НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Object-Oriented Programming in the Java language

TDD and JUnit

Yevhen Berkunskyi, NUoS
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua

History

- Kent Beck developed the first xUnit automated test tool for Smalltalk in mid-90's
- Beck and Gamma (of design patterns Gang of Four) developed JUnit on a flight from Zurich to Washington, D.C.
- Martin Fowler: "Never in the field of software development was so much owed by so many to so few lines of code."
- JUnit has become the standard tool for Test-Driven Development in Java (see junit.org)
- JUnit test generators now part of many Java IDEs (IntelliJ IDEA, NetBeans, Eclipse, BlueJ, …)

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

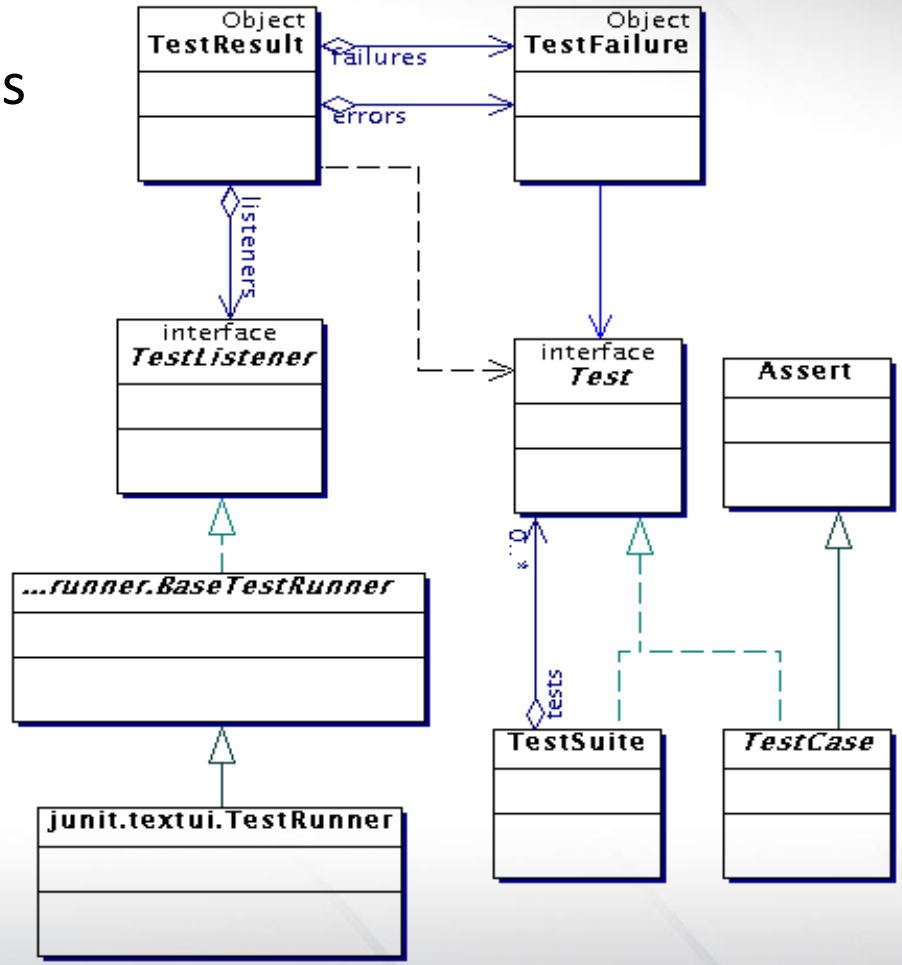
Why create a test suite?

- Obviously you have to test your code—right?
  - You can do *ad hoc* testing (running whatever tests occur to you at the moment), or
  - You can build a test suite (a thorough set of tests that can be run at any time)
- Disadvantages of a test suite
  - It's a lot of extra programming
    - True, but use of a good test framework can help quite a bit
  - You don't have time to do all that extra work
    - *False!* Experiments repeatedly show that test suites reduce debugging time more than the amount spent building the test suite
- Advantages of a test suite
  - Reduces total number of bugs in delivered code
  - Makes code much more maintainable and refactorable

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Architectural overview

- JUnit test framework is a package of classes that lets you write tests for each method, then easily run those tests
- **TestRunner** runs tests and reports **TestResult**s
- You test your class by extending abstract class ***TestCase*** (optional)
- To write test cases, you need to know and understand the **Assert** class



Writing a TestCase

- To start using JUnit, create a subclass of *TestCase*, (optional in JUnit 4 and 5) to which you add test methods
- Name of class is important – should be of the form MyClass***Test***
- This naming convention lets TestRunner automatically find your test classes

```
import org.junit.jupiter.api.BeforeEach;

import static org.junit.jupiter.api.Assertions.*;

class MainTest {
    @BeforeEach
    void setUp() {

    }

}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Writing methods in TestCase

- Pattern follows ***programming by contract*** paradigm:
  - Set up **preconditions**
  - Exercise functionality being tested
  - Check **postconditions**
- Example:

```
public void testEmptyList() {
     Bowl emptyBowl = new Bowl();
     assertEquals(0, emptyList.size(),
          “Size of an empty list should be zero.”);
     assertTrue(emptyBowl.isEmpty(),
          “An empty bowl should report empty.”);
}
```

- Things to notice:
  - Specific method signature – public void ***test***Whatever()
  - Coding follows pattern
  - Notice the assert-type calls…

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Assert methods

- Each assert method has parameters like these: *expected-value, actual-value, message*
- Assert methods dealing with floating point numbers get an additional argument, a tolerance
- Each assert method has an equivalent version that does not take a message – however, this use is not recommended because:
  - messages helps documents the tests
  - messages provide additional information when reading failure logs

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Assert methods

- assertTrue(Boolean *test*, String *message*)
- assertFalse(Boolean *test*, String *message*)
- assertNull(Object *object*, String *message*)
- assertNotNull(Object *object*, String *message*)
- assertEquals(Object *expected*, Object *actual*, String *message*)
  // uses equals method
- assertSame(Object *expected*, Object *actual*, String *message*)
  // uses == operator
- assertNotSame(Object *expected*, Object *actual*, String *message*)

More stuff in test classes

- Suppose you want to test a class Counter
- public class CounterTest {
  - This is the unit test for the Counter class
- public CounterTest() { } //Default constructor
- protected void setUp()
  - Test *fixture* creates and initializes instance variables, etc.
- protected void tearDown()
  - Releases any system resources used by the test fixture
- public void testIncrement(), public void testDecrement()
  - These methods contain tests for the Counter methods increment(), decrement(), etc.
  - Note capitalization convention

JUnit tests for Counter

```
public class CounterTest {
    Counter counter1;
    @BeforeEach
    protected void setUp() {  // creates a test fixture
        counter1 = new Counter();
    }
    @Test
    public void testIncrement() {
        assertTrue(counter1.increment() == 1);
        assertTrue(counter1.increment() == 2);
    }
    @Test
    public void testDecrement() {
        assertTrue(counter1.decrement() == -1);
    }
}
```

Note that each test begins with a *brand new* counter

This means you don't have to worry about the order in which the tests are run

TestSuites

- TestSuites collect a selection of tests to run them as a unit
- Collections automatically use TestSuites, however to specify the order in which tests are run, write your own:

```
public static Test suite() {
    suite.addTest(new TestBowl("testBowl"));
    suite.addTest(new TestBowl("testAdding"));
    return suite;
}
```

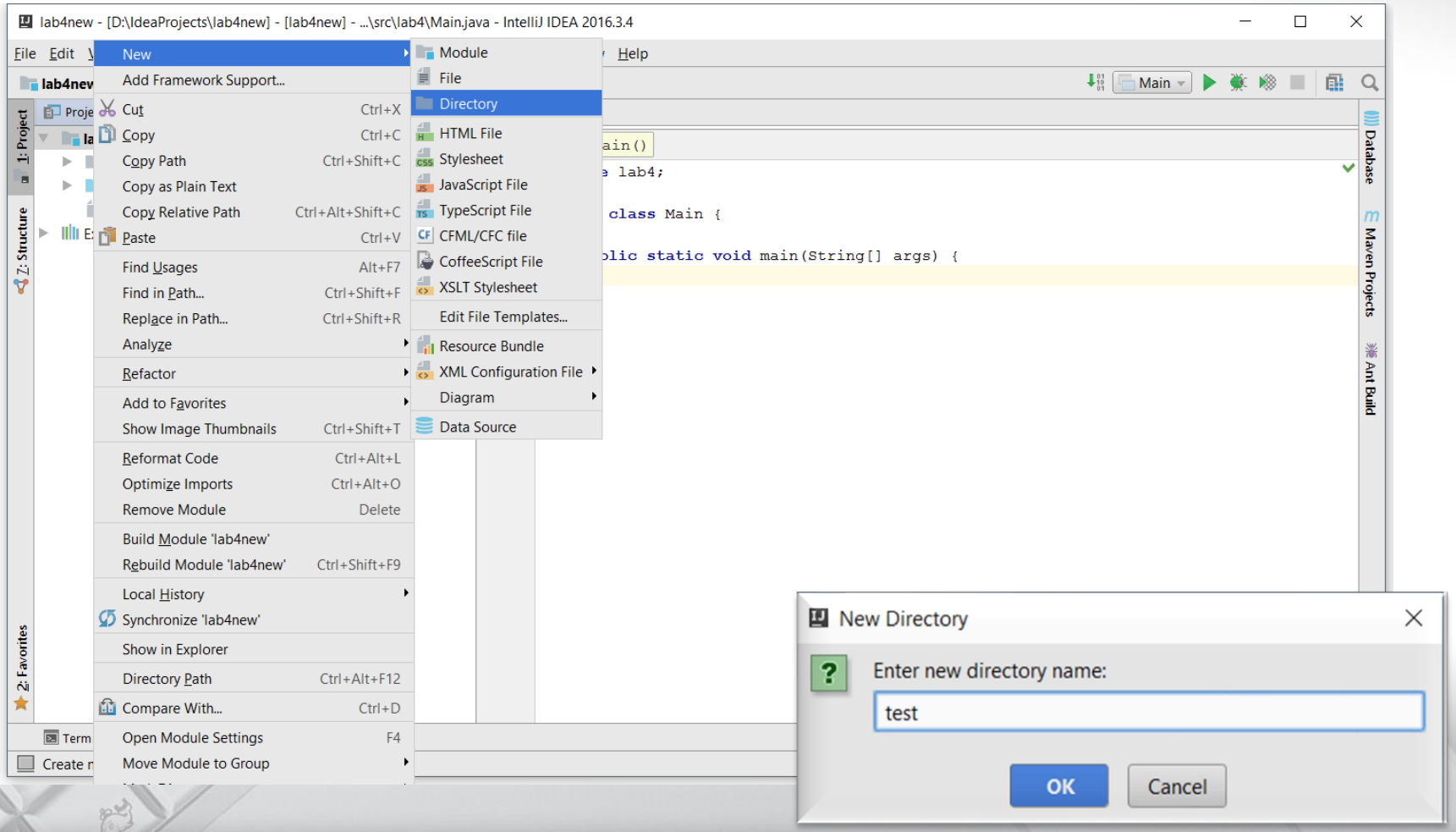
- Should seldom have to write your own TestSuites as each method in your TestCase should be independent of all others
- Can create TestSuites that test a whole package:

```
public static Test suite() {
    TestSuite suite = new TestSuite();
    suite.addTestSuite(TestBowl.class);
    suite.addTestSuite(TestFruit.class);
    return suite;
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

JUnit in IntelliJ IDEA

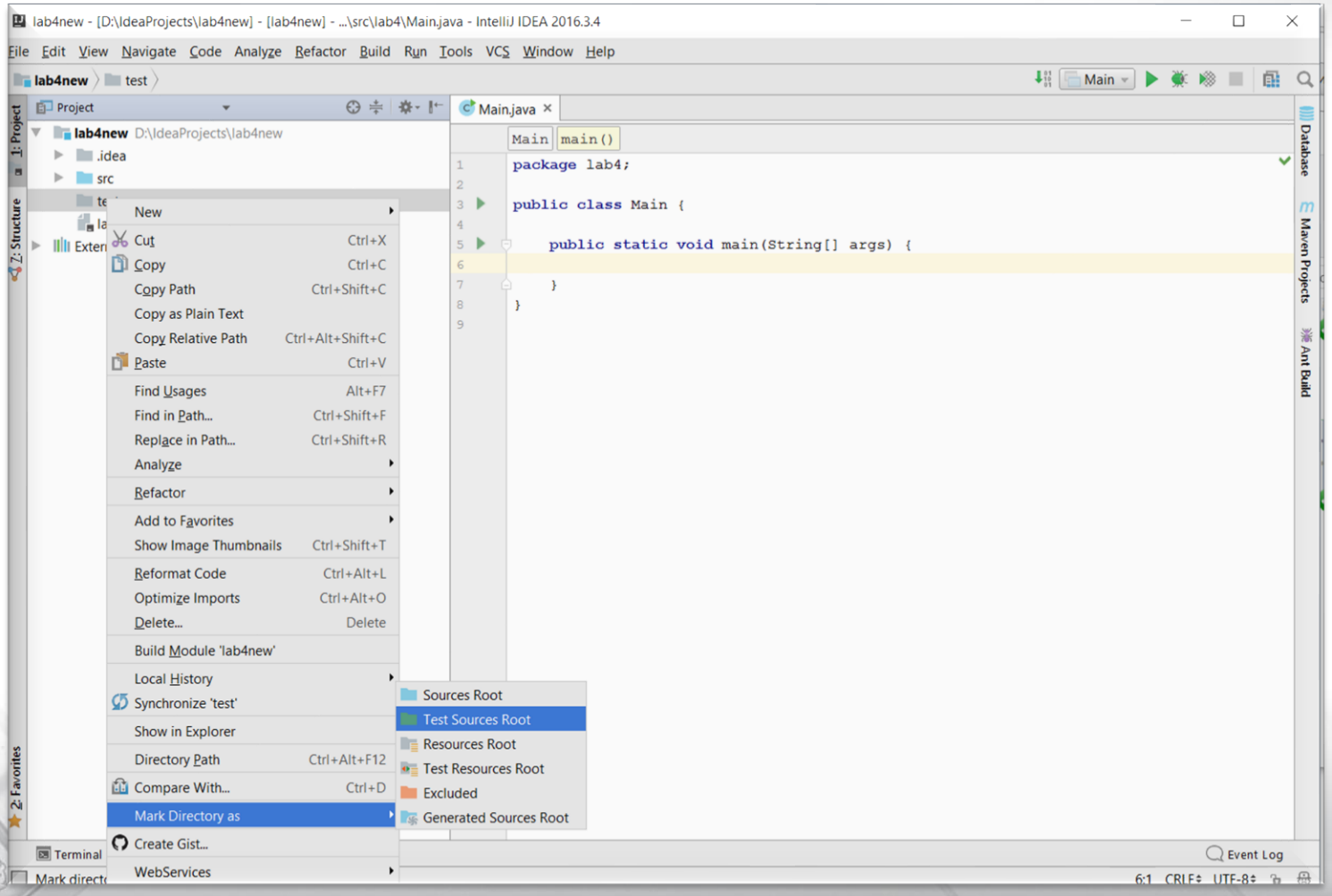
At first you have to create a directory for your tests



