

JUNIT

BY ANJUM ARA

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
 - Junit test generators now part of many Java IDEs (Eclipse, BlueJ, Jbuilder, DrJava)
 - Xunit tools have since been developed for many other languages (Perl, C++, Python, Visual Basic, C#, ...)
-

UNIT TESTING

- Unit Testing is one of the phase of software development process in which units(smallest part of the application) are separately examined for proper operation.
 - Generally programmers and occasionally white box testers created unit tests.
 - Its implementation can vary from being very manual (pencil and paper) to being formalized as part of build automation.
-

TESTS THAT ARE PERFORMED DURING THE UNIT TESTING ARE EXPLAINED BELOW

- **Module Interface test:** In module interface test, it is checked whether the information is properly flowing in to the program unit (or module) and properly happen out of it or not.
 - **Local data structures:** These are tested to inquiry if the local data within the module is stored properly or not.
 - **Boundary conditions:** It is observed that much software often fails at boundary related conditions. That's why boundary related conditions are always tested to make safe that the program is properly working at its boundary condition's.
 - **Independent paths:** All independent paths are tested to see that they are properly executing their task and terminating at the end of the program.
 - **Error handling paths:** These are tested to review if errors are handled properly by them or not.
-

FAMOUS JAVA UNIT TESTING TOOLS

- JMockit
 - Artima SuiteRunner
 - Cactus
 - Checkstyle
 - Cobertura
 - Continuous Testing
 - Daedalos JUnit Extensions
 - Dbunit
 - DDSteps
 - DepUnit
 - djUnit
 - Dumpster
 - EasyMock + ClassExtension
 - EclEmma
 - Ejb3unit
 - Emma
 - Feed4JUnit
 - Fest
 - Findbugs
 - Gretel
 - GroboUtils
 - Hansel 1.0
 - Haste
 - J2ME Unit Testing Toolkit
 - Jacareto
 - Jete
 - Jetif
 - JFCUnit
 - Jiffie
 - jMock
 - JOSIT
 - JsTester
 - JSystem
-

FAMOUS JAVA UNIT TESTING TOOLS

- JUnitPerf
- JXUnit
- Macker
- Mockito
- MockObjects
- Mockrunner
- MoMEUnit
- moreUnit
- NoUnit
- Pisces
- PMD
- Quilt
- Slim
- Sonar
- StrutsTestCase for JUnit v1.9.5
- T2
- TagUnit
- Testare
- TestGen4J
- TestNG
- UISpec4J
- Unitils
- XHTMLUnit
- Xtest
- Fest
- Findbugs
- Gretel
- GroboUtils
- Hansel 1.0
- Haste
- J2ME Unit Testing Toolkit
- Jailer
- JDepend
- JEasyTest
- JellyUnit
- Jester
- JTestCase
- Jumble
- JUnit
- JUnit-addons
- JUnitDoclet
- JUnitEE

JUNIT

- It is part of unit testing frameworks family collectively known as xUnit. Other examples are CppUnit, PHPUnit, JSUnit etc.
 - JUnit is linked as a JAR at compile-time; the framework resides under `packagesjunit.framework` for JUnit 3.8 and earlier and under `org.junit` for JUnit 4 and later.
 - JUnit has now become the standard tool for Test-Driven Development in Java.
 - JUnit test generators now part of many Java IDEs (Eclipse, BlueJ, Jbuilder, DrJava etc).
-

JUNIT (CONTD...)

- JUnit provides Annotation to identify the test methods and utility methods to assert expected results.
 - JUnit has Test runners for running tests.
 - JUnit tests can be organized into test suites which contains test cases and other test suites.
 - JUnit Framework can be easily integrated with either of the followings:
 - Eclipse
 - Ant
 - Maven
-

TESTING VIA JUNIT

- Can test JUnit by using Test Case & Test Suit .
- **Test Case:** A Test Case is a java class which is used to test a single class.
- **Test Suite :** A Test Suit is a java class which contains many test cases or java classes to test together in a single run.

