

# Test Driven Development!

Yes, we can!



# About me:



- Java enthusiastic.
  - Clean Code, SOLID, TDD, BDD, DevOps...
- Now in WATA FACTORY
- Keep the contact:
  - @geeksusma



# Why tests are needed?

 **Francisco Moreno**  
@morvader

De manera general, ¿escribir tests hace que entreguéis funcionalidad más rápido? **#PreguntaSeria**  
Se agradecen RTs

Sí	46.3%
No	43.6%
Igual	10.1%

337 votos · Resultados finales

5:47 p. m. · 25 may. 2020 · [Twitter Web App](#)

From 337 votes, some conclusions can be taken

# Why tests are needed?

- We are in the 21th century but...
- So many people don't writes tests!
- Or if they write tests their tests are not good enough!
- Or even worst, they think tests don't help to deliver or coding faster!

# Remember, What is a good Test?

- F.I.R.S.T principles:

- Fast



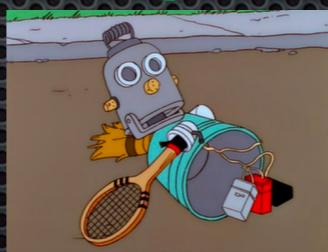
- Isolated/Independent



- Repeatable



- Self-Validating



- Thorough

# Ok, so now what the hell is TDD?

- Is a Software Engineering practice which involves:
  - TFD (Test First Development)
  - Refactoring

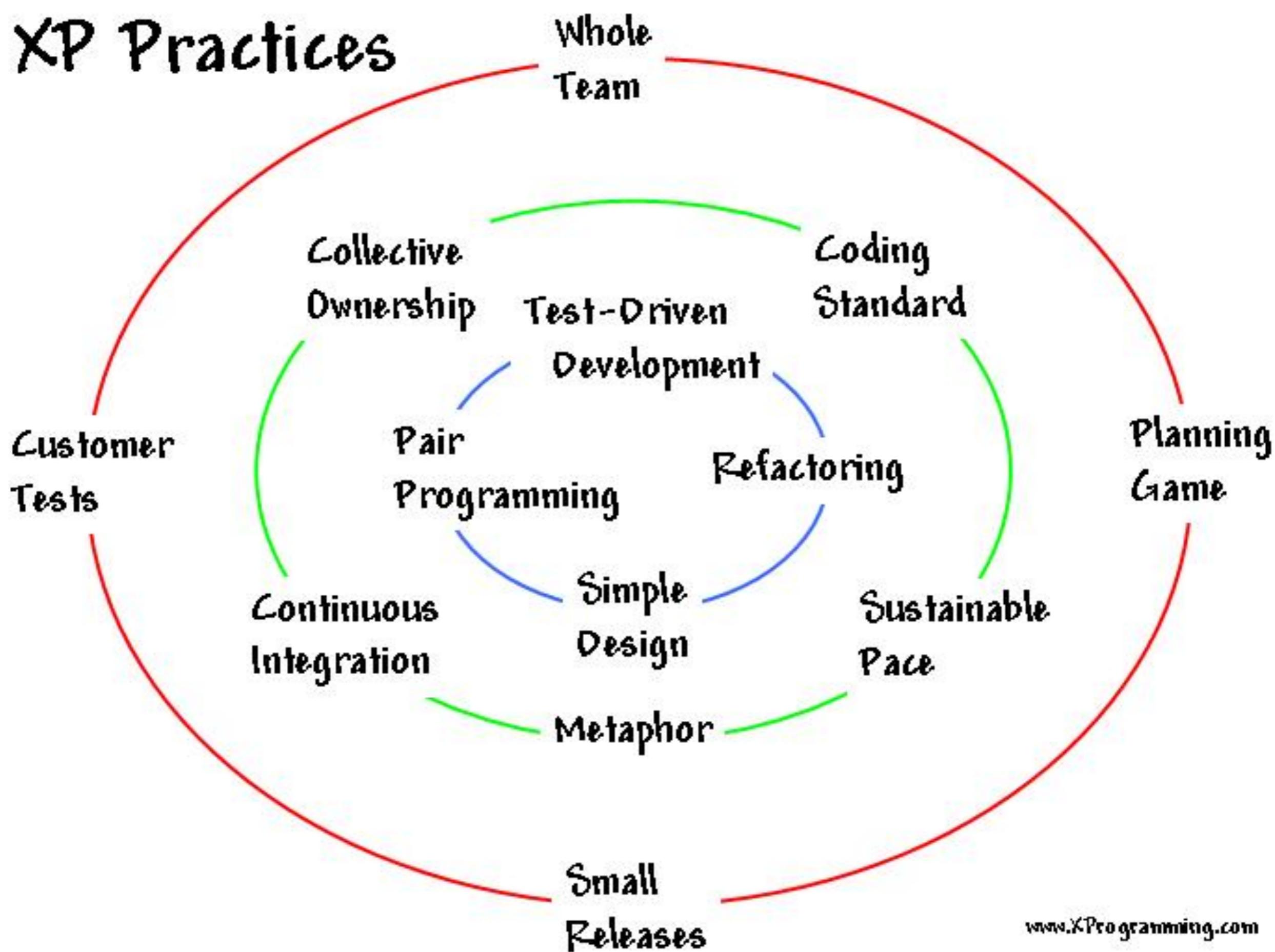
# Why TDD?

