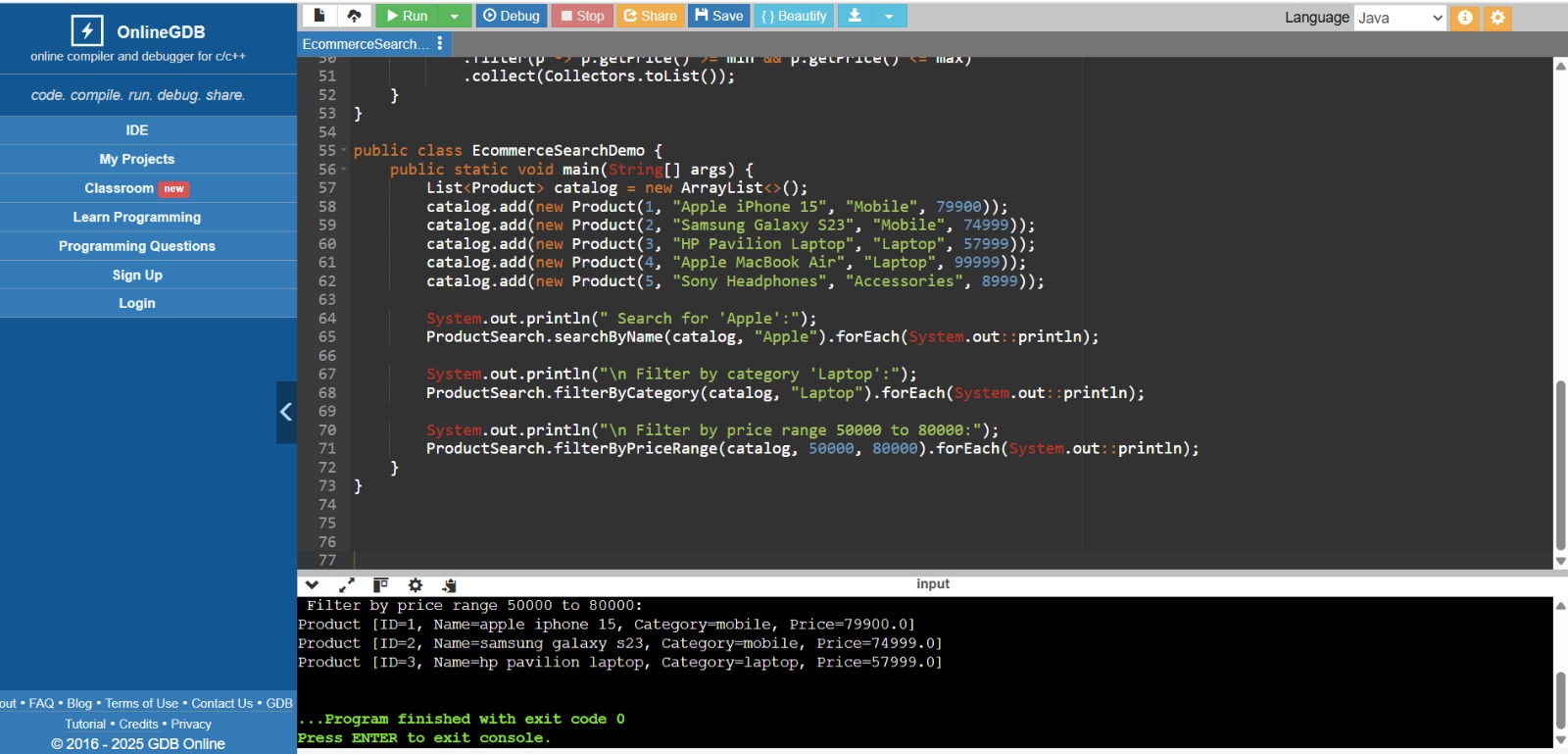
# Week1\_DataStructuresandAlgorithms\_HandsOn

## Exercise 2: E-commerce Platform Search Function

Code:

import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
class Product {  
 private int id;  
 private String name;  
 private String category;  
 private double price;  
  
 public Product(int id, String name, String category, double price) {  
 this.id = id;  
 this.name = name.toLowerCase();  
 this.category = category.toLowerCase();  
 this.price = price;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getCategory() {  
 return category;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public String toString() {  
 return "Product [ID=" + id + ", Name=" + name + ", Category=" + category + ", Price=" + price + "]";  
 }  
}  
  
class ProductSearch {  
 public static List<Product> searchByName(List<Product> products, String keyword) {  
 return products.stream()  
 .filter(p -> p.getName().contains(keyword.toLowerCase()))  
 .collect(Collectors.toList());  
 }  
  
 public static List<Product> filterByCategory(List<Product> products, String category) {  
 return products.stream()  
 .filter(p -> p.getCategory().equalsIgnoreCase(category))  
 .collect(Collectors.toList());  
 }  
  
 public static List<Product> filterByPriceRange(List<Product> products, double min, double max) {  
 return products.stream()  
 .filter(p -> p.getPrice() >= min && p.getPrice() <= max)  
 .collect(Collectors.toList());  
 }  
}  
  
public class EcommerceSearchDemo {  
 public static void main(String[] args) {  
 List<Product> catalog = new ArrayList<>();  
 catalog.add(new Product(1, "Apple iPhone 15", "Mobile", 79900));  
 catalog.add(new Product(2, "Samsung Galaxy S23", "Mobile", 74999));  
 catalog.add(new Product(3, "HP Pavilion Laptop", "Laptop", 57999));  
 catalog.add(new Product(4, "Apple MacBook Air", "Laptop", 99999));  
 catalog.add(new Product(5, "Sony Headphones", "Accessories", 8999));  
  
 System.out.println(" Search for 'Apple':");  
 ProductSearch.searchByName(catalog, "Apple").forEach(System.out::println);  
  
 System.out.println("\n Filter by category 'Laptop':");  
 ProductSearch.filterByCategory(catalog, "Laptop").forEach(System.out::println);  
  
 System.out.println("\n Filter by price range 50000 to 80000:");  
 ProductSearch.filterByPriceRange(catalog, 50000, 80000).forEach(System.out::println);  
 }  
}

**OUTPUT:**

****

------------------------------------------------------------

## Exercise 7: Financial Forecasting

Code:

import java.util.ArrayList;  
import java.util.List;  
  
public class Main {  
  
 public static List<Double> movingAverage(List<Double> prices, int windowSize) {  
 List<Double> forecast = new ArrayList<>();  
  
 if (prices.size() < windowSize) {  
 System.out.println("Window size is larger than data size.");  
 return forecast;  
 }  
  
 double sum = 0;  
  
 for (int i = 0; i < windowSize; i++) {  
 sum += prices.get(i);  
 }  
  
 forecast.add(sum / windowSize);  
  
 for (int i = windowSize; i < prices.size(); i++) {  
 sum += prices.get(i) - prices.get(i - windowSize);  
 forecast.add(sum / windowSize);  
 }  
  
 return forecast;  
 }  
  
 public static void main(String[] args) {  
 List<Double> stockPrices = new ArrayList<>();  
 stockPrices.add(100.0);  
 stockPrices.add(102.0);  
 stockPrices.add(104.0);  
 stockPrices.add(103.0);  
 stockPrices.add(101.0);  
 stockPrices.add(105.0);  
 stockPrices.add(107.0);  
 stockPrices.add(110.0);  
  
 int windowSize = 3;  
  
 List<Double> forecast = movingAverage(stockPrices, windowSize);  
  
 System.out.println("Original Prices: " + stockPrices);  
  
 System.out.println("Forecasted Moving Averages (window = " + windowSize + "):");  
 for (Double avg : forecast) {  
 System.out.printf("%.2f ", avg);  
 }  
 }  
}

**OUTPUT:**

