INTRODUCTION TO JAVA

Java 1.0

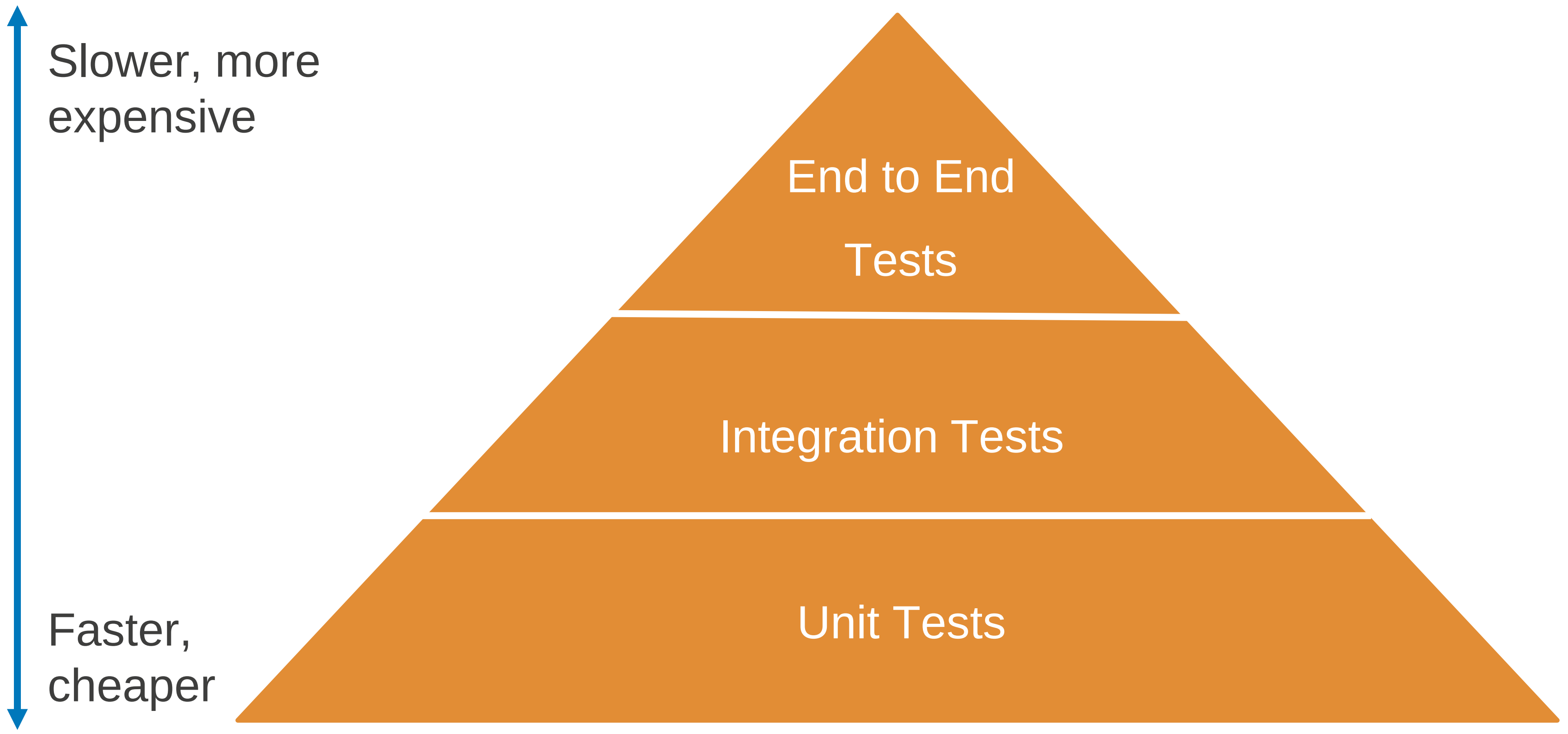
CODE TESTING

Lesson # 13

THE PURPOSE OF SOFTWARE TESTS

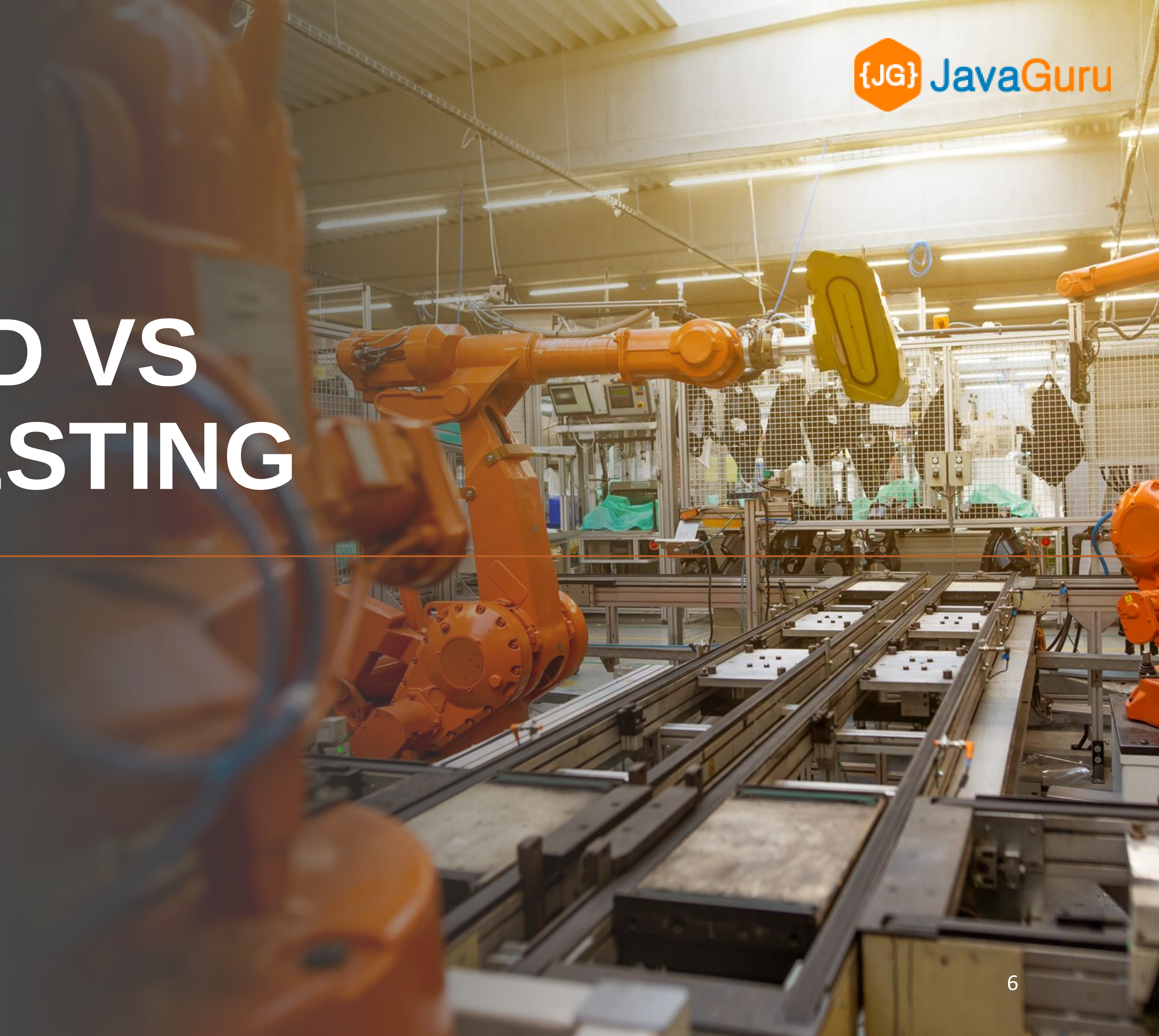
- A test is a piece of software that executes another piece of software in order to confirm that the code operates as expected
- A test can check
  - Expected state (state testing)
  - Expected sequence of events (behavior testing)
- Having high test coverage allows the development of new features without being afraid to break existing code

SOFTWARE TESTING SCOPES



SOFTWARE TESTING SCOPES

- Integration tests
  - Aims to test the behavior of a component or the integration between a set of components
  - Check that the whole system works as intended
- Unit tests
  - Targets a small unit of code (e.g., a method or a class)
  - External class dependencies should be replaced with test implementation objects (mocks)



AUTOMATED VS MANUAL TESTING

MANUAL TESTING

- Executing test cases manually without any tool support is known as manual testing
- It’s time-consuming and tedious
  - Since human resources execute test cases, it is very slow and tedious
- Huge investment in human resources
  - As test cases need to be executed manually, more testers are required for manual testing

MANUAL TESTING

- Less reliable
  - Manual testing is less reliable, as it has to account for human errors
- Non-programmable
  - No programming can be done to write sophisticated tests to fetch hidden information

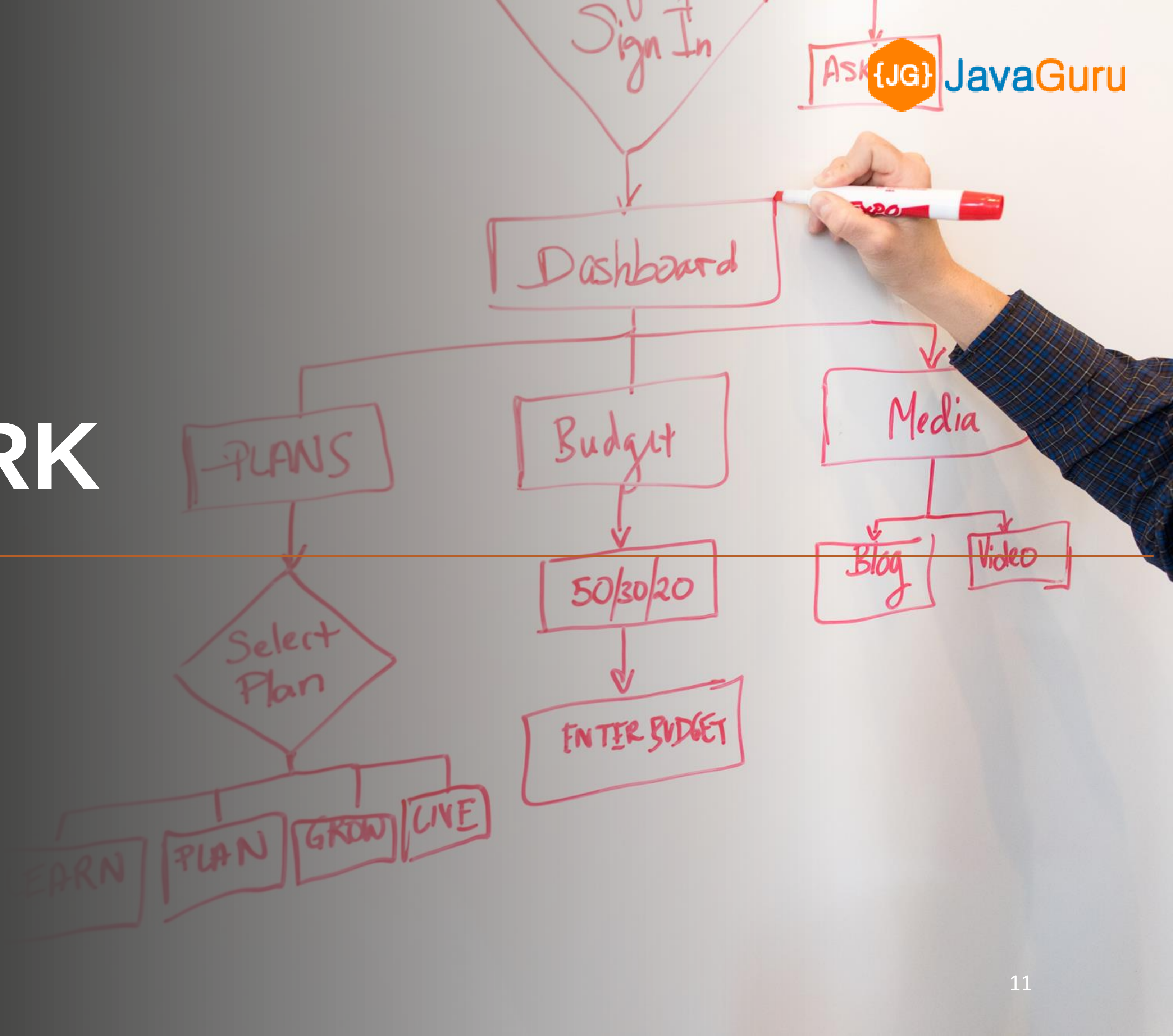
AUTOMATED TESTING

- Taking tool support and executing the test cases by using an automation tool is known as automation testing
- Fast
    - Automation runs test cases significantly faster than human resources
- Less investment in human resources
    - Test cases are executed using automation tools, so less number of testers are required in automation testing

AUTOMATED TESTING

- More reliable
  - Automation tests are precise and reliable
- Programmable
  - Testers can program sophisticated tests to bring out hidden information

JUNIT FRAMEWORK



JUNIT TESTING APROACH

- JUnit is a unit testing framework for the Java programming language
- JUnit test is a method contained in a class that is only used for testing (also called a test class)
- Formally written unit test case is characterized by:
  - Known input
  - Expected output

SYSTEM UNDER TEST

```java
public class Calculator {

    public int sum(int a, int b) {
        return a + b;
    }

}
```

TEST CLASS

```java
public class CalculatorTest {

    private Calculator calculator;

    @BeforeEach
    public void setUp() {
        calculator = new Calculator();
    }

   @Test
   public void shouldCalculateSum() {
       int result = calculator.sum(3, 5);
       assertEquals(8, result);
   }

}
```

TEST PREPARATION ANNOTATIONS

Annotation	Description
@Test **public void** testCase() {}	The @Test annotation indicates the following method as a test method
@Disabled **public void** testCase() {}	This annotation is useful when you want temporarily disable the execution of a specific test
@Test @Timeout(value = 500, unit = MILLISECONDS) **public void** testCase() {}	If the method takes longer than 500 milliseconds, the test will fail

TEST DECLARATION ANNOTATIONS

Annotation	Description
@BeforeEach **public void** setUp() {}	This method is executed before each test
@AfterEach **public void** tearDown() {}	This method is executed after each test
@BeforeAll **public static void** setUp(){}	The following static method is executed once, before the start of all tests
@AfterAll **public static void** tearDown(){}	The following static method is executed once after all tests have been completed

ASSERT STATEMENTS

Assertion	Desscription

Assertions.assertEquals(expected, actual);
Assertions.assertNotEquals(expected, actual);
	Asserts that expected and actual are equal or not equal

Assertions.assertTrue(actual);
Assertions.assertFalse(actual);
	Asserts that the supplied condition is true or not true

Assertions.assertNull(actual);
Assertions.assertNotNull(actual);
	Asserts that actual is null or not null

