CSE2115 Software Engineering Methods

Assignment 3

Authors

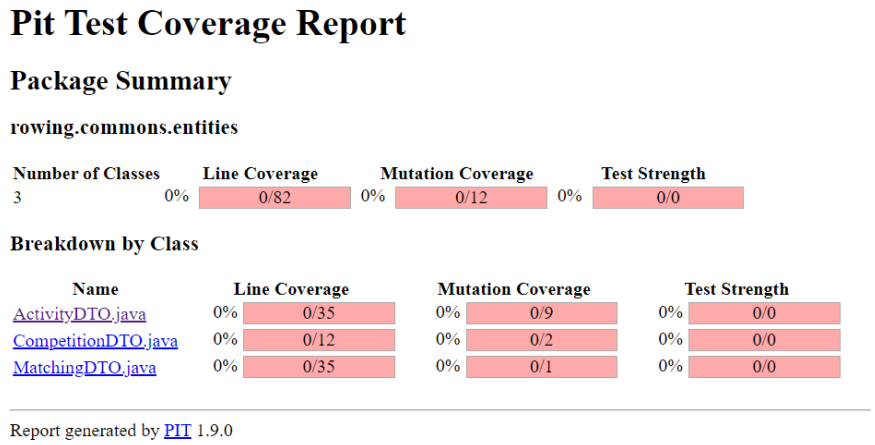
Bogdan-Andrei Bancuta - 5507340
Alexandru Gabriel Cojocaru - 5520967
Khalit Gulamov - 5491541
Alexandru-Nicolae Ojica - 5477484
Wing-Yan Joyce Sung - 5011825
Efe Unluyurt - 5452481

25 January 2023

Task 1

1.1 ActivityDTO

The test suite for the *Certificates* class was chosen for improvement. It was determined that the test suite had low coverage and a low mutation score. Previously, the mutation coverage was 0%, or more specifically, none of the 9 mutants were killed (fig. 1a). By adding 21 test cases, the mutation score was increased to 100% (fig. 1b). The tests can be viewed in the following commit: `https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM33b/-/commit/ef737051a4ad42c857a3ab934f786b9238564dc1`.



Pit Test Coverage Report

Package Summary

rowing.commons.entities

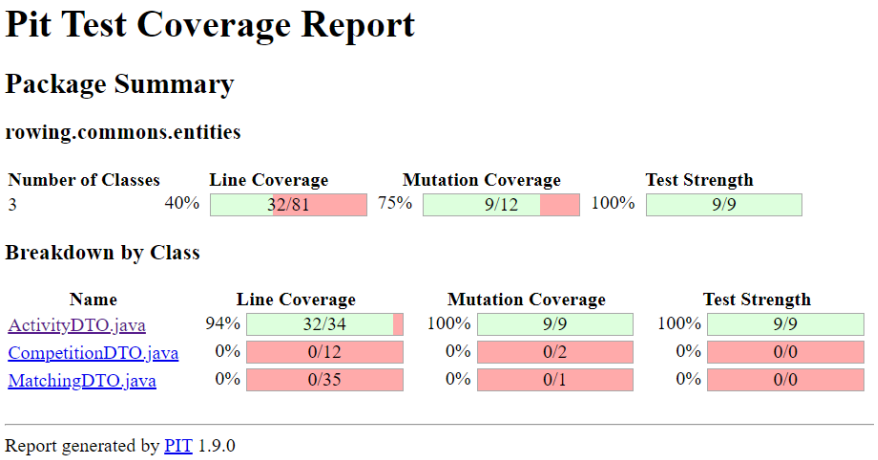
Number of Classes	Line Coverage	Mutation Coverage	Test Strength
3	0% 0/82	0% 0/12	0% 0/0

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
ActivityDTO.java	0% 0/35	0% 0/9	0% 0/0
CompetitionDTO.java	0% 0/12	0% 0/2	0% 0/0
MatchingDTO.java	0% 0/35	0% 0/1	0% 0/0

Report generated by PIT 1.9.0

(a) Mutation score before.



