**Exercise 1: Inventory Management System**

Program:

package inventorymanagement;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

class Product {

    private String productId;

    private String productName;

    private int quantity;

    private double price;

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

        this.productId = productId;

        this.productName = productName;

        this.quantity = quantity;

        this.price = price;

    }

    public String getProductId() {

        return productId;

    }

    public String getProductName() {

        return productName;

    }

    public int getQuantity() {

        return quantity;

    }

    public double getPrice() {

        return price;

    }

    public void setProductName(String productName) {

        this.productName = productName;

    }

    public void setQuantity(int quantity) {

        this.quantity = quantity;

    }

    public void setPrice(double price) {

        this.price = price;

    }

    @Override

    public String toString() {

        return "Product ID: " + productId +

               ", Name: " + productName +

               ", Quantity: " + quantity +

               ", Price: $" + String.format("%.2f", price);

    }

}

class InventoryManager {

    private Map<String, Product> inventory;

    public InventoryManager() {

        this.inventory = new HashMap<>();

        System.out.println("Inventory Manager initialized. Using HashMap for storage.");

    }

    public boolean addProduct(Product product) {

        if (inventory.containsKey(product.getProductId())) {

            System.out.println("Error: Product with ID " + product.getProductId() + " already exists.");

            return false;

        }

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

        System.out.println("Added: " + product.getProductName() + " (ID: " + product.getProductId() + ")");

        return true;

    }

    public boolean updateProduct(String productId, String newName, int newQuantity, double newPrice) {

        Product product = inventory.get(productId);

        if (product == null) {

            System.out.println("Error: Product with ID " + productId + " not found for update.");

            return false;

        }

        if (newName != null && !newName.isEmpty()) {

            product.setProductName(newName);

        }

        if (newQuantity >= 0) {

            product.setQuantity(newQuantity);

        }

        if (newPrice >= 0) {

            product.setPrice(newPrice);

        }

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

        return true;

    }

    public boolean deleteProduct(String productId) {

        if (inventory.remove(productId) != null) {

            System.out.println("Deleted product with ID: " + productId);

            return true;

        }

        System.out.println("Error: Product with ID " + productId + " not found for deletion.");

        return false;

    }

    public Product getProduct(String productId) {

        return inventory.get(productId);

    }

    public void displayAllProducts() {

        if (inventory.isEmpty()) {

            System.out.println("\nInventory is empty.");

            return;

        }

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

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

            System.out.println(product);

        }

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

    }

    public static void main(String[] args) {

        System.out.println("--- Starting Inventory Management System ---");

        InventoryManager manager = new InventoryManager();

        Scanner scanner = new Scanner(System.in);

        manager.addProduct(new Product("P001", "Laptop", 10, 1200.50));

        manager.addProduct(new Product("P002", "Mouse", 50, 25.00));

        manager.addProduct(new Product("P003", "Keyboard", 30, 75.00));

        manager.addProduct(new Product("P001", "Duplicate Product", 5, 100.00));

        manager.displayAllProducts();

        System.out.println("\n--- Updating Product P002 (Mouse) ---");

        manager.updateProduct("P002", "Wireless Mouse", 45, 29.99);

        manager.displayAllProducts();

        System.out.println("\n--- Attempting to Update Non-existent Product P004 ---");

        manager.updateProduct("P004", "Monitor", 5, 200.00);

        System.out.println("\n--- Deleting Product P003 (Keyboard) ---");

        manager.deleteProduct("P003");

        manager.displayAllProducts();

        System.out.println("\n--- Retrieving Product P001 ---");

        Product p001 = manager.getProduct("P001");

        if (p001 != null) {

            System.out.println("Found: " + p001);

        } else {

            System.out.println("Product P001 not found.");

        }

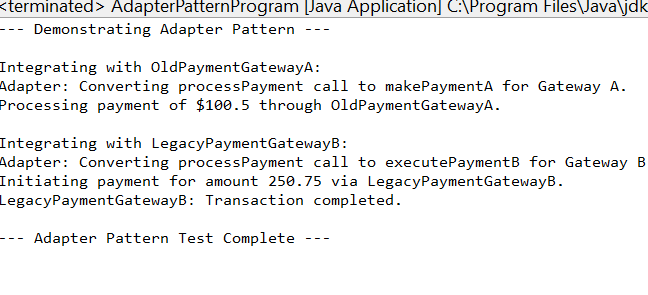
        System.out.println("\n--- Inventory Management System Ended ---");

        scanner.close();

    }

}

**OUTPUT**:



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

**Program:**

// Product.java for Exercise 2

**package** ecommerce;

// Implementing Comparable to allow sorting by productName for Binary Search

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

**private** String productId;

**private** String productName;

**private** String category;

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

**this**.productId = productId;

**this**.productName = productName;

**this**.category = category;

}

// Getters

**public** String getProductId() {

**return** productId;

}

**public** String getProductName() {

**return** productName;

}

**public** String getCategory() {

**return** category;

}

@Override

**public** String toString() {

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

}

/\*\*

\* Compares products based on their productName, essential for binary search.