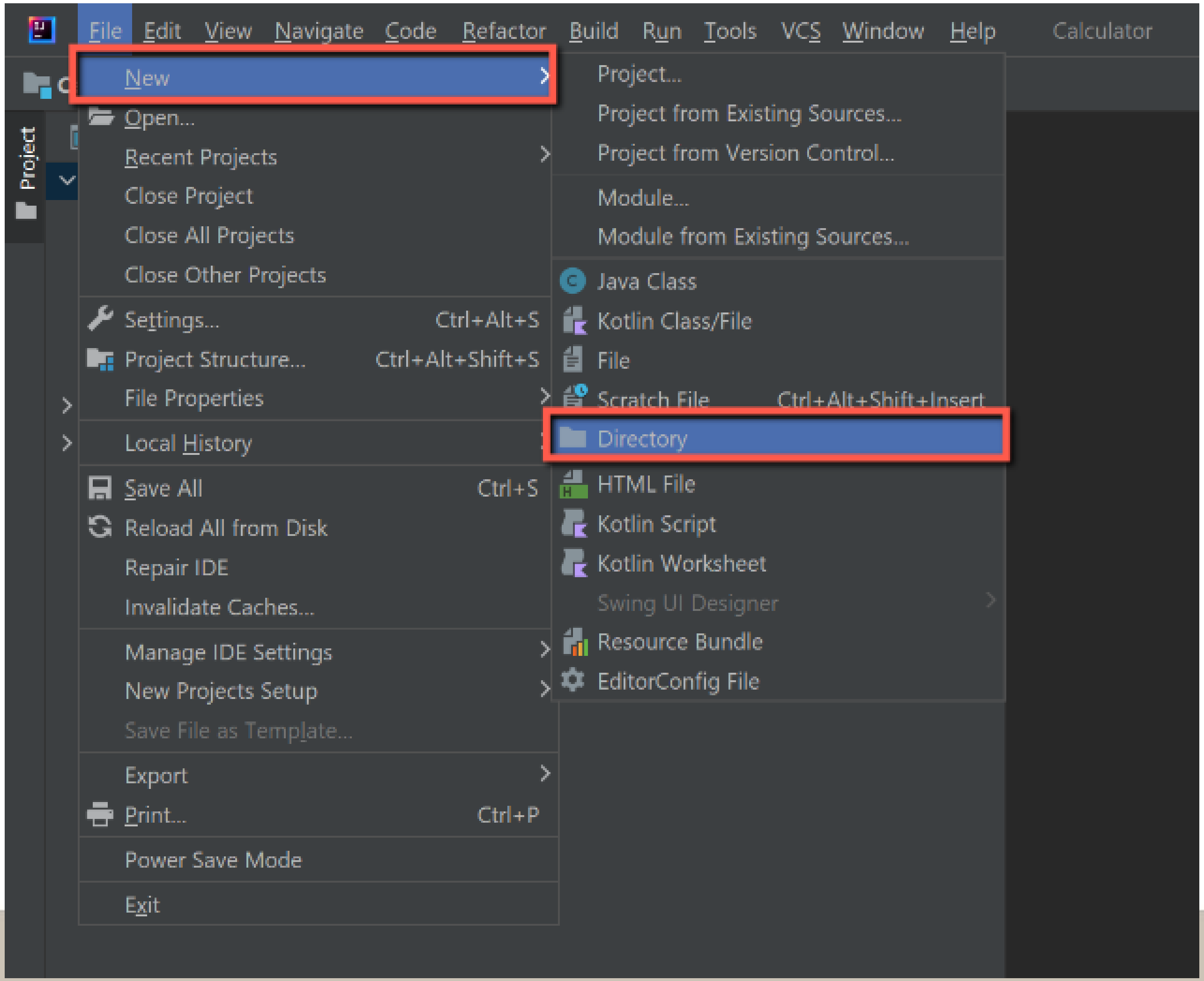
Assertions.assertSame(expected, actual);
Assertions.assertNotSame(expected, actual);
	Asserts that expected and actual refer to the same object

Assertions.assertThrows(expectedType, executable);	Asserts that execution of the supplied executable throws an exception of the expectedType and returns the exception

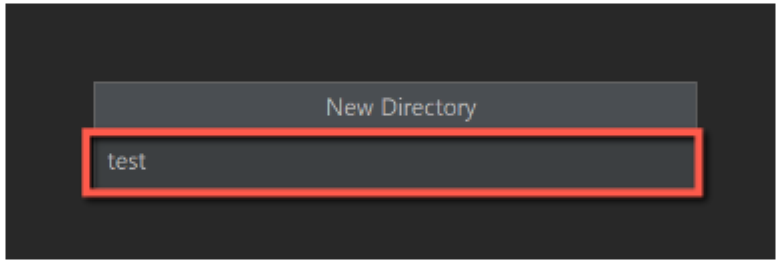
JavaGuru

MANUAL JUNIT PROJECT SETUP

MANUAL JUNIT SETUP

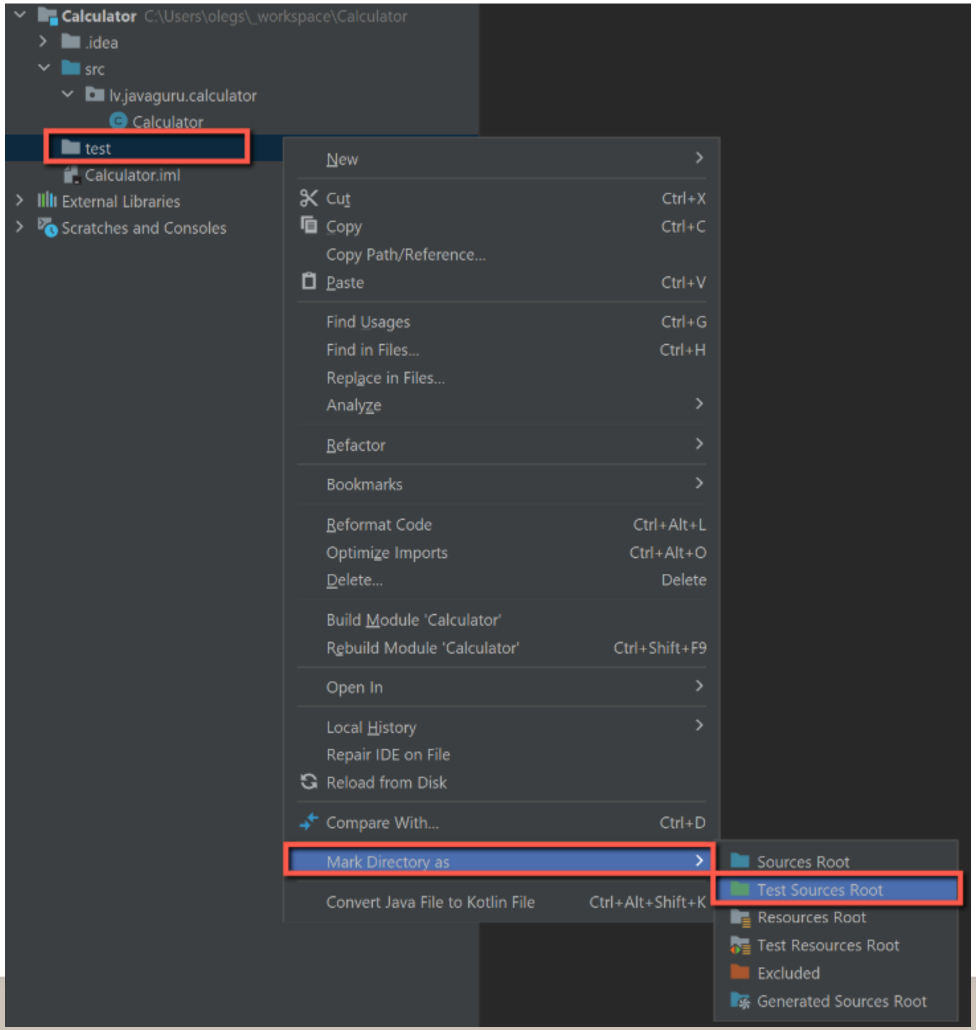


MANUAL JUNIT SETUP

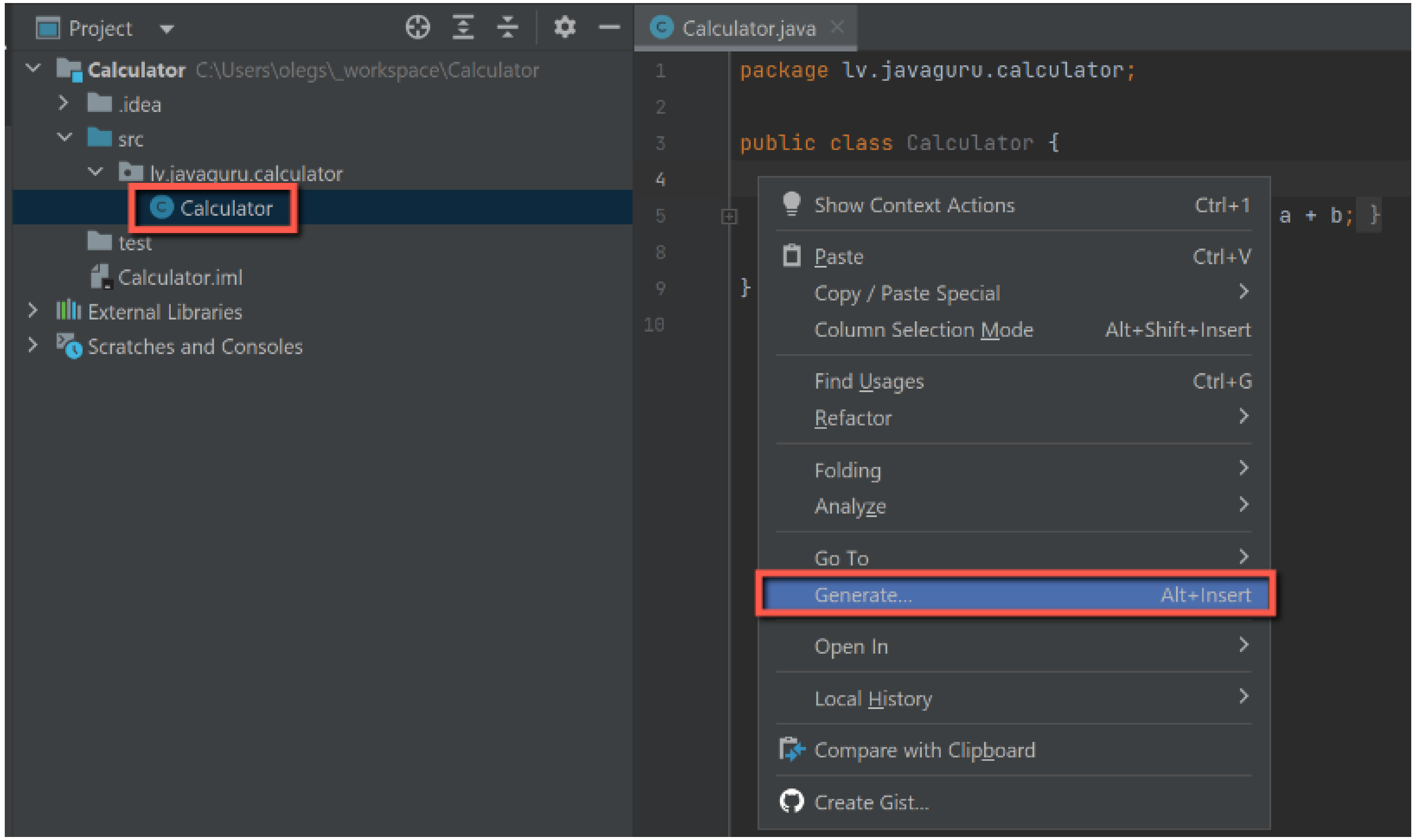


test

MANUAL JUNIT SETUP

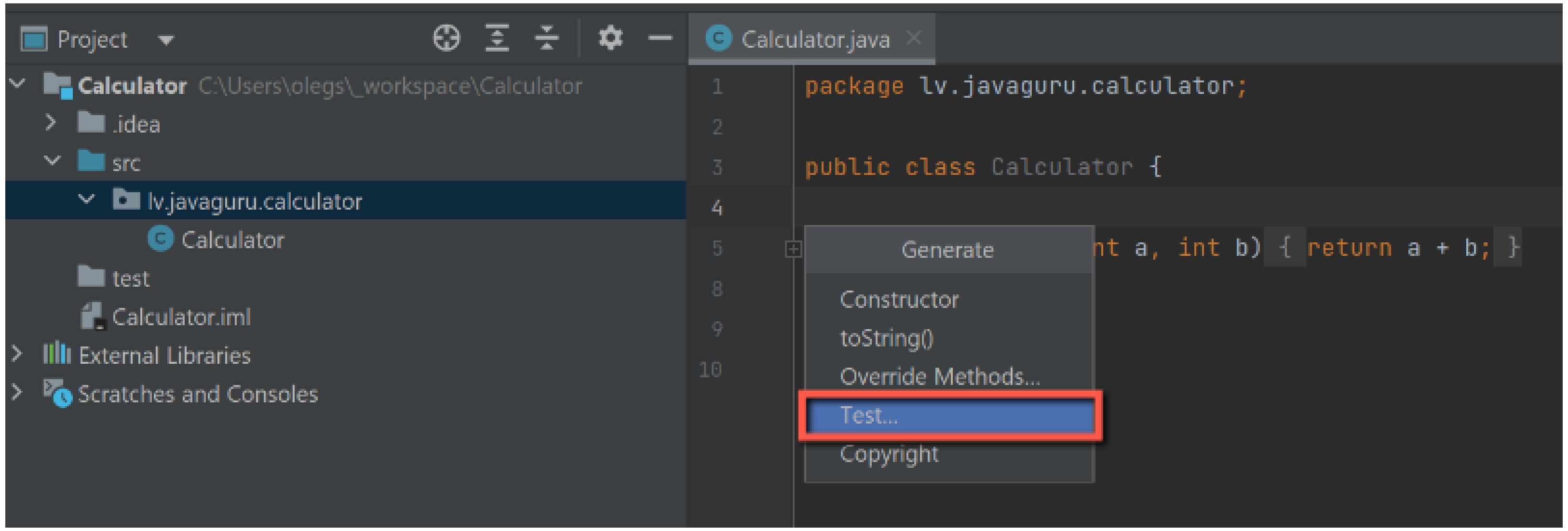


MANUAL JUNIT SETUP



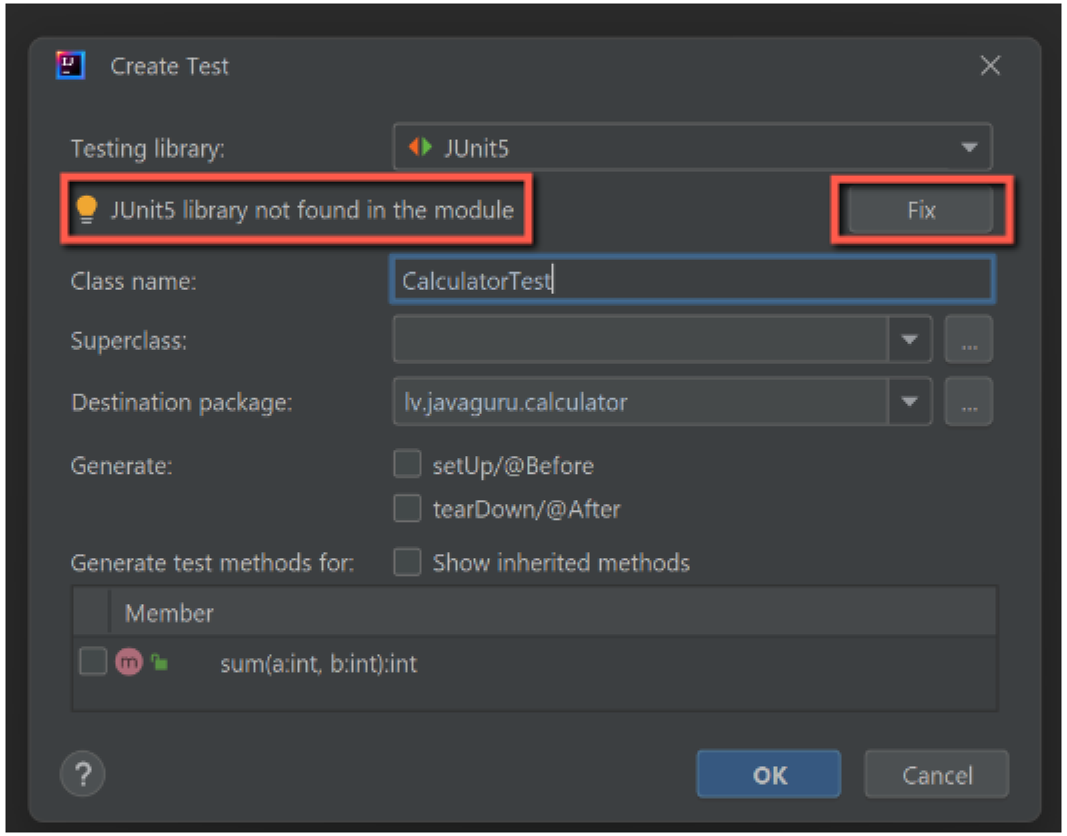
```
package lv.javaguru.calculator;

public class Calculator {

}
```

MANUAL JUNIT SETUP



```
package lv.javaguru.calculator;

public class Calculator {

    nt a, int b) { return a + b; }
```

MANUAL JUNIT SETUP



Testing library: JUnit5

JUnit5 library not found in the module

Class name: CalculatorTest

Superclass:

Destination package: lv.javaguru.calculator

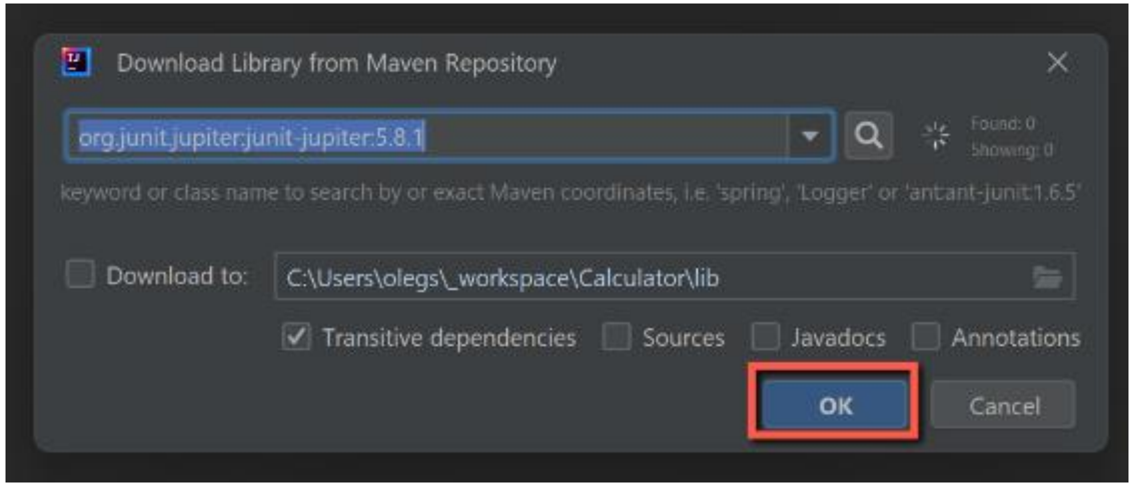
Generate: setUp/@Before, tearDown/@After

Generate test methods for: Show inherited methods

Member

sum(a:int, b:int):int

MANUAL JUNIT SETUP


Download Library from Maven Repository

org.junit.jupiter:junit-jupiter:5.8.1

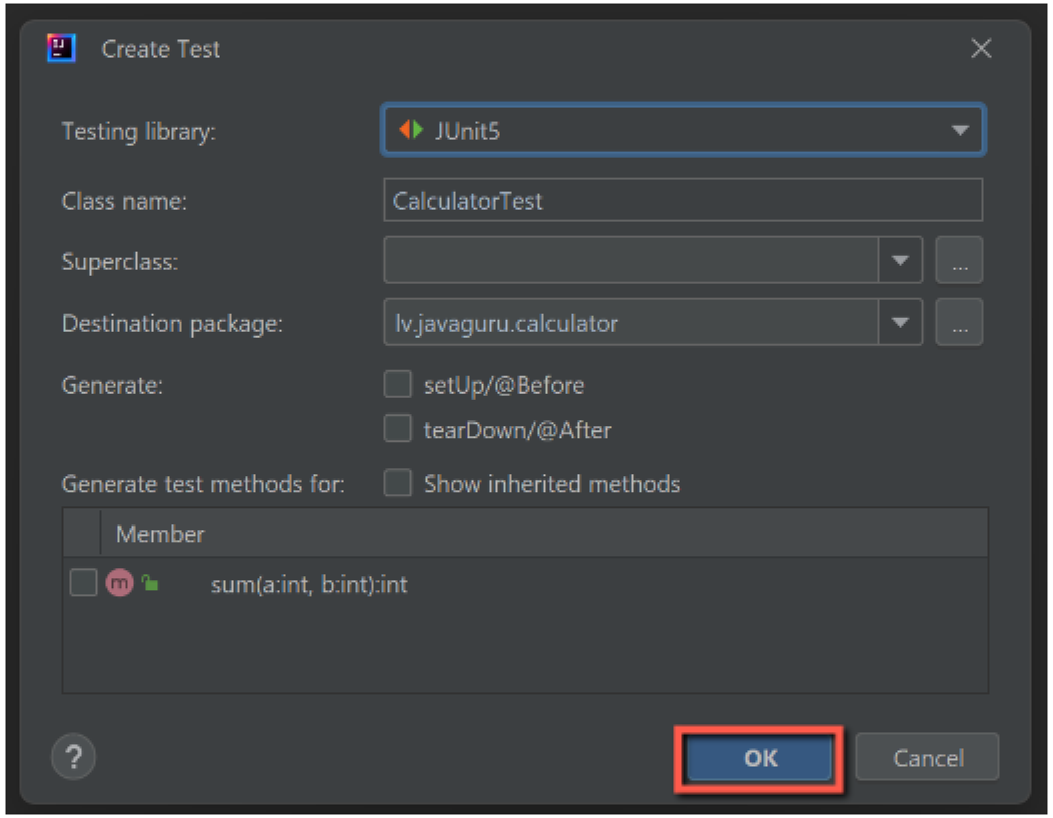
Found: 0
Showing: 0

keyword or class name to search by or exact Maven coordinates, i.e. 'spring', 'Logger' or 'ant:ant-junit:1.6.5'

☐ Download to: C:\Users\olegs_workspace\Calculator\lib

☑ Transitive dependencies ☐ Sources ☐ Javadocs ☐ Annotations

MANUAL JUNIT SETUP


Create Test

Testing library: JUnit5

Class name: CalculatorTest

Superclass:

Destination package: lv.javaguru.calculator

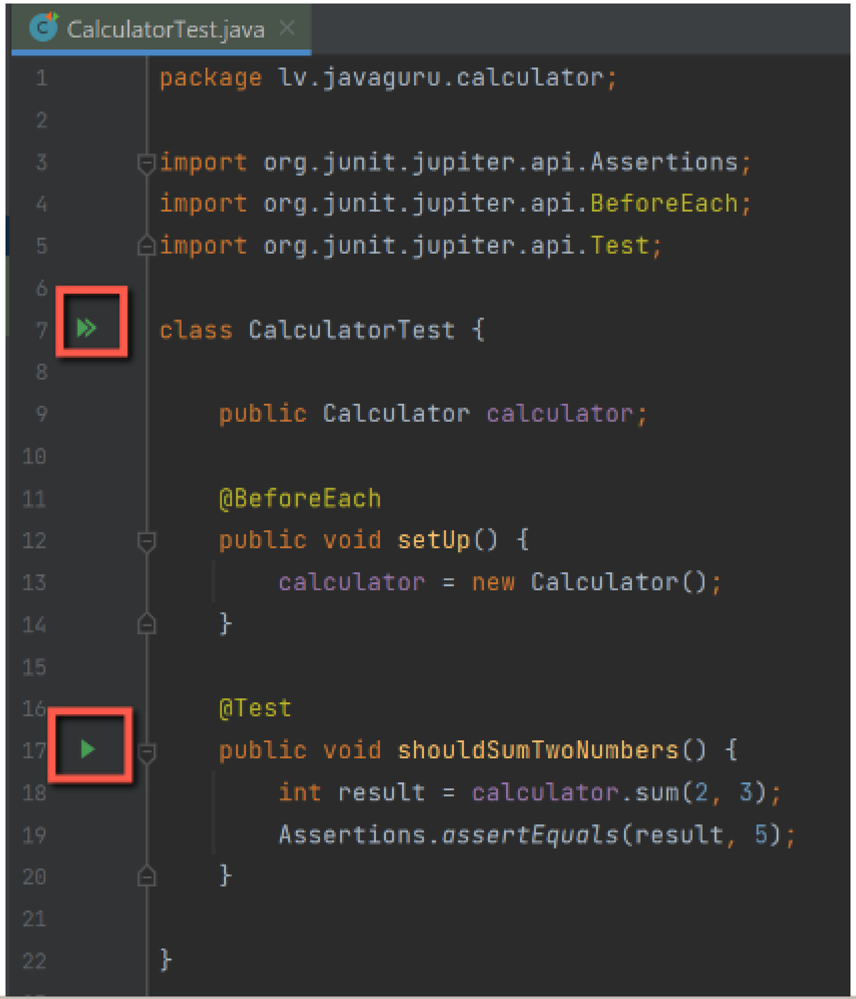
Generate: setUp/@Before, tearDown/@After

Generate test methods for: Show inherited methods

Member

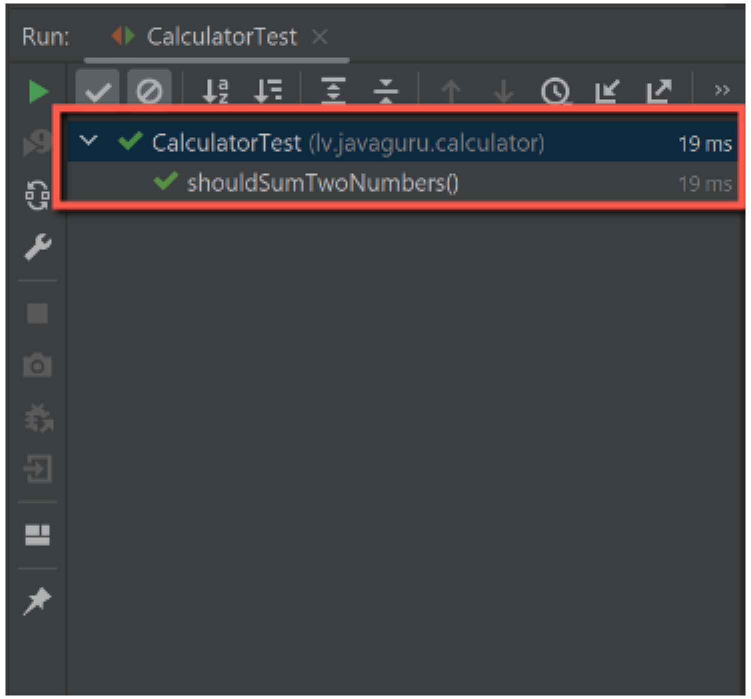
sum(a:int, b:int):int

MANUAL JUNIT SETUP


CalculatorTest.java

```
package lv.javaguru.calculator;

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

class CalculatorTest {

    public Calculator calculator;

    @BeforeEach
    public void setUp() {
        calculator = new Calculator();
    }

    @Test
    public void shouldSumTwoNumbers() {
        int result = calculator.sum(2, 3);
        Assertions.assertEquals(result, 5);
    }

}
```

MANUAL JUNIT SETUP



```
CalculatorTest (lv.javaguru.calculator)   19 ms
    shouldSumTwoNumbers()                 19 ms
```

REFERENCES

REFERENCES

- https://junit.org/junit4/
- https://www.tutorialspoint.com/junit/junit_basic_usage.htm
- https://www.vogella.com/tutorials/JUnit/article.html
- https://www.swtestacademy.com/junit4/
- https://dzone.com/articles/7-popular-unit-test-naming

QUESTIONS?

{JG} JavaGuru

THANK YOU!