BENEFITS

- Simple framework for writing automated, self-verifying tests in Java
 - Support for test assertions
 - Test suite development
 - Immediate test reporting
-

FEATURES

- Annotation to identify the test methods(After JUnit 4).
 - Assertions for testing expected results.
 - Test fixtures for sharing common test data.
 - Test runners for running tests.
 - Aggregating tests using suites.
-

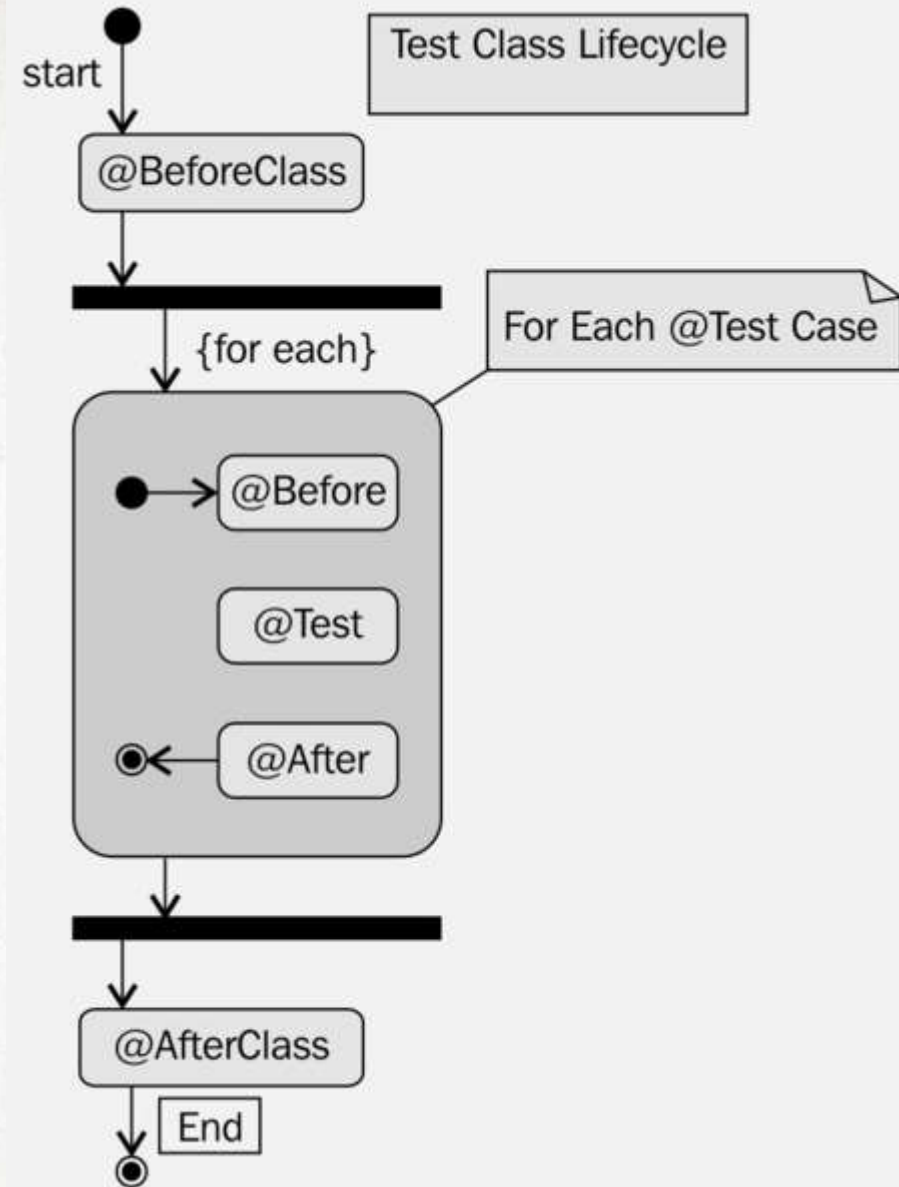
NEED OF JUNIT

- JUnit tests allow you to write code faster while increasing quality.
 - JUnit is elegantly simple. It is less complex & takes less time.
 - JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results.
 - JUnit tests can be organized into test suites containing test cases and even other test suites.
 - Writing JUnit tests is inexpensive.
-

JUNIT ANNOTATIONS

- **@Test** : The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case.
- **@Before** : Several tests need similar objects created before they can run. Annotating a public void method with @Before causes that method to be run before each Test method.
- **@After** : If you allocate external resources in a Before method you need to release them after the test runs. Annotating a public void method with @After causes that method to be run after the Test method.
- **@BeforeClass** : Annotating a public static void method with @BeforeClass causes it to be run once before any of the test methods in the class. This can be used to perform computationally expensive setup (like logging into a database).
- **@AfterClass** : Will perform the method after all tests have finished. This can be used to perform clean-up activities for example be used to disconnect to a database
- **@Ignore** : Sometimes you want to temporarily disable a test or a group of tests. Methods annotated with Test that are also annotated with @Ignore will not be executed as tests. Also, you can annotate a class containing test methods with @Ignore and none of the containing tests will be executed.

