FEATURE ENVY

RegistrySearch

```
private List<String> handleRegistrySearch(String text){
List<String> results = new ArrayList<>();
results.addAll(CardManager.getCardManager().searchIn
Cards(text));
results.addAll(HabitTracker.getHabitTracker().searchInH
abits(text));
results.addAll(TodoTracker.getInstance().searchInTodos(
text));
results.addAll(StudyTaskManager.getStudyTaskManager
().searchInRegistries(text));
this.searchLog.addSearchHistory(text);
this.searchLog.setNumUsages(this.searchLog.getNumU
sages() + 1);
results.add("\nLogged in: " +
this.searchLog.getLogName());
return results;
fix the feature envy code smell using the "extract
```

Here's the refactored code using the "extract method" technique to address the Feature Envy code smell:

method" refactoring technique

```
0
Java
private List<String> handleRegistrySearch(String text) {
   List<String> results = new ArrayList<>();
    results.addAll(getAllSearchResults(text));
    this.searchLog.addSearchHistory(text);
    this.searchLog.setNumUsages(this.searchLog.getNumUsages() + 1);
    results.add("\nLogged in: " + this.searchLog.getLogName());
    return results;
private List<String> getAllSearchResults(String text) {
   List<String> results = new ArrayList<>();
    results.addAll(CardManager.getCardManager().searchInCards(text));
    results.addAll(HabitTracker.getHabitTracker().searchInHabits(text));
    results.addAll(TodoTracker.getInstance().searchInTodos(text));
    results.addAll(StudyTaskManager.getStudyTaskManager().searchInRegistries(text))
    return results:
3
```

Explanation:

- Feature Envy: The original handleRegistrySearch method exhibited Feature Envy because it was overly concerned with the logic of searching within different modules (CardManager, HabitTracker, etc.). This logic should ideally reside within those modules themselves.
- Extract Method: By extracting the search logic into a separate private method getAllSearchResults , we improve cohesion. The handleRegistrySearch method now focuses on its primary responsibility: handling the search results and updating the search log.
- · Benefits:
 - Improved Readability: The code becomes more modular and easier to understand.
 - Reduced Code Duplication: If the search logic needs to be reused in other parts of the application, it can be easily done by calling the getAllSearchResults method.
 - Increased Maintainability: Changes to the search logic within any of the modules (CardManager, HabitTracker, etc.) only require modifications to the getAllSearchResults method.

this.searchLog.addSearchHistory(text);

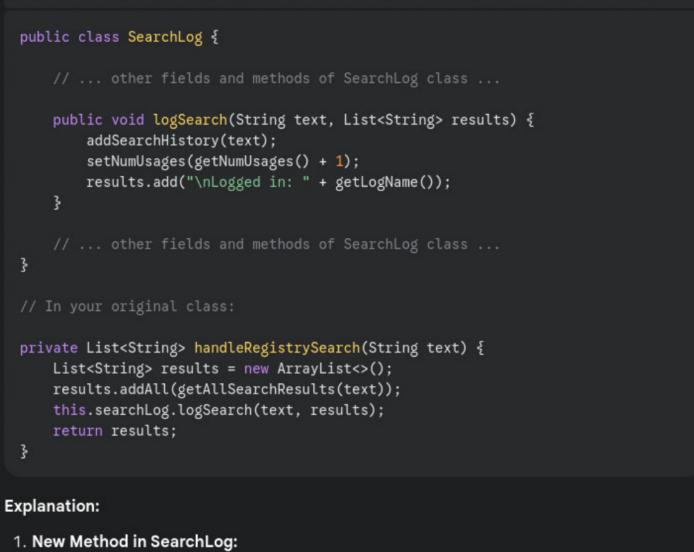
maintainability of the code.

0

This refactoring addresses the Feature Envy code smell and improves the overall design and

```
this.searchLog.setNumUsages(this.searchLog.getNumU
sages() + 1);
results.add("\nLogged in: " +
this.searchLog.getLogName());
now extract this lines to as new method in SearchLog
class
```

О



Java

- A new method logSearch is created within the SearchLog class.
 - This method encapsulates the logic of:
 - Adding the search text to the search history.
 - Incrementing the usage counter. · Adding the log-in information to the results list.

2. Call from handleRegistrySearch: In the handleRegistrySearch method, the call to

this.searchLog.addSearchHistory(text); and the subsequent lines are replaced with a single call to this.searchLog.logSearch(text, results); .

- Benefits:
- Improved Cohesion: The logic related to search logging is now properly placed within the SearchLog class, enhancing its responsibility and cohesion.
- application, it can be easily done by calling the logSearch method on the SearchLog instance. Increased Maintainability: Any changes to the search logging behavior (e.g., adding new log

Reduced Duplication: If the search logging logic needs to be reused in other parts of the

information) can be made within the SearchLog class without affecting other parts of the code.

This refactoring further improves the code's structure and maintainability by adhering to the principle of encapsulation and placing related functionalities within the appropriate class.