Karen Giselle Valdez Muñoz

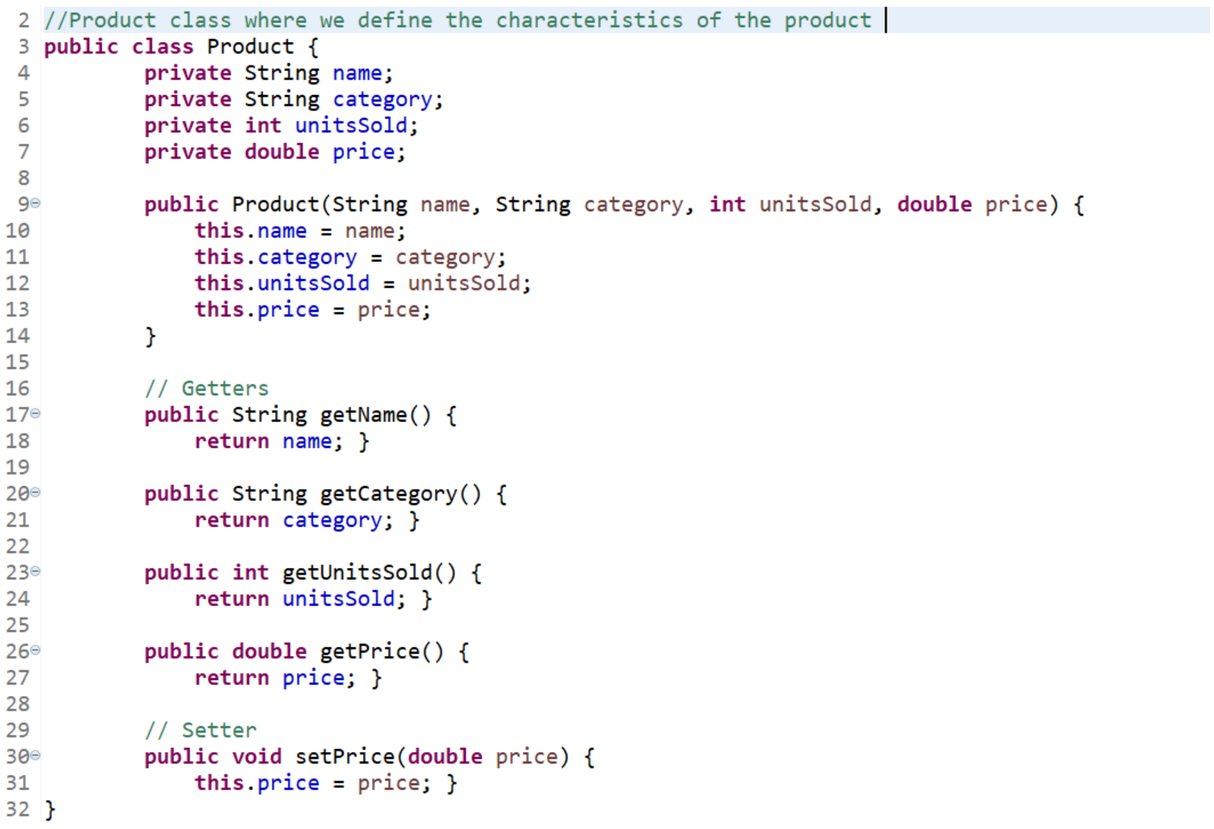
Sales and Products Analysis

Introduction

This document explains a Java program that processes a list of products to identify the best sellers in each category using Java Streams. The code performs several operations to filter, peek, sorted, flatmap, and collect product information.

