**Exercise 1: Inventory Management System**

Code:

package com.Inventory;

public class Product {

private int productId;

private String productName;

private int quantity;

private 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 int getProductId() {

return productId;

}

public String getProductName() {

return productName;

}

public void setProductName(String productName) {

this.productName = productName;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

*@Override*

public String toString() {

return "ID: " + productId + ", Name: " + productName + ", Qty: " + quantity + ", Price: Rs." + price;

}

}

package com.Inventory;

import java.util.\*;

public class InventoryManager {

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

//add

public void addProduct(Product product) {

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

}

//update

public boolean updateProduct(int id, String name, int quantity, double price) {

Product p = inventory.get(id);

if (p != null) {

p.setProductName(name);

p.setQuantity(quantity);

p.setPrice(price);

return true;

}

return false;

}

//delete

public boolean deleteProduct(int id) {

return inventory.remove(id) != null;

}

//view

public void displayInventory() {

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

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

}

}

}

package com.Inventory;

public class Main {

public static void main(String[] args) {

InventoryManager manager = new InventoryManager();

// add

manager.addProduct(new Product(101, "Keyboard", 50, 999.99));

manager.addProduct(new Product(102, "Mouse", 80, 499.50));

// view

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

manager.displayInventory();

// update

manager.updateProduct(101, "Gaming Keyboard", 45, 1299.99);

// delete

manager.deleteProduct(102);

// view after changes

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

manager.displayInventory();

}

}

Output:

A screenshot of a computer

AI-generated content may be incorrect.

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

Code:

package com.ecommerce;

public class 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 String toString() {

return "ID: " + productId + ", Name: " + productName + ", Category: " + category;

}

}

package com.ecommerce;

public class SearchAlgorithms {

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

for (Product p : products) {

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

return p;

}

}

return null;

}

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

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

while (left <= right) {

int mid = (left + right) / 2;

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

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

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

else right = mid - 1;

}

return null;

}

}

package com.ecommerce;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(1, "Keyboard", "Electronics"),

new Product(2, "Shoes", "Fashion"),

new Product(3, "Laptop", "Electronics"),

new Product(4, "Book", "Stationery")

};

Product result1 = SearchAlgorithms.*linearSearch*(products, "Laptop");

System.***out***.println("Linear Search Result: " + result1);

Arrays.*sort*(products, Comparator.*comparing*(Product::getProductName, String.***CASE\_INSENSITIVE\_ORDER***));

Product result2 = SearchAlgorithms.*binarySearch*(products, "Laptop");

System.***out***.println("Binary Search Result: " + result2);

}

}

Output:

A screenshot of a computer program

AI-generated content may be incorrect.

**Exercise 3: Sorting Customer Orders**

Code:

package com.customer;

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 "Order ID: " + orderId + ", Customer: " + customerName + ", Total: $" + totalPrice;

}

}

package com.customer;

public class SortAlgorithms {

public static void bubbleSort(Order[] orders) {

int n = orders.length;

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

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

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

Order temp = orders[j];

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

orders[j + 1] = temp;

}

}

}

}

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

if (low < high) {

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

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

*quickSort*(orders, pi + 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;

}

}

package com.customer;

public class Main {

public static void main(String[] args) {

Order[] orders = {

new Order(1, "Alice", 350.50),

new Order(2, "Bob", 1200.00),

new Order(3, "Charlie", 800.75),

new Order(4, "Diana", 450.20)

};

System.***out***.println("Before Sorting:");

for (Order o : orders) System.***out***.println(o);

System.***out***.println("\nAfter Bubble Sort:");

SortAlgorithms.*bubbleSort*(orders);

for (Order o : orders) System.***out***.println(o);

orders = new Order[] {

new Order(1, "Alice", 350.50),

new Order(2, "Bob", 1200.00),

new Order(3, "Charlie", 800.75),

new Order(4, "Diana", 450.20)

};

System.***out***.println("\nAfter Quick Sort:");

SortAlgorithms.*quickSort*(orders, 0, orders.length - 1);

for (Order o : orders)

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

}

}

Output:

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 7: Financial Forecasting**

Code:

package com.financial;

public class FinancialForecast {

public static double futureValue(double P, double r, int n) {

if (n == 0) {

return P;

}

return *futureValue*(P, r, n - 1) \* (1 + r);

}

public static void main(String[] args) {

double initialInvestment = 1000.0;

double growthRate = 0.10;

int years = 5;

double result = *futureValue*(initialInvestment, growthRate, years);

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

}

}

Output:

A screenshot of a computer

AI-generated content may be incorrect.