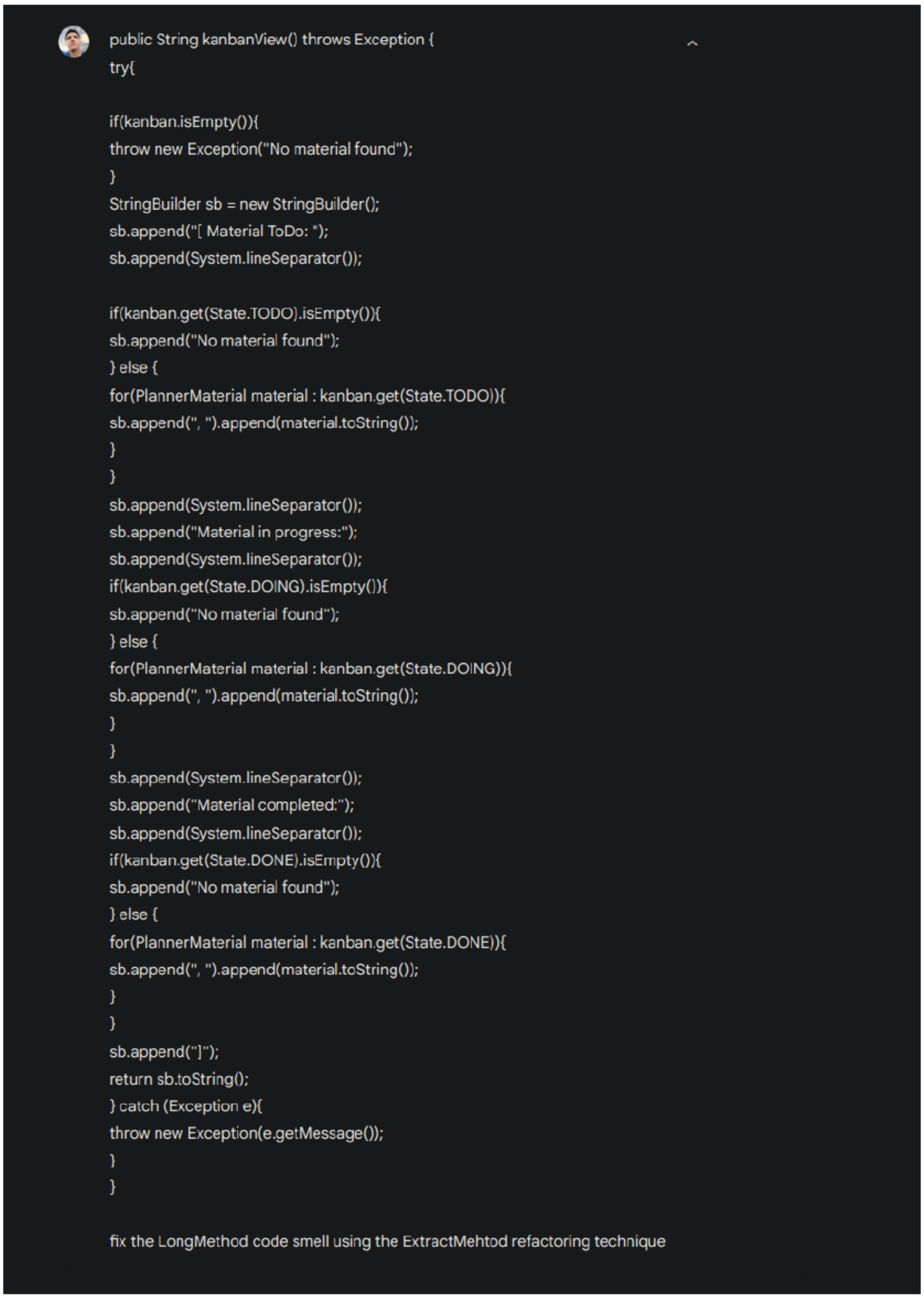
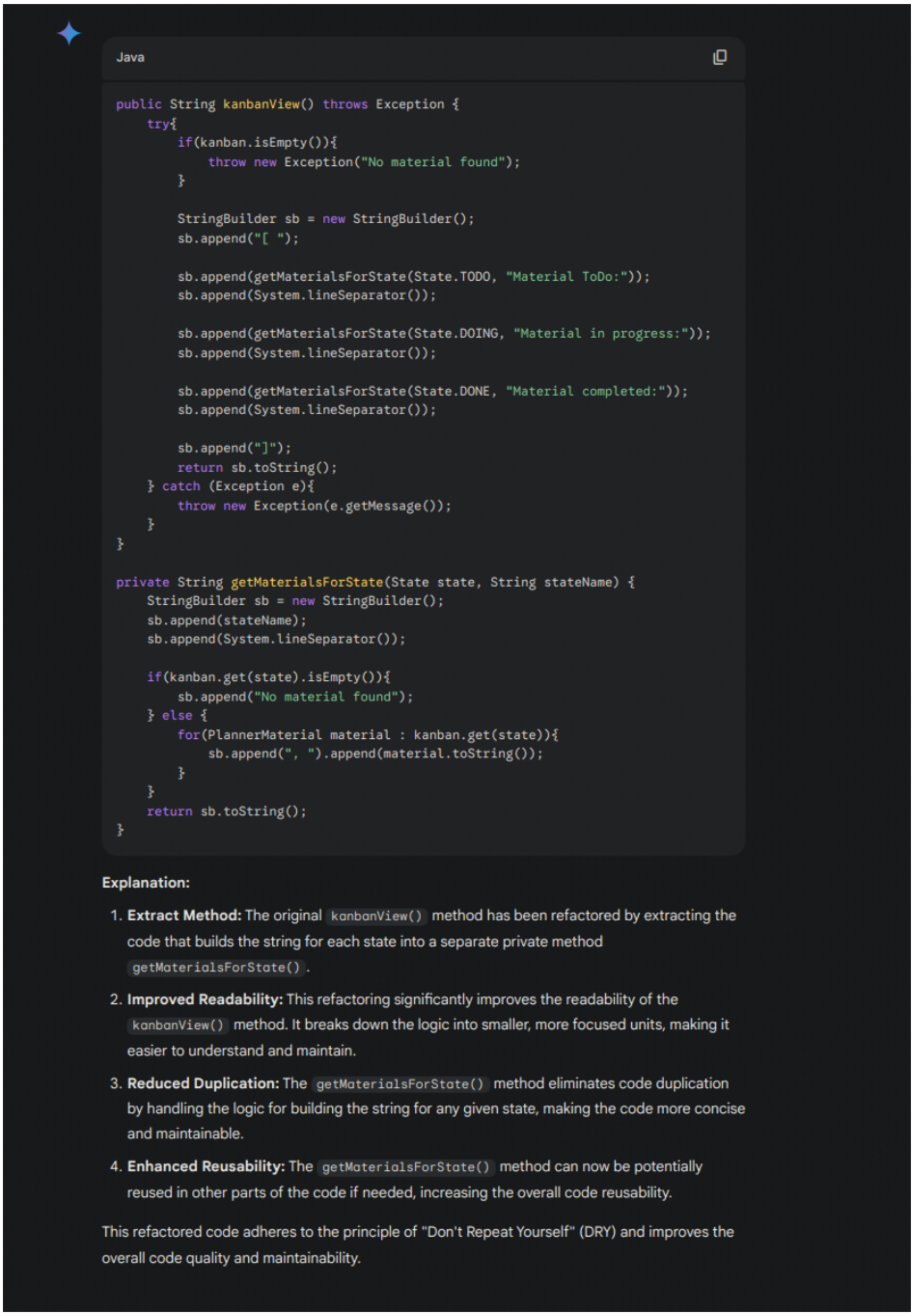
Long Method

