

# **СОВРЕМЕННЫЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПО**

**Практикум:**

**Модульное тестирование с  
JUnit**

# JUnit

- Библиотека для модульного тестирования
  - поставляется в виде jar-файла
  - в составе большинства IDE для Java
- JUnit запускает пакеты тестов и выдаёт отчёт о результатах
- Для каждого теста выполняет действия:
  - setUp() – настройка окружения
  - запуск тестового метода
    - тестовый метод проверяет какой-то аспект поведения: вызывает тестируемый метод класса и проверяет утверждения относительно результатов
  - tearDown() – уничтожение окружения

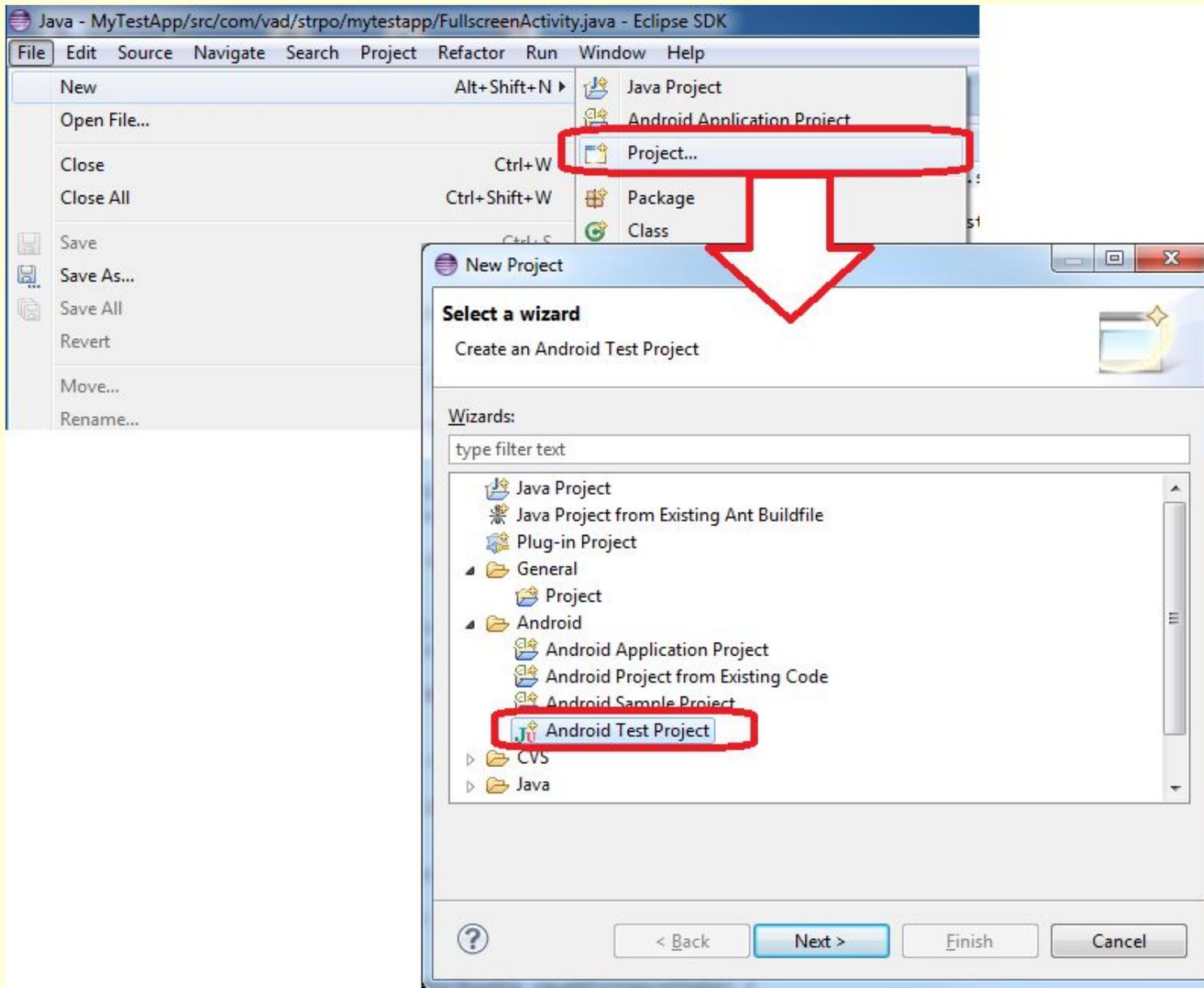


# Проект с тестами

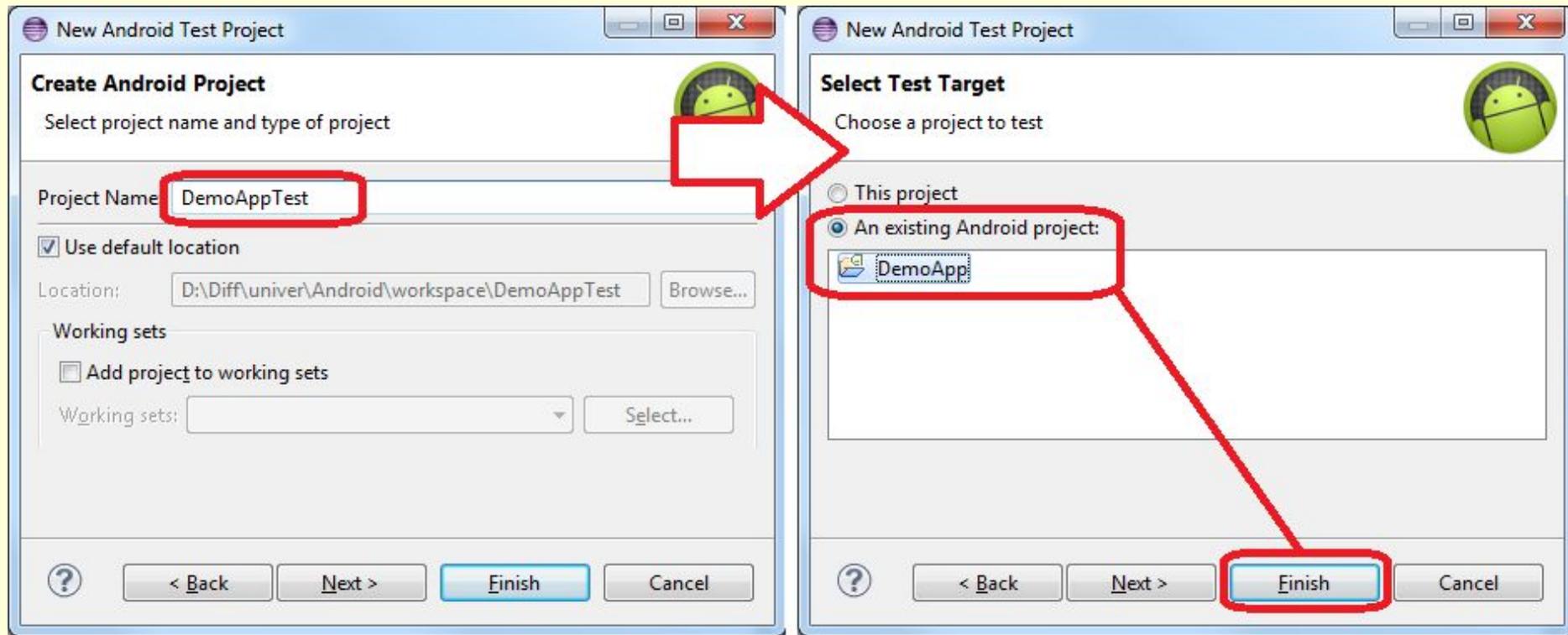
- Создаём новый проект
  - “Java Project” / “Android Test Project”
- Определяем зависимость от тестируемого проекта
- Создаём модули с тестами
  - Best practices: параллельная структура пакетов (дублирует структуру тестируемого проекта)



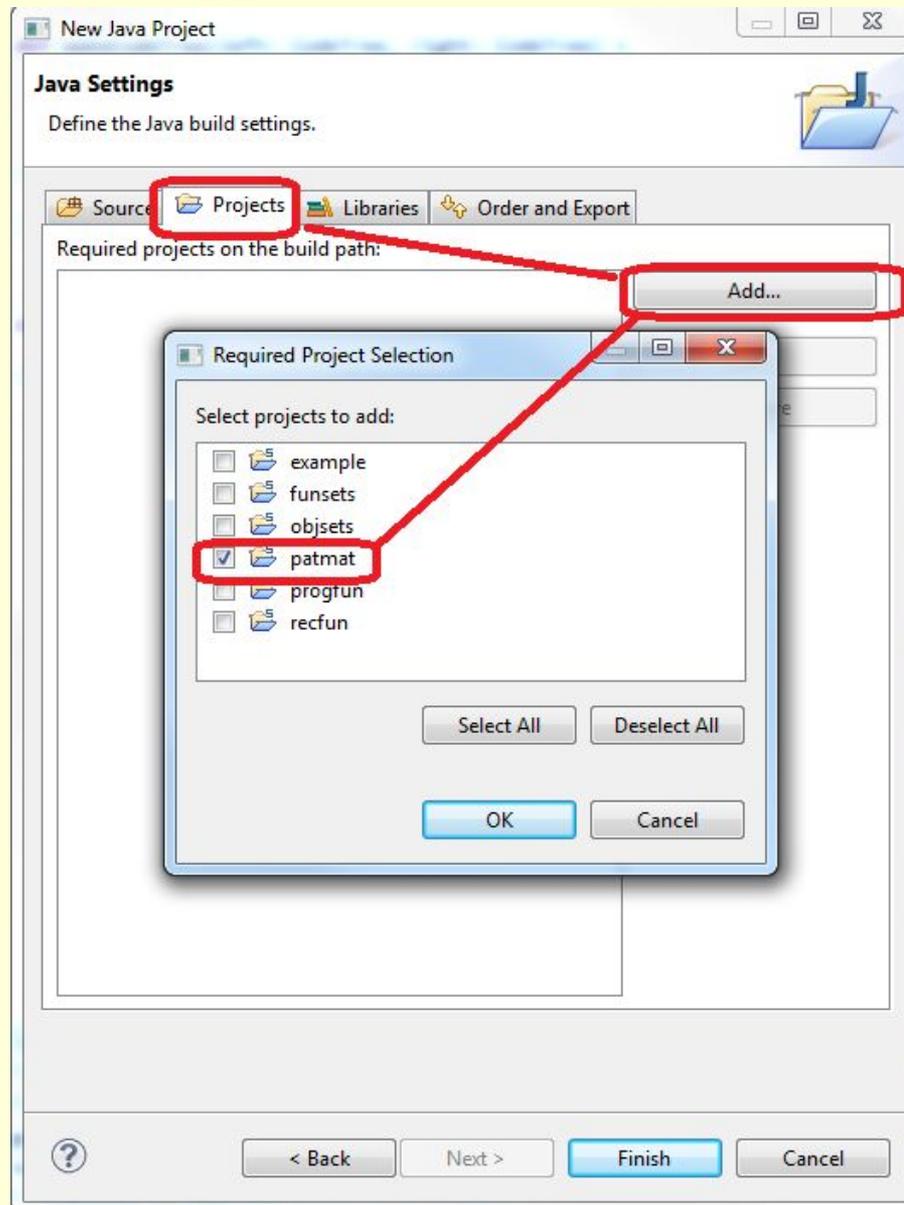
# Проект с тестами: Android



# Проект с тестами: Android



# Проект с тестами: Java Project



# Модули с тестами

- Классы, наследуемые от TestCase:
  - setUp() / tearDown()
  - public testXXX() – методы с тестами
    - наборы asserts – проверочных утверждений
- TestSuite – набор TestCases:
  - static suite() – фабричный метод для набора тестов
    - включает группу TestCase-ов и отдельных методов testXXX()
- Опционально – метод main() для запуска пакета тестов



# Создание нового тестового модуля

The image illustrates the steps to create a new JUnit test case in an IDE. On the left, the project structure shows the package `my.example.demoapp.test` selected. A context menu is open, with the **New** option highlighted. The **JUnit Test Case** option is also highlighted. A red arrow points from this option to the **New JUnit Test Case** dialog box on the right.

The **New JUnit Test Case** dialog box contains the following information:

- 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.
- Radio buttons**:  New JUnit 3 test,  New JUnit 4 test
- Source folder**: DemoAppTest/src (Browse...)
- Package**: my.example.demoapp.test (Browse...)
- Name**: DemoBotTest
- Superclass**: java.lang.Object (Browse...)
- 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**: my.example.demoapp.DemoBot (Browse...)

At the bottom of the dialog, there are buttons for **< Back**, **Next >**, **Finish**, and **Cancel**.



# Создание нового тестового модуля

```
DemoBot.java DemoBotTest.java
package my.example.demoapp;

import java.util.Locale;

public class DemoBot {
    public String talkTo(String messageToBot)
    {
        if (messageToBot.toLowerCase(Locale.getDefault()).contains("Hello")) {
            return new String("Hi!");
        }
        else {
            return new String("Well...");
        }
    }
}

DemoBot.java *DemoBotTest.java
package my.example.demoapp.test;

import static org.junit.Assert.*;

public class DemoBotTest {

    @Test
    public void test() {
        fail("Not yet implemented");
    }
}
```

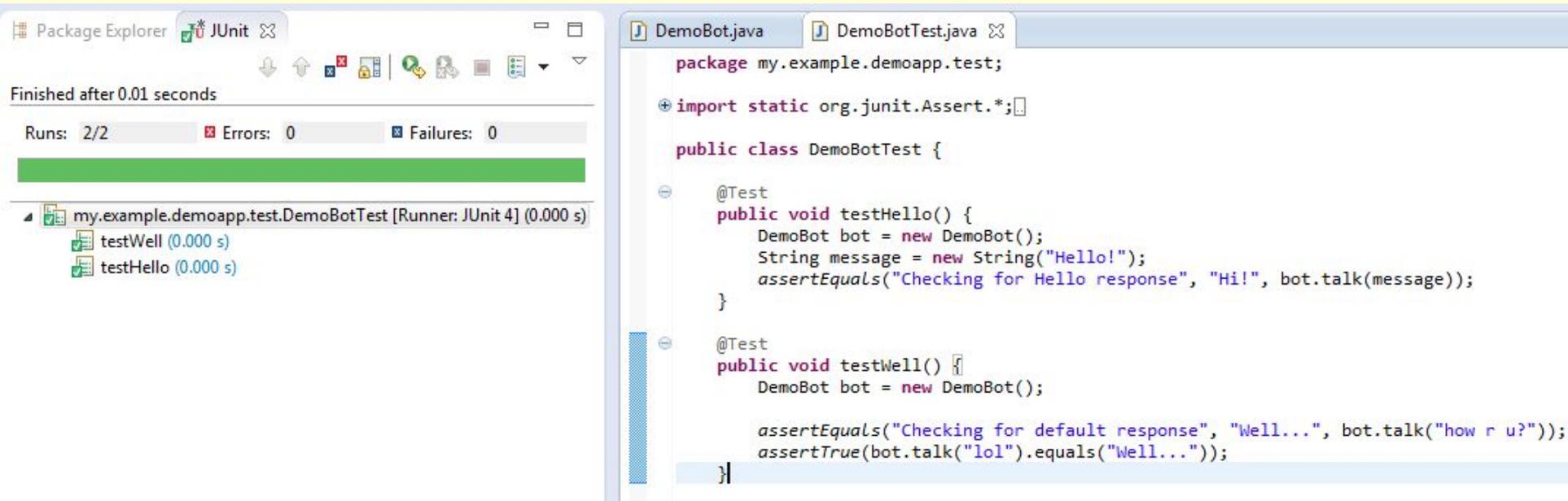
# Виды проверочных утверждений

- `assertXXX`:
  - `assertTrue(boolean test)`
  - `assertFalse(boolean test)`
  - `assertEquals(expected, actual)`
  - `assertSame(Object expected, Object actual)`
  - `assertNotSame(Object expected, Object actual)`
  - `assertNull(Object object)`
  - `assertNotNull(Object object)`
- `fail()`
  - провал теста(генерирует `AssertionFailedError`)
- Все вышеуказанные методы в варианте с текстовым сообщением
  - напр., `assertTrue(String message, boolean test)`



# Тестовые методы

- “Run As...->JUnit Test”
- формируется отчёт
  - «красный» - есть не прошедшие тесты
  - «зелёный» - все тесты пройдены



The screenshot displays an IDE interface. On the left, the Package Explorer shows a JUnit test runner. The main window shows the source code for DemoBotTest.java, which includes two test methods: testHello() and testWell().

JUnit Test Results:

- Finished after 0.01 seconds
- Runs: 2/2
- Errors: 0
- Failures: 0

Test Results:

- my.example.demoapp.test.DemoBotTest [Runner: JUnit 4] (0.000 s)
  - testWell (0.000 s)
  - testHello (0.000 s)

Source Code (DemoBotTest.java):

```
package my.example.demoapp.test;

import static org.junit.Assert.*;

public class DemoBotTest {

    @Test
    public void testHello() {
        DemoBot bot = new DemoBot();
        String message = new String("Hello!");
        assertEquals("Checking for Hello response", "Hi!", bot.talk(message));
    }

    @Test
    public void testWell() {
        DemoBot bot = new DemoBot();

        assertEquals("Checking for default response", "Well...", bot.talk("how r u?"));
        assertTrue(bot.talk("lol").equals("Well..."));
    }
}
```



# Fixtures

- Позволяют снизить дублирование кода
- Если для набора тестов нужны общие «предустановки», можно поместить их в метод `setUp`
  - в предыдущем примере, можно было бы сделать `DemoBot` полем класса `DemoBotTest` и инициализировать его в `setUp()`
    - чтобы к каждому тесту был создан «свежий» объект для тестирования

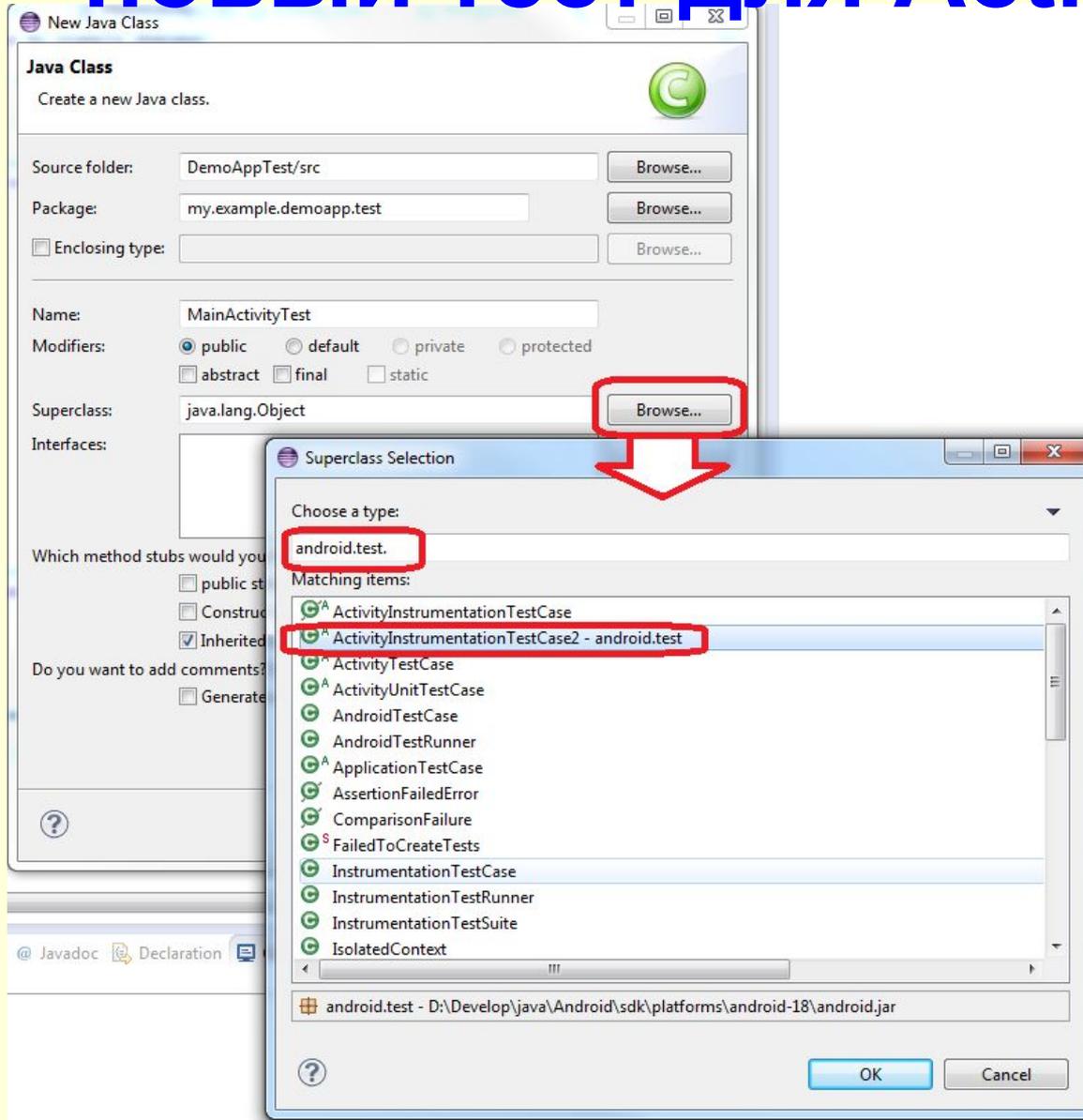


# Тестирование GUI в Android средствами JUnit + android.test

- android.test – фреймворк для написания специальных Android-тестов
  - GUI-тесты Android-компонент
    - большой набор assert-методов (см. документацию)
    - средства для изолированного тестирования GUI-классов (mock-и Android-компонент)
- Можно автоматизировать действия над интерфейсом и проверять:
  - видны ли элементы GUI, какие значения они содержат, как расположены и т.п.



# android.test: НОВЫЙ ТЕСТ для Activity



# android.test:

## запуск тестов

“Run As” -> “Android JUnit Test”

The screenshot shows the Eclipse IDE interface. The top toolbar includes a 'Run As' button with a JUnit icon. The Package Explorer on the left shows a test run for 'my.example.demoapp.test.MainActivityTest' with a green progress bar and a summary of 4/4 runs, 0 errors, and 0 failures. The main editor displays the source code for MainActivityTest.java, which includes assertions for the initial state and a test method 'testChat()' that simulates a button click and verifies the chat view content.

```
ViewAsserts.assertOnScreen(input.getRootView(), input);
ViewAsserts.assertOnScreen(chatView.getRootView(), chatView);
ViewAsserts.assertOnScreen(button.getRootView(), button);
}

public void testStartingEmpty() {
    assertEquals("Initial value of input is empty", "", input.getText().toString());
    assertEquals("Chat view contains '>', '>', chatView.getText().toString());
}

public void testChat() {
    activity.runOnUiThread(new Runnable() {
        public void run() {
            input.requestFocus();
        }
    });

    instrumentation.waitForIdleSync();
    instrumentation.sendStringSync("Hello!");
    instrumentation.waitForIdleSync();

    TouchUtils.clickView(this, button);

    assertTrue(chatView.getText().toString().contains(">Hello!\n"));
}
}
```



# Ссылки

- JUnit
  - документация: <https://github.com/junit-team/junit/wiki>
  - статья с примерами: <http://habrahabr.ru/post/120101/>
- Android Test:
  - Документация: <http://developer.android.com/intl/ru/reference/android/test/package-summary.html>
    - создание функциональных тестов: <http://developer.android.com/intl/ru/training/activity-testing/activity-functional-testing.html>
  - Тестирование Android-приложений: <http://habrahabr.ru/post/113584/>

