JAVA
Error Handling and Unit Test
LAB 9

1. Objectives:

- Design simple classes to illustrate catching error
- Design simple classes to illustrate Java Unitary testing
- Junit framework usage

2. Unit Test / Integration Test / System Tests / VIL Tests

2.1. Definition:

SIL: SOFTWARE IN THE LOOP
HIL: HARDWARE IN THE LOOP
VIL: VEHICULE IN THE LOOP
DUT: Device under test

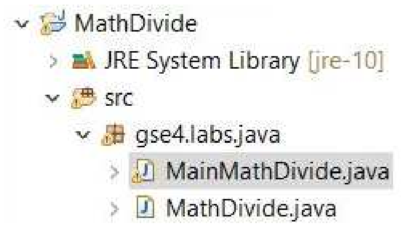
Different family of tests:

- Unitary tests
- Integration tests
- System tests
- Vehicle tests

Regression
Behavior testing
Test fixture is a test precondition: a determined specific input

3. Error Handling

- Create a new Java Project with Eclipse**: MathDivide**
- **Add 2 Classes:**
  - **.1. MathDivide.java**
  - **.2. MainMathDivide.java**



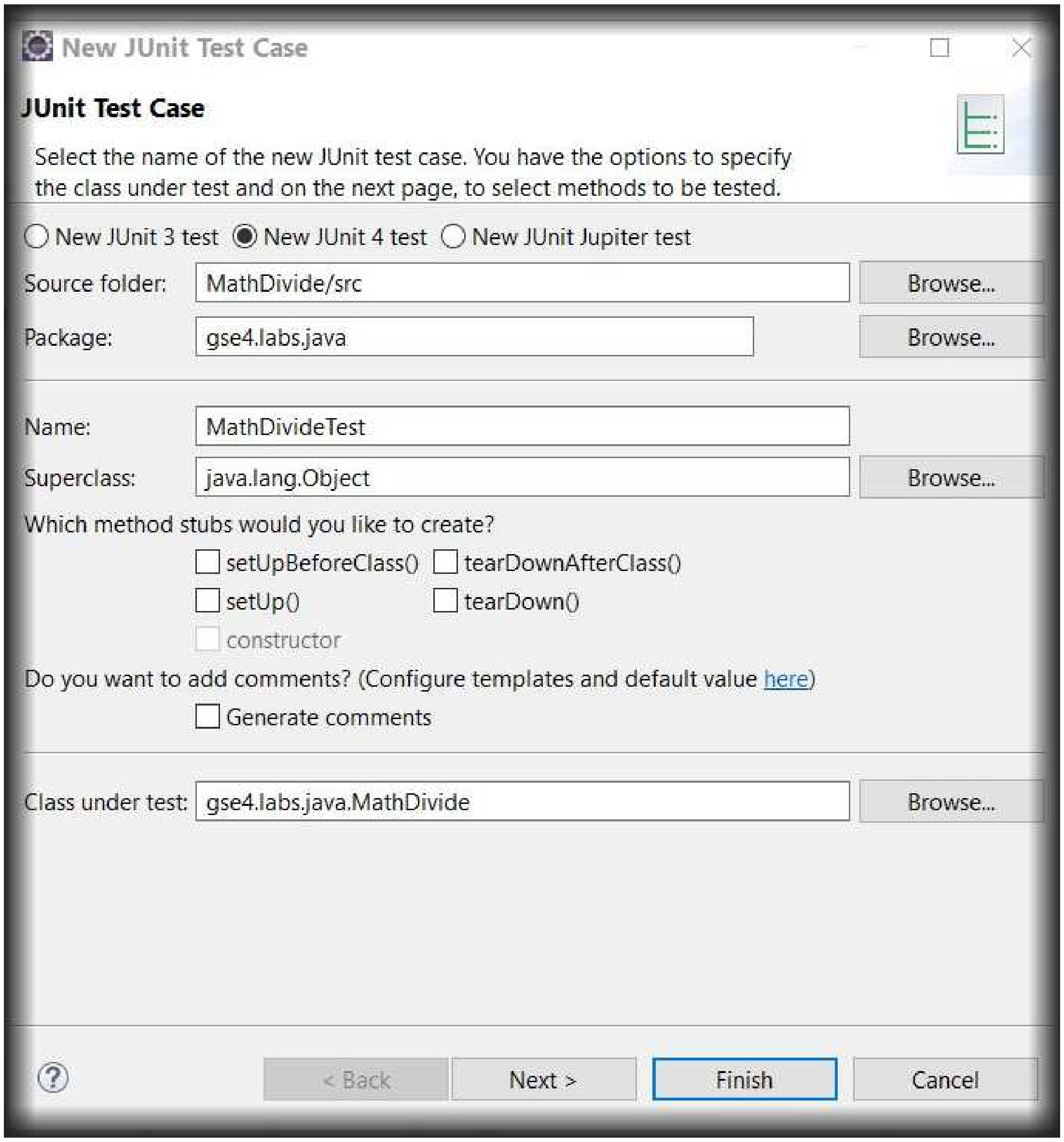
```
MathDivide
  JRE System Library [jre-10]
  src
    gse4.labs.java
      MainMathDivide.java
      MathDivide.java
```

- Add a new **MathDivide** class representing a Division object. The **MathDivide** class contains:
  - o A method to make integer division **`DivideNormal`**

- Test your MathDivide class by writing a simple program that creates a MathDivide instance
- Execute the **DivideNormal** division of 4 by 2 and provide the result
- Create Divide method to control yourself the division by 0 by using : throw new IllegalArgumentException, generate your own error message
- Improve the code by adding DivideCatchError method, that is making the same division but with caching the error: use Catch Error/Exception from course.

4. Unitary test

Right-click on **MathDivide.java class** in the Package select New / JUnit Test Case and fill as documented in the next picture


New JUnit Test Case

JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

○ New JUnit 3 test [✓ ● New JUnit 4 test] ○ New JUnit Jupiter test

Source folder: MathDivide/src

Package: gse4.labs.java

Name: MathDivideTest

Superclass: java.lang.Object

Which method stubs would you like to create?

☐ setUpBeforeClass() ☐ tearDownAfterClass()

☐ setUp() ☐ tearDown()

☐ constructor

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Class under test: gse4.labs.java.MathDivide

Use @Test which Identifies a method as a test method.

Create a test vector by using assertEquals checking the result of the division

Create a second test vector by using assertEquals checking a wrong output the result of the division

Create a third second test vector by using @Test(expected = IllegalArgumentException.class)

So testing the detection of division by 0