

TimeLineView

QualidadeDeSoftware > Feature Envy Refactoring 1 ▾

fix the Feature Envy code smell using the move method refactoring technique

```
package org.example.studyplanner;

import java.time.LocalDateTime;
import java.util.List;

public class TimelineView {

    public String habitDateViewAll(HabitTracker ht){
        List<Habit> habits = ht.getHabits();
        StringBuilder response = new StringBuilder();
        for(Habit habit : habits){
            response.append("[ Habit: ")
                .append(habit.getName())
                .append(". Records: ");
            List<LocalDateTime> records =
                ht.getHabitRecords(habit.getId());
            for(LocalDateTime record : records){
                response.append(ht.formatHabitDate(record)).append(" ");
            }
        }
    }
}
```

Envie uma mensagem para o ChatGPT



O ChatGPT pode cometer erros. Considere verificar informações importantes.

QualidadeDeSoftware > Feature Envy Refactoring 1 ▾

Here's a refactored version of the code:

Refactored Code

HabitTracker Class (with new methods)

java

Copiar código

```
package org.example.studyplanner;

import java.time.LocalDateTime;
import java.util.List;

public class HabitTracker {

    // Existing methods...

    public String formatHabitWithRecords(Habit habit) {
        StringBuilder response = new StringBuilder();
        response.append("[ Habit: ")
            .append(habit.getName())
            .append(". Records: ");
        List<LocalDateTime> records = getHabitRecords(habit.getId());
        for(LocalDateTime record : records){
            response.append(formatHabitDate(record)).append(" ");
        }
    }
}
```

Feature Envy StudySearch

QualidadeDeSoftware > Feature Envy Refactoring 2

fix the Feature Envy code smell using the move method refactoring technique without adding a class, just changing what is in the move to and method

```
package org.example.studysearch;

import org.example.studycards.CardManager;
import org.example.studyplanner.HabitTracker;
import org.example.studyplanner.TODOTracker;
import org.example.studyregistry.StudyMaterial;
import org.example.studyregistry.StudyTaskManager;

import java.util.ArrayList;
import java.util.List;

public class GeneralSearch implements Search<String> {
    private SearchLog searchLog = new SearchLog("General Search");

    public GeneralSearch() {}

    @Override
```

QualidadeDeSoftware > Feature Envy Refactoring 2

extensively. Here's the refactored code:

Refactored Code

java Copiar código

```
package org.example.studysearch;

import org.example.studycards.CardManager;
import org.example.studyplanner.HabitTracker;
import org.example.studyplanner.TODOTracker;
import org.example.studyregistry.StudyMaterial;
import org.example.studyregistry.StudyTaskManager;

import java.util.ArrayList;
import java.util.List;

public class GeneralSearch implements Search<String> {
    private SearchLog searchLog = new SearchLog("General Search");

    public GeneralSearch() {}

    @Override
```

Feature Envy MaterialSearch

QualidadeDeSoftware > Feature Envy Refactoring 3 ▾

fix the Feature Envy code smell using the move method refactoring technique without adding a class, making changes only to the "method" file and adding more things to the class that is already being used

```
package org.example.studysearch;

import org.example.studyregistry.StudyMaterial;

import java.util.ArrayList;
import java.util.List;

public class MaterialSearch implements Search<String>{

    private SearchLog searchLog = new SearchLog("Material Search");

    public MaterialSearch() {}

    @Override
    public List<String> search(String text) {
```

QualidadeDeSoftware > Feature Envy Refactoring 3 ▾



To fix the **Feature Envy** code smell using the **Move Method** refactoring technique, the `handleMaterialSearch` method from `MaterialSearch` can be moved into the `SearchLog` class. This minimizes the reliance of `MaterialSearch` on `SearchLog` for search handling logic. Below is the refactored code:

Updated `MaterialSearch` Class

java

Copiar código

```
package org.example.studysearch;

import java.util.List;

public class MaterialSearch implements Search<String> {

    private SearchLog searchLog = new SearchLog("Material Search");

    public MaterialSearch() {}

    @Override
```

Feature Envy RegistrySearch

QualidadeDeSoftware > Feature Envy Refactoring 4 ▾

fix the Feature Envy code smell using the move method refactoring technique without adding a class, making changes only to the "method" file and adding more things to the class that is already being used

```
package org.example.studysearch;

import org.example.studycards.CardManager;
import org.example.studyplanner.HabitTracker;
import org.example.studyplanner.TODOTracker;
import org.example.studyregistry.StudyTaskManager;

import java.util.ArrayList;
import java.util.List;

public class RegistrySearch implements Search<String>{
    private SearchLog searchLog = new SearchLog("Registry Search");
    public RegistrySearch(){}

    @Override
    public List<String> search(String text) {
```

QualidadeDeSoftware > Feature Envy Refactoring 4 ▾

eliminate the Feature Envy smell. The `handleRegistrySearch` method's logic is moved to `SearchLog`, leveraging the existing functionality of `SearchLog`. No new classes are introduced, and the changes are restricted to the provided files.

Refactored RegistrySearch

```
java Copiar código

package org.example.studysearch;

import java.util.List;

public class RegistrySearch implements Search<String> {
    private SearchLog searchLog = new SearchLog("Registry Search");

    public RegistrySearch() {}

    @Override
    public List<String> search(String text) {
        // Delegating the search and logging responsibility to SearchLog
        return searchLog.performSearchWithLogging(text);
    }
}
```

Feature Envy LeitnerSystem

fix the Feature Envy code smell using the move method refactoring technique without adding a class, making changes only to the "method" file and adding more things to the class that is already being used

```
public String getRandomCardFromBox() {
    String response = "";
    response += leitnerSystem.getMethodName();
    List<Box> boxes = leitnerSystem.getBoxes();
    response += leitnerSystem.getRandomCard(boxes);
    return response;
}
```

```
package org.example.studycards;
```

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
```



To address the **Feature Envy** code smell in the `getRandomCardFromBox` method using the **Move Method** refactoring technique, the method can be moved to the `LeitnerSystem` class. This ensures that the logic operates closer to the data it depends on, thereby improving cohesion and reducing the dependency of one class on the internals of another.

Here's the updated `LeitnerSystem` class with the `getRandomCardFromBox` method added, and the refactored version of the original class:

Refactored `LeitnerSystem` Class

java

Copiar código

```
package org.example.studycards;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class LeitnerSystem extends StudyMethod {
    List<Box> boxes = null;
```