**WEEK-1**

**Data Structures Algorithms**

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

**Product.java**

package exercise3\_search;

public class Product {

private String name;

private String category;

private double price;

public Product(String name, String category, double price) {

this.name = name;

this.category = category;

this.price = price;

}

public String getName() {

return name;

}

public String getCategory() {

return category;

}

public double getPrice() {

return price;

}

public void display() {

System.***out***.println("Name: " + name + " | Category: " + category + " | Price: ₹" + price);

}

}

**SearchEngine.java**

package exercise3\_search;

import java.util.\*;

public class SearchEngine {

private List<Product> products;

public SearchEngine() {

products = new ArrayList<>();

products.add(new Product("Samsung Smart TV", "Electronics", 45000));

products.add(new Product("MacBook Air M2", "Electronics", 98000));

products.add(new Product("Harry Potter Box Set", "Books", 2500));

products.add(new Product("Men's Leather Jacket", "Fashion", 3999));

products.add(new Product("Air Fryer", "Home Appliances", 6999));

products.add(new Product("The Alchemist", "Books", 350));

products.add(new Product("Sony PlayStation 5", "Gaming", 49990));

products.add(new Product("Women's Handbag", "Fashion", 1299));

products.add(new Product("Xbox Controller", "Gaming", 4990));

products.add(new Product("Refrigerator 190L", "Home Appliances", 15499));

}

public List<Product> search(String keyword, String category) {

List<Product> results = new ArrayList<>();

for (Product product : products) {

boolean matchesKeyword = product.getName().toLowerCase().contains(keyword.toLowerCase());

boolean matchesCategory = category.equalsIgnoreCase("all") ||

product.getCategory().equalsIgnoreCase(category);

if (matchesKeyword && matchesCategory) {

results.add(product);

}

}

return results;

}

public Set<String> getAvailableCategories() {

Set<String> categories = new HashSet<>();

for (Product p : products) {

categories.add(p.getCategory());}

return categories;

}

}

**Main.java**

package exercise3\_search;

import java.util.\*;

public class Main {

public static void main(String[] args) {

SearchEngine engine = new SearchEngine();

Scanner scanner = new Scanner(System.***in***);

System.***out***.print("Enter search keyword: ");

String keyword = scanner.nextLine();

System.***out***.println("Available Categories: " + engine.getAvailableCategories());

System.***out***.print("Enter category to filter (or type 'all'): ");

String category = scanner.nextLine();

List<Product> results = engine.search(keyword, category);

System.***out***.println("\nSearch Results:");

if (results.isEmpty()) {

System.***out***.println("No products found matching your criteria.");

} else {

for (Product p : results) {

p.display();

}}}}

A computer screen shot of a program

AI-generated content may be incorrect.

**Exercise 7: Financial Forecasting**

**Forcast.java**

package exercise7\_forecasting;

public class Forecast {

public double predictRevenue(double currentRevenue, double growthRate, int months) {

return currentRevenue \* Math.*pow*(1 + growthRate, months);

}

}

**Main.java**

package exercise7\_forecasting;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Forecast forecast = new Forecast();

Scanner scanner = new Scanner(System.***in***);

System.***out***.println("📈 Financial Forecasting Tool");

System.***out***.print("Enter current revenue (₹): ");

double currentRevenue = scanner.nextDouble();

System.***out***.print("Enter monthly growth rate (e.g. 0.05 for 5%): ");

double growthRate = scanner.nextDouble();

System.***out***.print("Enter number of months to forecast: ");

int months = scanner.nextInt();

double predictedRevenue = forecast.predictRevenue(currentRevenue, growthRate, months);

System.***out***.printf("Predicted revenue after %d months: ₹%.2f\n", months, predictedRevenue);}}

A screen shot of a computer

AI-generated content may be incorrect.