", Quantity: " + quantity +

", Price: " + price);

}

}

**Main.java**

public class Main {

public static void main(String[] args) {

Inventory inventory = new Inventory();

// Adding products into hash map

inventory.addProduct(new Product(101, "Laptop", 5, 55000));

inventory.addProduct(new Product(102, "Keyboard", 20, 1500));

inventory.addProduct(new Product(103, "Mouse", 50, 500));

// Updating a product in hash map

inventory.updateProduct(102, "Mechanical Keyboard", 15, 1800);

// Deleting a product from hash map

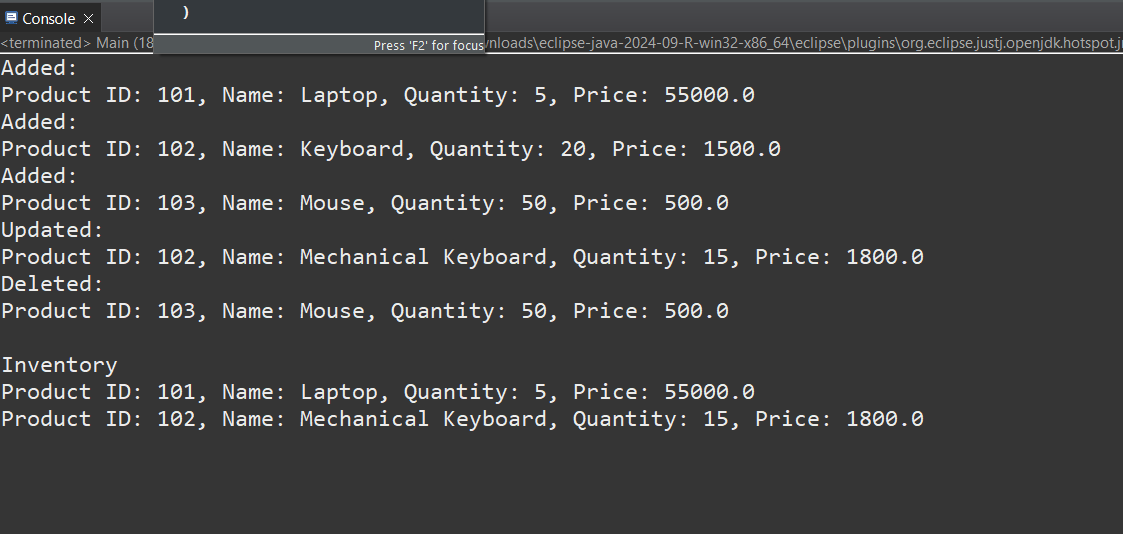
inventory.deleteProduct(103);

inventory.displayAll();

}

}

**Output:**

****

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

**Product.java**

package product;

public class Product {

private int productId;

private String productName;

private String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public int getProductId() {

return productId;

}

public void getProductDetails() {

System.***out***.println("Product ID: "+ productId + "\nProduct Name: "+ productName +"\nCategory: "+category+"\n");

}

}

**Searching.java**

package product;

public class Searching {

public Product linearSearch(Product[] ArrayOfProducts, int targetId) {

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

for(Product p : ArrayOfProducts) {

if(targetId == p.getProductId()) {

return p;

}

}

return null;

}

public Product binarySearch(Product[] ArrayOfProducts, int targetId) {

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

int s = 0;

int e = ArrayOfProducts.length-1;

if(s <= e) {

int mid = (s + e)/2;

Product p = ArrayOfProducts[mid];

int midId = p.getProductId();

if(midId == targetId) {

return p;

}

else if(midId < targetId) {

s = mid + 1;

}

else {

e = mid - 1;

}

}

return null;

}

}

**Main.java**

package product;

import java.util.Arrays;

import java.util.Comparator;

public class Main {

public static void main(String[] args) {

Product[] ArrayOfProducts = {

new Product(2,"Smart Watch","Electronics"),

new Product(4,"Jeans Pants","Clothing"),

new Product(1,"Shoes","Clothing"),

new Product(3,"TV","Appliances"),

new Product(5,"Teddy Bear","Toys")

};

int targetId = 3;

Searching search = new Searching();

//linear search

Product p = search.linearSearch(ArrayOfProducts, targetId);

if(p!=null) {

p.getProductDetails();

}

Arrays.sort(ArrayOfProducts, Comparator.comparingInt(Product::getProductId));

