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?

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 thing tests don’t help to deliver or coding faster!

Remember, What is a good Test?

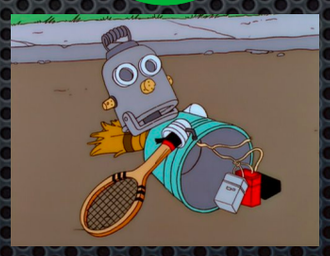
- F.I.R.ST principles:
  - Fast

  - 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

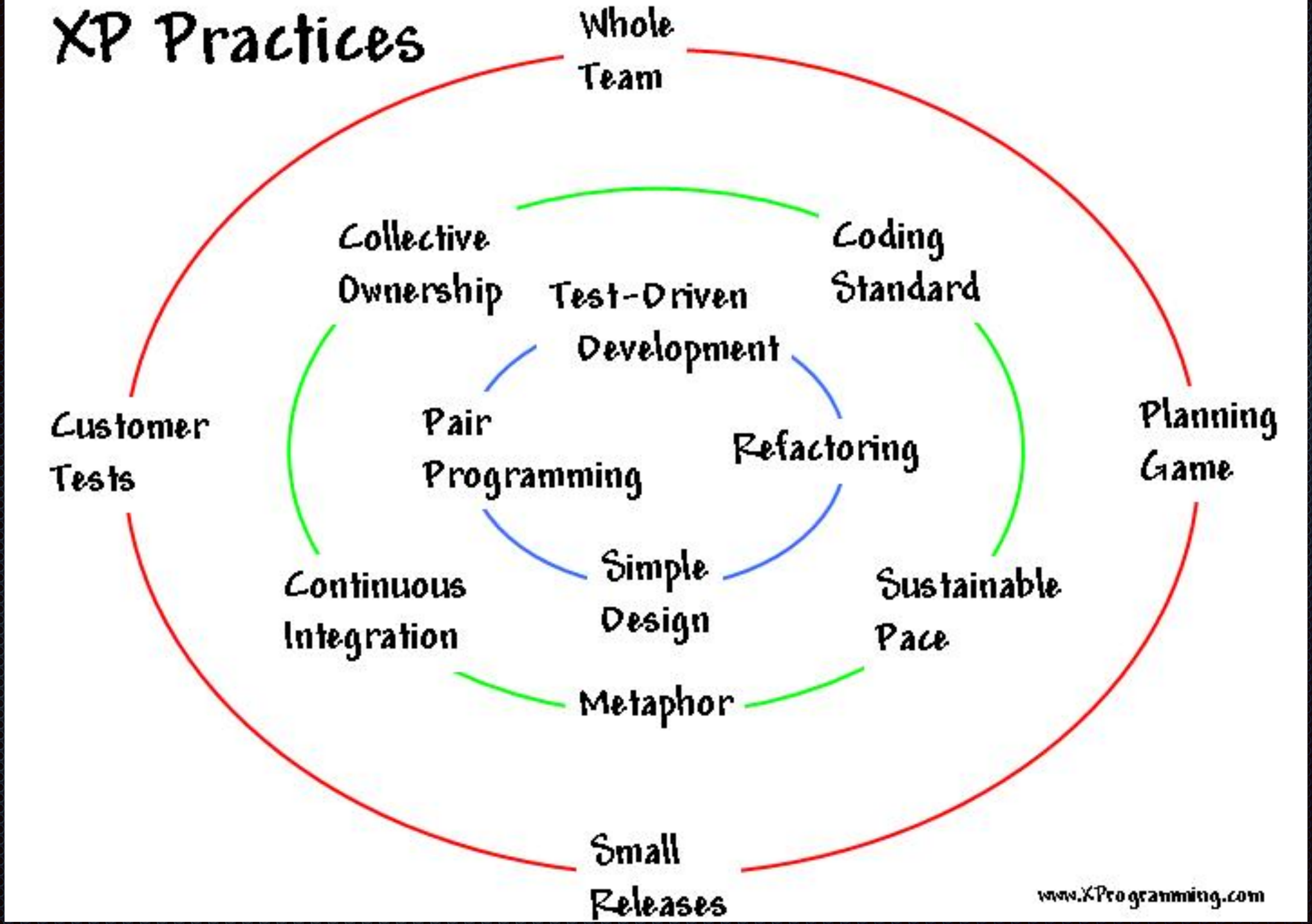
Why TDD?

