

Project Report

Code Reengineering

Kelompok 13

Bryan Valentino Wijaya - 2702301395
Cladio Bernard Octaviano - 2702362876
Kevin Jeremia - 2702292932
Marco Bennedict Makin - 2702336631



01

Overview




Overview

Source: <https://github.com/syedalisait/subtitle-downloader/>

Code smell yang kami temukan pada source tersebut :

- Comment
- Feature Envy
- Large Class
- Message Chain
- Speculative Generality
- Long Method
- Primitive Obsession



02

Refactoring Techniques



Refactoring Techniques

Teknik Refactoring? Metode atau cara sistematis untuk mengubah struktur internal code tanpa mengubah fungsi atau perilakunya. Dengan tujuan :

- Membuat *code* lebih mudah dibaca
- Membuat *code* lebih mudah dipelihara/maintenance
- Membuat *code* lebih mudah untuk dikembangkan lagi kedepannya



Teknik Refactoring yang Kami Pakai

Extract Method

Digunakan untuk refactor *long method*. Dengan cara memecah method

Extract Variable

Digunakan untuk memudahkan memahami kondisi kompleks ke suatu variabel

Move Method

Digunakan ketika ada *feature envy*. Dengan cara memindahkan method ke class yang relevan

Replace Parameter with Method Call

Digunakan untuk refactor *long parameter list*. Dengan cara memanggil method langsung

Hide Method

Mengubah method menjadi *private* supaya bisa diakses dalam kelas itu sendiri



03

Code Smells



Comments

```
95  /*
96  * Expectation:
97  *   movieFolderPath is something like E:/Movies or E:/Downloads/Movies where all the movies are present inside
98  *   separate folders
99  * Example:
100 *   E:/Downloads/Movies/TheLma/TheLma (2017).mp4
101 *   E:/Downloads/Elle/Elle (2016).mp4
102 * Logic:
103 *   1) Parse through the directories inside "movieFolderPath"
104 *   2) If there are any Movie Folders with ".srt" file present, Ignore them
105 *   3) Else, download the subtitle for the movie
106 */
107
108 // Lambda Expression for FileNameFilter
109 String[] files = new File(movieFolderPath).list((directory, name) -> getSubtitlesForAllMovies(directory, name));
```

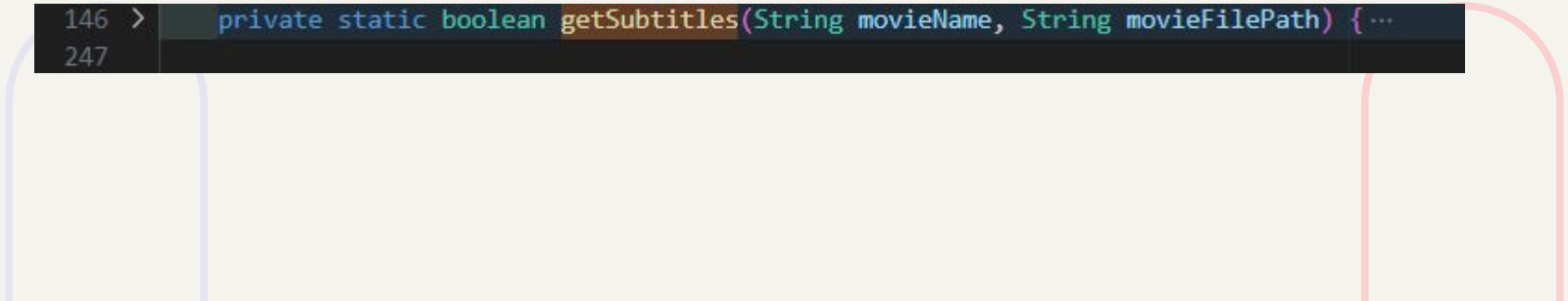
```
117  /**
118  * ***** Util Functions *****
119  * getSubtitleForAllMovies - Loops through the Movies Directory and calls getSubtitles function to Download the subtitles for the movies
120  * getSubtitles - Given a movieName, Downloads the subtitle for the movie
121  * getMovieName - Given the Full Path of Movie File location returns the Movie Name
122  * getSearchResults - Given a Movie Name, returns the search results in Vify Subtitle
123  * getMovieURL - Given a html content of "div class=media-body", gives the link to Movie URL page
124  * getMovieAndYear - Given a html content of the Movie, parses the movieName and Year of the Movie
125  * getSubtitleURL - Given a movieURL retrieves the Subtitle URL for the movie
126  * getMovieFolder - Given a full path of the movie, returns the path of the parent folder
127  * getHTMLContent - Given a URL, returns the Document which has the html content which can be used to parse
128  * downloadSubtitle - Given a downloadSubtitleURL and PathToDownload, downloads the zip file, extracts the contents and saves the files in the given Path to Download
129  * checkExtension - Given a File object and list of Extension, checks whether the file contains any of the extensions and returns a boolean
130  * getExtensions - Returns the list of video file extensions
131  * displayMovieAndYear - Given a html element and count variable -> Parses the movie name/year from the Html content and prints to console
132  * *****
133  */
134
```


Long Method

```
118 > private static boolean getSubtitlesForAllMovies(File directory, String name) {..  
145
```

```
262 > private static String getMovieName(String movieFilePath) {..  
306
```

```
146 > private static boolean getSubtitles(String movieName, String movieFilePath) {..  
247
```



The diagram illustrates the flow of control between the three code snippets. A light blue line starts from the bottom of the first snippet (line 145), goes down, then right, then up to the bottom of the second snippet (line 306). A light red line starts from the bottom of the second snippet (line 306), goes down, then right, then up to the bottom of the third snippet (line 247).