//binary search

p = search.binarySearch(ArrayOfProducts, targetId);

if(p!=null) {

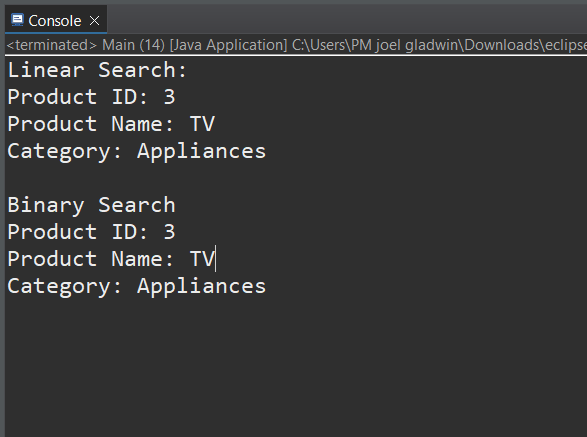
p.getProductDetails();

}

}

}

**Output:**



**Exercise 3: Sorting Customer Orders**

**Order.java**

public class Order {

public int orderId;

public String name;

public double price;

public Order(int orderId, String name, double price) {

this.orderId = orderId;

this.name = name;

this.price = price;

}

public void display() {

System.***out***.println("Order ID: " + orderId +

", Customer: " + name +

", Price:" + price);

}

}

**BubbleSort.java**

public class BubbleSort {

public static void sort(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].price > orders[j + 1].price) {

Order temp = orders[j];

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

orders[j + 1] = temp;

}

}

}

}

}

**QuickSort.java**

public class QuickSort {

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

if (low < high) {

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

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

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

}

}

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

double pivot = orders[high].price;

int i = low - 1;

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

if (orders[j].price <= 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**

public class Main {

public static void main(String[] args) {

Order[] orders = {

new Order(1, "Joel", 3500),

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

new Order(3, "raja", 5800),

new Order(4, "johan", 2200)

};

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

for (Order order : orders) {

order.display();

}

System.***out***.println("\n price Sorted using Bubble Sort:");

BubbleSort.*sort*(orders);

for (Order order : orders) {

order.display();

}

orders = new Order[]{

new Order(1, "Joel", 3500),

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

new Order(3, "raja", 5800),

new Order(4, "johan", 2200)

};

System.***out***.println("\nPrice Sorted using Quick Sort:");

QuickSort.*sort*(orders, 0, orders.length - 1);

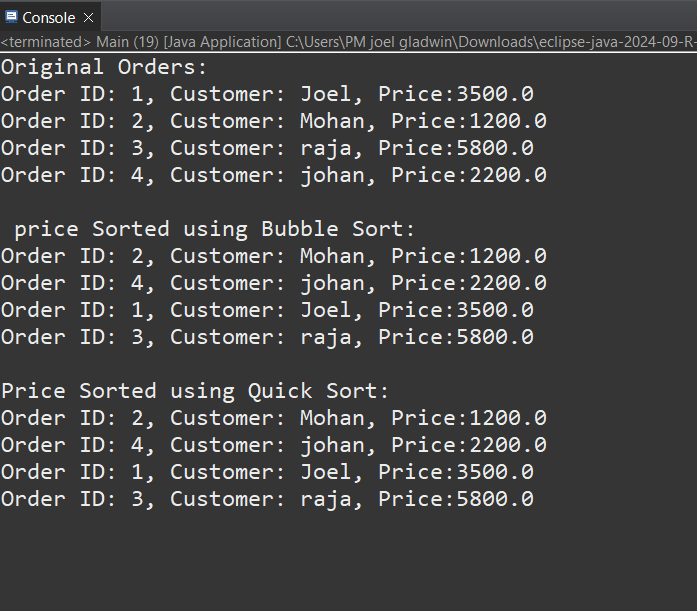
for (Order order : orders) {

order.display();

}

}

}

**OUTPUT:  
  
**

**Exercise 4: Employee Management System**

**Employee.java**

public class Employee {

public int employeeId;

public String name;

public String position;

public 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("ID: " + employeeId + ", Name: " + name +

", Position: " + position + ", Salary: " + salary);

}

}

**EmployeeManagementSystem.java**

public class EmployeeManagementSystem {

private Employee[] employees;

private int count;

public EmployeeManagementSystem(int size) {

employees = new Employee[size];

count = 0;

