

# **SOFTWARE ENGINEERING METHODS**

## **Assignment 3**

### **Group OP5-SEM10:**

Arjun Vilakathara

Stijn Coppens

Adriaan Brands

Gideon Bot

Ethan Keller

Noyan Toksoy

## Tools used

The tool used for this test refactoring is called PIT Test, <https://pitest.org/>. This tool is an industry standard for finding and killing mutants in your code and was recommended to us by the stakeholders of the project.

## AuthenticationService -> Domain -> Utility

Here's the mutation score of the test cases involved with this class:

### Pit Test Coverage Report

#### Package Summary

nl.tudelft.sem10.authenticationservice.domain

Number of Classes	Line Coverage	Mutation Coverage
8	98% 131/133	77% 50/65

#### Breakdown by Class

Name	Line Coverage	Mutation Coverage
<a href="#">JwtAuthenticationEntryPoint.java</a>	100% 4/4	100% 2/2
<a href="#">JwtRequest.java</a>	100% 10/10	67% 2/3
<a href="#">JwtResponse.java</a>	100% 4/4	100% 1/1
<a href="#">JwtTokenFilter.java</a>	100% 33/33	79% 15/19
<a href="#">JwtTokenUtil.java</a>	100% 24/24	94% 15/16
<a href="#">SecurityConfig.java</a>	100% 30/30	0% 0/7
<a href="#">UserDetailsImpl.java</a>	100% 20/20	100% 14/14
<a href="#">Utility.java</a>	75% 6/8	33% 1/3

Report generated by PIT 1.5.1

The mutation score for the Utility class was at 33% which is quite low. In order to improve this score and understand why there were mutants surviving, we further examined the mutants generated by Pitest. Here are the mutants for the hash() method, only one them was killed:

```
25 2 while (hashText.length() < 128) {
26     hashText = "0" + hashText;
27 }
28 1 return hashText;
29 }
30 }
```

#### Mutations

```
25 1. changed conditional boundary → SURVIVED
26 2. negated conditional → TIMED OUT
28 1. replaced return value with "" for nl.tudelft.sem10.authenticationservice.domain.Utility::hash → SURVIVED
```

The first mutant that survived had altered the conditional operator on line 25. The second one on line 28 returned an empty String instead of the local variable hashText. The main reason why these mutants weren't killed was because there were no test cases that made assertions on the functionality of this Utility method. This Utility method was called inside other test cases and those cases had assertions on other functionalities. So that's why we decided to create a test suite dedicated to this class, which would increase the quality of our tests tremendously. We created the test suite with this commit SHA:

"034340cdc6971ce939452bb042c5b12757fb9679". In order to assert that the hashing algorithm works properly we first picked an example String and then converted it to the SHA-512 format by using one of the well-known, reliable implementations of the algorithm. We asserted that the algorithm should produce the same output as the converted String and it indeed worked as expected. Here's the mutation score of the algorithm, according to PIT Test, after the new test case:

# Pit Test Coverage Report

## Package Summary

nl.tudelft.sem10.authenticationservice.domain

Number of Classes	Line Coverage	Mutation Coverage
8	98% 131/133	80% 52/65

### Breakdown by Class

Name	Line Coverage	Mutation Coverage
<a href="#">JwtAuthenticationEntryPoint.java</a>	100% 4/4	100% 2/2
<a href="#">JwtRequest.java</a>	100% 10/10	67% 2/3
<a href="#">JwtResponse.java</a>	100% 4/4	100% 1/1
<a href="#">JwtTokenFilter.java</a>	100% 33/33	79% 15/19
<a href="#">JwtTokenUtil.java</a>	100% 24/24	94% 15/16
<a href="#">SecurityConfig.java</a>	100% 30/30	0% 0/7
<a href="#">UserDetailsImpl.java</a>	100% 20/20	100% 14/14
<a href="#">Utility.java</a>	75% 6/8	100% 3/3

Report generated by [PIT](#) 1.5.1

As can be observed from this picture, we managed to achieve 100% percent mutation coverage, meaning that our new test case managed to kill all the mutants. Overall, the improvement to the testing of this class was necessary to prevent future vulnerabilities.