Long Method (Solution)

```
29 private boolean isValidMovieDirectory(File dir) {
30     return dir.isDirectory() && dir.listFiles() != null;
31 }
32
33 private boolean shouldDownloadSubtitle(File dir) {
34     return hasNoSubtitleFile(dir);
35 }
36
37 private void processMovie(File movieFile) {
38     String movieFilePath = movieFile.toString();
39     System.out.println("\nMovie File Path:\n\t" + movieFilePath);
40     String movieName = getMovName.getMovieName(movieFilePath);
41
42     boolean success = getSub.getSubtitles(movieName, movieFilePath);
43     if (!success) {
44         System.out.println("\nSkipping Downloading Subtitles for the Current Movie");
45         System.out.println("-----" +
46             "-----\n");
47     }
48 }
49
50 private Optional<File> getFirstMovieFile(File folder) {
51     File[] files = folder.listFiles();
52     if (files == null)
53         return Optional.empty();
54
55     return Arrays.stream(files)
56         .filter(file -> checkExtension(file, getFileExtensions()))
57         .findFirst();
58 }
59
60 private boolean hasNoSubtitleFile(File folder) {
61     return Arrays.stream(folder.listFiles())
62         .noneMatch(f -> f.toString().endsWith(".srt"));
63 }
64
```

```
65 private static List<String> getFileExtensions() {
66     return Arrays.asList(".avi", ".mp4", ".mkv", ".mpg", ".mpeg", ".mov", ".rm", ".vob", ".wmv", ".flv", ".3gp");
67 }
68
69 private static boolean checkExtension(File file, List<String> fileExtensions) {
70     int index = file.toString().lastIndexOf(".");
71     if (index == -1) {
72         return false;
73     }
74     String format = file.toString().substring(index);
75     return fileExtensions.contains(format);
76 }
```

Message Chain

```
132 private static boolean getSubtitlesForAllMovies(File directory, String name) {
133     try {
134         File temp = new File(directory, name);
135         if (temp.isDirectory() && temp.listFiles() != null) {
136             // Check if ".srt" file is already not present for the Movie
137             if (Arrays.stream(temp.listFiles()).noneMatch(f -> f.toString().endsWith(".srt"))) {
138                 // Check and get the Movie File with one of the video format extension from the Movie Folder
139                 Optional<File> file = Arrays.stream(temp.listFiles()).filter(f -> checkExtension(f, getFileExtensions())).findFirst();
```

```
118 private static boolean getSubtitlesForAllMovies(File directory, String name) {
119     try {
120         File temp = new File(directory, name);
121         if (temp.isDirectory() && temp.listFiles() != null) {
122             // Check if ".srt" file is already not present for the Movie
123             if (Arrays.stream(temp.listFiles()).noneMatch(f -> f.toString().endsWith(suffix:".srt"))) {
124                 // Check and get the Movie File with one of the video format extension from the Movie Folder
125                 Optional<File> file = Arrays.stream(temp.listFiles()).filter(f -> checkExtension(f, getFileExtensions())).findFirst();
126                 if (file.isPresent()) {
127                     String movieFilePath = file.get().toString();
128                     System.out.println("\nMovie File Path:\n\t" + movieFilePath);
129                     String movieName = getMovieName(movieFilePath);
130                     if(!getSubtitles(movieName, movieFilePath)) {
131                         System.out.println(x:"\nSkipping Downloading Subtitles for the Current Movie");
132                         System.out.println("-----\n");
133                     }
134                 }
135             }
136         }
137     }
```

Message Chain (Solution)

```
50     private Optional<File> getFirstMovieFile(File folder) {  
51         File[] files = folder.listFiles();  
52         if (files == null)  
53             return Optional.empty();  
54  
55         return Arrays.stream(files)  
56             .filter(file -> checkExtension(file, getFileExtensions()))  
57             .findFirst();  
58     }
```


```
60     private boolean hasNoSubtitleFile(File folder) {  
61         return Arrays.stream(folder.listFiles())  
62             .noneMatch(f -> f.toString().endsWith(".srt"));  
63     }
```



04


Summary





Smelly code adalah masalah yang dapat terjadi pada masa mendatang dimana akan berdampak pada proses pemeliharaan. Dalam proyek ini, kami berhasil melakukan refactor pada proyek java yang awalnya hanya satu file menjadi sembilan file.

Proyek kami juga menggunakan SonarQube untuk memvalidasi apakah masih terdapat smelly code dan beberapa metrics lain untuk menunjukkan secara detail tentang code yang sudah kami refactor.





**Sekian
Terimakasih!**