}

public void addEmployee(Employee e) {

if (count < employees.length) {

employees[count++] = e;

} 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 displayAll() {

if (count == 0) {

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

return;

}

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

employees[i].display();

}

}

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.");

}

}

**Main.java**

public class Main {

public static void main(String[] args) {

EmployeeManagementSystem ems = new EmployeeManagementSystem(5);

ems.addEmployee(new Employee(101, "Alice", "Manager", 75000));

ems.addEmployee(new Employee(102, "Bob", "Engineer", 55000));

ems.addEmployee(new Employee(103, "Charlie", "Analyst", 50000));

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

ems.displayAll();

System.*out*.println("\nSearching for Employee ID 102:");

Employee found = ems.searchEmployee(102);

if (found != null) found.display();

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

System.*out*.println("\nDeleting Employee ID 102:");

ems.deleteEmployee(102);

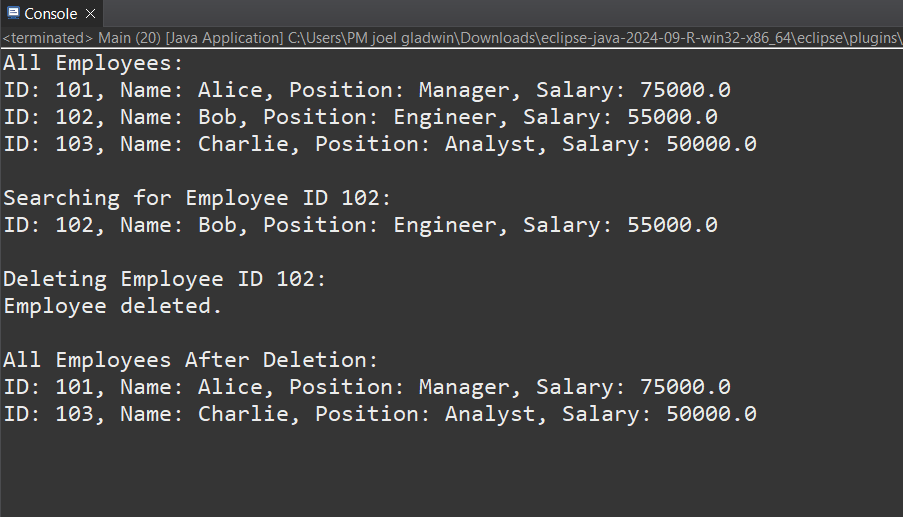
System.*out*.println("\nAll Employees After Deletion:");

ems.displayAll();

}

}

**Output:**

****

**Exercise 5: Task Management System**

**Task.java**

public class Task {

public int taskId;

public String taskName;

public String status;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

}

public void display() {

System.***out***.println("Task ID: " + taskId +

", Name: " + taskName +

", Status: " + status);

}

}

**Taskmanage.java**

import java.util.LinkedList;

public class TaskManager {

private LinkedList<Task> taskList;

public TaskManager() {

taskList = new LinkedList<>();

}

public void addTask(Task task) {

taskList.add(task);

}

public Task searchTask(int taskId) {

for (Task task : taskList) {

if (task.taskId == taskId)

return task;

}

return null;

}

public void deleteTask(int taskId) {

for (Task task : taskList) {

if (task.taskId == taskId) {

taskList.remove(task);

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

return;

}

}

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

}

public void displayTasks() {

if (taskList.isEmpty()) {

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

return;

}

for (Task task : taskList) {

task.display();

}

}

}

**Main.java**

public class Main {

public static void main(String[] args) {

TaskManager tm = new TaskManager();

tm.addTask(new Task(1, "Design Module", "Pending"));

tm.addTask(new Task(2, "Write Code", "In Progress"));

tm.addTask(new Task(3, "Test Application", "Pending"));

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

tm.displayTasks();

System.***out***.println("\nSearch Task ID 2:");

Task found = tm.searchTask(2);

if (found != null) found.display();

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

System.***out***.println("\nDelete Task ID 2:");

tm.deleteTask(2);

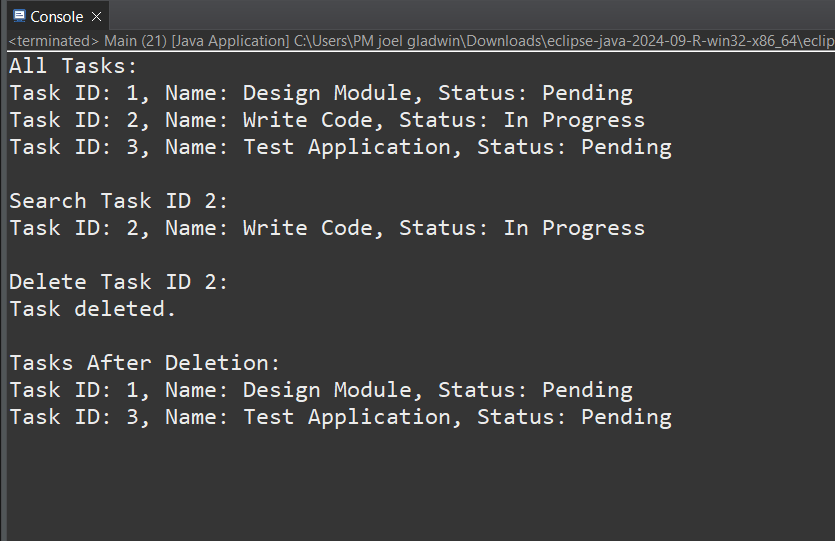
System.***out***.println("\nTasks After Deletion:");

tm.displayTasks();

}

}

**Output**

****

**Exercise 6: Library Management System**

**Book.java**

public class Book {

public int bookId;

public String title;

public 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("Book ID: " + bookId +

", Title: " + title +

", Author: " + author);

}

}

**BookSearching.java**

public class BookSearch {

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

while (low <= high) {

int mid = (low + high) / 2;

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

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

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

else high = mid - 1;

}

return null;

}

}

**Main.java**

import java.util.Arrays;

import java.util.Comparator;

public class Main {

public static void main(String[] args) {

Book[] books = {

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

new Book(2, "To Kill a Mockingbird", "Harper Lee"),

new Book(3, "1984", "George Orwell"),

new Book(4, "Brave New World", "Aldous Huxley"),

new Book(5, "Sapiens", "Yuval Noah Harari")

};

// Linear Search

System.out.println("Linear Search for '1984':");

Book result1 = BookSearch.linearSearch(books, "1984");

if (result1 != null) result1.display();

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

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

// Binary Search

System.out.println("\nBinary Search for '1984':");

Book result2 = BookSearch.binarySearch(books, "1984");

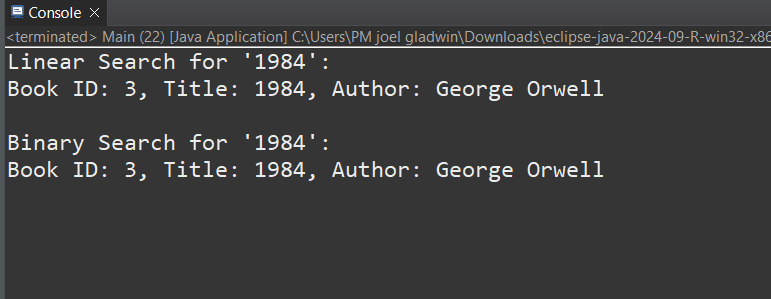
if (result2 != null) result2.display();

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

}

}

**Output**

****

**Exercise 7: Financial Forecasting**

**FinancialForecasting.java**

public class FinancialForecast {

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

if (years == 0) {

return initialAmount;

} else {

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

}

}

}

**Main.java**

public class Main {

public static void main(String[] args) {

double Amount = 10000;

double Rate = 0.08;

int years = 5;

double result = FinancialForecast.*futureValue*(Amount, Rate, years);

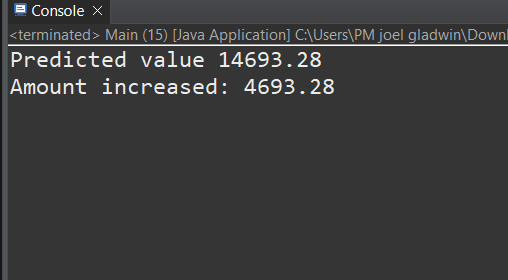
System.***out***.printf("Predicted value %.2f\n", result);

System.***out***.printf("Amount increased: %.2f", (result-Amount));

}

}

**Output**

****