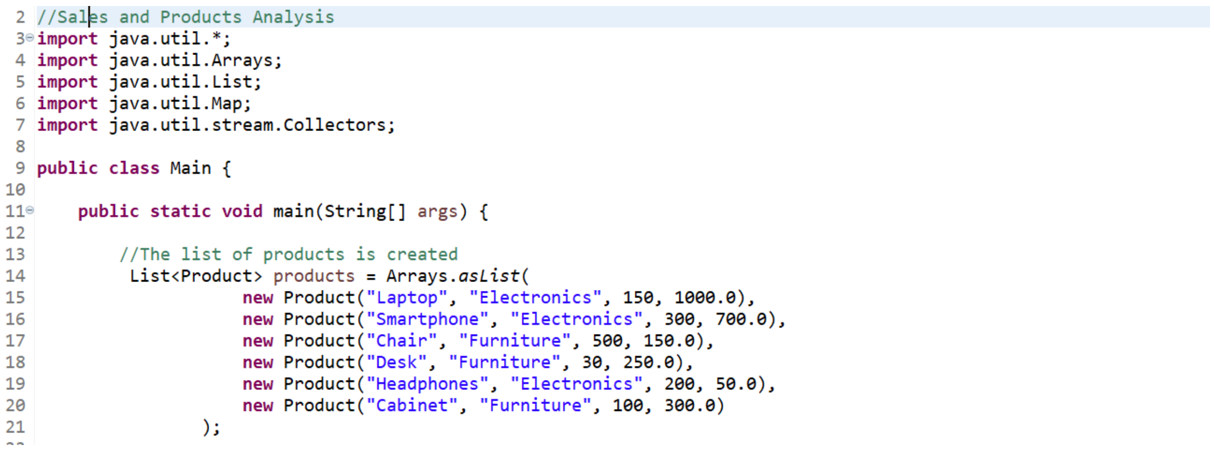
Code

First we create a class called Product used to define the characteristics of a product. It includes fields to store product information, a constructor to initialize these fields, and methods to access and modify the product's data.



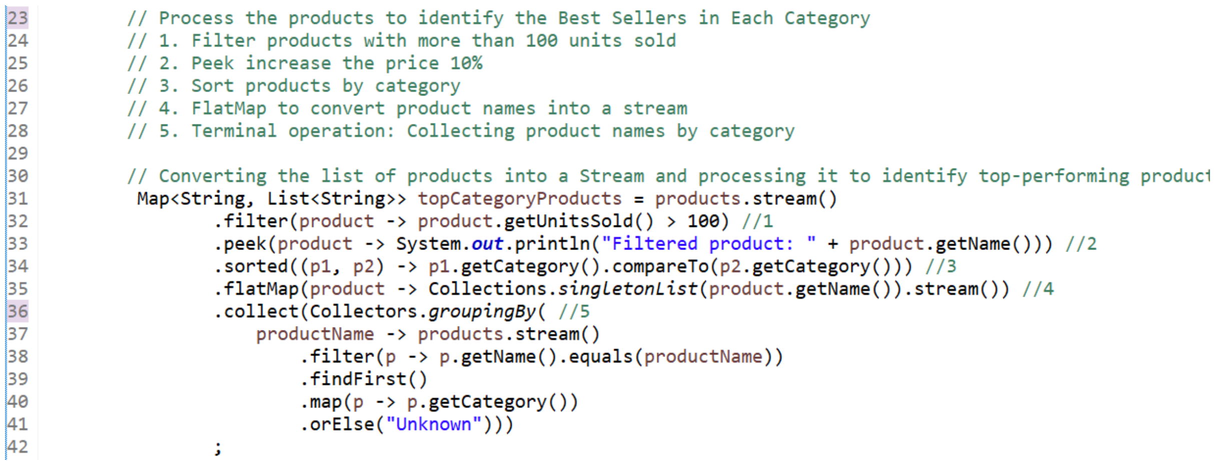
```java
//Product class where we define the characteristics of the product
public class Product {
        private String name;
        private String category;
        private int unitsSold;
        private double price;

        public Product(String name, String category, int unitsSold, double price) {
            this.name = name;
            this.category = category;
            this.unitsSold = unitsSold;
            this.price = price;
        }

        // Getters
        public String getName() {
            return name; }

        public String getCategory() {
            return category; }

        public int getUnitsSold() {
            return unitsSold; }

        public double getPrice() {
            return price; }

        // Setter
        public void setPrice(double price) {
            this.price = price; }
}
```

Then in the main class a List of Product objects is created, each having a name, category, units sold, and price.



```java
//Sales and Products Analysis
import java.util.*;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

public class Main {

    public static void main(String[] args) {

        //The list of products is created
         List<Product> products = Arrays.asList(
                    new Product("Laptop", "Electronics", 150, 1000.0),
                    new Product("Smartphone", "Electronics", 300, 700.0),
                    new Product("Chair", "Furniture", 500, 150.0),
                    new Product("Desk", "Furniture", 30, 250.0),
                    new Product("Headphones", "Electronics", 200, 50.0),
                    new Product("Cabinet", "Furniture", 100, 300.0)
                );
```