- It makes you code faster (yes trust me)
- It makes your code less tolerant to bugs (at least the trivial ones)
- It will encourage to you to write smaller classes
- It will help you to have useful tests beyond the test coverage. Good to refactor afterwards.

# Why TDD?

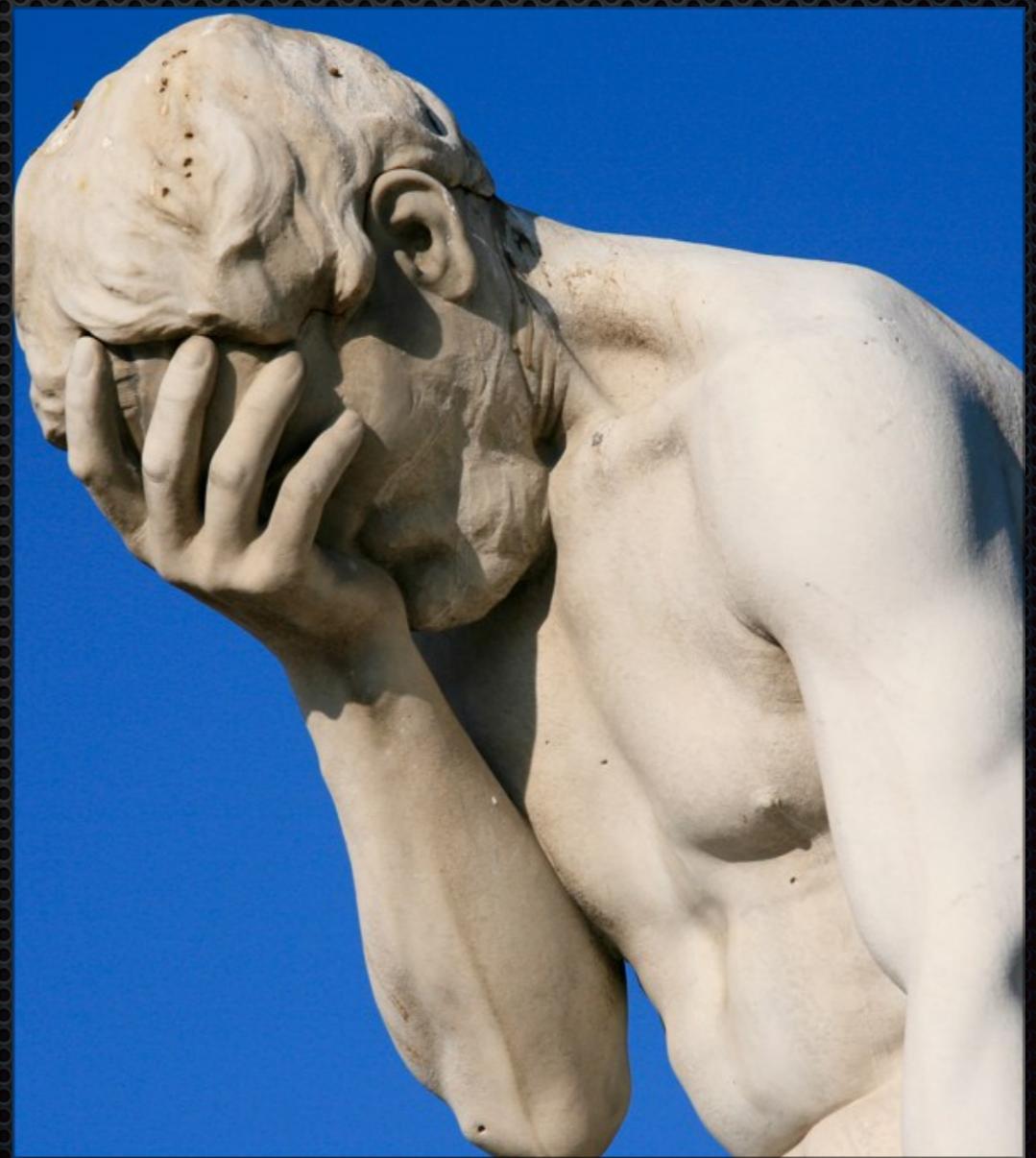
- It will make you to ask questions about your code at coding time
- Encourage for Pair Programming
  - One dev writes the test, the second fix the code
- Tests is the first milestone to become an XP Dev
- No test no party :-)

