Task: Working with Streams in Java

Objective: Create a Java program to manage and analyze a list of `Product` objects using streams.

Instructions

1. **Create a `Product` Class**:
   - Fields:
     - `String name`
     - `String category`
     - `double price`
     - `int quantity`
   - Constructor to initialize all fields.
   - Getters for each field.
   - Override the `toString` method for a readable representation of a product.
2. **Main Program**:
   - In the `App.java` file, perform the following:
     1. **Initialize Data**:
        - Create an `ArrayList<Product>` and add at least 6 products, ensuring they belong to at least 3 different categories.
     2. **Perform Stream Operations**:
        - **Display all products.**
        - **Filter**: Create a list of products where the price is greater than 50.
        - **Map**: Create a new list of product names only.
        - **Group**: Group products by category.
        - **Reduce**: Calculate the total value of all products (price × quantity).
     3. **Additional Challenges (Optional)**:
        - Find the product with the highest price.
        - Sort products by price in ascending order and display them.
3. **Testing**:
   - Include a basic test file (`AppTest.java`) that verifies at least one operation (e.g., filtering or total value calculation).

Expected Outcome

- The program should demonstrate the ability to:
  - Filter, map, and group data using streams.
  - Work with `reduce` for calculations.
  - Present data in a clear and readable format.

Explanation

Streams do not change the original (raw) data. Instead, they produce a new result or collection, leaving the original data intact. This is one of the key design principles of streams—**immutability**.

Why Does Nothing Change in the Raw Data?

1. **Immutability of Streams**:
   - Streams are designed to process data without altering the source. Instead of modifying the input data, streams create new objects or results. This makes the code predictable and safe, especially in multi-threaded environments.
2. **Functional Programming Approach**:
   - Streams follow the principles of functional programming, which avoids side effects (changes to existing data). This helps maintain **data integrity** and ensures the original data remains reusable.

What Happens During Stream Operations?

Here's a breakdown of what happens:

1. **Raw Data**:
   - You start with your original collection (e.g., `ArrayList` of numbers or students).
2. **Stream Pipeline**:
   - Operations like `map`, `filter`, `reduce`, etc., **do not alter the raw data**. They create a new result based on the stream's processing.
3. **New Results**:
   - The results are either collected into a new collection or returned as a single value.

For example:

```
List<Integer> numbers = Arrays.asList(20, 10, 5, 7, 15, 42);

// Original list stays the same
System.out.println("Original: " + numbers);

// New list after mapping
List<Integer> multipliedBy2 = numbers.stream()
                                      .map(n -> n * 2)
                                      .toList();
System.out.println("New List: " + multipliedBy2);
```

Output:

```
Original: [20, 10, 5, 7, 15, 42]
New List: [40, 20, 10, 14, 30, 84]
```

Benefits of Immutability in Streams

1. **Reusability**:
   - The original data can be reused for other operations without worrying about unintended changes.
2. **Thread-Safety**:
   - Since the raw data isn’t modified, streams are inherently safer to use in parallel operations (`parallelStream`).

3. **Predictability**:
    - When raw data doesn’t change, the logic becomes easier to debug and understand. The original input remains consistent.
4. **Clear Separation**:
    - There's a clear distinction between **input** (original collection) and **output** (stream results), making the flow of data cleaner.

Analogy

Imagine you have a bucket of apples. When you apply stream operations, you're not taking apples out of the original bucket and altering them. Instead:

1. You **copy apples** (apply operations like filtering or slicing).
2. You put the **processed apples in a new basket** (result collection).
3. The **original bucket stays untouched**, so you can use it again.