

 AndersonLima0

fix DataClass code smell using the move method refactoring technique

 Reference.java:60-97

 GitHub Copilot

> Used 1 reference

Let's fix the Data Class code smell in `Reference.java` by applying the Move Method refactoring technique. Here's the step-by-step plan:

1. Identify behaviors that should be moved into the Reference class
2. Add methods that encapsulate behavior rather than just get/set data
3. Consider common operations that would use this data together

Here's the refactored code:

```
public class Reference {  
    // ...existing code...  
  
    public void incrementViewCount() {  
        this.viewCount++;  
    }  
  
    public void incrementDownloadCount() {  
        this.downloadCount++;  
    }  
  
    public void incrementShareCount() {  
        this.shareCount++;  
    }  
  
    public boolean isPopular() {  
        return this.viewCount > 1000 || this.downloadCount > 100;  
    }  
  
    public boolean isHighlyRated() {  
        return this.rating >= 4;  
    }  
  
    public boolean isShareWorthy() {  
        return isHighlyRated() && isPopular();  
    }  
}
```

```

    public void incrementViewCount() {
        this.viewCount++;
    }

    public void incrementDownloadCount() {
        this.downloadCount++;
    }

    public void incrementShareCount() {
        this.shareCount++;
    }

    public boolean isPopular() {
        return this.viewCount > 1000 || this.downloadCount > 100;
    }

    public boolean isHighlyRated() {
        return this.rating >= 4;
    }

    public boolean isShareWorthy() {
        return isHighlyRated() && isPopular();
    }

    public String getEngagementSummary() {
        return String.format("Views: %d, Downloads: %d, Shares: %d, Rating: %d",
            viewCount, downloadCount, shareCount, rating);
    }

    // Keep essential getters/setters but prefer using behavior methods above
    // ...existing code...
}

```

Changes made:

- Added increment methods for counts instead of direct setters
- Added methods that combine data points to provide meaningful behaviors
- Introduced business logic methods that encapsulate decision making
- Maintained necessary getters/setters but emphasized behavior-focused methods

This refactoring moves the class from being a pure data holder to having meaningful behaviors and responsibilities.