Karen Giselle Valdez Muñoz

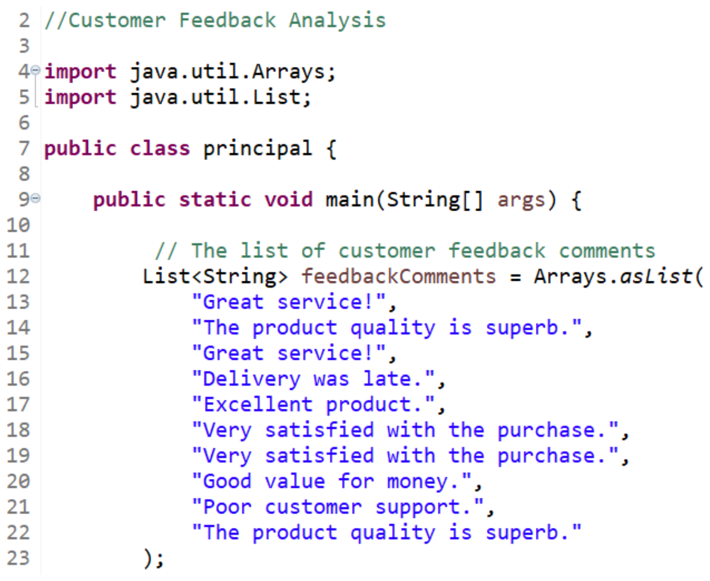
Customer Feedback Analysis

Introduction

In this code, we analyze customer feedback by processing a list of comments using Java Streams. The goal is to filter out comments that are too short, remove duplicates, sort the comments, and finally print each unique, sorted comment. The code employs various stream operations to achieve this.

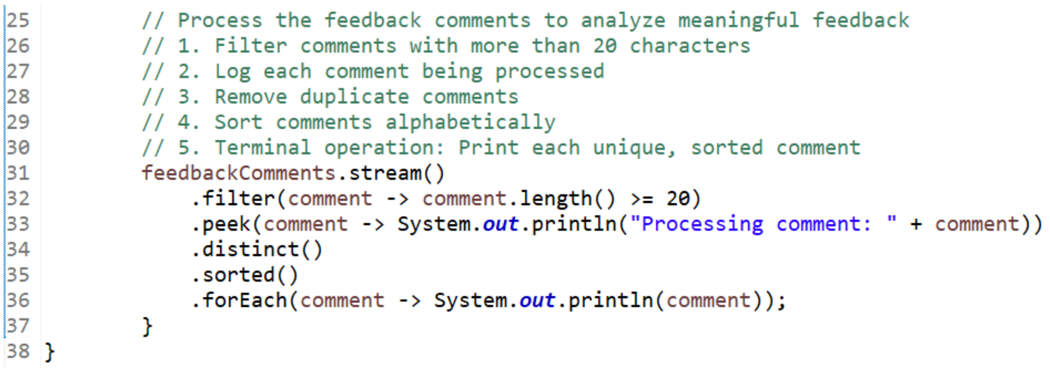
Code

We start initializing a list of customer feedback comments using Arrays.asList(...), which creates a fixed-size list of strings. Each string in the list represents an individual customer comment, including both positive and negative feedback. Some comments are repeated, which is intentional as the code later processes these comments to filter, log, remove duplicates, and sort them. The list feedbackComments is the primary data source that will be manipulated in the subsequent stream operations, forming the foundation for the analysis performed in the program.



```java
//Customer Feedback Analysis

import java.util.Arrays;
import java.util.List;

public class principal {

    public static void main(String[] args) {

         // The list of customer feedback comments
        List<String> feedbackComments = Arrays.asList(
            "Great service!",
            "The product quality is superb.",
            "Great service!",
            "Delivery was late.",
            "Excellent product.",
            "Very satisfied with the purchase.",
            "Very satisfied with the purchase.",
            "Good value for money.",
            "Poor customer support.",
            "The product quality is superb."
        );
```

Then we process the feedback comments to analyze meaningful feedback.



```java
        // Process the feedback comments to analyze meaningful feedback
        // 1. Filter comments with more than 20 characters
        // 2. Log each comment being processed
        // 3. Remove duplicate comments
        // 4. Sort comments alphabetically
        // 5. Terminal operation: Print each unique, sorted comment
        feedbackComments.stream()
            .filter(comment -> comment.length() >= 20)
            .peek(comment -> System.out.println("Processing comment: " + comment))
            .distinct()
            .sorted()
            .forEach(comment -> System.out.println(comment));
        }
}
```

feedbackComments.stream(): Converts the list of customer feedback comments (feedbackComments) into a stream, allowing us to perform a sequence of operations on the data.

1. .filter(comment -> comment.length() >= 20): Filters the comments based on their length, only allowing comments that have 20 or more characters to pass through. (Intermediate Operation)
2. .peek(comment -> System.out.println("Processing comment: " + comment)): Logs each comment that passes through the stream after filtering. (Intermediate Operation)
3. .distinct(): Removes duplicate comments from the stream. (Intermediate Operation)
4. .sorted(): Sorts the comments alphabetically. (Intermediate Operation)
5. .forEach(comment -> System.out.println(comment)): Prints each unique, sorted comment to the console. (Terminal Operation)

Conclusion

This code effectively processes customer feedback by filtering out irrelevant comments, logging each relevant comment, removing duplicates, sorting the comments, and finally printing them. It demonstrates the power of Java Streams in handling collections of data in a clear and concise manner.