[Date]

FIT5171 Assignment 3

Measuring and Assessing Quality of the Codebase

Team 27

Neel Shridharani

Rashmitha Karkera

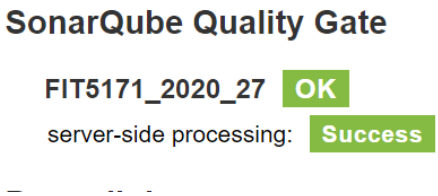
Rebecca Christi

Contents

1. Integration and Setup

1.1. SonarQube SetUp

The SonarQube quality gate has been successfully SetUp and integrated with Jenkins server. Such that, for every commit that has been done into git, post a successful build, the SonarQube server measures, monitors and conducts analysis of the committed code quality.



1.2. Code Base Integration

The team has successfully integrated both the code base, the merged project compiles without errors and test cases from both code bases passed without errors or assertion failures.

In comparison, to the team's previous logs there are additional test cases in each model class. Also, in addition we have Track and TrackUnitTest suite integrated from other teams' Codebase. The team has also made considerable changes to attain full coverage which is explained in detail in further sections.

1.2.1. New Model Class

- **Track.java -** to check for multiple tracks in an albumNew Test Class
- **TrackUnitTest.java** – to test track model class

1.2.2. New Test Cases

- **TrackUnitTest.java:**
  trackNameCannotBeNull() - to check if track name is null
  trackLengthCannotBeNull() - to check if track content is null
- **ECMMiner.java:**
  mostSoldAlbum() - to get most sold album
  highestRatedAlbum()- to get highest rated album
- **ECMMinerIntegrationTest.java:**
  shouldSearchAlbumByRecordNumberAndReturnBusiestYears()- returns busiest years by record number.
- **ECMMinerUnitTest.java:**
  shouldReturnAllAlbumsSimilarToGiven() - returns all similar album list

1.3. Jenkins Integration

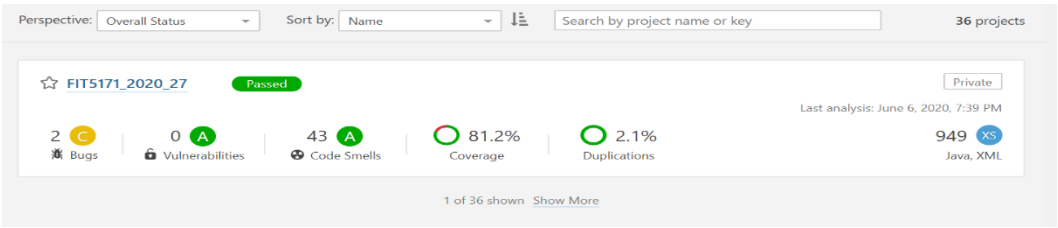
```
[INFO] Scanning for projects...
[INFO] --------------------< allaboutecm:FIT5171_2020_27 >-----------------
[INFO] Building FIT5171_2020_27 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------
 [INFO]  T E S T S
[INFO] Running allaboutecm.model.MusicianUnitTest
[INFO] Tests run: 25, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.32 s - in allaboutecm.model.MusicianUnitTest
[INFO] Running allaboutecm.model.TrackUnitTest
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.042 s - in allaboutecm.model.TrackUnitTest
```

```
[INFO] Running allaboutecm.model.MusicalInstrumentUnitTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.002 s - in allaboutecm.model.MusicalInstrumentUnitTest
[INFO] Running allaboutecm.model.AlbumUnitTest
[INFO] Tests run: 38, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.12 s - in allaboutecm.model.AlbumUnitTest
[INFO] Running allaboutecm.model.MusicianInstrumentTest
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.019 s - in allaboutecm.model.MusicianInstrumentTest
[INFO] Running allaboutecm.mining.ECMMinerIntegrationTest
 [INFO] Tests run: 11, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
12.976 s - in allaboutecm.mining.ECMMinerIntegrationTest
[INFO] Running allaboutecm.mining.ECMMinerUnitTest
[INFO] Tests run: 15, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.163 s - in allaboutecm.mining.ECMMinerUnitTest
[INFO] Running allaboutecm.dataaccess.neo4j.URLConvertorUnitTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.021 s - in allaboutecm.dataaccess.neo4j.URLConvertorUnitTest
[INFO] Running allaboutecm.dataaccess.neo4j.Neo4jDAOUnitTest
[INFO] Tests run: 32, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
7.083 s - in allaboutecm.dataaccess.neo4j.Neo4jDAOUnitTest
[INFO] Tests run: 139, Failures: 0, Errors: 0, Skipped: 0
[JENKINS] Recording test results
```

2. Code Analysis:

2.1. Initial Code Analysis After Integration

Below is the initial report on Sonarqube:



Bugs	Code Smells	Duplication	Coverage	Vulnerability	Complexity
2	43	2.1%	79.6%	0	949

- In terms of **complexity** the code contains 949 lines of code, spread over 7 different test classes with 117 different unit tests.
- Less **coverage** due to missing test cases (uncovered conditions/ lines) for some model classes. The code has 79.6 % of coverage with a total 71 uncovered lines and 57 uncovered conditions which is already above the accepted coverage value.

Coverage 79.6%

	Coverage	Uncovered Lines	Uncovered Conditions
src/main/java/allaboutecm/model/Track.java	42.9%	8	–
src/main/java/allaboutecm/dataaccess/neo4j/URLConverter.java	46.2%	5	2
src/main/java/allaboutecm/dataaccess/neo4j/Neo4jDAO.java	73.1%	23	16
src/main/java/allaboutecm/model/MusicianInstrument.java	78.8%	2	5
src/main/java/allaboutecm/mining/ECMMiner.java	81.3%	31	25
src/main/java/allaboutecm/model/Musician.java	84.0%	2	2
src/main/java/allaboutecm/model/MusicalInstrument.java	89.5%	0	2
src/main/java/allaboutecm/model/Album.java	93.2%	0	5
src/main/java/allaboutecm/model/Entity.java	100%	0	–

9 of 9 shown

- **Duplication** in ECM miner due to similarity in mostSimilarAlbums method.
- High **Code Smells** because of following reasons:
    - Unused variables/ objects
    - Unused packages(import statements)

2.2. Code Improvement

To improve the codebase following changes were made to the production/ or test classes. They are as given below.

2.2.1. Model Class

Unused packages or comment code have been removed from model classes, to reduce **Code Smells.**

To increase coverage certain methods in classes were modified. They are as follows:

1. Methods to cover (if) clauses, were extended in order to verify and draw comparison of **Musicians** with null objects.
2. Methods to cover (if) clauses, were extended in order to verify and draw comparison of **Albums** with null objects.
3. Methods to cover (if) clauses, were extended in order to verify and draw comparison of **MusicianInstrument** with null objects.
4. Removal of useless assignments were fixed, which helped in improving the code quality.
5. Iteration of the set was re-coded for efficient usage.

2.2.2. Test Class

Unused packages, commented code and unused variable / objects have been removed from most classes. This helped greatly to reduce **Code Smells**.

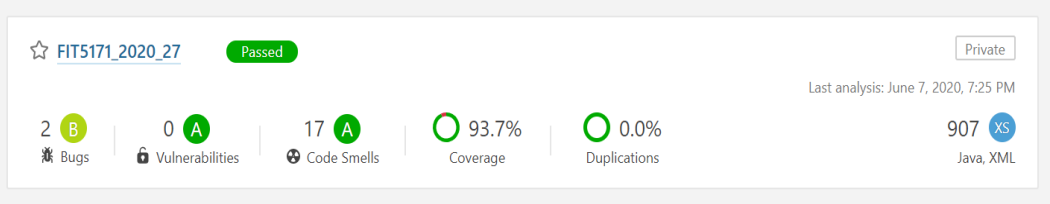
To increase coverage certain test cases in the test classes were modified/ added. They are as follows:

1. Unit Test cases were written to verify and validate **musicians**, **albums** and **musician Instruments** and draw comparisons with other parameters.

2. **URL class** was extended and written unit test cases to satisfy the conditions of the same.
3. **Track class** which was implemented by the other team, was extended and implemented in methods of ours to serve the code coverage.
4. Errors related to assertions were fixed to improve the code quality.
5. Changes on **ECMIner Unit test** and **Integration** test were done to accommodate verification on busiest year, check for popular musicians.

2.3. Code Analysis After Improvement

Below is the final report after improving code quality.



Bugs	Code Smells	Duplication	Coverage	Vulnerability	Complexity
2	17	0.0 %	93.7 %	0	907

- The code in terms of complexity contains 907 lines of code, spread over 7 different test classes containing over 139 different unit tests. This is reasonable in terms of code **complexity** for the given amount of test cases.
- **Code coverage** between 70-80 % is a reasonable goal for testing any system. Code coverage has been **increased** from 79.6% to 93.7% with only 15 uncovered lines and 22 uncovered conditions.

- There are no existing vulnerabilities in terms of **security** apart from two small bugs caused due to extended implementation in test case.
- **Duplication** has been completely removed.
- **Technical Debt** currently after the improvement on codebase is 3 hours, which is great considering the approaches used to improve the program.

3. Contribution

Team Member	Contribution	Task
Neel Shridharani	33%	Integration + Code Quality Analysis and Code extension + Report
Rashimitha Karkera	33%	Integration + Code Quality Analysis and Code extension + Report
Rebecca Christi	33%	Integration + Code Quality Analysis and Code extension + Report