JUNIT LIFE CYCLE



ASSERTION

- All the assertion are in the Assert class.

```
public class Assert extends java.lang.Object
```

- This class provides a set of assertion methods useful for writing tests.
 - Only failed assertions are recorded.
-

ASSERTION METHODS

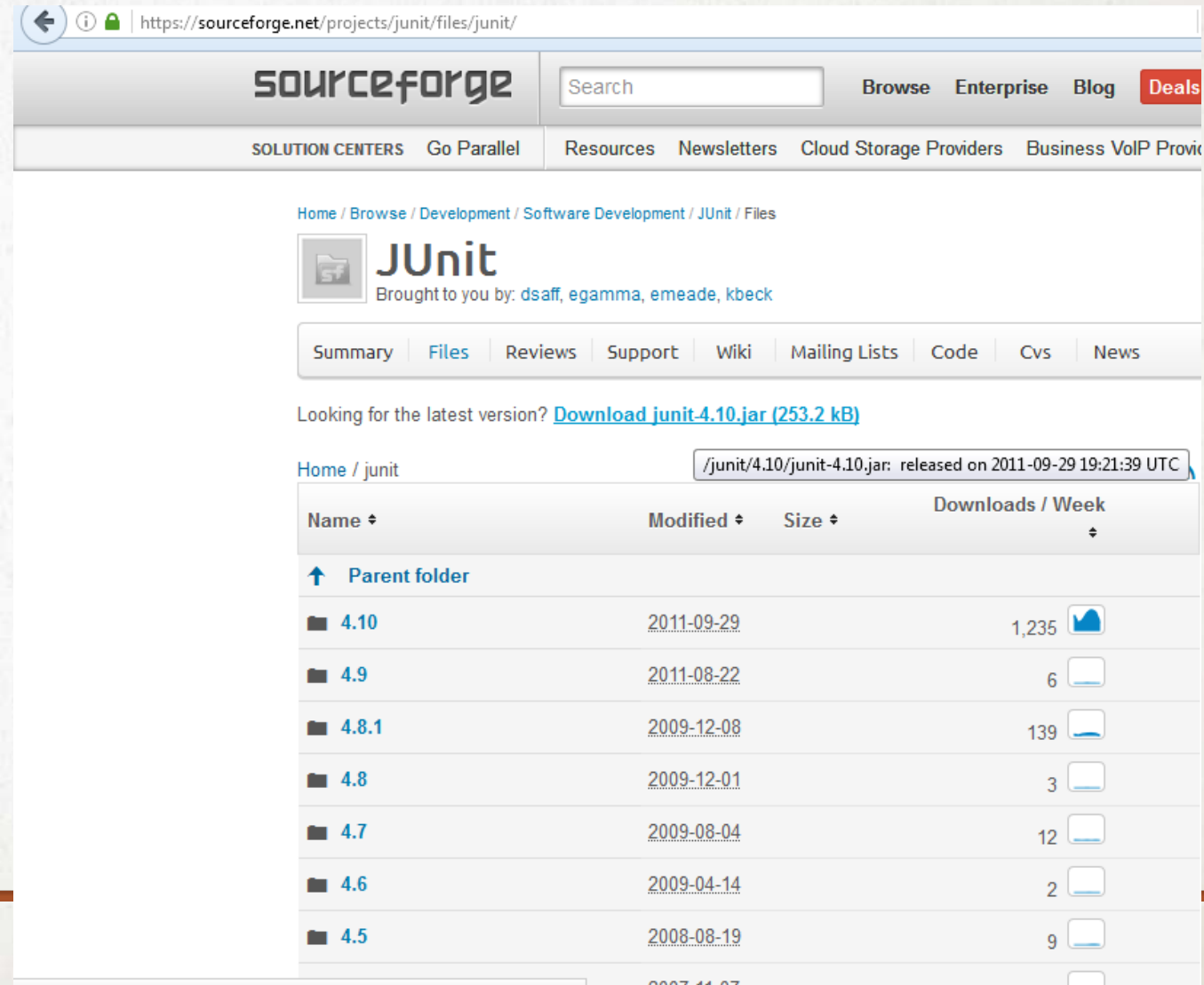
- **void assertEquals(boolean expected, boolean actual)** Check that two primitives/Objects are equal
 - **void assertTrue(boolean expected, boolean actual)** Check that a condition is true
 - **void assertFalse(boolean condition)** Check that a condition is false
 - **void assertNotNull(Object object)** Check that an object isn't null.
 - **void assertNull(Object object)** Check that an object is null
-

ASSERTION METHODS (CONTD...)

