
# Automation Testing

# Writing Tests

# A Day in the Life of an Automation Test Engineer

Marcelo now understands the purpose of using of software testing and unit testing with Junit.

Now, he has to understand how to write tests with accurate procedures and steps, recognize how  test structure works, what is lifecycle to write test in Junit and what are the test hierarchy and different types of levels of testing lie under the scenario.

This lesson will help Marcelo to perform the tests to complete his exertion.

# Learning Objectives

By the end of this lesson, you will be able to:

- Analyze how to write tests in Junit.
- Describe what is a test structure and how it process.
- Recognize lifecycle method of writing the tests.
- Classify the test hierarchies and its levels.

# Introduction

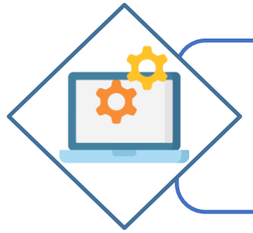
# Introduction to Write Tests

A unit test in Java gives the developer the opportunity to test an object. The most common way to write unit tests in Java is to use JUnit, a free, open sources unit-testing Java application.

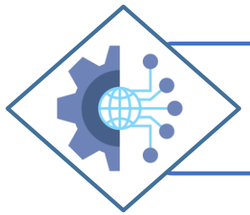
# Introduction to Write Tests

We have some following steps to how to write test in JUnit:

First, users need to download JUnit 5.8.1. In users browser, go to this page on "https://junit.org" and click junit.jar. Another web page will be displayed.



Click the jar option for Junit 5.8.1. This action will start the download of junit-5.8.1.jar to users computer.



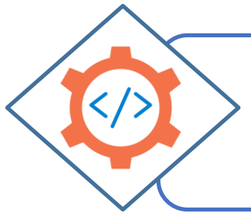
Now user will write the program that contains the JUnit tests.

# Introduction to Write Tests

Save users file as WriteAUnitTest.java.

Copy junit-5.8.2.jar to the directory that contains users Java program.

Open a command prompt and navigate to the directory containing user's Java program.

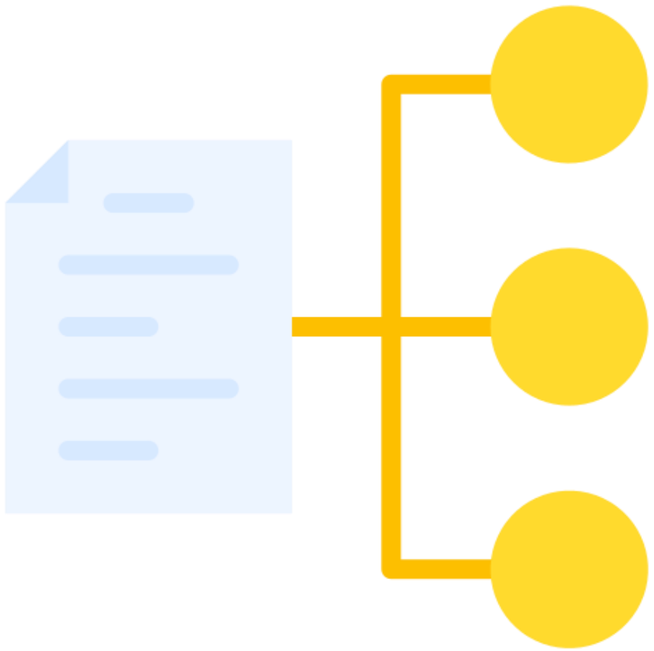


Type in the command to run the JUnit test runner using user's program and hit Enter.

# Test Structure

# Test Structure

Structuring your tests in an understandable way is not an easy task. JUnit 5 helps you create a sensible test structure by grouping related tests together.



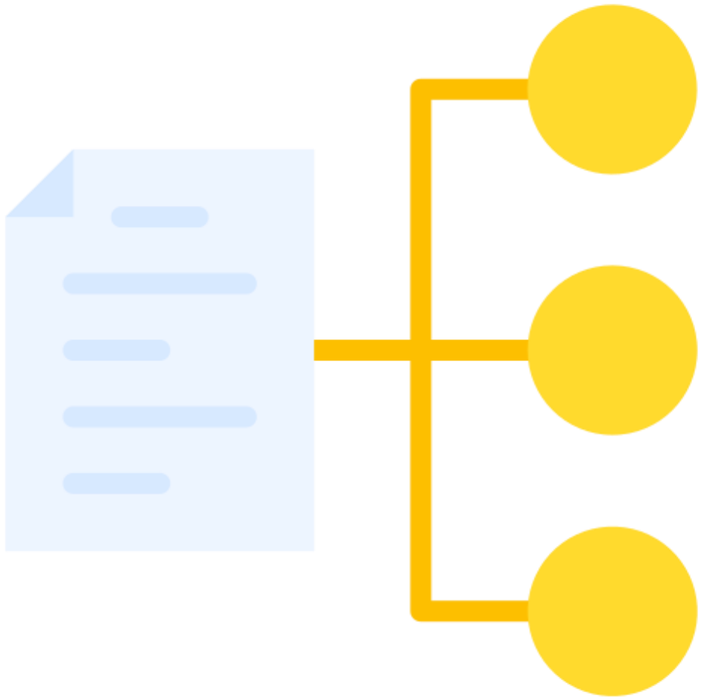
It is a value object.

Wraps a String value.

Value can’t be NULL, empty, or otherwise blank.

To demonstrate a well structure test we have some points which are mentioned above :

# Test Structure



Values are automatically trimmed..

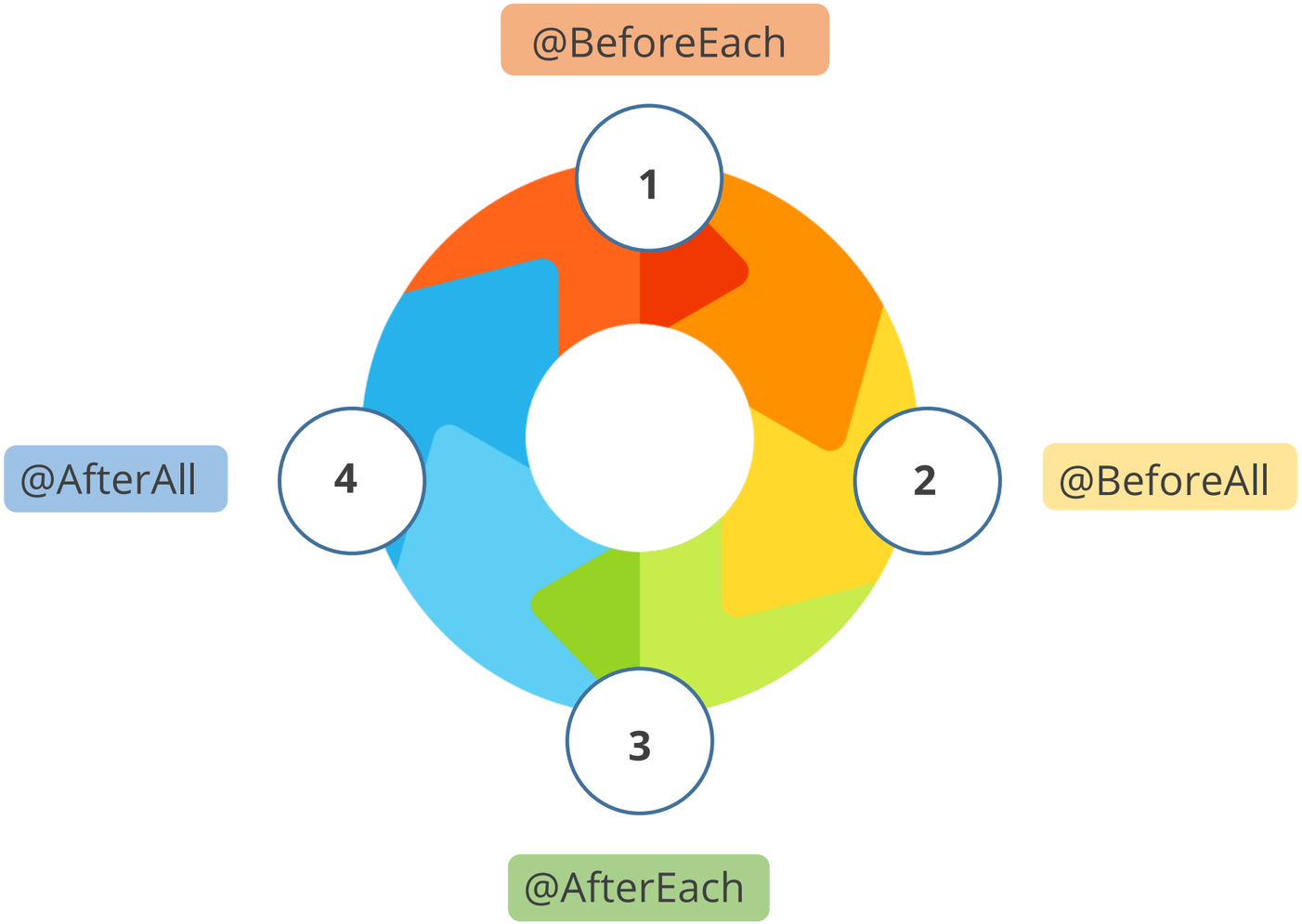
Equality is based on the wrapped value.

Instances can only be created using the factory method.

# Lifecycle Methods

# Lifecycle Methods

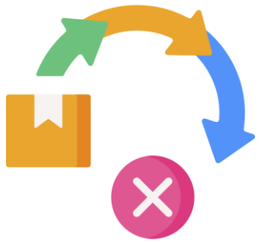
In JUnit 5, the test lifecycle is driven by four primary annotations which are as follows:



# Lifecycle Methods

Along with it, each test method must be marked with @Test annotation from package org.junit.Jupiter.

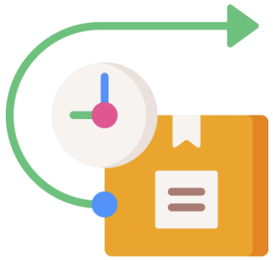
**@BeforeEach-** is used to signal that the annotated method should be executed before each @Test method in the current test class.



**@BeforeAll-** is used to signal that the annotated method should be executed before all the @Test, @RepeatedTest, @ParameterizedTest, and @TestFactory methods in the current class.

# Lifecycle Methods

**@AfterEach-** is used to signal that the annotated method should be executed after each @Test method in the current test class.



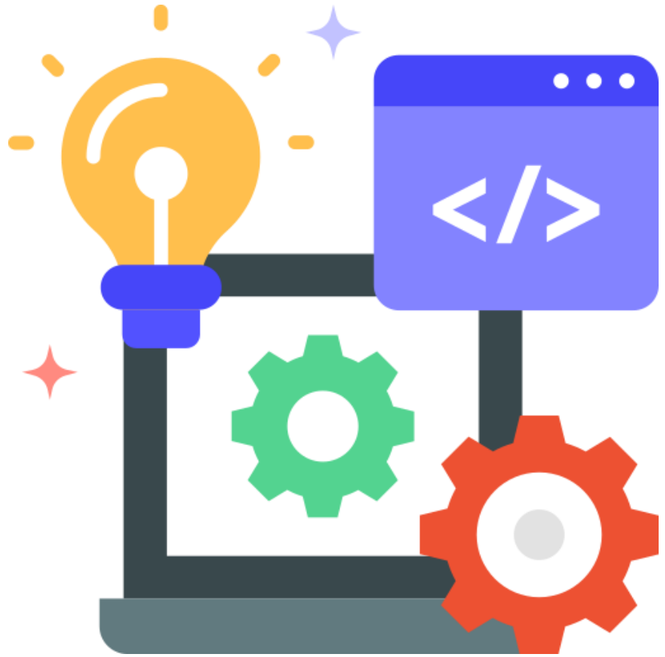
**@AfterAll -** annotation is used to signal that the annotated method should be to executed after all tests in the current test class.

# Test Hierarchies

# Test Hierarchies

Testing is typically used to discover errors left over from earlier phases, as well as problems created during the coding process itself. As a result, several levels of testing are utilized in the testing process, with each level aiming to examine different components of the system.

- Unit testing
- Integration testing



- System testing
- Acceptance testing

# Test Hierarchies

**Unit testing:** The first level of testing is called unit testing. Unit testing is essentially for verification of the code produced by individual programmers, and is typically done by the programmer of the module.

**Integration testing:** Integration testing is the next level of testing. Several unit-tested modules are integrated into distinct subsystems, which are then tested. The goal here is to see if the modules can be properly integrated.

# Test Hierarchies

**System testing:** Here the entire software system is tested. The reference document for this process is the requirements document, and the goal is to see if the software meets its requirements.

**Acceptance testing:** Acceptance testing is often performed with realistic data of the client to demonstrate that the software is working satisfactorily. It may be done in the setting in which the software is to eventually function.

# Key Takeaways

- A unit test in Java gives the developer the opportunity to test an object.
- In JUnit 5, the test lifecycle is driven by four primary annotations which are: @BeforeAll, @BeforeEach, @AfterEach and @AfterAll.
- Testing is typically used to discover errors left over from earlier phases, as well as problems created during the coding process itself.
- Unit testing is essentially for verification of the code produced by individual program. Acceptance testing is often performed with realistic data of the client to demonstrate that the software is working satisfactorily.