# XP Practices

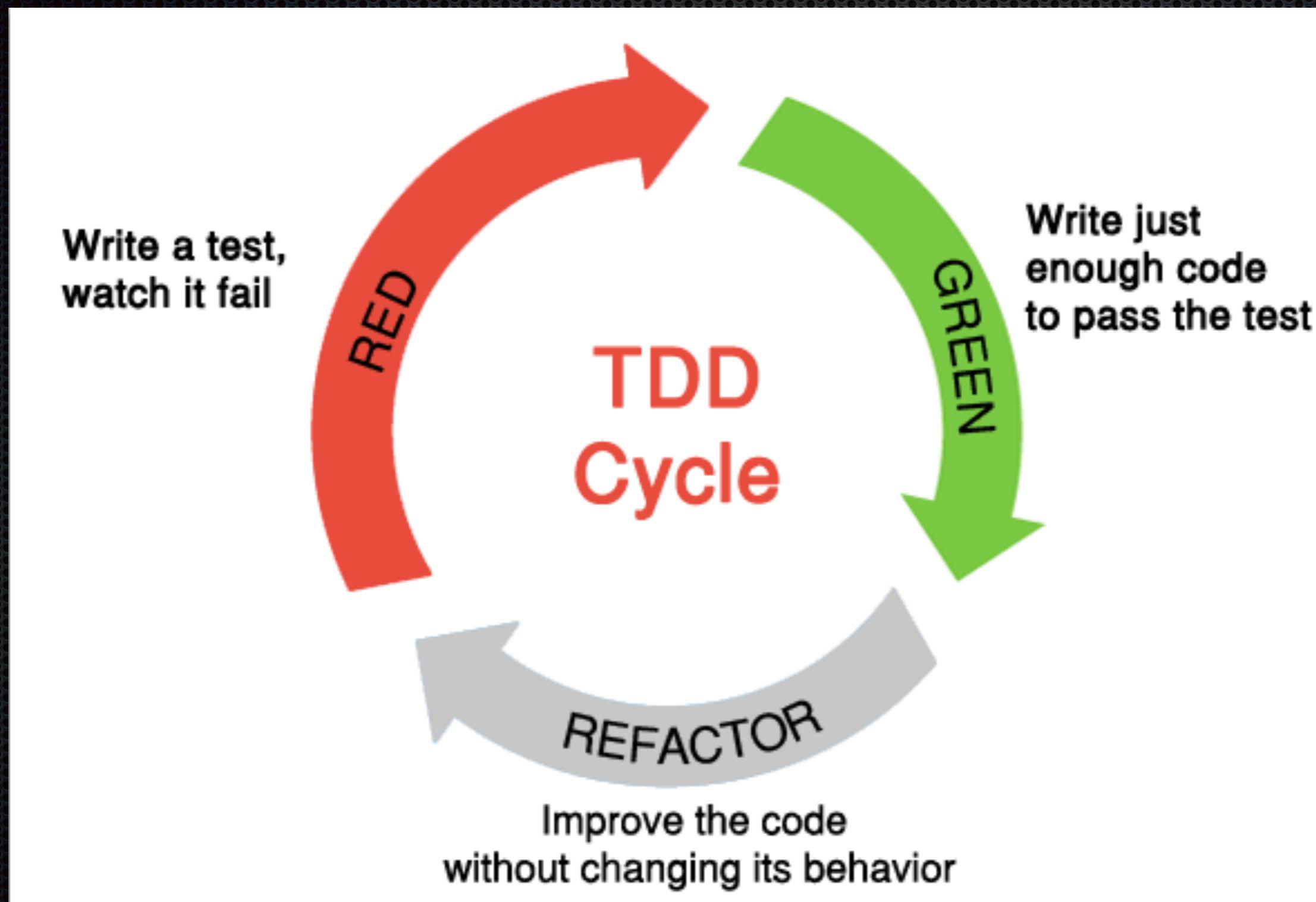


# What is not and some fallacies

- A myth because no one really do TDD
- A way to commit the test coverage (writing tests after)
- Tests make us code slower
- Only for Java devs because “they have Junit”
- Is not a testing task, is just coding!
- Not for everything!



# The TDD Cycle



# In details



- Write a test, watch it failing:
  - Use your favorite IDE
  - Add only the minimum production code to make your code compiling
  - Run the test
- “Never trust in a test if you never watched it failing first”



# In details



- Write just enough code to pass the test:
  - Use as less code lines as you can
  - No matter how stupid your code looks in this step  
(KISS -> Keep It Simple and Stupid)
  - Do not forget to re-run the old tests!

# In details



- Refactor:
  - For code duplicity
  - For test redundant even repeated
    - (!) Is quite important to have readable tests!
  - And go back to the starting point

# In action (Simple use case)

- We need a way (using best practices of course) to fetch an user by his id.
- Please, don't return the User Entity (just return User as the domain object)
- Ensure not illegal fetching is allowed

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Bar:** Shows tabs for HELP.md, pom.xml, .gitignore, and GetUserDAOtest.java (which is the active tab).
- Project Structure:** The project is named "tdd-example" located at "/wata/tdd-example". It contains .idea, .mvn, src (with main and test), and a pom.xml file.
- Editor:** The code for GetUserDAOtest.java is displayed. The class defines a private field getUserDAO and a setup method. It contains a test method named should\_throwIllegalUse\_when\_idNotGiven that uses catchThrowable to assert that a NoSuchElementException is thrown when null is passed to the DAO's getById method.

```
1 package es.geeksusma.tddexample.user;
2
3 import org.junit.jupiter.api.BeforeEach;
4 import org.junit.jupiter.api.Test;
5
6 import static org.assertj.core.api.Assertions.assertThat;
7 import static org.assertj.core.api.Assertions.catchThrowable;
8
9 class GetUserDAOtest {
10
11     private GetUserDAO getUserDAO;
12
13     @BeforeEach
14     void setUp() {
15         getUserDAO = new GetUserDAO();
16     }
17
18     @Test
19     void should_throwIllegalUse_when_idNotGiven() {
20
21         //when
22         final Throwable raisedException = catchThrowable(() -> getUserDAO.byId(null));
23
24         //then
25         assertThat(raisedException).isInstanceOf(IllegalArgumentException.class)
26             .hasMessageContaining("Id to fetch is mandatory");
27     }
28 }
```

```
ge es.geeksusma.tddexample.user;

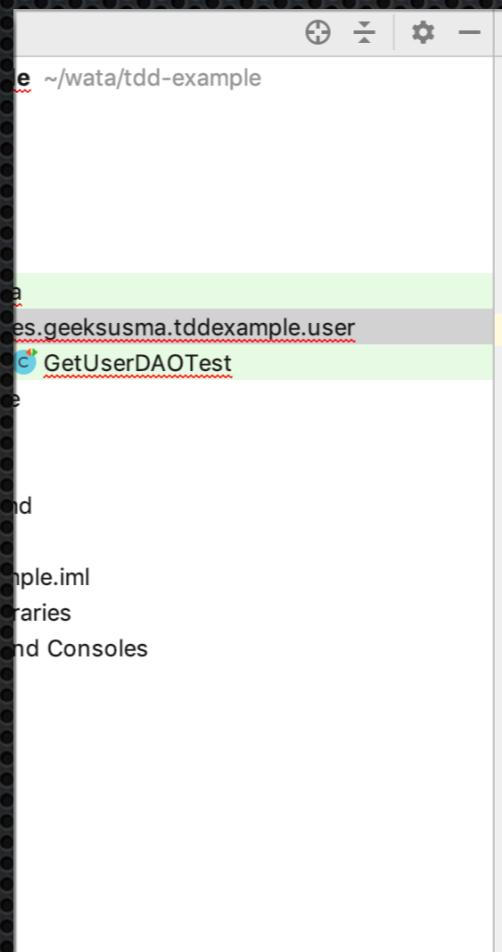
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.assertj.core.api.Assertions.assertThat;
import static org.assertj.core.api.Assertions.catchThrowable;

 GetUserDAOTest {

    private GetUserDAO getUserDAO;
    @BeforeEach
    void setUp() {
        getUserDAO = new GetUserDAO();
    }

    @Test
    void should_th ...
    //when
    final Throwable raisedException = catchThrowable(() - ...
    //then
    assertThat(raisedException).isInstanceOf(IllegalArgumentException)
        .hasMessageContaining("Id to fetch is mandatory");
}
```



The screenshot shows an IDE interface with a code editor and a navigation bar at the top. The code editor displays Java test code. A tooltip is open over the variable 'getUserDAO', listing several options:

- Create class 'GetUserDAO'
- Create enum 'GetUserDAO'
- Create inner class 'GetUserDAO'
- Create interface 'GetUserDAO'
- Create type parameter 'GetUserDAO'
- Add Maven Dependency...
- Change access modifier

The code in the editor is as follows:

```
package es.geeksusma.tddexample.user;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.assertj.core.api.Assertions.assertThat;
import static org.assertj.core.api.Assertions.catchThrowable;

class GetUserDAOTest {

    private GetUserDAO getUserDAO;

    @BeforeEach
    void setUp() {
        getUserDAO = new GetUserDAO();
    }

    @Test
    void should_throwIllegalUse_when_idNotGiven() {
        //when
        final Throwable raisedException = catchThrowable(() -> getUserDAO ...

        //then
        assertThat(raisedException).isInstanceOf(IllegalArgumentException)
            .hasMessageContaining("Id to fetch is mandatory");
    }
}
```

Tests failed: 1 of 1 test – 94 ms

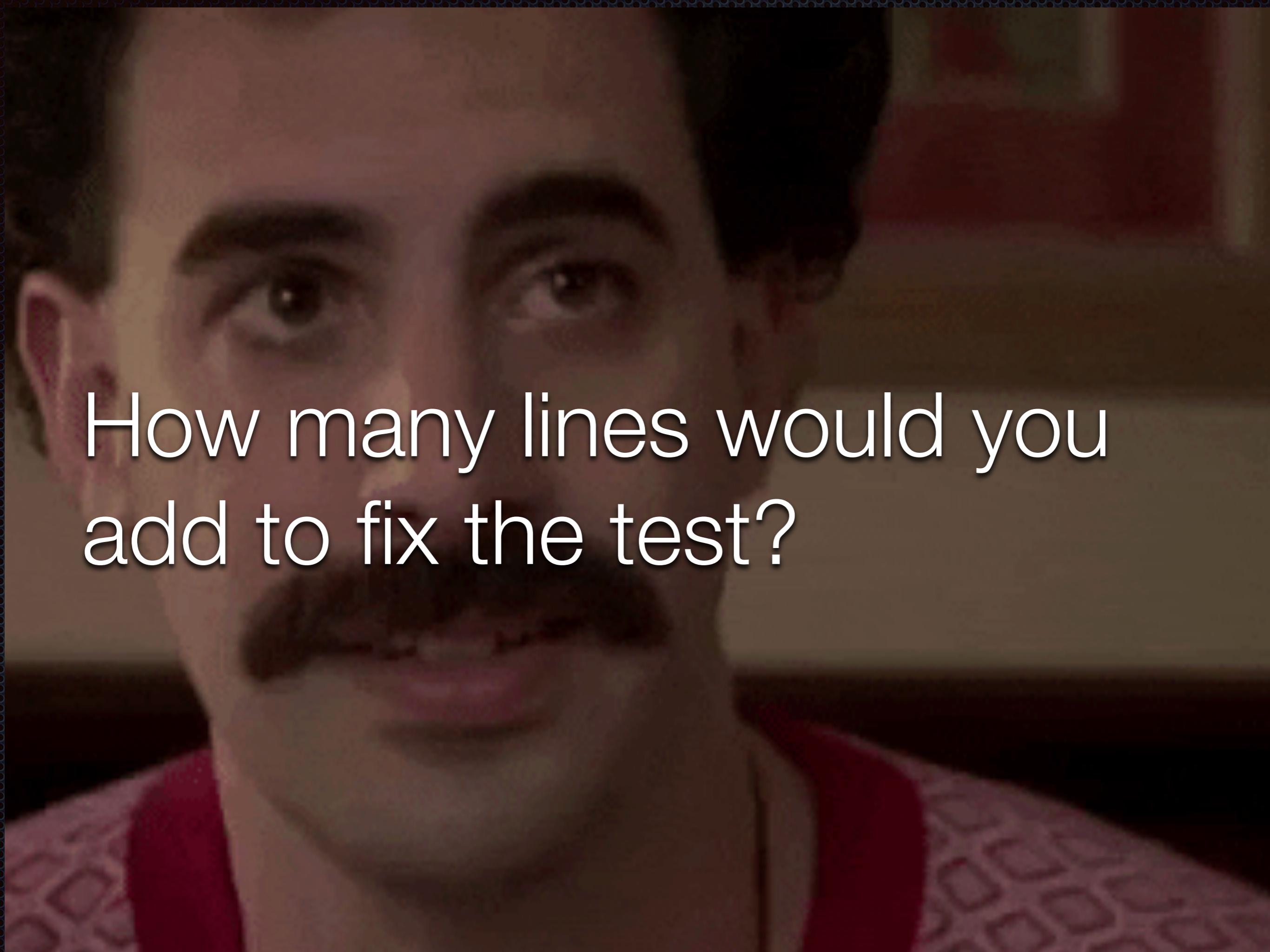
Test Results

- GetUserDAOTest
  - should\_throwIllegalUse\_when\_idNotGiven()

94 ms 94 ms 94 ms

/Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home/bin/java ...

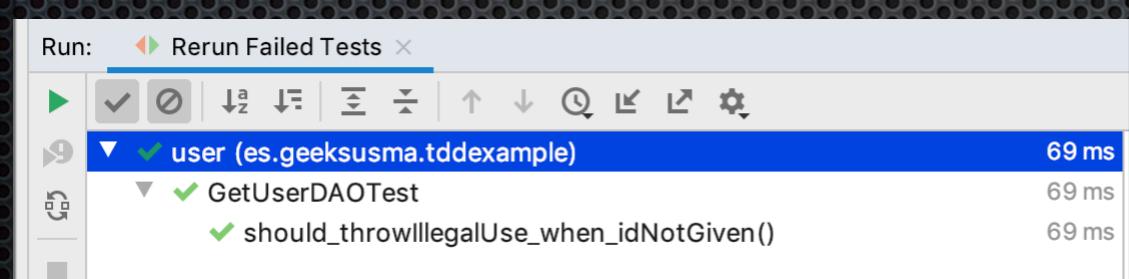
```
java.lang.AssertionError:  
Expecting:  
  <java.lang.UnsupportedOperationException: implement me!>  
to be an instance of:  
  <java.lang.IllegalArgumentException>  
but was:  
  <"java.lang.UnsupportedOperationException: implement me!"  
  at es.geeksusma.tddexample.user.GetUserDAO.byId(GetUserDAO.java:5)
```



How many lines would you  
add to fix the test?

A screenshot of an IDE showing a Java file named  `GetUserDAO.java`. The code defines a class  `GetUserDAO` with a method  `getById` that throws an `IllegalArgumentException` if the  `id` parameter is null. A yellow warning icon is present on the line where the exception is thrown.

```
1 package es.geeksusma.tddexample.user;
2
3 class GetUserDAO {
4     void getById(final Long id) {
5         throw new IllegalArgumentException("Id to fetch is mandatory");
6     }
7 }
8
```



```
@Test
void should_returnEmpty_when_notFound() {
    //given
    final Long notExistingId = 123L;

    //when
    final Optional<User> found = getUserDAO.byId(notExistingId);

    //then
    assertThat(found).isEmpty();
}
```

Tests failed: 1 of 1 test – 20 ms

/Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/H

java.lang.IllegalArgumentException: Id to fetch is mandatory

at es.geeksusma.tddexample.user.GetUserDAO.byId(GetUserDAO.j  
at es.geeksusma.tddexample.user.GetUserDAOTest.should\_return  
at java.base/java.util.ArrayList.forEach(ArrayList.java:1540)

```
1 package es.geeksusma.tddexample.user;
2
3 import java.util.Optional;
4
5 class GetUserDAO {
6     Optional<User> getById(final Long id) {
7         if (id == null) {
8             throw new IllegalArgumentException("Id to fetch is mandatory");
9         }
10        return Optional.empty();
11    }
12 }
13
```

GetUserDAOTest > should\_returnEmpty\_when\_notFound()

Run: GetUserDAOTest



Tests passed: 2 of 2 tests – 84 ms



Test Results

- GetUserDAOTest
  - should\_returnEmpty\_when\_notFound()
  - should\_throwIllegalUse\_when\_idNotGiven()

84 ms

84 ms

79 ms

5 ms

/Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home/bin/java ...

Process finished with exit code 0

```
47
48     @Test
49     void should_fetchFromRepository_when_userIdIsValid() {
50         //given
51         final Long userId = 321L;
52
53         //when
54         getRepository().byId(userId);
55
56         //then
57         then(repository).should().findById();
58     }

```

GetUserDAOTest > setUp()

Run: Rerun Failed Tests



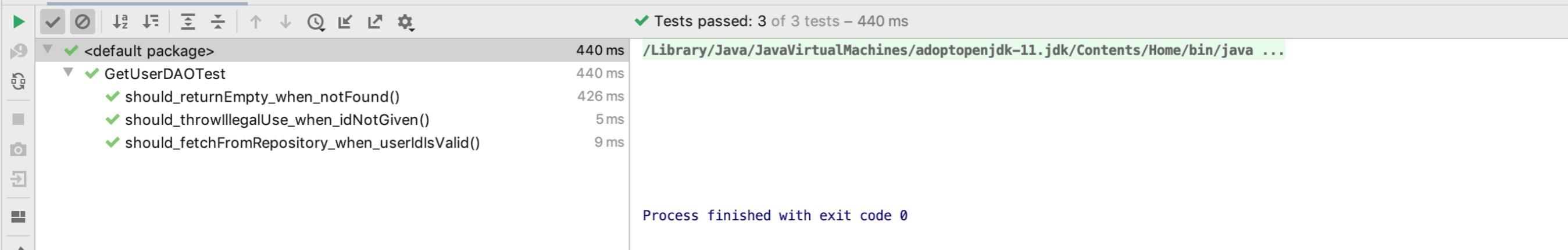
Tests failed: 1 of 1 test – 459 ms

9	user (es.geeksusma.tddexample)	459 ms	/Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home/bin/java ...
	GetUserDAOTest	459 ms	
	should_fetchFromRepository_when_userIdIsValid()	459 ms	<p>Wanted but not invoked: userRepository.findById(); -&gt; at es.geeksusma.tddexample.user.GetUserDAOTest.should_fetchFromRepository_when_userIdIsValid(GetUserDAOTest.java:49) Actually, there were zero interactions with this mock.</p> <p>Wanted but not invoked: userRepository.findById(); -&gt; at es.geeksusma.tddexample.user.GetUserDAOTest.should_fetchFromRepository_when_userIdIsValid(GetUserDAOTest.java:49) Actually, there were zero interactions with this mock.</p> <p>+ at es.geeksusma.tddexample.user.GetUserDAOTest.should_fetchFromRepository_when_userIdIsValid(GetUserDAOTest.java:49) + at java.base/java.util.ArrayList.forEach(ArrayList.java:1540) &lt;9 internal calls&gt; + at java.base/java.util.ArrayList.forEach(ArrayList.java:1540) &lt;21 internal calls&gt;</p>

```
4
5     class GetUserDAO {
6         private final UserRepository userRepository;
7
8     @   GetUserDAO(final UserRepository userRepository) {
9
10        this.userRepository = userRepository;
11    }
12
13    Optional<User> getById(final Long id) {
14        if (id == null) {
15            throw new IllegalArgumentException("Id to fetch is mandatory");
16        }
17        userRepository.findById();
18        return Optional.empty();
19    }
}
```

GetUserDAO > byId()

Run: ◀▶ GetUserDAOTest (1) ×



```
60 }
61
62     @Test
63     void should_mapEntity_when_found() {
64         //given
65         final Long existingUserId = 67676L;
66         final UserEntity foundUser = new UserEntity();
67
68         given(userRepository.findById(existingUserId)).willReturn(Optional.of(foundUser));
69
70         //when
71         getUserDAO.findById(existingUserId);
72
73         //then
74         then(userDomainMapper).should().map(foundUser);
75     }
76 }
77
```

GetUserDAOTest > should\_mapEntity\_when\_found()

Run: GetUserDAOTest.should\_mapEntity\_when\_f...

Tests failed: 1 of 1 test – 441 ms

Test Results

GetUserDAOTest

should\_mapEntity\_when\_found() 441 ms

org.mockito.exceptions.misusing.NullInsteadOfMockException:  
Argument passed to verify() should be a mock but is null!  
Examples of correct verifications:  
 verify(mock).someMethod();  
 verify(mock, times(10)).someMethod();  
 verify(mock, atLeastOnce()).someMethod();  
 not: verify(mock.someMethod());  
Also, if you use @Mock annotation don't miss initMocks()

```
11     this.userRepository = userRepository,
12     this.userDomainMapper = userDomainMapper;
13   }
14
15   Optional<User> getById(final Long id) {
16     if (id == null) {
17       throw new IllegalArgumentException("Id to fetch is mandatory");
18     }
19     userRepository.findById(id).map( u -> userDomainMapper.map(u));
20     return Optional.empty();
21   }
22
23 }
```

GetUserDAO &gt; getById()

Run: GetUserDAOTest (2)

	Tests	/Library
▶	✓	
◀	✗	
↓↑		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		
↑↓		
⌚		
↶ ↽		
⚙️		
▶	✓ Tests	
◀		
⌚		
☰		

HELP.md × pom.xml × .gitignore × GetUserDAOTest.java × UserRepository.java × UserEntity.java × User.java ×

```
77
78     @Test
79     void should_returnUser_when_found() {
80         //given
81         final Long existingUserId = 77777L;
82         final UserEntity foundUser = new UserEntity();
83         final User expectedUser = new User();
84
85         given(userRepository.findById(existingUserId)).willReturn(Optional.of(foundUser));
86         given(userDomainMapper.map(foundUser)).willReturn(expectedUser);
87
88         //when
89         final Optional<User> fetchedUser = getUserDAO.findById(existingUserId);
90
91         //then
92         assertEquals(fetchedUser.get(), expectedUser);
93     }
94 }
```

GetUserDAOTest > should\_returnUser\_when\_found()

Run: GetUserDAOTest.should\_returnUser\_when\_f...

Tests failed: 1 of 1 test – 429 ms

Test Results

Test	Time
GetUserDAOTest should_returnUser_when_found()	429 ms

/Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home

java.util.NoSuchElementException: No value present

at java.base/java.util.Optional.get(Optional.java:148)  
at es.geeksusma.tddexample.user.GetUserDAOTest.should\_returnUser\_when\_found(GetUserDAOTest.java:42)  
at java.base/java.util.ArrayList.forEach(ArrayList.java:1540)  
at java.base/java.util.ArrayList.forEach(ArrayList.java:1540)

```
MD HELP.md × pom.xml × .gitignore × GetUserDAOTest.java × UserDomainMapper.java × UserRepository.java × UserEntity.java × User.java ×
9 @GetUserDAO(final UserRepository userRepository, final UserDomainMapper userDomainMapper) {
10
11     this.userRepository = userRepository;
12     this.userDomainMapper = userDomainMapper;
13 }
14
15 Optional<User> getById(final Long id) {
16     if (id == null) {
17         throw new IllegalArgumentException("Id to fetch is mandatory");
18     }
19     return userRepository.findById(id).map( u -> userDomainMapper.map(u));
20 }
21 }
22
GetUserDAO > getById()
```

Run: **GetUserDAOTest**

Tests passed: 5 of 5 tests – 457 ms

▶	457 ms
✓	457 ms
⌚	/Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Ho
⌄	<default package>
⌄	✓ GetUserDAOTest
⌄	✓ should_returnUser_when_found()
⌄	✓ should_returnEmpty_when_notFound()
⌄	✓ should_throwIllegalUse_when_idNotGiven()
⌄	✓ should_fetchFromRepository_when_userIdIsValid()
⌄	✓ should_mapEntity_when_found()

# Questions:

- Can we refactor a little bit more?
- Can we remove a redundant test?
- Do you think we have done?