Pit Test Coverage Report

Package Summary

rowing.commons.entities

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
3	40% 32/81	75% 9/12	100% 9/9

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
ActivityDTO.java	94% 32/34	100% 9/9	100% 9/9
CompetitionDTO.java	0% 0/12	0% 0/2	0% 0/0
MatchingDTO.java	0% 0/35	0% 0/1	0% 0/0

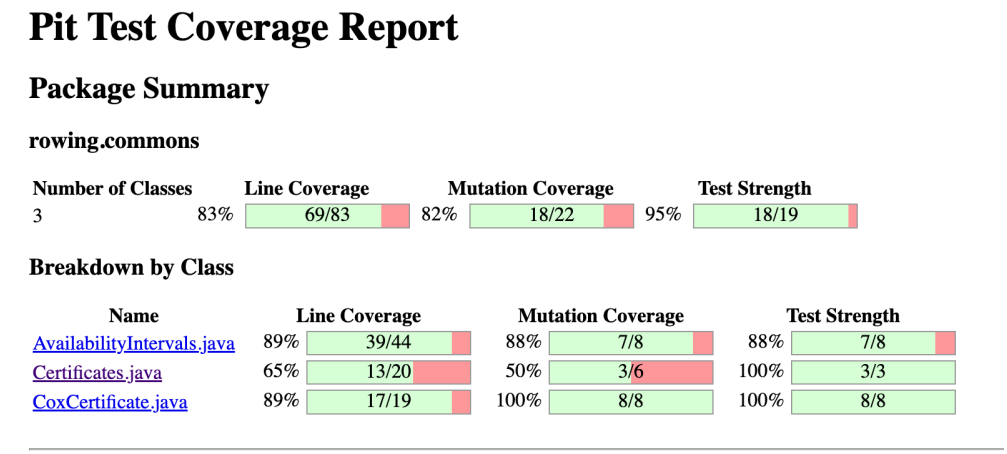
Report generated by PIT 1.9.0

(b) Mutation score of after.

Figure 1: Mutation scores of the ***ActivityDTO*** class.

1.2 *Certificates*

The test suite for the *Certificates* class was picked to improve. The test suite was found to be a bit lacking as it showed to low coverage and low mutation score. Before, the mutation coverage was 0%, or to be precise 0 out of 9 mutants were killed (fig. 2a). By adding two 21 test cases the mutation score was improved to 100% (fig. 2b). The tests can be reviewed in the following commit: `https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM33b/-/merge_requests/63/diffs?commit_id=0dbbd5b7eabec133d9c0da12a1f6cfb986b6cb2c`.



Pit Test Coverage Report

Package Summary

rowing.commons

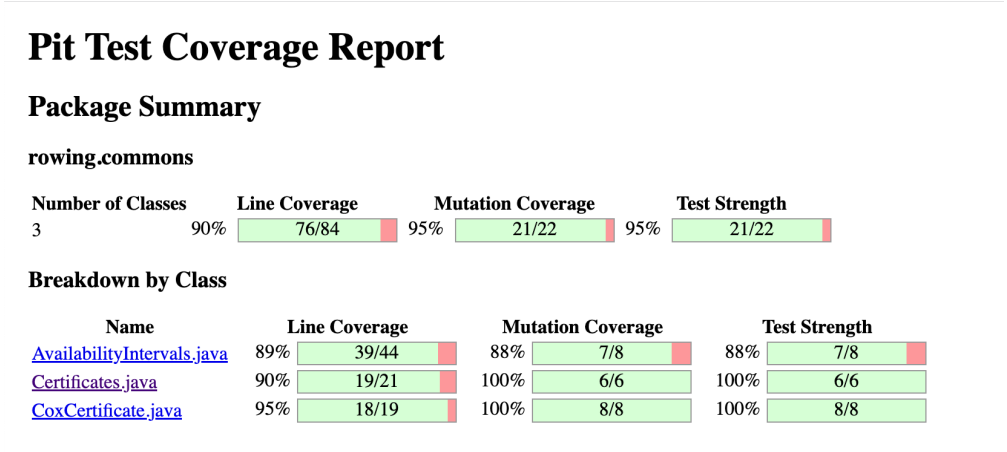
Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
3	83%	69/83	82%	18/22	95%	18/19

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
AvailabilityIntervals.java	89%	39/44	88%	7/8	88%	7/8
Certificates.java	65%	13/20	50%	3/6	100%	3/3
CoxCertificate.java	89%	17/19	100%	8/8	100%	8/8

Report generated by PIT 1.9.0

(a) Mutation score before.



Pit Test Coverage Report

Package Summary

rowing.commons

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
3	90%	76/84	95%	21/22	95%	21/22

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
AvailabilityIntervals.java	89%	39/44	88%	7/8	88%	7/8
Certificates.java	90%	19/21	100%	6/6	100%	6/6
CoxCertificate.java	95%	18/19	100%	8/8	100%	8/8

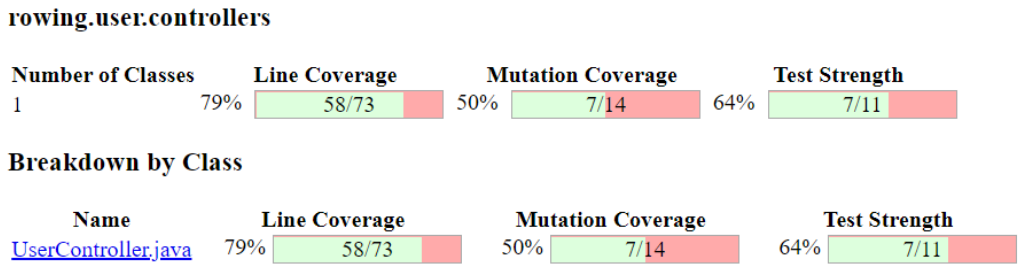
Report generated by PIT 1.9.0

(b) Mutation score of after.

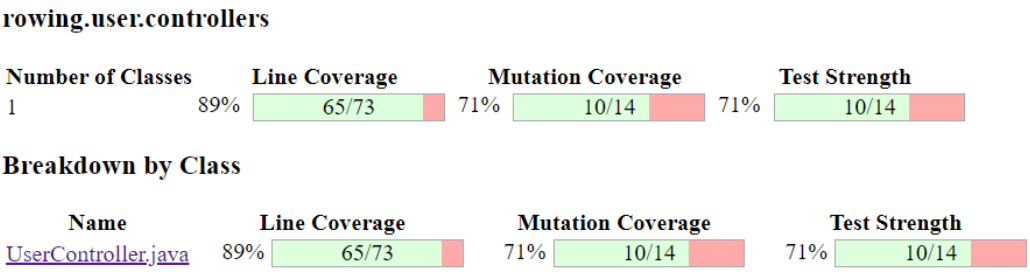
Figure 2: Mutation scores of the ***Certificates*** class.

1.3 *UserController*

The mutation score of the *UserController* class was chosen to improve. Before the mutation coverage was 50%, where 7 out of 14 mutants were killed (fig. 3a). Due to the addition of three new tests 10 out of 14 mutants were killed, improving the mutation score to 71% (fig. 3b). The test can be reviewed in the following commit: `https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM33b/-/merge_requests/62/diffs?commit_id=d6d5400ee95262d4859b4c74eb06761bb6b2c8df`.



(a) Mutation score before.



(b) Mutation score of after.

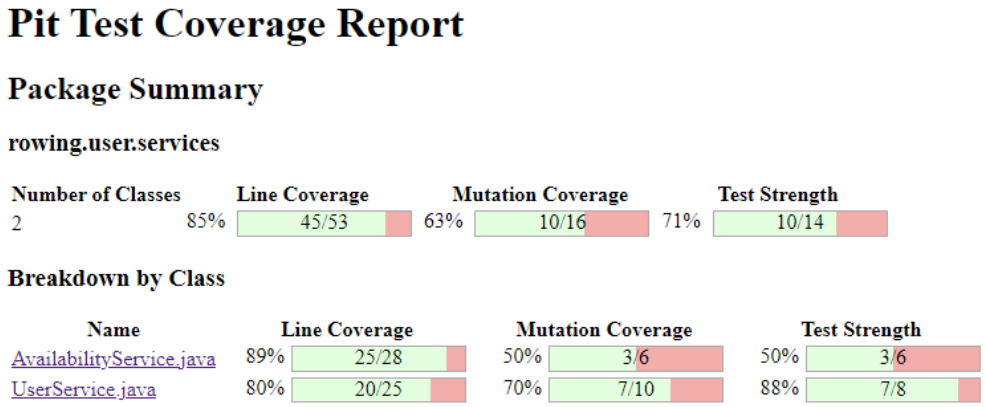
Figure 3: Mutation scores of the ***UserController*** class.

The four remaining mutants were all equivalent mutants. One mutant removes a line in which something was printed to the console log, this line does not contribute anything to the functionality of the program and was most likely added during development to check the correctness of the function. The three other equivalent mutants all replace the content of an http response with null, however, the content was never defined and is thus null already.