## AuthenticationService -> Domain -> JwtRequest

The mutation score for this class was 67% for this class as depicted on the image below:

### Breakdown by Class

Name	Line Coverage	Mutation Coverage
<a href="#">JwtAuthenticationEntryPoint.java</a>	100% 4/4	100% 2/2
<a href="#">JwtRequest.java</a>	100% 10/10	67% 2/3

One mutant was not killed, namely a mutant that caused a mutation of the return type of the getType() method. The mutation caused the method to return 0 because that might be a special number for edge cases. But for us '0' is an identifier for one of the user types/roles in our application. This was not covered broadly enough in the corresponding test suite. Thus the tests were altered as can be seen in the commit with SHA:

1e8317c4cd6b23b905a2137f8cc45bd6cc0f1520

```
52  /**
53   * Getter for the type.
54   *
55   * @return type of user
56   */
57  public int getType() {
58  1  return type;
59  }
60
61 }
```

### Mutations

```
49  1. replaced return value with "" for nl/tudelft/sem10/authenticationservice/domain/JwtRequest::getNetId → KILLED
49  1. replaced return value with "" for nl/tudelft/sem10/authenticationservice/domain/JwtRequest::getPassword → KILLED
58  1. replaced int return with 0 for nl/tudelft/sem10/authenticationservice/domain/JwtRequest::getType → SURVIVED
```

This change made sure that the mutation was covered and that subsequently, the mutant was killed. We now have 100% mutant coverage for this class as can be seen in the image below:

### Breakdown by Class

Name	Line Coverage	Mutation Coverage
<a href="#">JwtAuthenticationEntryPoint.java</a>	100% <div>4/4</div>	100% <div>2/2</div>
<a href="#">JwtRequest.java</a>	100% <div>10/10</div>	100% <div>3/3</div>

### AuthenticationService -> Domain -> UserDetailsServiceImpl

The only surviving mutant in the AuthenticationService was a mutation removing a print statement on line 30, making the mutation score 67% as can be seen below:

```

29 1 if (user == null) {
30 1   System.out.println("Could not find a user with netId: " + username);
31   return null;

```

<a href="#">UserDetailsServiceImpl.java</a>	100% <div>7/7</div>	67% <div>2/3</div>
---	---------------------	--------------------

After making sure this statement is executed by *Mockito verify*, we kill the remaining mutant resulting in a mutation score of 100%. This test was implemented in commit b88d49905f2bdb9c9828b9a7394179cd9ec55a600 and was subsequently improved to comply with PMD rules in commit dedbf9378267c6390c383eef955d8baf511ba4d4.

```

30 1   System.out.println("Could not find a user with netId: " + username);

```

<a href="#">UserDetailsServiceImpl.java</a>	100% <div>7/7</div>	100% <div>3/3</div>
---	---------------------	---------------------

### CourseService -> Application -> AbstractRepositoryService

The initial mutation score score was 29%:

Name	Line Coverage	Mutation Coverage
<a href="#">AbstractRepositoryService.java</a>	100% <div>14/14</div>	29% <div>2/7</div>

Out of the seven mutants killed, five of them were left alive. The mutants that were left alive as follows:

```

24
25 @Override
26 public Iterable<T> get() {
27 1 return getRepository().findAll();
28 }
29
30 @Override
31 public T get(U id) {
32 // Get an entry by ID or null if no such entry exists
33 1 return getRepository().findById(id).orElse(null);
34 }
35
36 @Override
37 public T add(T data) {
38 // Duplicate entry
39 1 if (getRepository().existsById(idMapper.apply(data))) {
40 return null;
41 }
42
43 // Save the entry
44 getRepository().save(data);
45
46 1 return data;
47 }
48
49 @Override
50 public T remove(U id) {
51 T t = getRepository().findById(id).orElse(null);
52
53 // No such entry
54 1 if (t == null) {
55 return null;
56 }
57
58 // Delete the entry
59 1 getRepository().deleteById(id);
60
61 // Return the deleted entry
62 1 return t;
63 }
64
65 /**
66 * Get the backing repository.
67 *
68 * @return the backing repository.
69 */
70 protected abstract JpaRepository<T, U> getRepository();
71 }

