



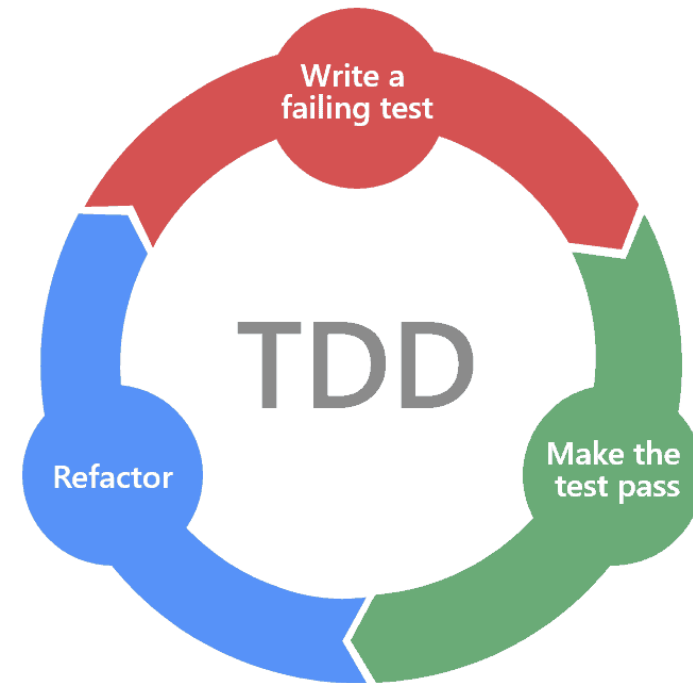
Back-End Development

Test Driven Development

G. Jongen, J. Pieck, E. Steegmans, B. Van Impe

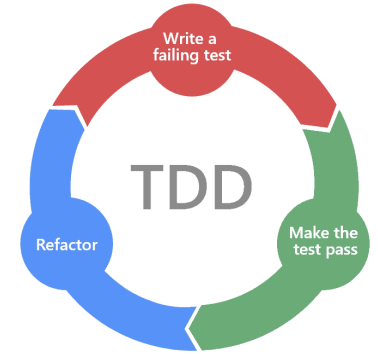
# Test Driven Development (TDD)

- is a software development process relying on software requirements being converted to test cases before software is fully developed
  1. Write a failing test
  2. Make the test pass
  3. Refactor