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

JUnit in IntelliJ IDEA

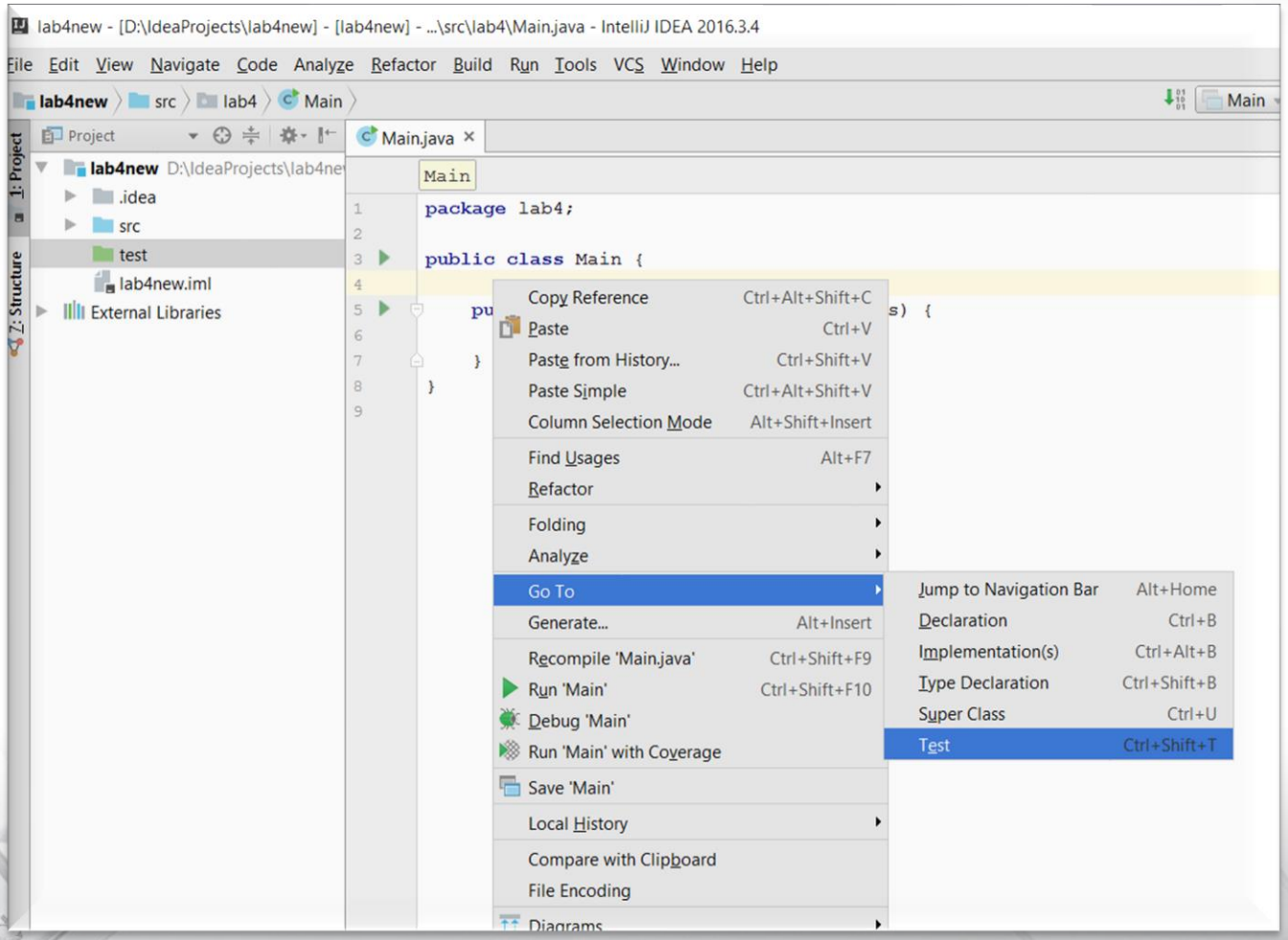
Then, mark it as Test Sources Root



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

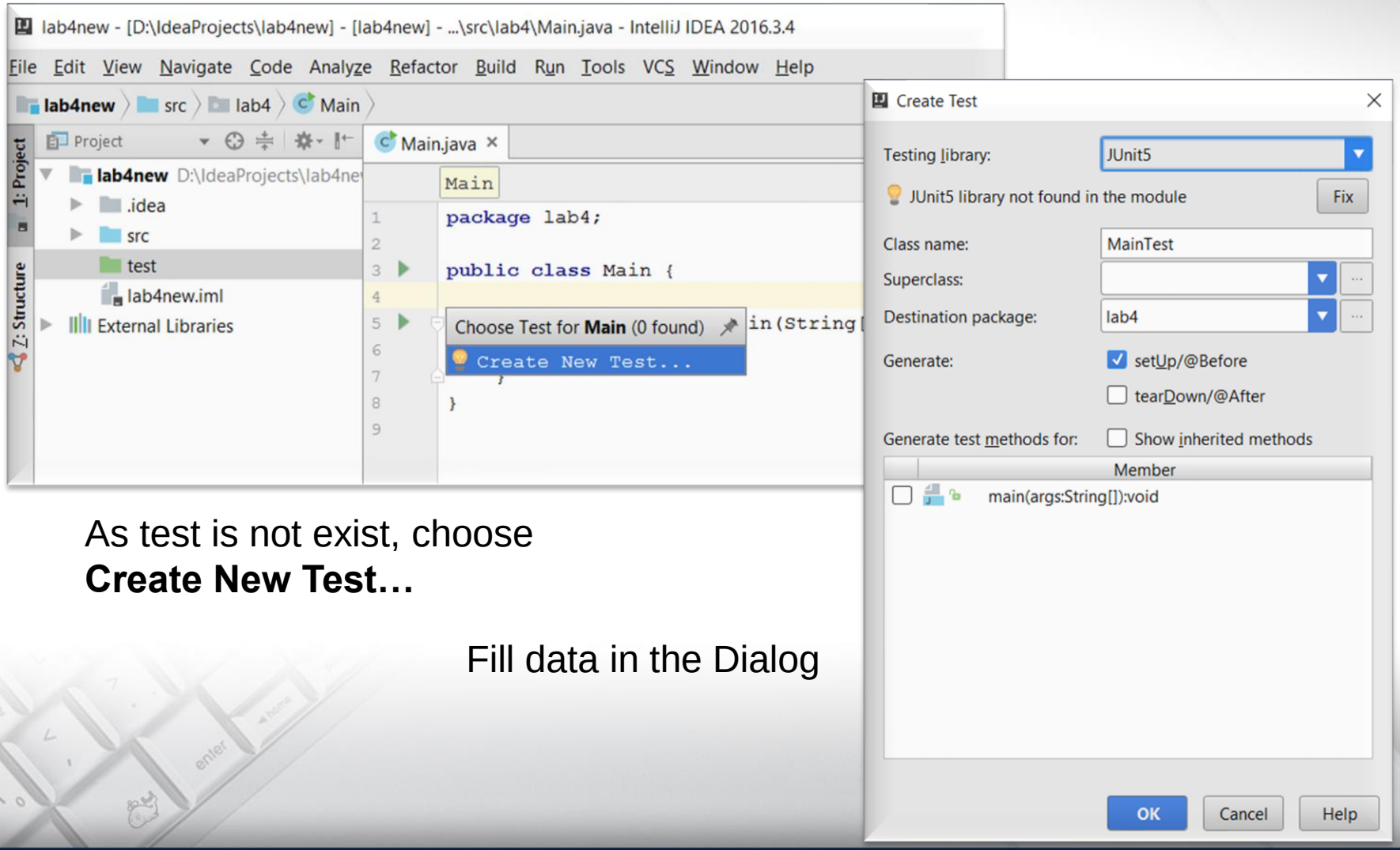
JUnit in IntelliJ IDEA