KanbanView



```
public String kanbanView() throws Exception {
try{

if(kanban.isEmpty()){
throw new Exception("No material found");
}
StringBuilder sb = new StringBuilder();
sb.append("[ Material ToDo: ");
sb.append(System.lineSeparator());

if(kanban.get(State.TODO).isEmpty()){
sb.append("No material found");
} else {
for(PlannerMaterial material : kanban.get(State.TODO)){
sb.append(", ").append(material.toString());
}
}
sb.append(System.lineSeparator());
sb.append("Material in progress:");
sb.append(System.lineSeparator());
if(kanban.get(State.DOING).isEmpty()){
sb.append("No material found");
} else {
for(PlannerMaterial material : kanban.get(State.DOING)){
sb.append(", ").append(material.toString());
}
}
sb.append(System.lineSeparator());
sb.append("Material completed:");
sb.append(System.lineSeparator());
if(kanban.get(State.DONE).isEmpty()){
sb.append("No material found");
} else {
for(PlannerMaterial material : kanban.get(State.DONE)){
sb.append(", ").append(material.toString());
}
}
sb.append("]");
return sb.toString();
} catch (Exception e){
throw new Exception(e.getMessage());
}
}
```

fix the LongMethod code smell using the ExtractMehtod refactoring technique



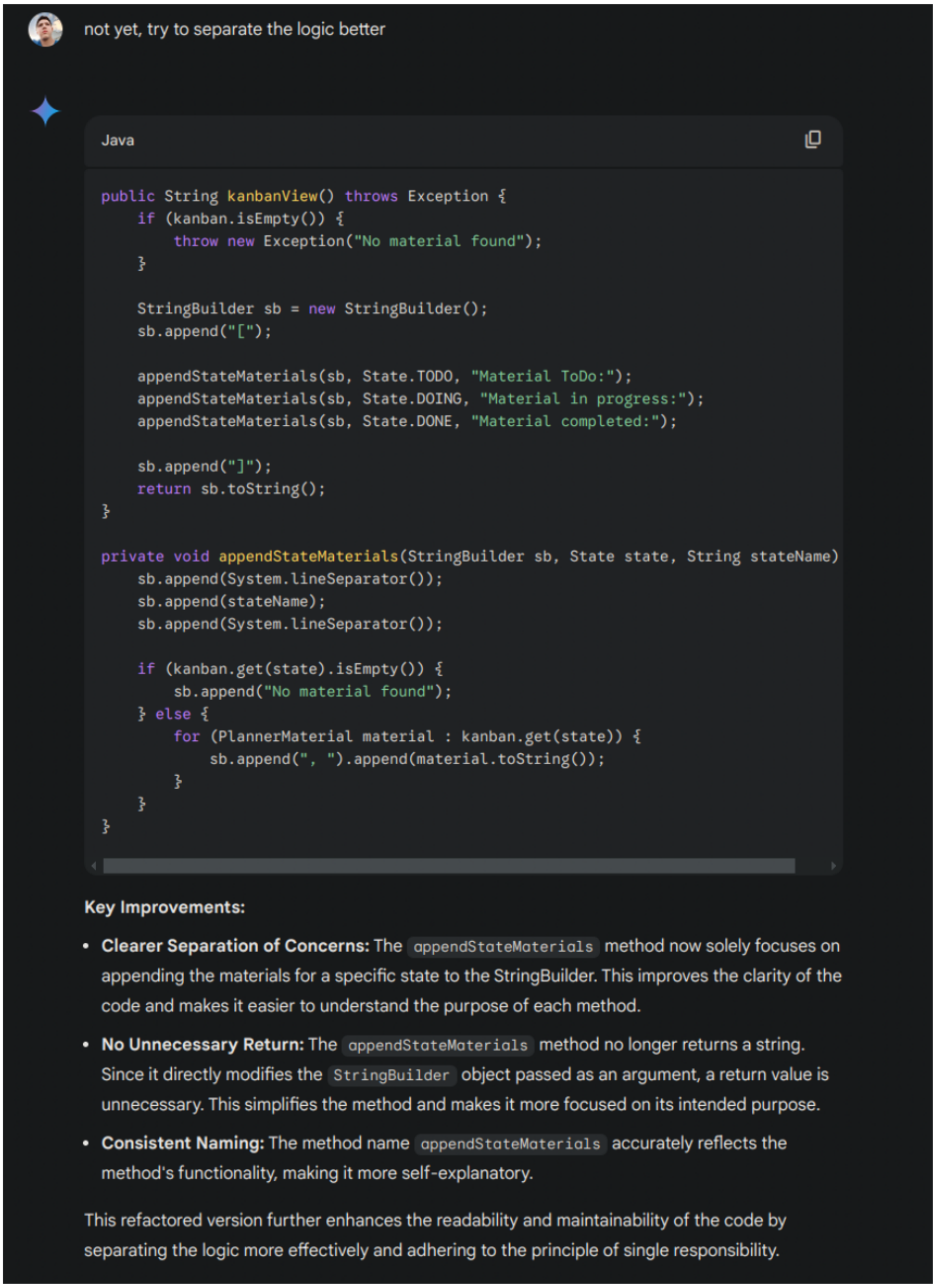
Java

```java
public String kanbanView() throws Exception {
    try{
        if(kanban.isEmpty()){
            throw new Exception("No material found");
        }

        StringBuilder sb = new StringBuilder();
        sb.append("[ ");

        sb.append(getMaterialsForState(State.TODO, "Material ToDo:"));
        sb.append(System.lineSeparator());

        sb.append(getMaterialsForState(State.DOING, "Material in progress:"));
        sb.append(System.lineSeparator());

        sb.append(getMaterialsForState(State.DONE, "Material completed:"));
        sb.append(System.lineSeparator());

        sb.append("]");
        return sb.toString();
    } catch (Exception e){
        throw new Exception(e.getMessage());
    }
}

private String getMaterialsForState(State state, String stateName) {
    StringBuilder sb = new StringBuilder();
    sb.append(stateName);
    sb.append(System.lineSeparator());

    if(kanban.get(state).isEmpty()){
        sb.append("No material found");
    } else {
        for(PlannerMaterial material : kanban.get(state)){
            sb.append(", ").append(material.toString());
        }
    }
    return sb.toString();
}
```

Explanation:

1. **Extract Method:** The original `kanbanView()` method has been refactored by extracting the code that builds the string for each state into a separate private method `getMaterialsForState()`.
2. **Improved Readability:** This refactoring significantly improves the readability of the `kanbanView()` method. It breaks down the logic into smaller, more focused units, making it easier to understand and maintain.
3. **Reduced Duplication:** The `getMaterialsForState()` method eliminates code duplication by handling the logic for building the string for any given state, making the code more concise and maintainable.
4. **Enhanced Reusability:** The `getMaterialsForState()` method can now be potentially reused in other parts of the code if needed, increasing the overall code reusability.

This refactored code adheres to the principle of "Don't Repeat Yourself" (DRY) and improves the overall code quality and maintainability.



not yet, try to separate the logic better

Java

```java
public String kanbanView() throws Exception {
    if (kanban.isEmpty()) {
        throw new Exception("No material found");
    }

    StringBuilder sb = new StringBuilder();
    sb.append("[");

    appendStateMaterials(sb, State.TODO, "Material ToDo:");
    appendStateMaterials(sb, State.DOING, "Material in progress:");
    appendStateMaterials(sb, State.DONE, "Material completed:");

    sb.append("]");
    return sb.toString();
}

private void appendStateMaterials(StringBuilder sb, State state, String stateName)
    sb.append(System.lineSeparator());
    sb.append(stateName);
    sb.append(System.lineSeparator());

    if (kanban.get(state).isEmpty()) {
        sb.append("No material found");
    } else {
        for (PlannerMaterial material : kanban.get(state)) {
            sb.append(", ").append(material.toString());
        }
    }
}
```

Key Improvements:

- **Clearer Separation of Concerns:** The `appendStateMaterials` method now solely focuses on appending the materials for a specific state to the StringBuilder. This improves the clarity of the code and makes it easier to understand the purpose of each method.
- **No Unnecessary Return:** The `appendStateMaterials` method no longer returns a string. Since it directly modifies the `StringBuilder` object passed as an argument, a return value is unnecessary. This simplifies the method and makes it more focused on its intended purpose.
- **Consistent Naming:** The method name `appendStateMaterials` accurately reflects the method's functionality, making it more self-explanatory.

This refactored version further enhances the readability and maintainability of the code by separating the logic more effectively and adhering to the principle of single responsibility.