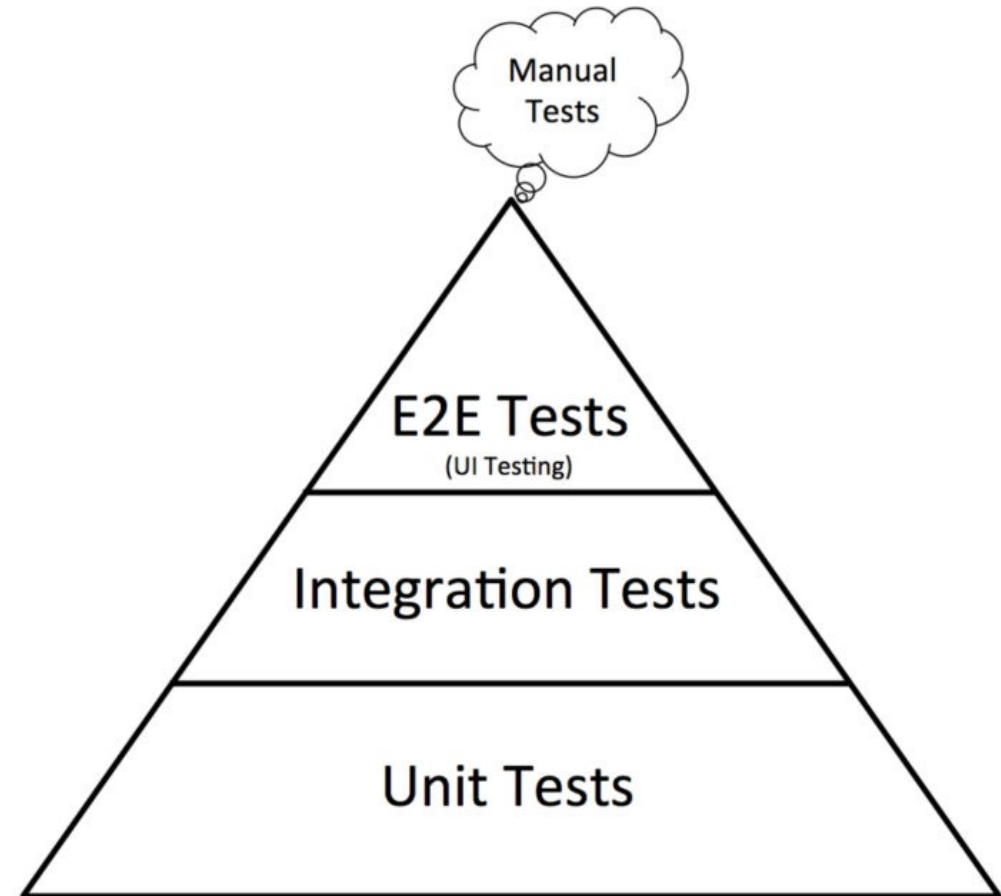
# TDD@UCLL

- PHASE 1
  - Students are given the test class and test methods
  - Students are writing code to make the test class and test methods pass

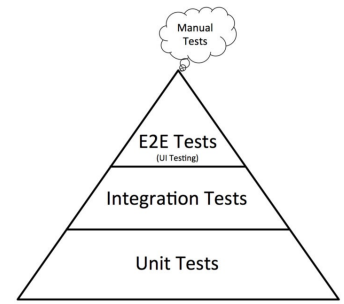


# Testing Pyramid

- Unit tests
- Integration tests
- E2E tests
- Manual tests



# Unit Testing



- is the process of testing small isolated portions of a software application called units
- you only focus on that little part or unit in your unit test
  - unit is e.g. a class, a method, a happy case, an unhappy case, ...
  - should be very fast
  - should only test 1 functionality/scenario

# Unit test

- Test method
  - Tests a method of a class
    - Happy or unhappy case
  - 3 parts
    - Given: the context or input
    - When: the action (or method under test)
    - Then: the expected outcome
  - Name of test method should be human readable
    - given...\_when...\_then...

# JUnit 5 for Java

- pom.xml

```
<dependency>  
  <groupId>org.junit.jupiter</groupId>  
  <artifactId>junit-jupiter-engine</artifactId>  
  <version>5.6.2</version>  
  <scope>test</scope>  
</dependency>
```

# STEP 0 – Read the test method

## Constructor – Happy case

```
//given
private String validNameElke = "Elke";
private int validAgeElke = 44;

//constructor
//happy case
@Test
void givenValidValues_whenCreatingUser_thenUserIsCreatedWithTheseValues() {
    //when
    User elke = new User(validNameElke, validAgeElke);

    //then
    assertNotNull(elke);
    assertEquals(validNameElke, elke.getName());
    assertEquals(validAgeElke, elke.getAge());
    assertEquals(0, elke.countYearsOfMembership());
}
```



# @Test

- Indicates that it is a test method
- It is a method that can be executed
  - It is like a little main method that you can run/execute ...

```
@Test  
void givenValidValues_whenCreatingUser_thenUserIsCreatedWithTheseValues() {
```

# assert methods

- Methods to test the state of objects
  - boolean assertEquals(expected, actual)
    - assertEquals(44, elke.getAge())
      - true if the instance variable age of the elke object has the value 44, false otherwise
  - boolean assertNotNull(object)
    - assertNotNull(elke)
      - true if the object with name elke is created with the default values in the instance variables, false otherwise

```
assertNotNull(elke);  
assertEquals(validAgeElke, elke.getAge());
```

# STEP 1 – RUN THE TESTMETHOD

The screenshot shows an IDE with a test failure. The test method `givenValidValues_whenCreatingUser_thenUserIsCreatedWithTheseValues()` is highlighted. The failure message is `Expected [Elke] but was [null]`. The test run history on the right shows several failed and passed tests.

```
14 //constructor
15 //happy case
16 @Test
17 void givenValidValues_whenCreatingUser_thenUserIsCreatedWithTheseValues() {
18     //when
19     User elke = new User(validNameElke, validAgeElke);
20
21     //then
22     assertNotNull(elke);
23     assertEquals(validNameElke, elke.getName());
```

Expected [Elke] but was [null] givenValidValues\_whenCreatingUser\_thenUserIsCreatedWithTheseValues()

Expected	Actual
-Elke	+null

Test run at 2/19/2023, 5:46:42 AM  
givenValidValues\_whenCreatingUser\_thenUserIsCreatedWithTheseValues() Expected [Elke] but was [null]  
org.opentest4j.AssertionFailedError: expected: [Elke] but was: [null]

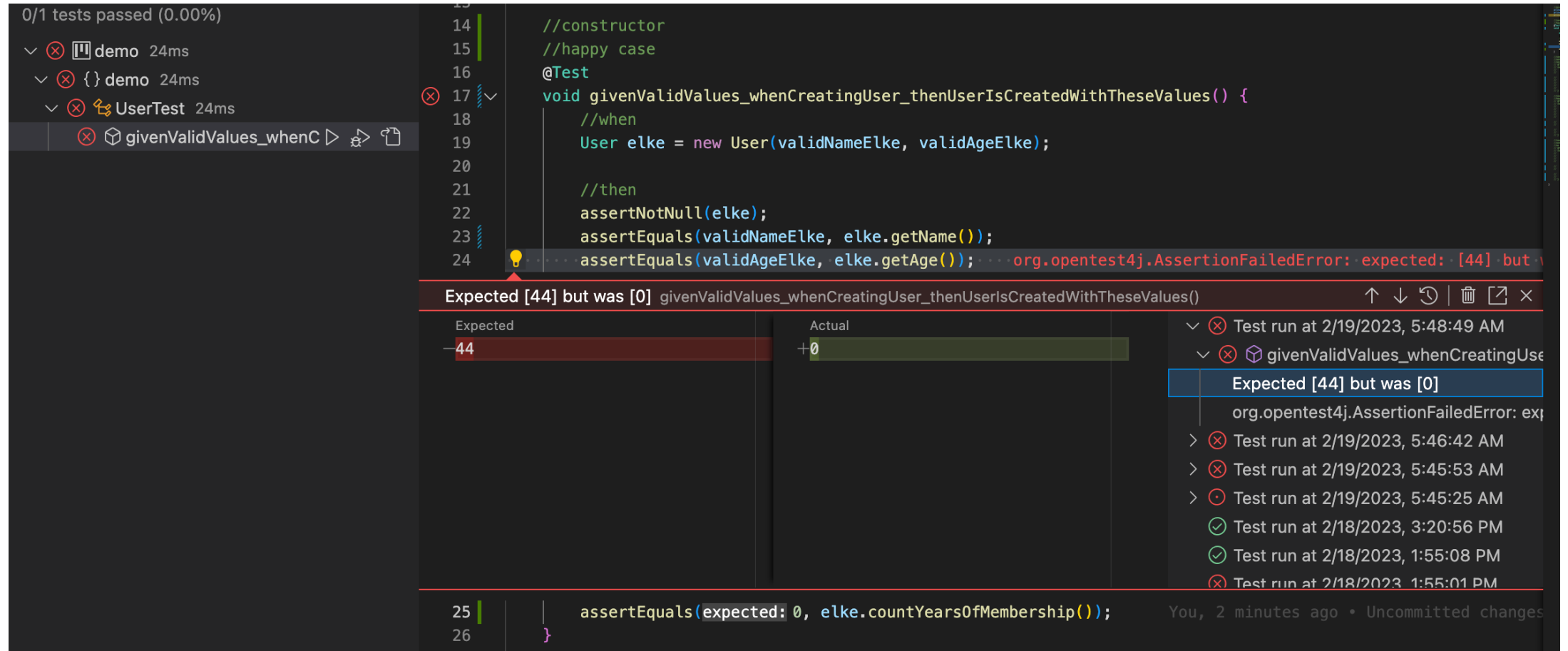
Test run at 2/19/2023, 5:45:53 AM  
Test run at 2/19/2023, 5:45:25 AM  
Test run at 2/18/2023, 3:20:56 PM  
Test run at 2/18/2023, 1:55:08 PM  
Test run at 2/18/2023, 1:55:01 PM  
Test run at 2/18/2023 11:26:27 AM

```
24 assertEquals(validAgeElke, elke.getAge());
25 assertEquals(expected: 0, elke.countYearsOfMembership());
26 }
```

# STEP 2 – WRITE CODE FOR THE METHOD UNDER TEST

```
public class User {  
  
    private String name;  
    private int age;  
    private List<Integer> membershipYears;  
  
    public User(String name, int age) {  
        this.name = name;  
    }  
}
```

# STEP 1 – RE-RUN THE TESTMETHOD



0/1 tests passed (0.00%)

- demo 24ms
- { } demo 24ms
- UserTest 24ms
- givenValidValues\_whenC

```
//constructor
//happy case
@Test
void givenValidValues_whenCreatingUser_thenUserIsCreatedWithTheseValues() {
    //when
    User elke = new User(validNameElke, validAgeElke);

    //then
    assertNotNull(elke);
    assertEquals(validNameElke, elke.getName());
    assertEquals(validAgeElke, elke.getAge());
}
```

Expected [44] but was [0] givenValidValues\_whenCreatingUser\_thenUserIsCreatedWithTheseValues()

Expected	Actual
-44	+0

- Test run at 2/19/2023, 5:48:49 AM
- givenValidValues\_whenCreatingUser
- Expected [44] but was [0]
- org.opentest4j.AssertionFailedError: exp
- Test run at 2/19/2023, 5:46:42 AM
- Test run at 2/19/2023, 5:45:53 AM
- Test run at 2/19/2023, 5:45:25 AM
- Test run at 2/18/2023, 3:20:56 PM
- Test run at 2/18/2023, 1:55:08 PM
- Test run at 2/18/2023 1:55:01 PM

```
assertEquals(expected: 0, elke.countYearsOfMembership());
}
```

You, 2 minutes ago • Uncommitted changes

# STEP 2 – WRITE CODE FOR THE METHOD UNDER TEST UNTILL TEST PASSES

```
public class User {  
  
    private String name;  
    private int age;  
    private List<Integer> membershipYears;  
  
    public User(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
}
```

# STEP 1 – RE-RUN THE TESTMETHOD

The screenshot shows an IDE interface with a test failure. On the left, a sidebar shows a project tree with 'demo' (32ms), 'UserTest' (32ms), and 'givenValidValues\_whenC' (32ms). The main editor displays a Java test method:

```
15 //happy case
16 @Test
17 void givenValidValues_whenCreatingUser_thenUserIsCreatedWithTheseValues() {
18     //when
19     User elke = new User(validNameElke, validAgeElke);
20
21     //then
22     assertNotNull(elke);
23     assertEquals(validNameElke, elke.getName());
24     assertEquals(validAgeElke, elke.getAge());
25     assertEquals(expected: 0, elke.countYearsOfMembership());
```

A red error bar at the bottom of the code editor indicates a `java.lang.NullPointerException: Cannot invoke "java.util.List.size()" because "this.membershipY...`. Below the code editor, a detailed error message is shown:

```
java.lang.NullPointerException: Cannot invoke "java.util.List.size()" because "this.member:
at demo.User.countYearsOfMembership(User.java:28)
at demo.UserTest.givenValidValues_whenCreatingUser_thenUserIsCreatedWithTheseValu
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

On the right side, a list of test runs is displayed:

- Test run at 2/19/2023, 5:52:35 AM
- givenValidValues\_whenCreatingUse
- java.lang.NullPointerException: Canno
- Test run at 2/19/2023, 5:52:28 AM
- Test run at 2/19/2023, 5:52:14 AM
- Test run at 2/19/2023, 5:51:57 AM
- Test run at 2/19/2023, 5:48:49 AM
- Test run at 2/19/2023, 5:46:42 AM
- Test run at 2/19/2023, 5:45:53 AM
- Test run at 2/19/2023, 5:45:25 AM
- Test run at 2/18/2023, 3:20:56 PM
- Test run at 2/18/2023, 1:55:08 PM

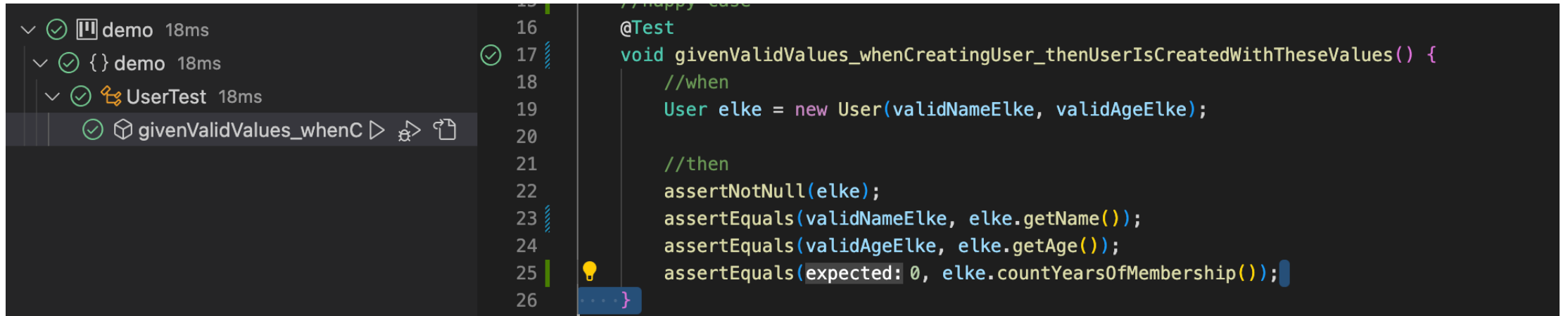
At the bottom of the IDE, the status bar shows '26' and 'You, 2 weeks ago • Initial commit'.

# STEP 2 – WRITE CODE FOR THE METHOD UNDER TEST UNTILL TEST PASSES

```
public class User {  
  
    private String name;  
    private int age;  
    private List<Integer> membershipYears = new ArrayList<Integer>();  
  
    public User(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
}
```



# STEP 1 – RE-RUN THE TESTMETHOD



The screenshot shows an IDE interface. On the left, a test runner panel displays a list of tests: 'demo' (18ms), '{ } demo' (18ms), 'UserTest' (18ms), and 'givenValidValues\_whenC'. The 'givenValidValues\_whenC' test is selected and highlighted. To the right, the source code for the test is visible. It is a JUnit-style test method named 'givenValidValues\_whenCreatingUser\_thenUserIsCreatedWithTheseValues()' with a line number from 16 to 26. The code includes a 'User' object creation and several assertions. A lightbulb icon is present next to line 25, indicating a suggestion or warning.

```
16 //happy case
17 @Test
18 void givenValidValues_whenCreatingUser_thenUserIsCreatedWithTheseValues() {
19     //when
20     User elke = new User(validNameElke, validAgeElke);
21
22     //then
23     assertNotNull(elke);
24     assertEquals(validNameElke, elke.getName());
25     assertEquals(validAgeElke, elke.getAge());
26     assertEquals(expected: 0, elke.countYearsOfMembership());
27 }
```

# STEP 0 – Read the test method

## Constructor – Unhappy case

```
//given
private String validNameElke = "Elke";

//constructor
//unhappy case
//invalid negative age
@Test
void givenInvalidNegativeAge_whenCreatingUser_thenUserIsCreatedWithAge0() {
    //when
    User elke = new User(validNameElke, -5);

    //then
    assertNotNull(elke);
    assertEquals(validNameElke, elke.getName());
    assertEquals(0, elke.getAge());
    assertEquals(0, elke.countYearsOfMembership());
}
```

# STEP 1 – RUN THE TESTMETHOD

```
29 //invalid negative age
30 @Test
31 void givenInvalidNegativeAge_whenCreatingUser_thenUserIsCreatedWithAge0() {
32     //when
33     User elke = new User(validNameElke, -5);
34
35     //then
36     assertNotNull(elke);
37     assertEquals(validNameElke, elke.getName());
38     assertEquals(expected: 0, elke.getAge());
```

Expected [0] but was [-5] givenInvalidNegativeAge\_whenCreatingUser\_thenUserIsCreatedWithAge0()

Expected	Actual
-0	+ -5

Test run at 2/19/2023, 5:59:25 AM

givenInvalidNegativeAge\_whenCrea

Expected [0] but was [-5]

org.opentest4j.AssertionFailedError: exp

Test run at 2/19/2023, 5:56:25 AM

Test run at 2/19/2023, 5:52:35 AM

Test run at 2/19/2023, 5:52:28 AM

Test run at 2/19/2023, 5:52:14 AM

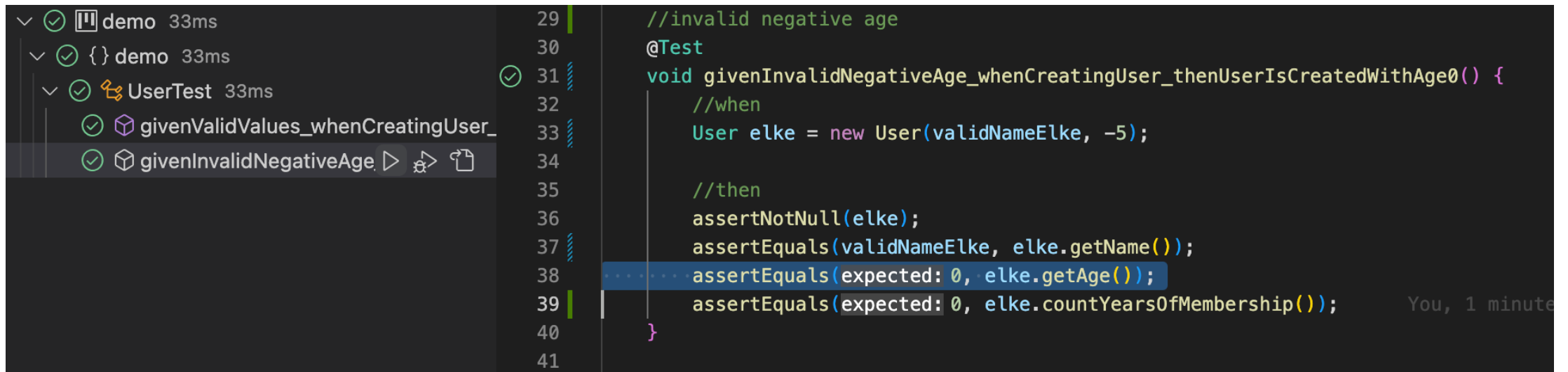
Test run at 2/19/2023, 5:51:57 AM

Test run at 2/19/2023, 5:48:49 AM

# STEP 2 – WRITE CODE FOR THE METHOD UNDER TEST

```
public User(String name, int age) {  
    this.name = name;  
    if (age >= 0)  
        this.age = age;  
}
```

# STEP 3 – RE-RUN THE TESTMETHOD



The screenshot shows an IDE interface. On the left, a test runner panel displays a tree of test results. The 'demo' test is expanded, showing 'UserTest' with two sub-tests: 'givenValidValues\_whenCreatingUser\_' and 'givenInvalidNegativeAge'. The 'givenInvalidNegativeAge' test is highlighted with a play button icon, indicating it is the current focus. On the right, the source code for the 'givenInvalidNegativeAge' test method is displayed. The code is in Java and uses JUnit 5 annotations. The method is named 'givenInvalidNegativeAge\_whenCreatingUser\_thenUserIsCreatedWithAge0()'. It creates a 'User' object with a valid name and a negative age of -5. It then asserts that the user is not null, that the name is correct, and that the age is 0. The line 'assertEquals(expected: 0, elke.getAge());' is highlighted with a blue selection bar. The code is as follows:

```
29 //invalid negative age
30
31 @Test
32 void givenInvalidNegativeAge_whenCreatingUser_thenUserIsCreatedWithAge0() {
33     //when
34     User elke = new User(validNameElke, -5);
35
36     //then
37     assertNotNull(elke);
38     assertEquals(validNameElke, elke.getName());
39     assertEquals(expected: 0, elke.getAge());
40     assertEquals(expected: 0, elke.countYearsOfMembership());
41 }
```

On the far right, a small text snippet 'You, 1 minute' is visible.

# IMPORTANT

- Always re-run your green coloured test methods
  - Because you could have broken other parts of your code at some point

# Unit test

- Test class
  - Tests all methods of that class
    - All happy and all unhappy cases are tested in separate test methods

# References

- <https://marsner.com/blog/why-test-driven-development-tdd/>