In your class choose “Go To ► Test



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

JUnit in IntelliJ IDEA

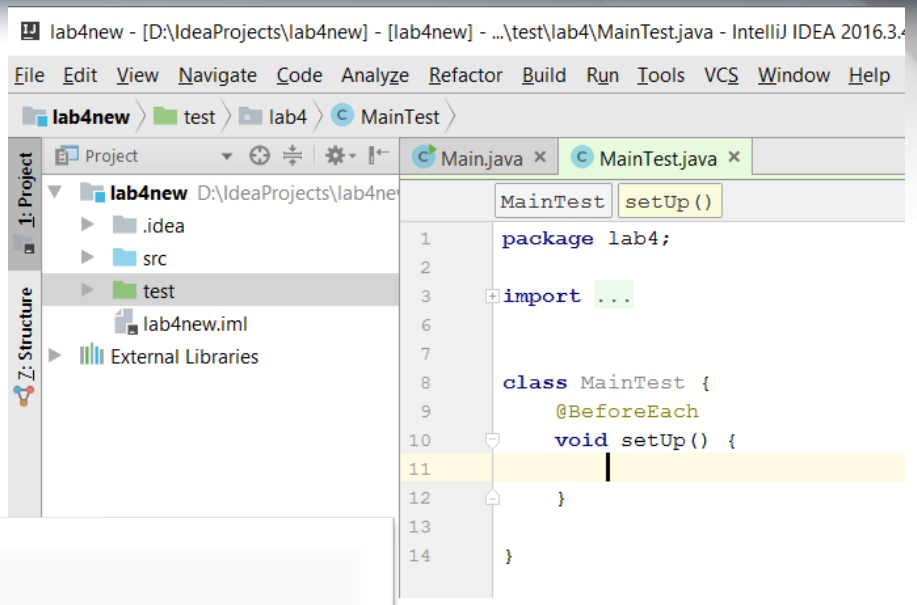


As test is not exist, choose
Create New Test…

Fill data in the Dialog

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