# Questions



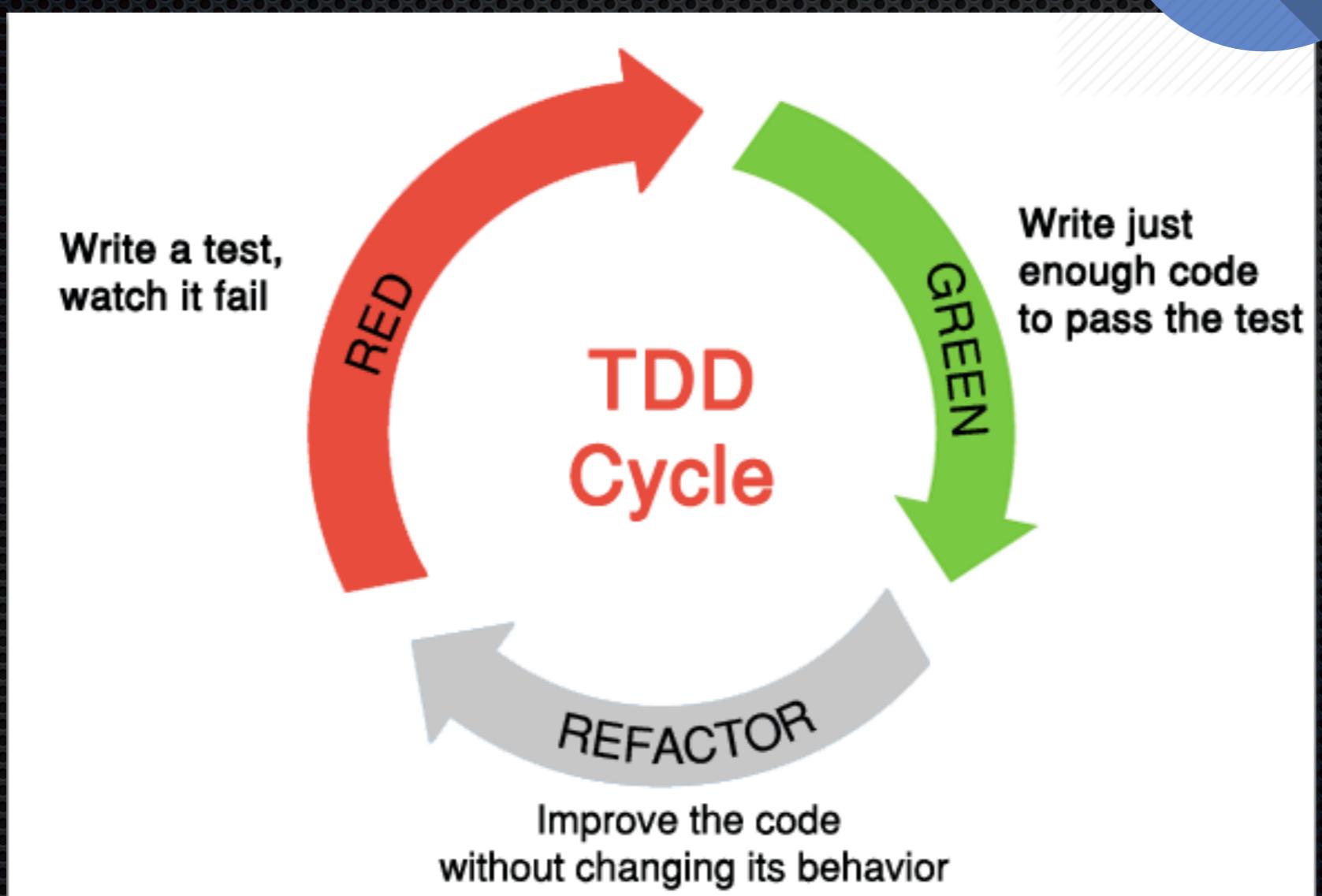
Some ideas:

- We could extract the logic of validating the input to an external Validator class to decouple the logic of validating the id
- Seems the next tests are redundant now even they were useful:
  - `should_fetchFromRepository_when_userIdIsValid`
  - `should_mapEntity_when_found`
  - `should_returnUser_when_found`

# Questions

- We're not done yet!:
  - UserRepository is not tested!
  - UserDomainMapper is not implemented!
  - Domain and Model Entity are not filled yet!
  - Both layers DAO and Repository are not tested!
- ... so we have some things to do, but we don't have to look back again to our DAO since it is fully tested at coding time!

# .... Questions?



# Resources

<https://github.com/geeksusma/tdd-example>

<https://github.com/geeksusma/disco-library-be>

<https://www.youtube.com/watch?v=MkpvaPcYvqk> (in Spanish)

Clean Code Chapter 9 (Robert. C. Martin)

TDD book by Kent Beck