```

### Mutations

```

27 1. replaced return value with null for nl/tudelft/sem10/courseservice/application/AbstractRepositoryService::get → SURVIVED
33 1. replaced return value with null for nl/tudelft/sem10/courseservice/application/AbstractRepositoryService::get → SURVIVED
39 1. negated conditional → KILLED
46 1. replaced return value with null for nl/tudelft/sem10/courseservice/application/AbstractRepositoryService::add → KILLED
54 1. negated conditional → SURVIVED
59 1. removed call to org/springframework/data/jpa/repository/JpaRepository::deleteById → SURVIVED
62 1. replaced return value with null for nl/tudelft/sem10/courseservice/application/AbstractRepositoryService::remove → SURVIVED

```

As can be seen from which mutations survived the first two survive because there were no checks to assert that they would indeed be not null. Adding these two tests allowed us to kill the mutants. The other 3 mutants could be killed by adding actual tests and checking if the value is what we expected when inserting/deleting/getting values from the mocked database. If it's replaced by null it's clearly not what we expected or if the method wasn't called at all the desired effect won't have happened.

As a result we achieved the following mutation score:

Name	Line Coverage		Mutation Coverage	
<a href="#">AbstractRepositoryService.java</a>	100%	14/14	100%	7/7
<a href="#">AuthService.java</a>	100%	7/7	100%	2/2
<a href="#">AuthServiceImpl.java</a>	36%	5/14	0%	0/4
<a href="#">CategoryServiceImpl.java</a>	100%	6/6	33%	1/3
<a href="#">CourseServiceImpl.java</a>	100%	3/3	100%	1/1
<a href="#">DataSourceConfig.java</a>	100%	12/12	0%	0/9



## AbstractRepositoryService.java

```
1 package nl.tudelft.sem10.courseservice.application;
2
3 import java.util.Objects;
4 import java.util.function.Function;
5 import org.springframework.data.jpa.repository.JpaRepository;
6
7 /**
8  * A shell {@link RepositoryService} implementation using a {@link JpaRepository}.
9  */
10 * @param <T> - Data type.
11 * @param <U> - Data identifier type.
12 */
13 public abstract class AbstractRepositoryService<T, U> implements RepositoryService<T, U> {
14     private final transient Function<T, U> idMapper;
15
16     /**
17      * Create a repository service.
18      *
19      * @param idMapper - Function<T, U> Mapping function to get a key from a data entity.
20      */
21     protected AbstractRepositoryService(Function<T, U> idMapper) {
22         this.idMapper = Objects.requireNonNull(idMapper);
23     }
24
25     @Override
26     public Iterable<T> get() {
27         return getRepository().findAll();
28     }
29
30     @Override
31     public T get(U id) {
32         // Get an entry by ID or null if no such entry exists
33         return getRepository().findById(id).orElse(null);
34     }
35
36     @Override
37     public T add(T data) {
38         // Duplicate entry
39         if (getRepository().existsById(idMapper.apply(data))) {
40             return null;
41         }
42
43         // Save the entry
44         getRepository().save(data);
45
46         return data;
47     }
48
49     @Override
50     public T remove(U id) {
51         T t = getRepository().findById(id).orElse(null);
52
53         // No such entry
54         if (t == null) {
55             return null;
56         }
57
58         // Delete the entry
59         getRepository().deleteById(id);
60
61         // Return the deleted entry
62         return t;
63     }
64
65     /**
66      * Get the backing repository.
67      *
68      * @return the backing repository.
69      */
70     protected abstract JpaRepository<T, U> getRepository();
71 }
```

### Mutations

```
27 1. replaced return value with null for nl/tudelft/sem10/courseservice/application/AbstractRepositoryService::get → KILLED
33 1. replaced return value with null for nl/tudelft/sem10/courseservice/application/AbstractRepositoryService::get → KILLED
39 1. negated conditional → KILLED
46 1. replaced return value with null for nl/tudelft/sem10/courseservice/application/AbstractRepositoryService::add → KILLED
54 1. negated conditional → KILLED
```

After adding some basic tests in commit *2b11cbe6c5a8e81b22f0253de59545942fc11e1b*, subsequently improved in commits *f7566c7fd3bb73e8d7638385bc4e3490cea3bf10* and *dedbf9378267c6390c383eef955d8baf511ba4d4*, where we managed to kill all of the mutants that were still left alive and get our mutation coverage up to a hundred percent.