- 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 :-)

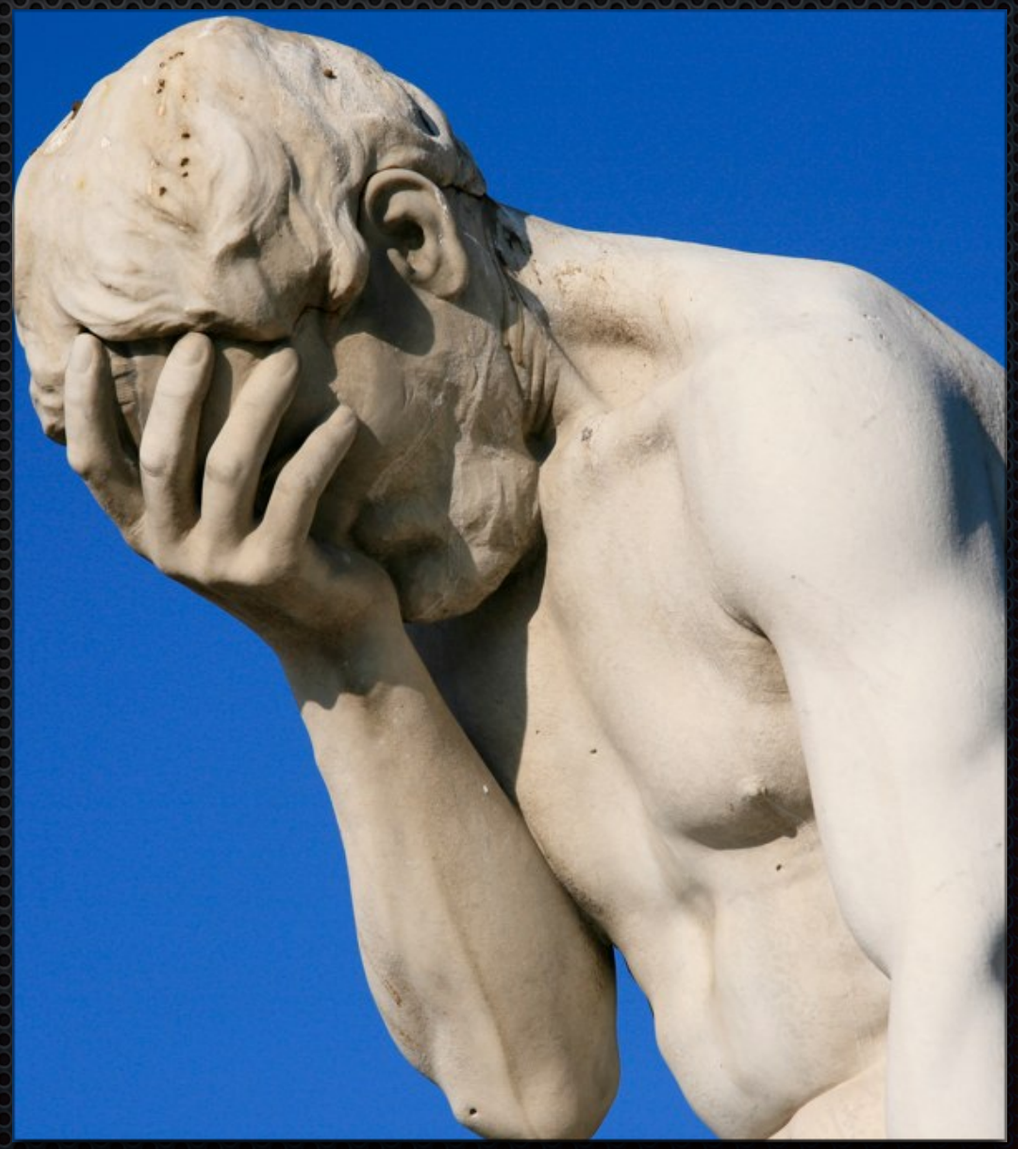
XP Practices



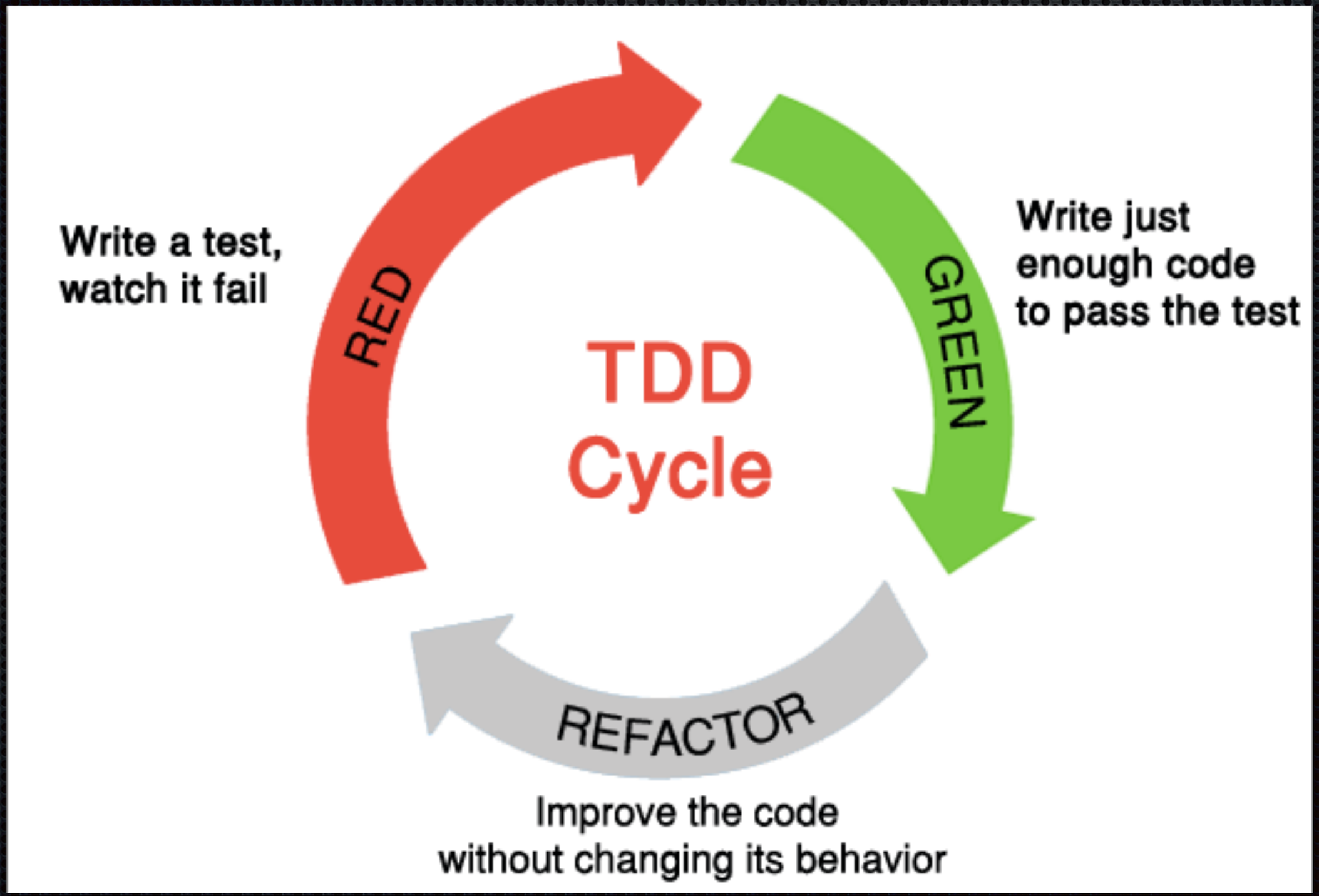
eXtreme Programming Practices

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 you 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

Project

- tdd-example ~/wata/tdd-example
  - .idea
  - .mvn
  - src
    - main
    - test
      - java
        - es.geeksusma.tddexample.user
          - GetUserDAOTest
  - .gitignore
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml
  - tdd-example.iml
- External Libraries
- Scratches and Consoles

HELP.md | pom.xml | .gitignore | GetUserDAOTest.java

```java
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.byId(null));

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

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

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

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

 GetUserDAOTest {

rivate GetUserDAO getUserDAO;

BeforeEach
oid setUp() {
   getUserDAO

Test
oid should_th

   //when
   final Throwable raisedException = catchThrowable(() -

   //then
   assertThat(raisedException).isInstanceOf(IllegalArgum
           .hasMessageContaining("Id to fetch is mandato
```