JUnit in IntelliJ IDEA

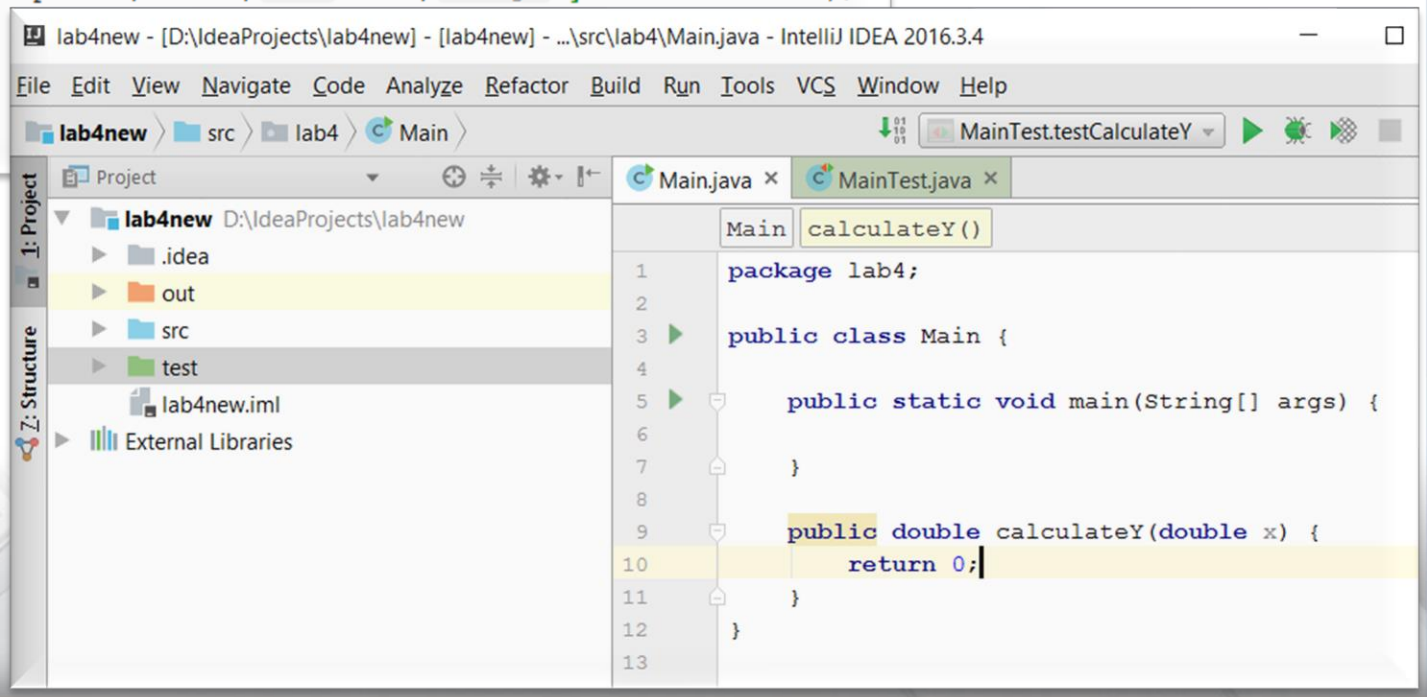



```java
class MainTest {
    Main main;

    @BeforeEach
    void setUp() { main = new Main(); }

    @Test
    void testCalculateY() {
        double x = 2.3;
        double expected = 5.29;