1.4 *UserService*

The class *UserService* had 70% mutation score (fig 4a)as reported by the mutation tool Pitest and being an important class for our application it has been chosen to be tested further. In total, three mutants created by Pitest have survived, all inside the *validateUpdateUserDto* (fig 5a). The mutants negated the equal conditions inside the *if* statement that is validating a user's first name, last name, and email, to not be null (fig 6a). In order to kill the mutants,

I created three more tests that would try to update one at a time, these fields with null values, and assert the correctness of the response given by the program, in this case, a bad request. After the three tests have been added, the Pitest tool reported that the three identified mutants were successfully killed (fig 5b) (fig 6b), and that the *UserService* class has achieved 100% mutation score (fig 4b). The commit that is responsible for these tests can be found here `https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM33b/-/merge_requests/60/diffs?commit_id=31d53a801b93fe536ec5230465f3d244c91458ca`



Pit Test Coverage Report

Package Summary

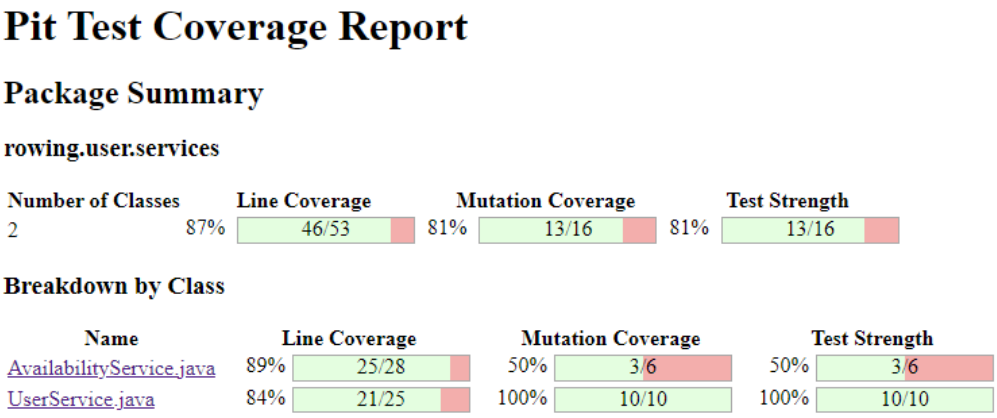
rowing.user.services

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
2	85%	45/53	63%	10/16	71%	10/14

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
AvailabilityService.java	89%	25/28	50%	3/6	50%	3/6
UserService.java	80%	20/25	70%	7/10	88%	7/8

(a) Mutation score before.



Pit Test Coverage Report

Package Summary

rowing.user.services

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
2	87%	46/53	81%	13/16	81%	13/16

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
AvailabilityService.java	89%	25/28	50%	3/6	50%	3/6
UserService.java	84%	21/25	100%	10/10	100%	10/10

(b) Mutation score of after.

Figure 4: Mutation scores of the ***UserService*** class.

```
83        */
84       private void validateUpdateUserDTO(UpdateUserDTO updateUserDTO) {
85 2         if (updateUserDTO.getFirstName() == null || updateUserDTO.getEmail() == null
86 1                 || updateUserDTO.getLastName() == null) {
87                 throw new IllegalArgumentException("First name, last name and email cannot be empty");
88             }
89         }
```

(a) Alive mutants.

```
83        */
84       private void validateUpdateUserDTO(UpdateUserDTO updateUserDTO) {
85 2         if (updateUserDTO.getFirstName() == null || updateUserDTO.getEmail() == null
86 1                 || updateUserDTO.getLastName() == null) {
87                 throw new IllegalArgumentException("First name, last name and email cannot be empty");
88             }
89         }
90
```

(b) Killed mutants.

Figure 5: Presence of mutants in the ***if*** statement.

Mutations

38	1. negated conditional → KILLED
40	1. replaced return value with null for rowing/user/services/UserService::findUserById → KILLED
54	1. replaced return value with null for rowing/user/services/UserService::getUser → KILLED
71	1. removed call to rowing/user/services/UserService::validateUpdateUserDTO → KILLED
73	1. removed call to rowing/user/domain/user/User::setParams → KILLED
76	1. replaced return value with null for rowing/user/services/UserService::updateUser → KILLED
85	1. negated conditional → SURVIVED 2. negated conditional → NO_COVERAGE
86	1. negated conditional → NO_COVERAGE
98	1. replaced return value with null for rowing/user/services/UserService::getUserSelected → KILLED

(a) Description of alive mutants.

Mutations

38	1. negated conditional → KILLED
40	1. replaced return value with null for rowing/user/services/UserService::findUserById → KILLED
54	1. replaced return value with null for rowing/user/services/UserService::getUser → KILLED
71	1. removed call to rowing/user/services/UserService::validateUpdateUserDTO → KILLED
73	1. removed call to rowing/user/domain/user/User::setParams → KILLED
76	1. replaced return value with null for rowing/user/services/UserService::updateUser → KILLED
85	1. negated conditional → KILLED 2. negated conditional → KILLED
86	1. negated conditional → KILLED
98	1. replaced return value with null for rowing/user/services/UserService::getUserSelected → KILLED

(b) Description of killed mutants.

Figure 6: Description of mutants in ***UserService*** class.

Task 2

For doing Task 2 of this assignment, we choose to use the domain Activity. This is because this domain was one of the core domains in our application, and we selected the classes which we are going to mutate in this task from the classes inside of this domain.

2.1 ActivityController

One of the classes we chose to inject a mutation is the class textitActivityController. This is because this class is responsible for our endpoints in this domain and mutations can lead application to break, which makes this class a critical class in the application. We inserted the mutation in the textitacceptUser() method, which is responsible for allowing the owner to accept a user to their activity. We inserted the mutation inside of the method where the certificates of the user are checked when the activity owner puts the user in the Cox position. We commented a line such that the check isn't done properly. When we wrote tests for this case, we saw that the test fails in the mutated code when it should kill the mutant. You can see the test cases and the line we commented in the following commit: `https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM33b/-/merge_requests/61/diffs?commit_id=d47b146794d86dc31e1a4f3299fc4f6b92554676`.
After we uncommented the line (reverted back to the original code), we saw that the test passed and the mutation is killed successfully, which you can see in the commit: `https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM33b/-/merge_requests/61/diffs?commit_id=58f5bc785dd09ec860d2bda8d67c863e39809f1d`.

2.2 ActivityService

Another class we chose to inject mutations in is the class textitActivityService. *ActivityService* is an extremely important class for our application because it not only handles the communication of the requests with the repositories but also is a level closer to them compared to the *activityController*. This class is critical and having mutants that pass or unchecked conditions can affect our application tremendously, even break the build completely. The mutant we injected was changing the exception message for the case where an user that already signed up, signs up again. This can be reviewed in this commit: `https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM33b/-/merge_requests/66/diffs?commit_id=fb0b828a6a40681f4f286dfd9db62169e36823e5`. This behaviour has not been specifically killed so injecting the mutant managed to pass the whole test suite. In order to kill it we implemented a case where we verify whether an user signed up already and also check if the exception message matches as shown in the link below `https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM33b/-/merge_requests/66/diffs?commit_id=2595dc687d60937e913e3283c2e29d0df6b400f7`

2.3 Director

One of the less functional but more important classes we had to verify was the *Director* class. The *Director* class is crucial to our application because it sets the parameters that every builder uses to create new objects. That is why any issues we found not tested could've broken the whole application. So in order to inject mutants we've found that our code does not check the type requirements for competitions so we added a mutant by commenting the line in the director so that the type could be whatever as shown in this commit `https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM33b/-/merge_requests/65/diffs?commit_id=728ee44d776c348cb942bc511fe47a658a9f413a`. By doing this our test suite passed so in order to catch this mutant we added a specific test that checks whether the type attribution for a competition was done correct as follows : `https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM33b/-/merge_requests/65/diffs?commit_id=6e6a9485b96ab677664ee1b30351db7d8268ac4a`

2.4 Match

The last class we chose to inject a mutation is the class textitMatch. This is because the class is responsible for the matching of the applicants of a particular activity and that activity itself. This class is a critical class in the domain because without this class, the users wouldn't be able to sign up, which would broke the whole purpose of the application. We inserted the mutant in one of the controller of the classes, which we receive a MatchingDTO object as a parameter. In this method, we have a case where the ID of the MatchingDTO object is null. In this case, our application assigns a random ID when creating a Match object. Again, for the mutated code we commented this condition and let the activity ID always assigned as the same with MatchingDTO object's ID. When we wrote a test for this case, we saw that the test fails and the mutant is not killed when it should be killed. You can see the test cases and the commented condition in the following commit: `https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM33b/-/merge_requests/59/diffs?commit_id=a3a384cecb503561b80d7b2318a3f7706dd0442a`.
In the original code (when we return to the first state) we saw that the test passed and the mutation is killed, like we expected, this can be seen in this commit: `https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM33b/-/merge_requests/59/diffs?commit_id=e15bb5349c4cbd98ea3d609545c7e2b870ddbf5e`.