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

```
e ~/wata/tdd-example

es.geeksusma.tddexample.user
GetUserDAOTest

nd
nple.iml
raries
nd Consoles
```

HELP.md | pom.xml | .gitignore | GetUserDAOTest.java

```
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
23
24         //then
25         assertThat(raisedException).isInstanceOf(IllegalArgumentException
26                 .hasMessageContaining("Id to fetch is mandatory");
27     }
28 }
```

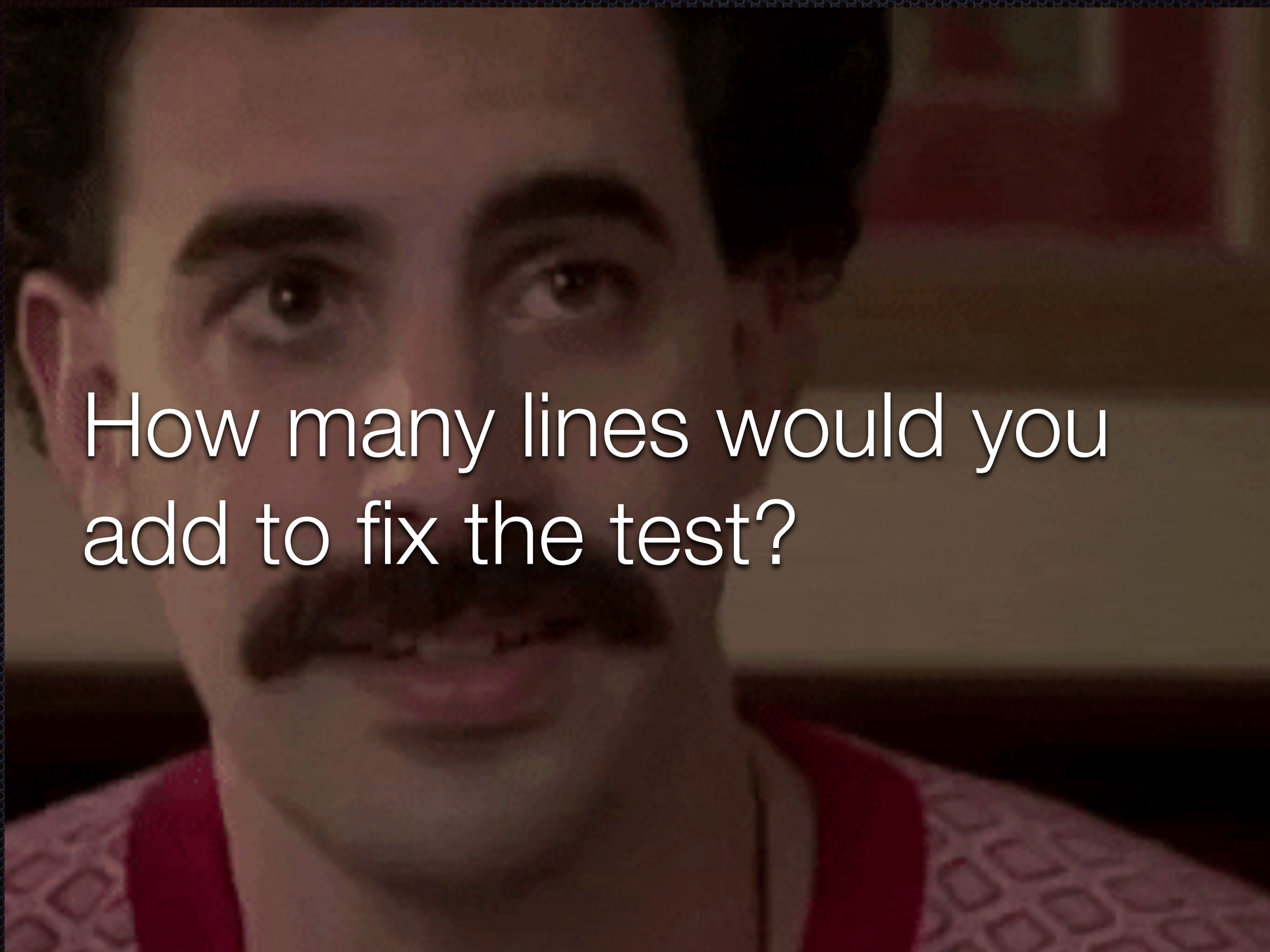
Tests failed: 1 of 1 test – 94 ms

Test Results	94 ms
GetUserDAOTest	94 ms
should_throwIllegalUse_when_idNotGiven()	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?

HELP.md | pom.xml | .gitignore | GetUserDAOTest.java | GetUserDAO.java

1: Project

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

GetUserDAO › byId()

Run: Rerun Failed Tests

- user (es.geeksusma.tddexample) — 69 ms
  - GetUserDAOTest — 69 ms
    - should_throwIllegalUse_when_idNotGiven() — 69 ms

```java
@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
```

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

import java.util.Optional;

class GetUserDAO {
    Optional<User> byId(final Long id) {
        if (id == null) {
            throw new IllegalArgumentException("Id to fetch is mandatory");
        }
        return Optional.empty();
    }
}

```

GetUserDAOTest › should_returnEmpty_when_notFound()

Run: GetUserDAOTest

Tests passed: 2 of 2 tests – 84 ms

Test Results	84 ms
GetUserDAOTest	84 ms
should_returnEmpty_when_notFound()	79 ms
should_throwIllegalUse_when_idNotGiven()	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          getUserDAO.byId(userId);
55
56          //then
57          then(userRepository).should().findById();
58      }
```

GetUserDAOTest › setUp()

Run: Rerun Failed Tests

Tests failed: 1 of 1 test – 459 ms

- user (es.geeksusma.tddexample) 459 ms
  - GetUserDAOTest 459 ms
    - should_fetchFromRepository_when_userIdIsValid() 459 ms

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



Wanted but not invoked:
userRepository.findById();
-> at es.geeksusma.tddexample.user.GetUserDAOTest.should_fetchFromRepository_when_userIdIsValid(GetUserDAOTest.java:
Actually, there were zero interactions with this mock.

Wanted but not invoked:
userRepository.findById();
-> at es.geeksusma.tddexample.user.GetUserDAOTest.should_fetchFromRepository_when_userIdIsValid(GetUserDAOTest.java:
Actually, there were zero interactions with this mock.

    at es.geeksusma.tddexample.user.GetUserDAOTest.should_fetchFromRepository_when_userIdIsValid(GetUserDAOTest.java
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1540) <9 internal calls>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1540) <21 internal calls>
```

```java
class GetUserDAO {
    private final UserRepository userRepository;

    GetUserDAO(final UserRepository userRepository) {

        this.userRepository = userRepository;
    }

    Optional<User> byId(final Long id) {
        if (id == null) {
            throw new IllegalArgumentException("Id to fetch is mandatory");
        }
        userRepository.findById();
        return Optional.empty();
    }
```

GetUserDAO › byId()

Run: GetUserDAOTest (1)

Tests passed: 3 of 3 tests – 440 ms

<default package>	440 ms
GetUserDAOTest	440 ms
should_returnEmpty_when_notFound()	426 ms
should_throwIllegalUse_when_idNotGiven()	5 ms
should_fetchFromRepository_when_userIdIsValid()	9 ms

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



Process finished with exit code 0
```

```
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.byId(existingUserId);
72
73          //then
74          then(userDomainMapper).should().map(foundUser);
75      }
76  }
77
```