\*/

@Override

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

**return** **this**.productName.compareToIgnoreCase(other.productName);

}

}

// SearchAlgorithms.java

package ecommerce;

import java.util.Arrays;

import java.util.Comparator;

public class SearchAlgorithms {

/\*\*

\* Implements linear search to find a product by its name.

\* @param products The array of Product objects to search through.

\* @param targetName The name of the product to find.

\* @return The Product object if found, null otherwise.

\* Time Complexity: O(N)

\*/

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

System.out.println("Performing Linear Search for: " + targetName);

for (Product product : products) {

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

return product;

}

}

return null; // Product not found

}

/\*\*

\* Implements binary search to find a product by its name.

\* Requires the array to be sorted by product name.

\* @param products The sorted array of Product objects to search through.

\* @param targetName The name of the product to find.

\* @return The Product object if found, null otherwise.

\* Time Complexity: O(log N)

\*/

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

System.out.println("Performing Binary Search for: " + targetName);

int low = 0;

int high = products.length - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

int comparison = products[mid].getProductName().compareToIgnoreCase(targetName);

if (comparison == 0) {

return products[mid]; // Product found

} else if (comparison < 0) {

low = mid + 1; // Target is in the right half

} else {

high = mid - 1; // Target is in the left half

}

}

return null; // Product not found

}

public static void main(String[] args) {

SearchAlgorithms searcher = new SearchAlgorithms();

// Products for Linear Search (can be unsorted)

Product[] productsLinear = {

new Product("P001", "Laptop", "Electronics"),

new Product("P002", "Mouse", "Electronics"),

new Product("P003", "Keyboard", "Electronics"),

new Product("P004", "Monitor", "Electronics"),

new Product("P005", "Webcam", "Accessories")

};

System.out.println("--- Linear Search Tests ---");

Product foundProductLinear = searcher.linearSearch(productsLinear, "Keyboard");

if (foundProductLinear != null) {

System.out.println("Found: " + foundProductLinear);

} else {

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

}

foundProductLinear = searcher.linearSearch(productsLinear, "Speakers");

if (foundProductLinear != null) {

System.out.println("Found: " + foundProductLinear);

} else {

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

}

// Products for Binary Search (must be sorted by productName)

Product[] productsBinary = {

new Product("P005", "Webcam", "Accessories"),

new Product("P003", "Keyboard", "Electronics"),

new Product("P001", "Laptop", "Electronics"),

new Product("P004", "Monitor", "Electronics"),

new Product("P002", "Mouse", "Electronics")

};

// Sort the array by product name before performing binary search

Arrays.sort(productsBinary); // Uses Product's compareTo method

System.out.println("\n--- Sorted Products for Binary Search ---");

for (Product p : productsBinary) {

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

}

System.out.println("\n--- Binary Search Tests ---");

Product foundProductBinary = searcher.binarySearch(productsBinary, "Laptop");

if (foundProductBinary != null) {

System.out.println("Found: " + foundProductBinary);

} else {

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

}

foundProductBinary = searcher.binarySearch(productsBinary, "Printer");

if (foundProductBinary != null) {

System.out.println("Found: " + foundProductBinary);

} else {

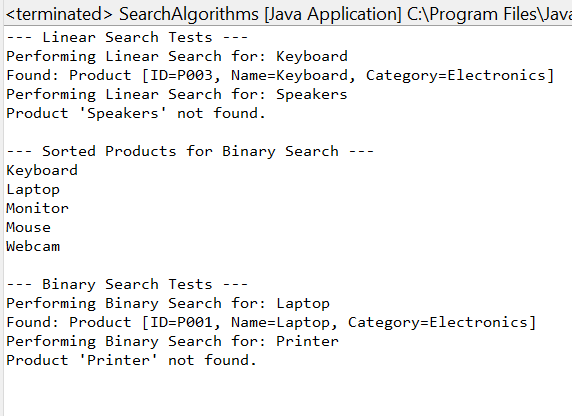
System.out.println("Product 'Printer' not found.");

}

}

}

**OUTPUT:**



**Exercise 6: Library Management System**

Program:

package library\_management;

import java.util.Arrays;

import java.util.Comparator;

import java.util.List;

import java.util.ArrayList;

class Book {

    private String bookId;

    private String title;

    private String author;

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

        this.bookId = bookId;

        this.title = title;

        this.author = author;

    }

    public String getBookId() {

        return bookId;

    }

    public String getTitle() {

        return title;

    }

    public String getAuthor() {

        return author;

    }

    @Override

    public String toString() {

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

    }

}

class LibrarySearch {

    public Book linearSearchByTitle(Book[] books, String targetTitle) {

        System.out.println("Performing Linear Search for Title: \"" + targetTitle + "\"");

        for (int i = 0; i < books.length; i++) {

            if (books[i].getTitle().equalsIgnoreCase(targetTitle)) {

                System.out.println("Linear Search: Found at index " + i);

                return books[i];

            }

        }

        System.out.println("Linear Search: Book with title \"" + targetTitle + "\" not found.");

        return null;

    }

    public Book binarySearchByTitle(Book[] books, String targetTitle) {

        System.out.println("Performing Binary Search for Title: \"" + targetTitle + "\"");

        int low = 0;

        int high = books.length - 1;

        while (low <= high) {

            int mid = low + (high - low) / 2;

            int comparison = books[mid].getTitle().compareToIgnoreCase(targetTitle);

            if (comparison == 0) {

                System.out.println("Binary Search: Found at index " + mid);

                return books[mid];

            } else if (comparison < 0) {

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

        System.out.println("Binary Search: Book with title \"" + targetTitle + "\" not found.");

        return null;

    }

}

public class LibraryManagementProgram {

    public static void main(String[] args) {

        System.out.println("--- Library Management System Search Function ---");

        Book[] libraryBooks = {

            new Book("B005", "The Hitchhiker's Guide to the Galaxy", "Douglas Adams"),

            new Book("B001", "1984", "George Orwell"),

            new Book("B008", "To Kill a Mockingbird", "Harper Lee"),

            new Book("B003", "Brave New World", "Aldous Huxley"),

            new Book("B007", "Dune", "Frank Herbert"),

            new Book("B002", "Pride and Prejudice", "Jane Austen"),

            new Book("B006", "The Lord of the Rings", "J.R.R. Tolkien"),

            new Book("B004", "Foundation", "Isaac Asimov")

        };

        System.out.println("\nInitial Library Books (unsorted for linear search):");

        for (Book book : libraryBooks) {

            System.out.println(book);

        }

        Book[] sortedBooks = Arrays.copyOf(libraryBooks, libraryBooks.length);

        Arrays.sort(sortedBooks, Comparator.comparing(Book::getTitle, String.CASE\_INSENSITIVE\_ORDER));

        System.out.println("\nLibrary Books (sorted by Title for binary search):");

        for (Book book : sortedBooks) {

            System.out.println(book);

        }

        LibrarySearch searcher = new LibrarySearch();

        System.out.println("\n--- Demonstrating Linear Search ---");

        System.out.println("Result: " + searcher.linearSearchByTitle(libraryBooks, "1984"));

        System.out.println("Result: " + searcher.linearSearchByTitle(libraryBooks, "The Martian"));

        System.out.println("Result: " + searcher.linearSearchByTitle(libraryBooks, "Dune"));

        System.out.println("\n--- Demonstrating Binary Search ---");

        System.out.println("Result: " + searcher.binarySearchByTitle(sortedBooks, "1984"));

        System.out.println("Result: " + searcher.binarySearchByTitle(sortedBooks, "The Martian"));

        System.out.println("Result: " + searcher.binarySearchByTitle(sortedBooks, "Pride and Prejudice"));

        System.out.println("Result: " + searcher.binarySearchByTitle(sortedBooks, "The Lord of the Rings"));

        System.out.println("\n--- Library Search Function Test Complete ---");

    }

}

**OUTPUT**:



**Exercise 7: Financial Forecasting**

**Program:**

package financial\_forecasting;

class FinancialForecasting {

    public double calculateFutureValue(double presentValue, double annualGrowthRate, int years) {

        if (years == 0) {

            return presentValue;

        }

        return calculateFutureValue(presentValue \* (1 + annualGrowthRate), annualGrowthRate, years - 1);

    }

    public static void main(String[] args) {

        System.out.println("--- Financial Forecasting Tool ---");

        double initialInvestment = 1000.0;

        double growthRate = 0.05; // 5% annual growth

        int forecastYears = 10;

        FinancialForecasting forecaster = new FinancialForecasting();

        System.out.println("Initial Investment: $" + String.format("%.2f", initialInvestment));

        System.out.println("Annual Growth Rate: " + (growthRate \* 100) + "%");

        System.out.println("Forecast Years: " + forecastYears);

        double futureValue = forecaster.calculateFutureValue(initialInvestment, growthRate, forecastYears);

        System.out.println("Future Value after " + forecastYears + " years: $" + String.format("%.2f", futureValue));

        System.out.println("\n--- Testing with different values ---");

        System.out.println("Initial: $500, Rate: 10%, Years: 5 -> Future Value: $" + String.format("%.2f", forecaster.calculateFutureValue(500, 0.10, 5)));

        System.out.println("Initial: $2000, Rate: 3%, Years: 2 -> Future Value: $" + String.format("%.2f", forecaster.calculateFutureValue(2000, 0.03, 2)));

        System.out.println("Initial: $100, Rate: 7%, Years: 0 -> Future Value: $" + String.format("%.2f", forecaster.calculateFutureValue(100, 0.07, 0)));

        System.out.println("\n--- Financial Forecasting Test Complete ---");

    }

}

OUTPUT:

