Copilot

Let's consider a real-time example to better understand the code. Imagine we have a list of customer feedback records, and we want to clean this data by removing records with null values, eliminating duplicates, and correcting any errors in the feedback text.

Step-by-Step Example

1. **Define the `FeedbackRecord` and `CleansedRecord` classes**:

```
class FeedbackRecord {
    private String customerId;
    private String feedback;
    private Integer rating;

    // Constructor
    public FeedbackRecord(String customerId, String feedback, Integer rating) {
        this.customerId = customerId;
        this.feedback = feedback;
        this.rating = rating;
    }

    // Getters
    public String getCustomerId() { return customerId; }
    public String getFeedback() { return feedback; }
    public Integer getRating() { return rating; }
}

class CleansedRecord {
    private String customerId;
    private String correctedFeedback;
    private Integer rating;

    // Constructor
    public CleansedRecord(String customerId, String correctedFeedback, Integer rating) {
        this.customerId = customerId;
        this.correctedFeedback = correctedFeedback;
        this.rating = rating;
    }

    @Override
    public String toString() {
        return "CleansedRecord{" +
                "customerId='" + customerId + '\'' +
                ", correctedFeedback='" + correctedFeedback + '\'' +
                ", rating=" + rating +
                '}';
    }
}
```

1. **Implement the `correctErrors` method**:

```
private static FeedbackRecord correctErrors(FeedbackRecord record) {
    // Implement error correction logic here
    String correctedFeedback = record.getFeedback().replaceAll("bad", "not good");
    return new FeedbackRecord(record.getCustomerId(), correctedFeedback, record.getRating());
}
```

1. **Create a list of `FeedbackRecord` objects and cleanse the data**:

```
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
```

```java
public class Main {
    public static void main(String[] args) {
        // Sample raw data
        List<FeedbackRecord> rawData = Arrays.asList(
            new FeedbackRecord("C001", "Service was bad", 2),
            new FeedbackRecord("C002", "Excellent service", 5),
            new FeedbackRecord("C003", null, 3),
            new FeedbackRecord("C001", "Service was bad", 2)
        );

        // Cleanse the data
        List<CleansedRecord> cleansedData = rawData.stream()
            .filter(record -> record.getFeedback() != null)
            .distinct()
            .map(record -> new CleansedRecord(
                record.getCustomerId(),
                correctErrors(record).getFeedback(),
                record.getRating()
            ))
            .collect(Collectors.toList());

        // Print the cleansed records
        cleansedData.forEach(System.out::println);
    }

    private static FeedbackRecord correctErrors(FeedbackRecord record) {
        String correctedFeedback = record.getFeedback().replaceAll("bad", "not good");
        return new FeedbackRecord(record.getCustomerId(), correctedFeedback, record.getRating());
    }
}
```

Explanation

- **rawData**: A list of `FeedbackRecord` objects with sample values.
- **Stream Processing**:
  - **filter(record -> record.getFeedback() != null)**: Removes records with null feedback.
  - **distinct()**: Eliminates duplicate records.
  - **map(record -> new CleansedRecord(…))**: Transforms each `FeedbackRecord` into a `CleansedRecord` with corrected feedback.
- **Collecting**: The cleansed `CleansedRecord` objects are collected into a list.
- **Printing**: The resulting `CleansedRecord` objects are printed to the console.

Output

The output will look something like this:

```
CleansedRecord{customerId='C001', correctedFeedback='Service was not good', rating=2}
CleansedRecord{customerId='C002', correctedFeedback='Excellent service', rating=5}
```

This example should help you understand how the code cleanses and transforms raw feedback data using Java Streams. If you have any more questions or need further clarification, feel free to ask!