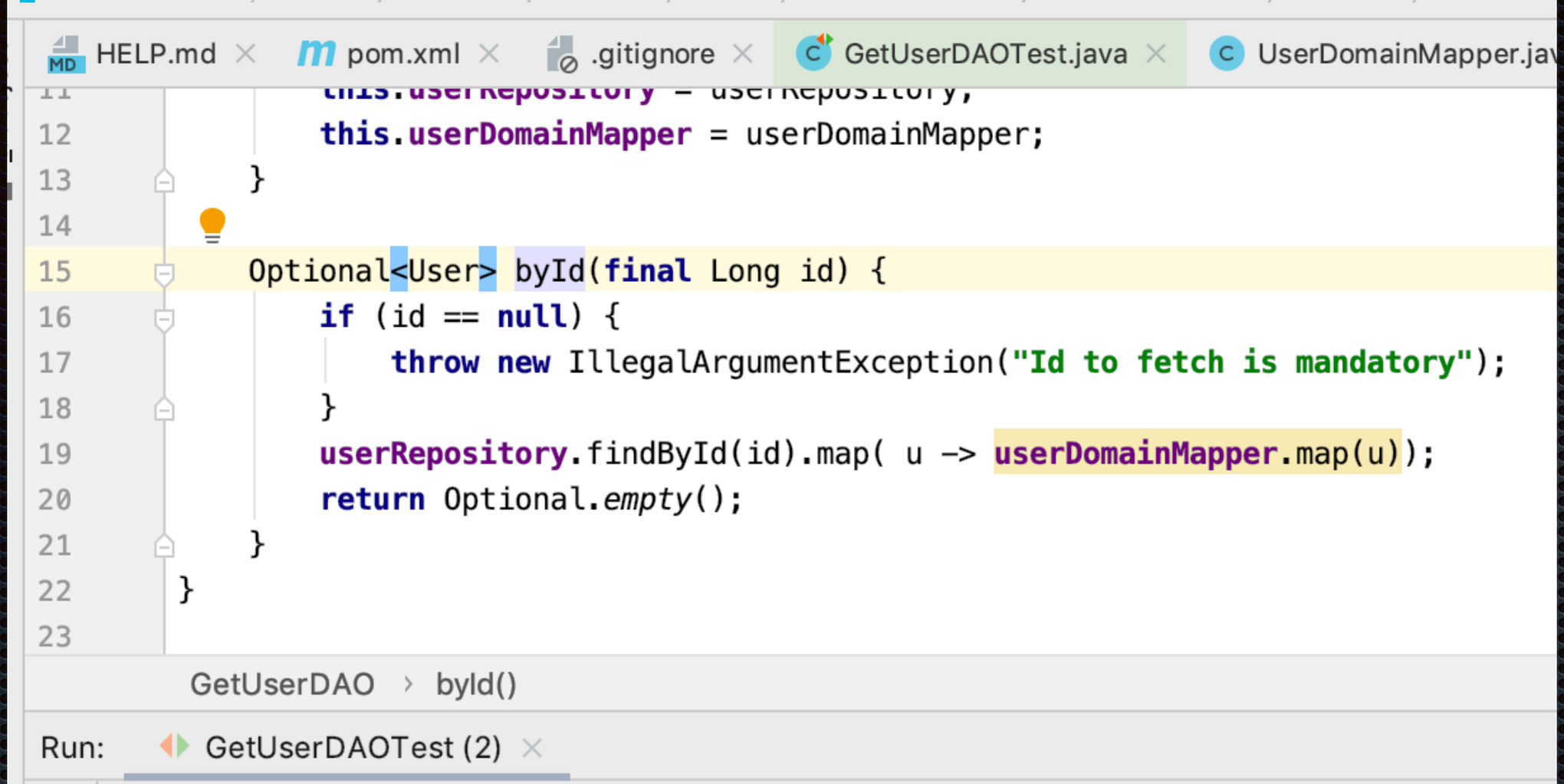
GetUserDAOTest › should_mapEntity_when_found()