In line 31 we start by creating a stream from the products list, with products.stream(), we use the Map interface for the data structure that stores key-value pairs.



```java
23         // Process the products to identify the Best Sellers in Each Category
24         // 1. Filter products with more than 100 units sold
25         // 2. Peek increase the price 10%
26         // 3. Sort products by category
27         // 4. FlatMap to convert product names into a stream
28         // 5. Terminal operation: Collecting product names by category
29 
30         // Converting the list of products into a Stream and processing it to identify top-performing product
31          Map<String, List<String>> topCategoryProducts = products.stream()
32                 .filter(product -> product.getUnitsSold() > 100) //1
33                 .peek(product -> System.out.println("Filtered product: " + product.getName())) //2
34                 .sorted((p1, p2) -> p1.getCategory().compareTo(p2.getCategory())) //3
35                 .flatMap(product -> Collections.singletonList(product.getName()).stream()) //4
36                 .collect(Collectors.groupingBy( //5
37                     productName -> products.stream()
38                         .filter(p -> p.getName().equals(productName))
39                         .findFirst()
40                         .map(p -> p.getCategory())
41                         .orElse("Unknown")))
42                 ;
```

Then from lines 32-35 we apply the Intermediate Operations: filter, peek, sorted, and flatmap.

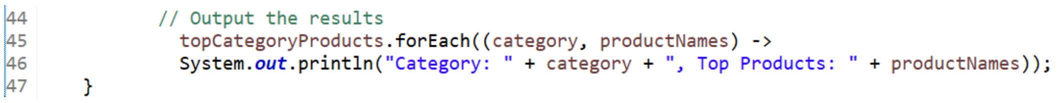
- .filter operation keeps only those Product objects where the number of units sold is greater than 100 and help us filter the stream of products.
- .peek operation was used for printing the filtered products.
- .sorted sorts the products by their category in ascending order.
    - Sorting products by category ensures that products with the same category are grouped together in the stream.
    - Having products sorted by category simplifies the grouping process and subsequent analysis, such as creating a map that classifies products by category.
- .flatmap The flatMap operation transforms each element of the stream into another stream and then flattens the results into a single stream.

- .flatMap(product -> Collections.singletonList(product.getName()).stream())
- Collections.singletonList(product.getName()): Creates a List containing a single element, which is the name of the product.
- .stream(): Converts the list containing the product's name into a Stream<String>.
- The flatMap operation takes all these individual streams (one for each product) and flattens them into a single stream of product names.

Then from the line 36- 41 we group by category:

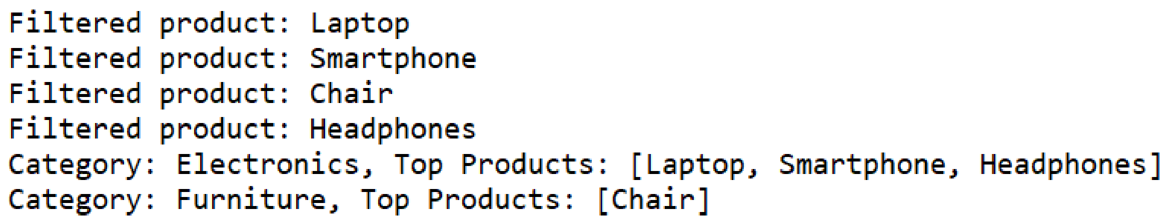
- The collect operation gathers the stream elements into a collection, in this case, a Map. Collectors.groupingBy creates a Map where each key is a category, and the value is a list of product names that belong to that category.
- Classifier Function: The lambda function determines the category for each product name by looking it up in the original products list. If the name is not found, it defaults to "Unknown".
- productName ->: products.stream():
    - Converts the original products list into a stream. This allows us to search through the list to find information about each product.
- .filter(p -> p.getName().equals(productName)):
    - Filters the products stream to find the product that matches the productName.
- .findFirst():
    - Retrieves the first product that matches the filter condition.
- .map(p -> p.getCategory()):
    - Maps the found product to its category if a product was found.
- .orElse("Unknown"):
    - Provides a default value if no product was found in the previous steps.

The lines from 44-47 are for printing the filtered stream



```
44          // Output the results
45            topCategoryProducts.forEach((category, productNames) ->
46            System.out.println("Category: " + category + ", Top Products: " + productNames));
47      }
```

As an output we obtain



```
Filtered product: Laptop
Filtered product: Smartphone
Filtered product: Chair
Filtered product: Headphones
Category: Electronics, Top Products: [Laptop, Smartphone, Headphones]
Category: Furniture, Top Products: [Chair]
```

The filtered products are the ones obtained by the peek method and the last two lines printed are from the lines 44-47, that is the list filtered with the best selling products and the category.