        double result = main.calculateY(x);
        assertEquals(expected,result, delta: 0.001, message: "y must be a x*x");
    }

}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

JUnit in IntelliJ IDEA

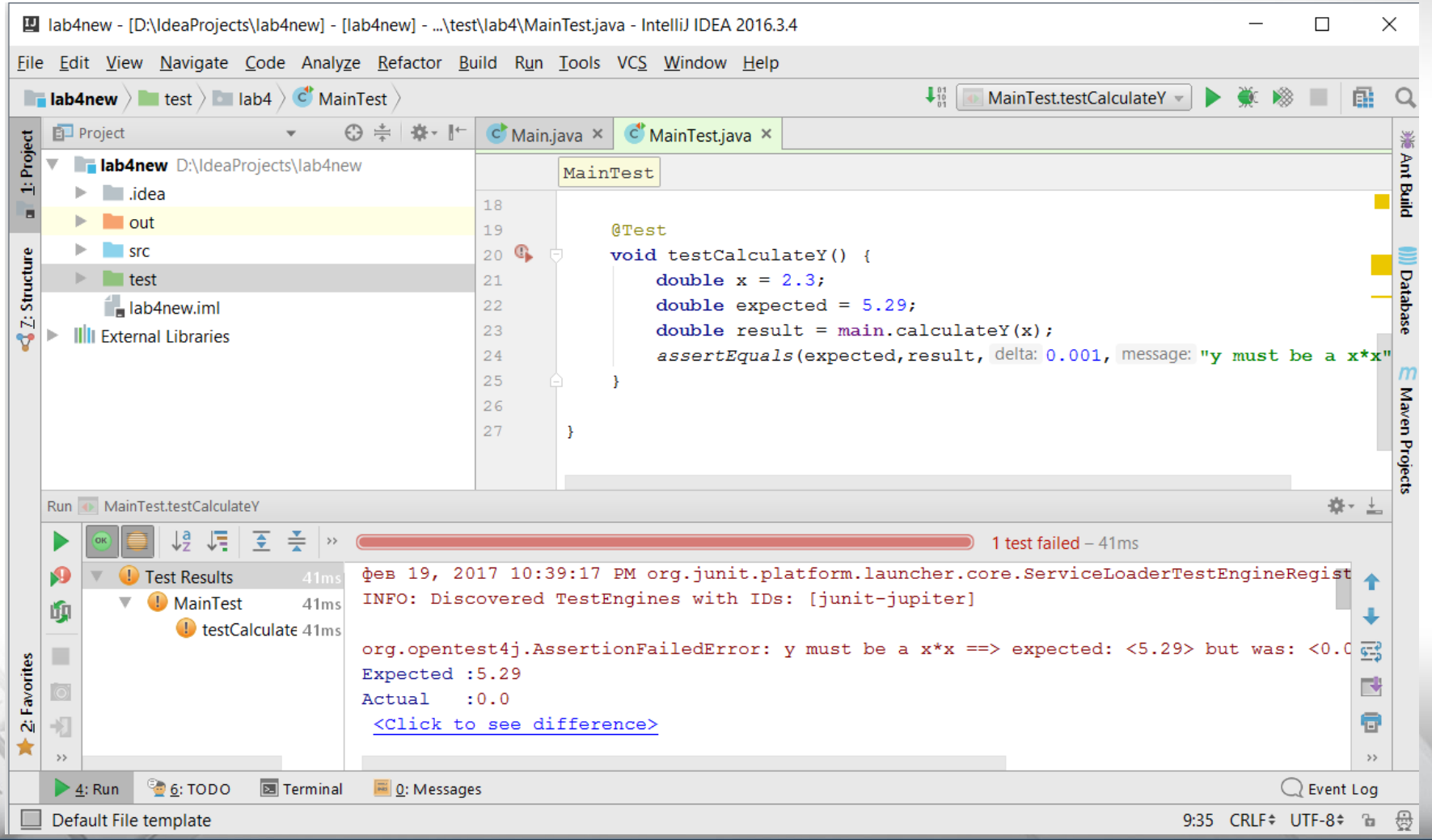
```
class MainTest {
    Main main;