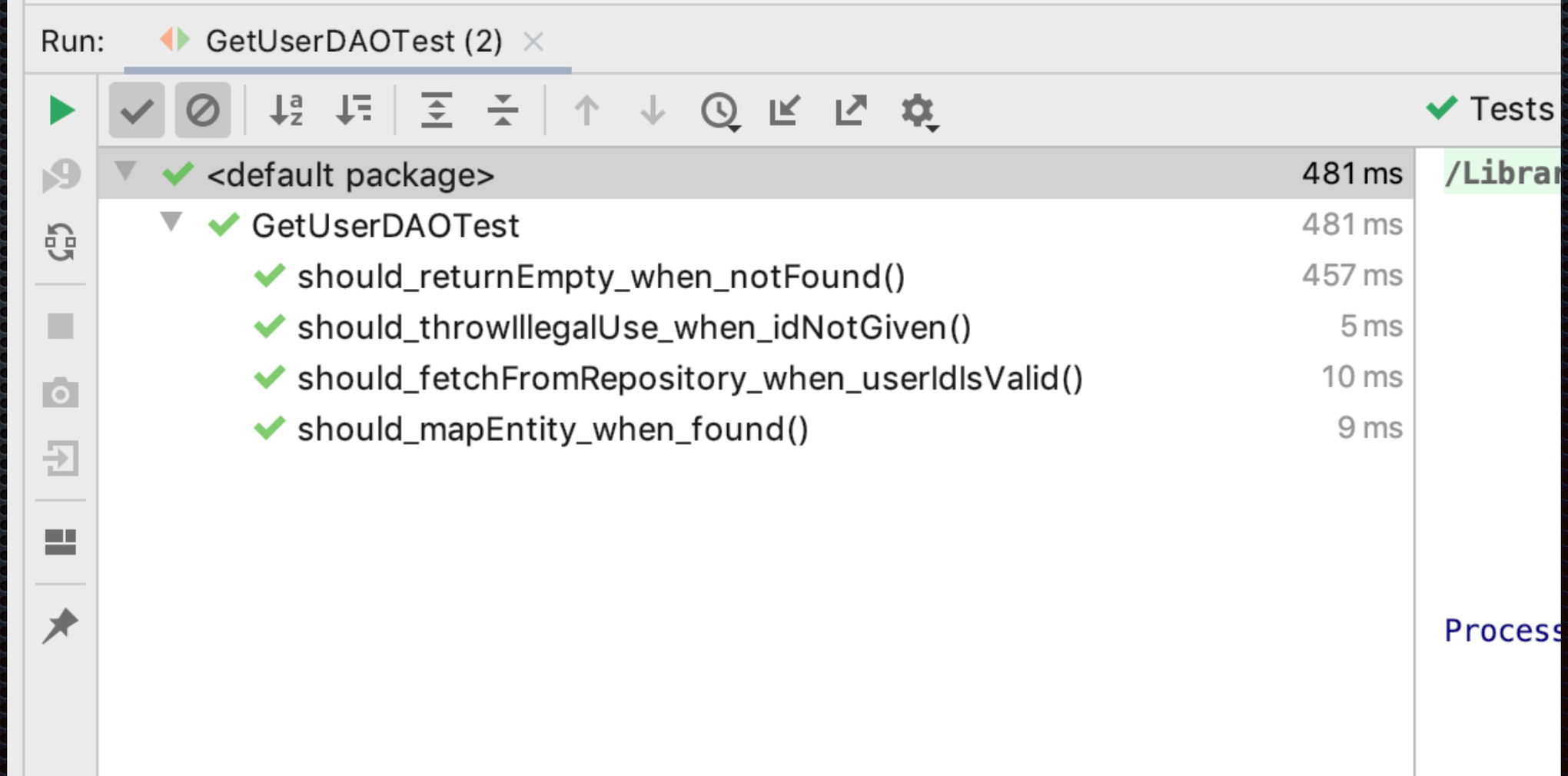
Run: GetUserDAOTest.should_mapEntity_when_f...

Tests failed: 1 of 1 test – 441 ms

Test Results	441 ms
GetUserDAOTest	441 ms
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()
```


HELP.md
pom.xml
.gitignore
GetUserDAOTest.java
UserDomainMapper.jav
this.userRepository = userRepository;
this.userDomainMapper = userDomainMapper;
}
Optional<User> byId(final Long id) {
if (id == null) {
throw new IllegalArgumentException("Id to fetch is mandatory");
}
userRepository.findById(id).map(u -> userDomainMapper.map(u));
return Optional.empty();
}
}
GetUserDAO › byId()


Run:
GetUserDAOTest (2)
Tests
<default package> 481 ms
GetUserDAOTest 481 ms
should_returnEmpty_when_notFound() 457 ms
should_throwIllegalUse_when_idNotGiven() 5 ms
should_fetchFromRepository_when_userIdIsValid() 10 ms
should_mapEntity_when_found() 9 ms

HELP.md | pom.xml | .gitignore | GetUserDAOTest.java | UserDomainMapper.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.byId(existingUserId);
90
91          //then
92          assertThat(fetchedUser.get()).isEqualTo(expectedUser);
93      }
94  }
```

GetUserDAOTest › should_returnUser_when_found()

Run: GetUserDAOTest.should_returnUser_when_f...

Tests failed: 1 of 1 test – 429 ms

Test Results	429 ms
GetUserDAOTest	429 ms
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_returnUse
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1540)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1540)
```

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> byId(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 }
```

GetUserDAO › byId()

Run: GetUserDAOTest

Tests passed: 5 of 5 tests – 457 ms

<default package>	457 ms
GetUserDAOTest	457 ms
should_returnUser_when_found()	437 ms
should_returnEmpty_when_notFound()	4 ms
should_throwIllegalUse_when_idNotGiven()	5 ms
should_fetchFromRepository_when_userIdIsValid()	8 ms
should_mapEntity_when_found()	3 ms

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

Questions:

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

Questions

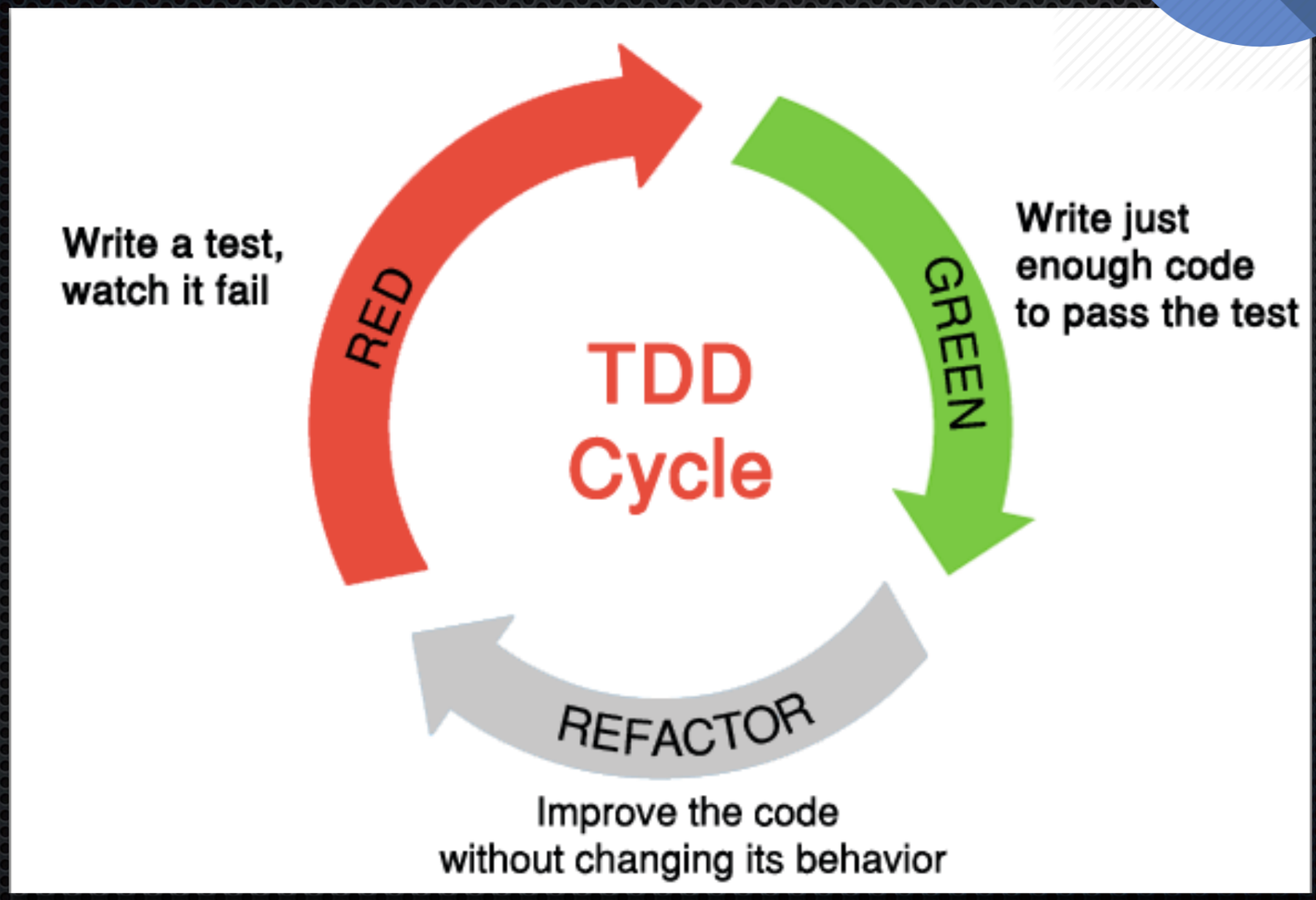
Some ideas:

- 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?


Write a test,
watch it fail
RED
TDD
Cycle
GREEN
Write just
enough code
to pass the test
REFACTOR
Improve the code
without changing its behavior

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