Assignment 3

Task 1: Automated Mutation Testing

1.matcher-microservice/src/main/java/nl/tudelft/sem/template/example/domain/Competition.java

Pit Test Coverage Report

Package Summary

nl.tudelft.sem.template.example.domain

Number of Classes	Line Coverage		Mutation Coverage	
14	63%	101/161	57%	31/54

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
Activity.java	93%	13/14	0%	0/1
Certificate.java	100%	14/14	86%	6/7
CertificateAttributeConverter.java	100%	3/3	100%	2/2
Competition.java	100%	10/10	50%	1/2

Pit Test Coverage Report

Package Summary

nl.tudelft.sem.template.example.domain

Number of Classes	Line Coverage		Mutation Coverage	
14	63%	101/161	59%	32/54

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
Activity.java	93%	13/14	0%	0/1
Certificate.java	100%	14/14	86%	6/7
CertificateAttributeConverter.java	100%	3/3	100%	2/2
Competition.java	100%	10/10	100%	2/2

The mutation score for the competition class in the Matcher microservice has been improved by 50% as it can be observed above. The commit of these changes can be found in the “assignment3Mutation” branch as “Changes to competition class for assignment 3”.
Link for the commit:
https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM31c/-/merge_requests/47/diffs?commit_id=217330b1a54c521744cb65c0535644e14dad098f

2.example-microservice/src/main/java/nl/tudelft/sem/template/example/controllers/UserDetailsController.java

Pit Test Coverage Report

Package Summary

nl.tudelft.sem.template.example.controllers

Number of Classes	Line Coverage		Mutation Coverage	
2	21%	11/53	10%	1/10

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
DefaultController.java	29%	7/24	25%	1/4
UserDetailsController.java	14%	4/29	0%	0/6

Pit Test Coverage Report

Package Summary

nl.tudelft.sem.template.example.controllers

Number of Classes	Line Coverage		Mutation Coverage	
2	66%	35/53	80%	8/10

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
DefaultController.java	67%	16/24	50%	2/4
UserDetailsController.java	66%	19/29	100%	6/6

Link for the commit:
https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM31c/-/merge_requests/48/diffs?commit_id=8d0045fbe7dcd169524eca09158d8caca212e46f

3.activity-microservice/src/main/java/nl/tudelft/sem/template/example/domain/Username.java

Package Summary

nl.tudelft.sem.template.example.domain

Number of Classes	Line Coverage		Mutation Coverage	
12	84%	152/182	87%	62/71

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
ActivityRequestModel.java	44%	11/25	80%	4/5
ActivityServiceCreateDelete.java	100%	30/30	100%	10/10
ActivityServiceEdit.java	97%	34/35	88%	23/26
ActivityServiceGet.java	100%	30/30	100%	10/10
Competition.java	100%	10/10	100%	2/2
NetId.java	100%	7/7	100%	1/1
NetIdAttributeConverter.java	100%	3/3	100%	2/2
PositionListConverter.java	100%	3/3	100%	2/2
TimeSlot.java	71%	17/24	67%	4/6
TimeSlotConverter.java	100%	3/3	100%	2/2
Training.java	67%	4/6	50%	2/4
Username.java	0%	0/6	0%	0/1

Package Summary

nl.tudelft.sem.template.example.domain

Number of Classes	Line Coverage		Mutation Coverage	
12	91%	165/182	94%	67/71

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
ActivityRequestModel.java	44%	11/25	80%	4/5
ActivityServiceCreateDelete.java	100%	30/30	100%	10/10
ActivityServiceEdit.java	97%	34/35	92%	24/26
ActivityServiceGet.java	100%	30/30	100%	10/10
Competition.java	100%	10/10	100%	2/2
NetId.java	100%	7/7	100%	1/1
NetIdAttributeConverter.java	100%	3/3	100%	2/2
PositionListConverter.java	100%	3/3	100%	2/2
TimeSlot.java	100%	24/24	83%	5/6
TimeSlotConverter.java	100%	3/3	100%	2/2
Training.java	100%	6/6	100%	4/4
Username.java	67%	4/6	100%	1/1

The mutation score for the Username class has been improved by 100%, the mutation score of the Training class has been improved by 50% and the mutation score of the TimeSlot class has been improved by 16%.
Link for the commit:
https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM31c/-/merge_requests/49/diffs?commit_id=66cbf6146ca882d2cbc5d4da63c866b55ce6ad7c

4.matcher-microservice\src\main\java\nl\tudelft\sem\template\example\domain\Activity.java

Activity.java	93%	13/14	0%	0/1
Certificate.java	100%	14/14	86%	6/7
CertificateAttributeConverter.java	100%	3/3	100%	2/2
Competition.java	100%	10/10	100%	2/2

Name	Line Coverage		Mutation Coverage	
Activity.java	100%	14/14	100%	1/1
Certificate.java	100%	14/14	86%	6/7
CertificateAttributeConverter.java	100%	3/3	100%	2/2
Competition.java	100%	10/10	100%	2/2

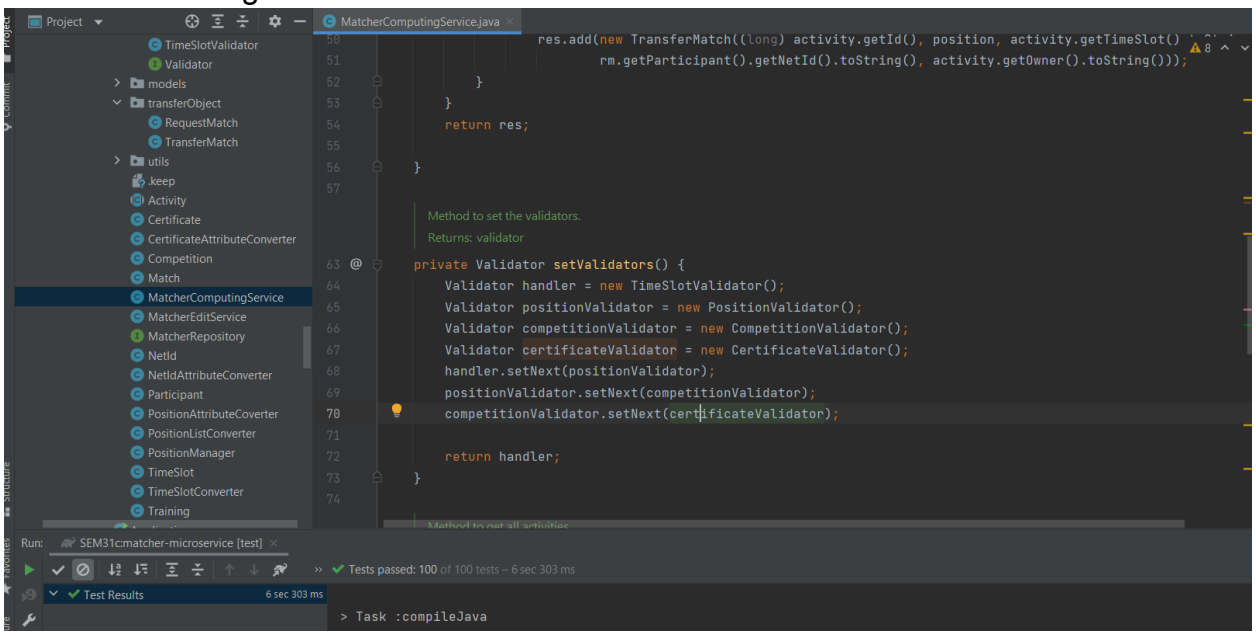
The mutation score for the Activity class has been improved by 100%.
Link for the commit:
https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM31c/-/merge_requests/50/diffs?commit_id=7933ff1f3931bae67af88af3d61b0b5e7d03b4de

Task 2: Manual Mutation Testing

1.matcher-microservice/src/main/java/nl/tudelft/sem/template/example/domain/MatcherComputingService.java

The MatcherComputingService is a critical component of our system because it encapsulates the main functionality of our application: filtering activities based on different conditions. The method that we decided to introduce a bug in is setValidators(), which initializes the filtering process, implemented using the Chain of Responsibility design pattern. The purpose of this method is to add Validators to the chain.
We thought that while adding the Validators to the chain, the developer might introduce the same validator twice, letting one of the desired validators out. In our case, the TimeSlotValidator was added twice to the chain and the developer forgot to add the PositionValidator because he might have just copied and pasted the initialisation of the first validator. The first snippet(line 65) shows the method before introducing the bug and second shows the method after introducing it.

Before introducing the mutant:



After introducing the mutant:

```
                res.add(new TransferMatch((long) activity.getId(), position, activity.getTimeSlot().toString()
                        rm.getParticipant().getNetId().toString(), activity.getOwner().toString()));
            }
        }
        return res;

    }

    Method to set the validators.
    Returns: validator
    public Validator setValidators() {
        Validator handler = new TimeSlotValidator();
        Validator positionValidator = new TimeSlotValidator();
        Validator competitionValidator = new CompetitionValidator();
        Validator certificateValidator = new CertificateValidator();
        handler.setNext(positionValidator);
        positionValidator.setNext(competitionValidator);
        competitionValidator.setNext(certificateValidator);

        return handler;
    }

    Method to get all activities.
```

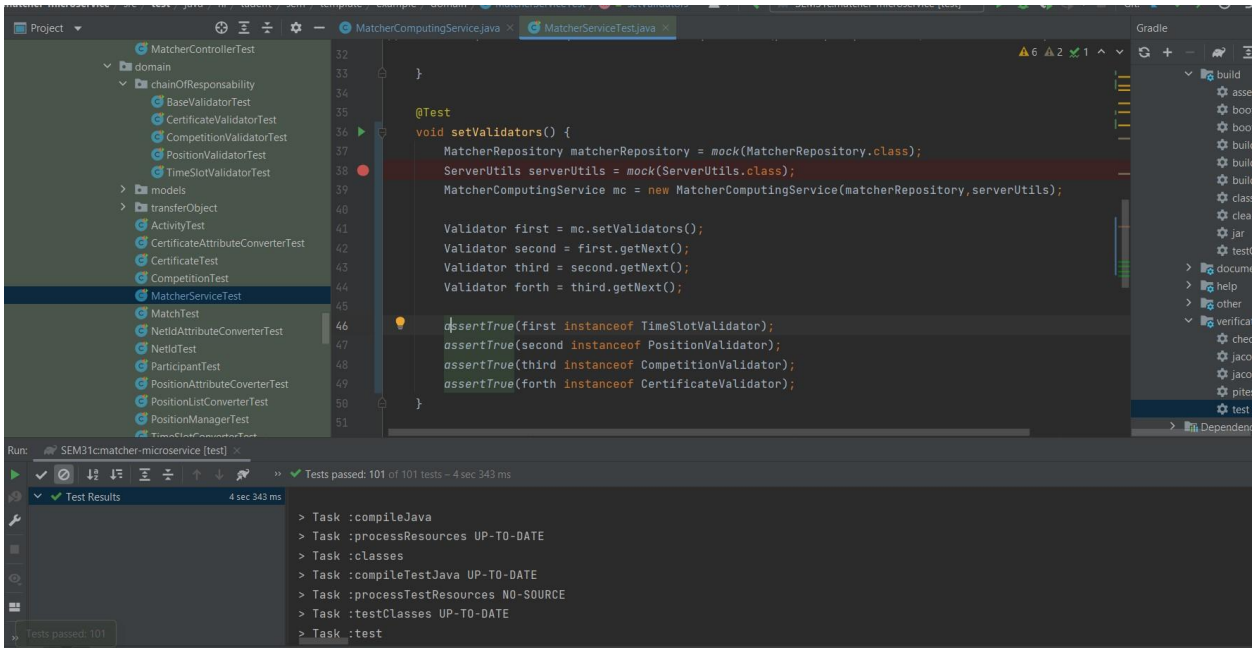
```
Run: SEM31c:matcher-microservice [test]
Tests passed: 101 of 101 tests – 4 sec 359 ms
Test Results 4 sec 359 ms
> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
```

It can be observed that even though we introduced a bug, it was not killed by the old set of tests. That is why we decided to add one more test that will verify the order and the type of the validators, which will fail in the bugged version of the code as it can be observed below.

```
}

@Test
void setValidators() {
    MatcherRepository matcherRepository = mock(MatcherRepository.class);
    ServerUtils serverUtils = mock(ServerUtils.class);
    MatcherComputingService mc = new MatcherComputingService(matcherRepository,serverUtils);

    Validator first = mc.setValidators();
    Validator second = first.getNext();
    Validator third = second.getNext();
    Validator forth = third.getNext();

    assertTrue(first instanceof TimeSlotValidator);
    assertTrue(second instanceof PositionValidator);
    assertTrue(third instanceof CompetitionValidator);
    assertTrue(forth instanceof CertificateValidator);
}
```

```
Run: MatcherServiceTest.setValidators
Tests failed: 1 of 1 test – 492 ms
Test Results 492 ms
MatcherServiceTest 492 ms
setValidators() 492 ms
> Task :matcher-microservice:compileJava UP-TO-DATE
> Task :matcher-microservice:processResources UP-TO-DATE
> Task :matcher-microservice:classes UP-TO-DATE
> Task :matcher-microservice:compileTestJava
> Task :matcher-microservice:processTestResources NO-SOURCE
> Task :matcher-microservice:testClasses
> Task :matcher-microservice:test FAILED
```

Lastly, after the mutant was killed, we can observe that all tests pass in the correct version of the code.



All the changes presented above can be found in the commit “Add test for Assg 3 part 2”.
Link for the commit:
https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM31c/-/merge_requests/47/diffs?commit_id=14441327a7ab869b310d7c93e25fb05b49695deb

2.matcher-microservice/src/main/java/nl/tudelft/sem/template/example/domain/MatcherEditService.java

When the owner of an activity accepts a list of participants from the requested users, all the matches that were created between the activity and the user should be deleted from the database. This process is done when an endpoint sendAcceptedUsers calls removeMatches which is a method in MatcherEditService. This is the reason why MatcherEditService class is important since it does the work of deletion of matches. In the method removeMatches, it looks for the list of matches to be deleted which is done in another method of this class called findMatch. FindMatch loops over the list of matches and seeks for the match that has the same id as the transfermatch object. In this process, we introduced a mutant that forgets to put a negation in if condition. In the for loop of the original code, it puts the match that does not exist in the list. But in the code with the inserted bug, the correct matches that should be deleted are not appended to the list.

Before introducing the mutant:

```
70 @    public List<Match> findMatch(TransferMatch tr,List<Match> matches){
71          List<Match> toDeletMatches= new ArrayList<>();
72          for(Match m : matches){
73              if(m.getActivityId().equals(tr.getActivityId()))
74                  if(!toDeletMatches.contains(m))
75                      toDeletMatches.add(m);
76          }
77          return toDeletMatches;
78      }
```

After introducing the mutant:

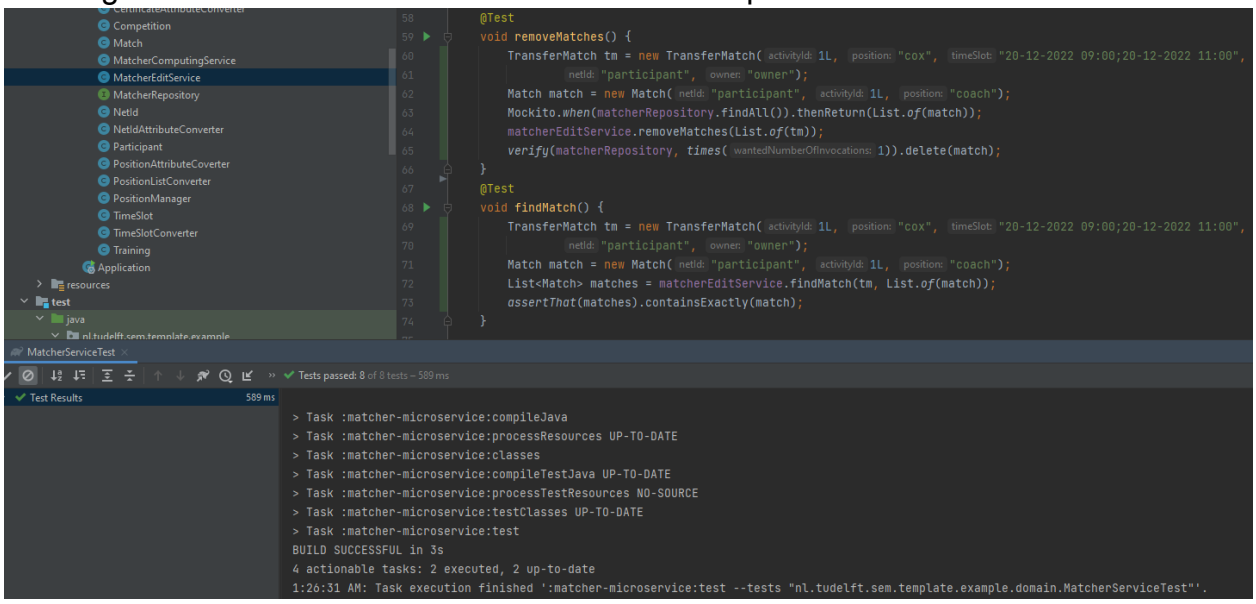
```
70 @    public List<Match> findMatch(TransferMatch tr,List<Match> matches){
71          List<Match> toDeletMatches= new ArrayList<>();
72          for(Match m : matches){
73              if(m.getActivityId().equals(tr.getActivityId()))
74                  if(toDeletMatches.contains(m))
75                      toDeletMatches.add(m);
76          }
77          return toDeletMatches;
78      }
```

In line 74, negation operator is deleted.

```
Match
MatcherComputingService
MatcherEditService
MatcherRepository
NetId
NetIdAttributeConverter
Participant
PositionAttributeCoverter
PositionListConverter
PositionManager
TimeSlot
TimeSlotConverter
Training
Application
resources
test
java
nl.tudelft.sem.template.example
MatcherServiceTest

58      @Test
59      void removeMatches() {
60          TransferMatch tm = new TransferMatch( activityId: 1L,  position: "cox",  timeSlot: "20-12-2022 09:00;20-12-2022 11:00",
61                  netId: "participant",  owner: "owner");
62          Match match = new Match( netId: "participant",  activityId: 1L,  position: "coach");
63          Mockito.when(matcherRepository.findAll()).thenReturn(List.of(match));
64          matcherEditService.removeMatches(List.of(tm));
65          verify(matcherRepository, times( wantedNumberOfInvocations: 1)).delete(match);
66      }
67      @Test
68      void findMatch() {
69          TransferMatch tm = new TransferMatch( activityId: 1L,  position: "cox",  timeSlot: "20-12-2022 09:00;20-12-2022 11:00",
70                  netId: "participant",  owner: "owner");
71          Match match = new Match( netId: "participant",  activityId: 1L,  position: "coach");
72          List<Match> matches = matcherEditService.findMatch(tm, List.of(match));
73          assertThat(matches).containsExactly(match);

Tests failed: 2, passed: 6 of 8 tests – 577 ms
Test Results                577 ms
MatcherServiceTest          577 ms
findMatch()                  68 ms
removeMatches()              37 ms

> Task :matcher-microservice:compileJava UP-TO-DATE
> Task :matcher-microservice:processResources UP-TO-DATE
> Task :matcher-microservice:classes UP-TO-DATE
> Task :matcher-microservice:compileTestJava
> Task :matcher-microservice:processTestResources NO-SOURCE
> Task :matcher-microservice:testClasses
> Task :matcher-microservice:test FAILED
```

We see that the mutant was killed by having two tests failed. First test is for the method with mutant which is findMatch and second test was for the method that uses findMatch which is removeMatch.

Running the test with the correct version of the code passes all the tests.



Link for the commit:
https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM31c/-/merge_requests/48/diffs?commit_id=ff5e24e5c833e7ba2f6fac3fcd7d4e82acdd04fb

Unfortunately, we have ran out of classes to mutate on in the matcher microservice, so we have moved on to the Notification microservice.

3.notification-microservice/src/main/java/nl/tudelft/sem/template/example/domain/NotificationEditService.java

The NotificationEditService is crucial to both the functioning and maintaining of our system, since it includes functionality for creating, storing and deleting notification from our database. These methods are mainly used in the NotificationController, class which is in charge of the API endpoints. Whenever a match is sent to the Notification controller, the createNotification method is called. On the other hand, when a user is accepted to participate in an activity, that specific user will receive a Notification, which should at some point (maybe a day after the specific activity) be deleted, in order to keep the database clean and speed up the query process.

Since we have concluded that deleting Notifications is integral to the microservice and it is not properly tested, we will introduce a mutant here. The method will instead save the notification instead of deleting it.

Before introducing the mutant :

```
/**
 * Deletes a notification.
 * @param n
 */
2 usages  alexandrumarin *
public void deleteNotification(Notification n) {
    notificationRepository.delete(n);
}
```

```
NotificationServiceTest
Tests passed: 8 of 8 tests
Test Results                498 ms    > Task :notificat
```

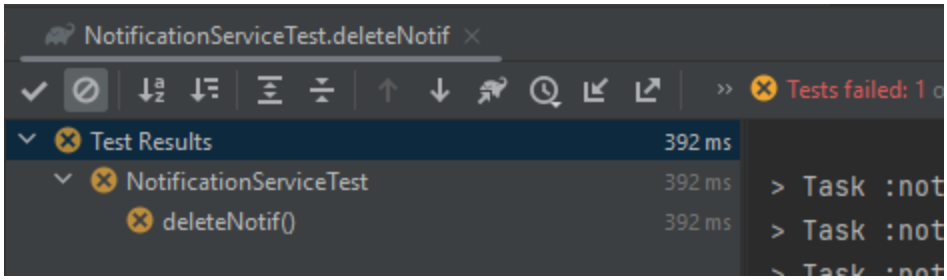
After introducing the mutant :

```
/**
 * Deletes a notification.
 * @param n
 */
2 usages  alexandrumarin *
public void deleteNotification(Notification n) {
    notificationRepository.save(n);
}
```

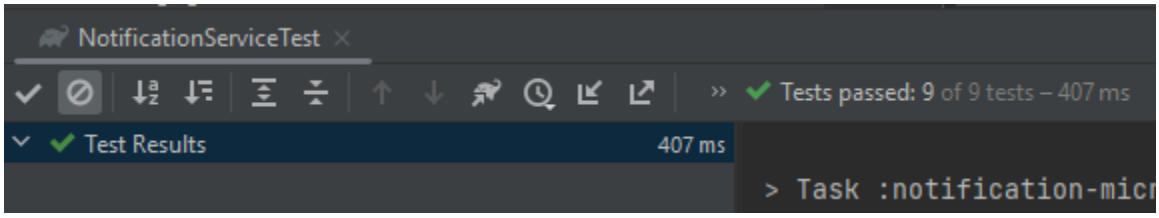
```
NotificationServiceTest
Tests passed: 8 of 8 tests
Test Results                498 ms    > Task :notificat
```

Test that fails in the mutant-infested code:

```
no usages  new *
@Test
public void deleteNotif(){
    ArgumentCaptor<Notification> captor = ArgumentCaptor.forClass(Notification.class);
    Notification notif = new Notification(new ActivityId("1"), paula, owner, message: "message", ownerNotification: false);
    serviceEdit.deleteNotification(notif);
    verify(notificationRepo).delete(captor.capture());
}
```



Running all tests with clean code passes everything:



Link for the commit:
https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM31c/-/commit/167972b96c2784380707bee14e8b441458156597

4.matcher-microservice/src/main/java/nl/tudelft/sem/template/example/domain/Certificate.java

The Certficate class is a crucial part of our system as it takes a part in matching the users with activities that require a specific certificate level. The isBetterCertificate checks if the user applying for an activity has a certificate that is equal or better than the one required, as could be seen in the code below:

Code before introducing the mutant

```
/**
 * Check if the certificate is better than the other.
 * @param other
 * @return true if the certificate is better than the other
 */
public boolean isBetterCertificate(Certificate other) {
    Map<String,Integer> hm = Map.of( k1: "C4", v1: 1,  k2: "4+",  v2: 2, k3: "8+", v3: 3);

        return hm.get(this.getCertificateType())>=hm.get(other.getCertificateType());
}
```

As we can see, all test cases pass:

```
CertificateTest
Tests passed: 5 of 5 tests – 47 ms
Test Results 47 ms
> Task :matcher-microserv
```

Now I introduced a mutant, and the method checks if the certificate is better than the other, but does not include equal certificates.

Code after the mutant

```
/**
 * Check if the certificate is better than the other.
 * @param other
 * @return true if the certificate is better than the other
 */
public boolean isBetterCertificate(Certificate other) {
    Map<String,Integer> hm = Map.of( k1: "C4", v1: 1,  k2: "4+",  v2: 2, k3: "8+", v3: 3);

        return hm.get(this.getCertificateType())>hm.get(other.getCertificateType());
}
```

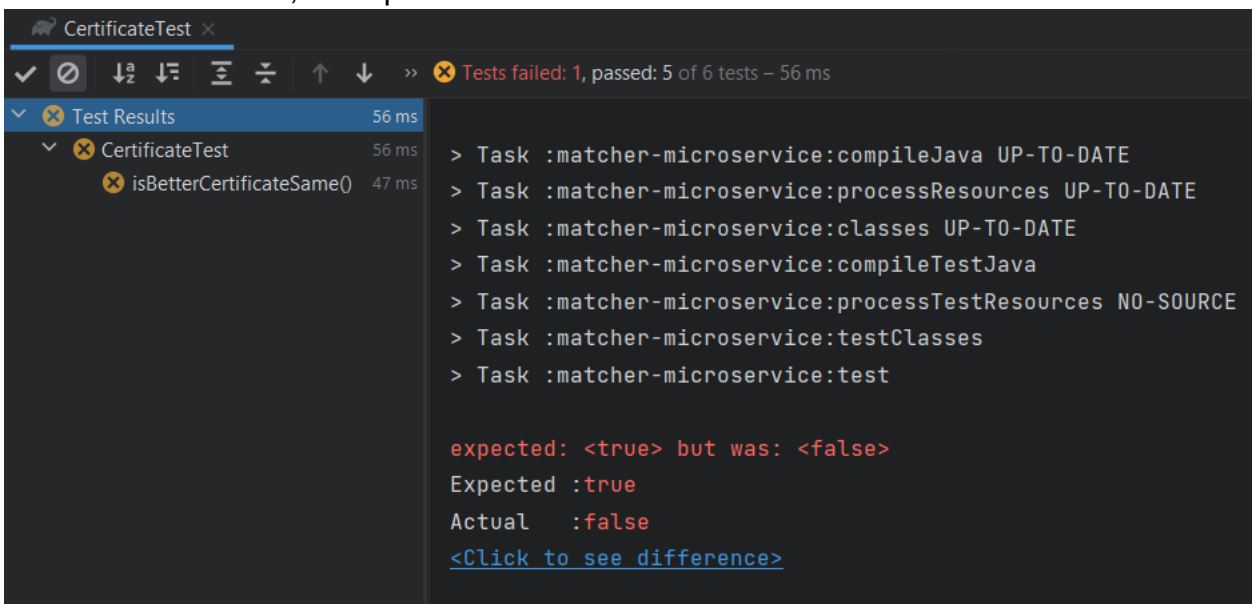
As we can see, all test cases still pass:

```
CertificateTest
Tests passed: 5 of 5 tests – 53 ms
Test Results 53 ms
> Task :matcher-microservice:com
```

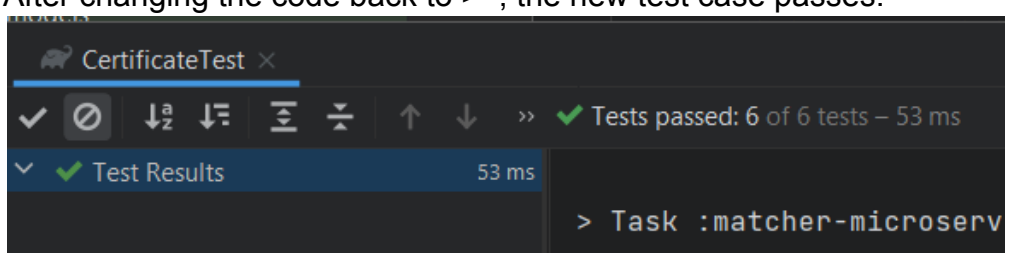
To take care of that, I added a new test case that checks equal certificates:

```
@Test
void isBetterCertificateSame() {
    Certificate certificate= new Certificate( certificateType: "C4");
    Certificate certificate1= new Certificate( certificateType: "C4");
    assertTrue(certificate1.isBetterCertificate(certificate));
}
```

The test for that fails, as expected:



After changing the code back to >=, the new test case passes:



Link for the commit:
https://gitlab.ewi.tudelft.nl/cse2115/2022-2023/SEM31c/-/commit/59af87dccadaac0fc9893c15a54f1068d6bd7637