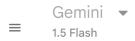
13/01/2025, 16:36 Gemini



Testar o Gemini Advanced



fix the code smell Feature envy using the move method

public class TimelineView {...



Certainly, let's address the "Feature Envy" code smell in your TimelineView class by using the "Move Method" refactoring technique.

1. Identify the Envious Method:

 The habitDateViewAll() method in TimelineView exhibits feature envy because it relies heavily on methods of the HabitTracker class (getHabits(), getHabitRecords(), formatHabitDate()) to perform its core functionality.

2. Move the Envious Behavior:

• Create a new method within the HabitTracker class to encapsulate the logic of generating the habit date view.

3. Refactored Code:

```
Java
public class TimelineView {
    public String habitDateViewAll(HabitTracker ht){
        return ht.getHabitDatesView();
3
public class HabitTracker {
    private List<Habit> habits;
    private Map<Integer, List<LocalDateTime>> tracker;
    private Integer nextId;
    private static HabitTracker instance;
    public String getHabitDatesView() {
        StringBuilder response = new StringBuilder();
        for(Habit habit: habits) {
            response.append("[ Habit: ")
                    .append(habit.getName())
                    .append(". Records: ");
            List<LocalDateTime> records = tracker.get(habit.getId());
            for(LocalDateTime record : records){
                response.append(formatHabitDate(record)).append(", ");
            3
```

13/01/2025, 16:36 Gemini

```
response.append("]");

Gemini ▼

1.5 Flash

Testar o Gemini Advanced

private String formatHabitDate(LocalDateTime record) {

// Implementation of formatHabitDate
}
```

Explanation:

- The habitDateViewAll() method in TimelineView now simply delegates the responsibility of generating the view to the getHabitDatesView() method in HabitTracker.
- The getHabitDatesView() method within HabitTracker encapsulates the logic for building the habit date view, utilizing its own data and methods.

Benefits of this Refactoring:

- Improved Cohesion: The HabitTracker class now has better cohesion as it encapsulates all the logic related to habit tracking, including the generation of its own views.
- **Reduced Coupling:** The TimelineView class is now less tightly coupled to the internal implementation details of HabitTracker.
- Increased Maintainability: Changes to how habit dates are formatted or how habits are stored can be made within HabitTracker without affecting TimelineView.
- Improved Readability: The TimelineView class becomes more concise and easier to understand as it delegates the complex logic to the responsible class.

This refactoring effectively addresses the "Feature Envy" code smell and improves the overall design and maintainability of your code.

少 夕 C % :

Powered by Gemini All Chat Downloader