    @BeforeEach
    void setUp() { main = new Main(); }

    @Test
    void testCalculateY() {
        double x = 2.3;
        double expected = 5.29;
        double result = main.calculateY(x);
        assertEquals(expected,result, delta: 0.001, message: "y must be a x*x");
    }

}
```



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

JUnit in IntelliJ IDEA

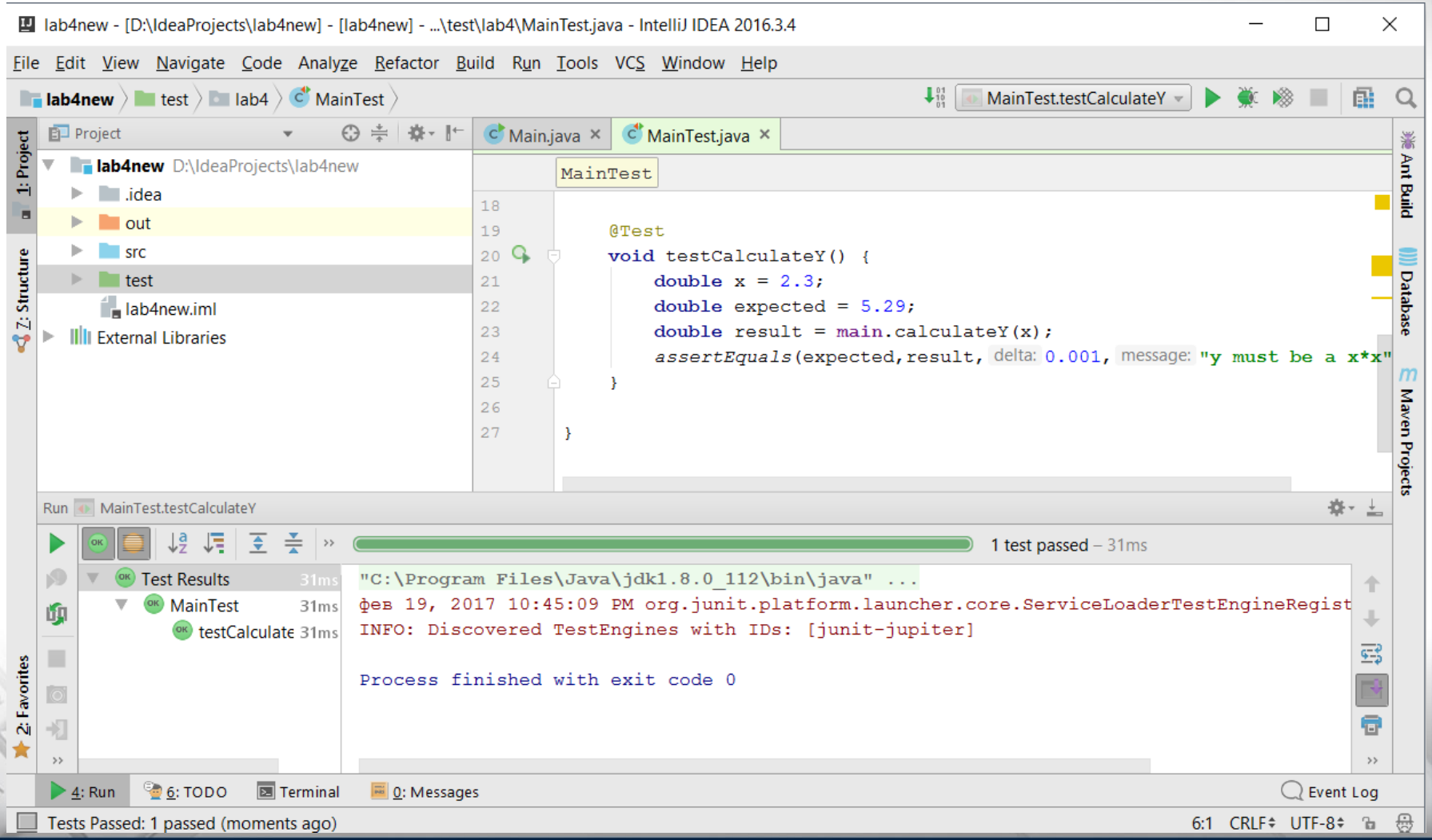
Run test of the generated method. It fails



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

JUnit in IntelliJ IDEA

Write correct method body. Run test of the generated method. It should be OK



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

More Information

- http://www.junit.org
  - Download of JUnit
  - Lots of information on using JUnit
- http://sourceforge.net/projects/cppunit
  - C++ port of Junit
- http://www.thecoadletter.com
  - Information on Test-Driven Development

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Example

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Questions?

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Object-Oriented Programming in the Java language

Part 2. TDD and JUnit

Yevhen Berkunskyi, NUoS
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua