**Exercise 1: Inventory Management System**

**Code:**

**Product.java**

**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** "[" + productId + "] " + productName + " - Qty: " + quantity + " | Price: ₹" + price;

}

}

**InventorySystem.java**

import java.util.\*;

public class InventorySystem {

ArrayList<Product> inventory = new ArrayList<>();

public void addProduct(Product p) {

inventory.add(p);

}

public void updateProduct(int productId, String newName, int newQty, double newPrice) {

for (Product p : inventory) {

if (p.productId == productId) {

p.productName = newName;

p.quantity = newQty;

p.price = newPrice;

return;

}

}

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

}

public void deleteProduct(int productId) {

inventory.removeIf(p -> p.productId == productId);

}

public void displayAll() {

for (Product p : inventory) {

System.*out*.println(p);

}

}

public ArrayList<Product> getInventory() {

return inventory;

}

}

**LinearSearch.java**

**import** java.util.ArrayList;

**public** **class** LinearSearch {

**public** **static** Product search(ArrayList<Product> list, **int** productId) {

**for** (Product p : list) {

**if** (p.productId == productId) {

**return** p;

}

}

**return** **null**;

}

}

**BinarySearch.java**

import java.util.\*;

public class BinarySearch {

public static Product search(ArrayList<Product> list, int productId) {

list.sort(Comparator.*comparingInt*(p -> p.productId));

int low = 0, high = list.size() - 1;

while (low <= high) {

int mid = (low + high) / 2;

Product midProduct = list.get(mid);

if (midProduct.productId == productId) {

return midProduct;

} else if (midProduct.productId < productId) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return null;

}

}

**Main.java**

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

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

InventorySystem inv = **new** InventorySystem();

// Add Products

inv.addProduct(**new** Product(103, "RAM", 5, 6999.00));

inv.addProduct(**new** Product(101, "GPU", 10, 799.99));

inv.addProduct(**new** Product(102, "Processor", 25, 399.50));

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

inv.displayAll();

// Update

System.***out***.println("\nUpdating Product 102:");

inv.updateProduct(102, "Wireless Mouse", 30, 499.00);

inv.displayAll();

// Delete

System.***out***.println("\nDeleting Product 101:");

inv.deleteProduct(101);

inv.displayAll();

// Linear Search

System.***out***.println("\nLinear Search for 103:");

Product found1 = LinearSearch.*search*(inv.getInventory(), 103);

System.***out***.println(found1 != **null** ? found1 : "Product not found");

// Binary Search

System.***out***.println("\nBinary Search for 102:");

Product found2 = BinarySearch.*search*(inv.getInventory(), 102);

System.***out***.println(found2 != **null** ? found2 : "Product not found");

}

}

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 7: Financial Forecasting**

**Code:**

**InvestmentForecast.java**

import java.util.\*;

public class InvestmentForecast {

// Step 1: Calculate average growth rate

public static double calculateAverageGrowthRate(List<Double> pastData) {

double totalGrowth = 0.0;

for (int i = 1; i < pastData.size(); i++) {

double growth = (pastData.get(i) - pastData.get(i - 1)) / pastData.get(i - 1);

totalGrowth += growth;

}

return totalGrowth / (pastData.size() - 1);

}

// Step 2: Forecast investment using recursion

public static double forecastInvestment(int yearsToPredict, double initialInvestment, double growthRate) {

if (yearsToPredict == 0) {

return initialInvestment;

}

return *forecastInvestment*(yearsToPredict - 1, initialInvestment, growthRate) \* (1 + growthRate);

}

public static void main(String[] args) {

// Past investment value data

List<Double> pastData = Arrays.*asList*(10000.0, 11000.0, 12100.0, 13000.0, 14300.0);

double averageGrowthRate = *calculateAverageGrowthRate*(pastData);

System.*out*.printf("Average Annual Growth Rate: %.2f%%\n", averageGrowthRate \* 100);

// Initial investment

double initialInvestment = 10000.0;

// Years to forecast

int forecastYears = 5;

System.*out*.println("\nForecasting future investment growth:");

for (int i = 1; i <= forecastYears; i++) {

double futureValue = *forecastInvestment*(i, initialInvestment, averageGrowthRate);

System.*out*.printf("Year +%d: ₹%.2f\n", i, futureValue);

}

}

}

**Output:**

A white background with black text

AI-generated content may be incorrect.