

Long Parameter List Audio Reference

QualidadeDeSoftware > Builder Pattern Refactoring audio reference 1 ▾

fix the Feature Long Parameter List smell using the builder pattern refactoring method

try not to create more code smells

```
package org.example.studymaterial;
```

```
import java.util.List;
```

```
public class AudioReference extends Reference {  
    public enum AudioQuality {  
        LOW, MEDIUM, HIGH, VERY_HIGH;  
    }  
    private AudioQuality audioQuality;
```

```
    public AudioReference(AudioQuality quality){  
        this.audioQuality = quality;  
    }
```

```
    public AudioQuality getAudioQuality() {
```

QualidadeDeSoftware > Builder Pattern Refactoring audio reference 1 ▾

LU



To resolve the **Feature Long Parameter List** smell using the **Builder Pattern**, we can refactor the `editAudio` method in the `AudioReference` class to encapsulate the parameters into a builder class. This approach improves readability and makes the code easier to maintain without introducing additional code smells.

Here's the refactored version of the code:

Updated `AudioReference` Class

java

Copiar código

```
package org.example.studymaterial;  
  
public class AudioReference extends Reference {  
    public enum AudioQuality {  
        LOW, MEDIUM, HIGH, VERY_HIGH;  
    }  
    private AudioQuality audioQuality;  
  
    public AudioReference(AudioQuality quality) {  
        this.audioQuality = quality;  
    }
```

Envie uma mensagem para o ChatGPT

Long Parameter List habit tracker

QualidadeDeSoftware > Refactoring HabitTracker Builder

fix the Feature Long Parameter List smell using the builder pattern refactoring use this method but without changing the behavior of other classes, okay?

```
package org.example.studyplanner;

import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.*;

public class HabitTracker {
    private List<Habit> habits;
    private Map<Integer, List<LocalDateTime>> tracker;
    private Integer nextId;

    private static HabitTracker instance;

    public static HabitTracker getHabitTracker() {
        if (instance == null) {
            instance = new HabitTracker();
        }
    }
}
```

Envie uma mensagem para o ChatGPT

QualidadeDeSoftware > Refactoring HabitTracker Builder

here's the refactored code.

Habit Tracker Refactor

```
1 // Builder Class for Habit
2 package org.example.studyplanner;
3
4 import java.time.LocalDateTime;
5 import java.time.LocalTime;
6
7 public class HabitBuilder {
8     private String name;
9     private String motivation;
10    private LocalTime dailyDedicationTime;
11    private Integer id;
12    private LocalDateTime startDate;
13    private Boolean isConcluded;
14}
```

Responder no chat >

Explanation of Changes

1. Added a `HabitBuilder` Class:

Encapsulates the habit creation process

Long Parameter List StudyObjective

QualidadeDeSoftware > Builder Pattern Refactoring ▾

fix the Feature Long Parameter List smell using the builder pattern refactoring use this method but without changing the behavior of other classes, okay?

```
package org.example.studyregistry;

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

public class StudyObjective extends Registry{
    private String title;
    private String description;
    private String topic;
    private Integer practicedDays;
    private LocalDateTime startDate;
    private Double duration;
    private String objectiveInOneLine;
    private String objectiveFullDescription;

    public String get...e() {
        return title;
    }
}
```

QualidadeDeSoftware > Builder Pattern Refactoring ▾



To address the "Feature Long Parameter List" smell using the Builder Pattern in the `Studyobjective` class without altering the behavior of other classes, follow these steps:

Refactored Code

Here's the refactored version of the `Studyobjective` class:

```
java Copiar código

package org.example.studyregistry;

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

public class StudyObjective extends Registry {
    private String title;
    private String description;
    private String topic;
    private Integer practicedDays;
    private LocalDateTime startDate;
    private Double duration;
}
```

Long Parameter List Study Task Manager

QualidadeDeSoftware > Refatoração Builder Pattern

fix the Feature Long Parameter List smell using the builder pattern refactoring method

Do not change existing methods

Resolve the problem sem gerar erros em outras classes

Reasons for the Problem

A long list of parameters might happen after several types of algorithms are merged in a single method. A long list may have been created to control which algorithm will be run and how.

Long parameter lists may also be the byproduct of efforts to make classes more independent of each other. For example, the code for creating specific objects needed in a method was moved from the method to the code for calling the method, but the created objects are passed to the method as parameters. Thus the original class no longer knows about the relationships between objects, and dependency has decreased. But if several of these objects are created, each of them will require its own parameter, which means a longer parameter list.

QualidadeDeSoftware > Refatoração Builder Pattern

java Copiar código

```
package org.example.studyregistry;

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

public class StudyTaskManager {
    private static StudyTaskManager instance;
    private StudyMaterial studyMaterial = StudyMaterial.getStudyMaterial();
    List<Registry> registryList;
    List<String> weekResponsibilities = List.of();

    private StudyTaskManager() {
        this.registryList = new ArrayList<>();
    }

    public static StudyTaskManager getStudyTaskManager() {
        if (instance == null) {
            instance = new StudyTaskManager();
        }
    }
}
```

Long Parameter List

QualidadeDeSoftware > Refatoração Builder Pattern ▾

But if these parameters are coming from different sources, you can pass them as a single parameter object via Introduce Parameter Object.


```
package org.example.studyregistry;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class StudyPlan extends Registry{
    private StudyObjective objective;
    private List<String> steps;


    public StudyPlan(String planName, StudyObjective objective,
List<StudyMaterial> materials) {
    this.name = planName;
    this.objective = objective;
    this.steps = new ArrayList<>();
```

QualidadeDeSoftware > Refatoração Builder Pattern ▾

 To address the Feature Long Parameter List smell in the `assignSteps` method using the Builder pattern, we can encapsulate the parameters into a builder class. This will improve readability and maintainability without modifying existing methods, thus avoiding breaking other classes.

Below is the refactored code using the Builder Pattern:

Refactored Code

java  Copiar código

```
package org.example.studyregistry;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class StudyPlan extends Registry {
    private StudyObjective objective;
    private List<String> steps;

    public StudyPlan(String planName, StudyObjective objective, List<StudyMaterial> materi
```