- **void assertSame(boolean condition)** The `assertSame()` method tests if two object references point to the same object
 - **void assertNotSame(boolean condition)** The `assertNotSame()` method tests if two object references not point to the same object
 - **void assertEquals(expectedArray, resultArray);** The `assertEquals()` method will test whether two arrays are equal to each other.
 - **fail(String message)** Fails a test with the given message.
 - **assertEquals(String message, expected array, actual array)** Asserts that two byte arrays are equal. If they are not, an `AssertionError` is thrown with the given message.
-

JUNIT SET UP

- download the latest version of JUnit, referred as junit4.x.x.zip from given below link :
- <http://sourceforge.net/projects/junit/files/>



The screenshot displays the SourceForge project page for JUnit. The page includes a search bar, navigation links, and a list of available versions. The latest version, 4.10, is highlighted with a download icon and a download count of 1,235.

SourceForge

Search

Browse Enterprise Blog Deals

SOLUTION CENTERS Go Parallel Resources Newsletters Cloud Storage Providers Business VoIP Providers

Home / Browse / Development / Software Development / JUnit / Files

JUnit

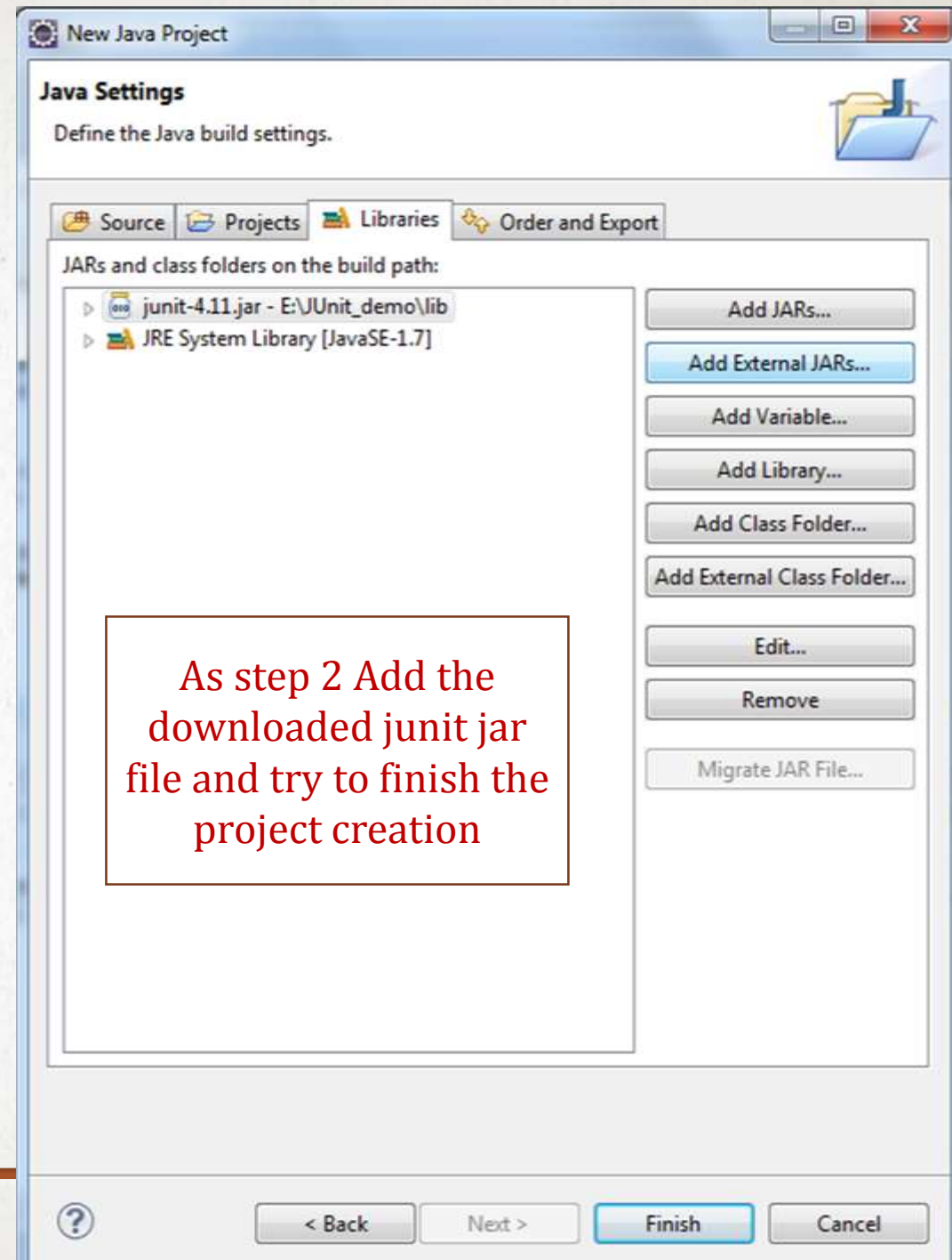
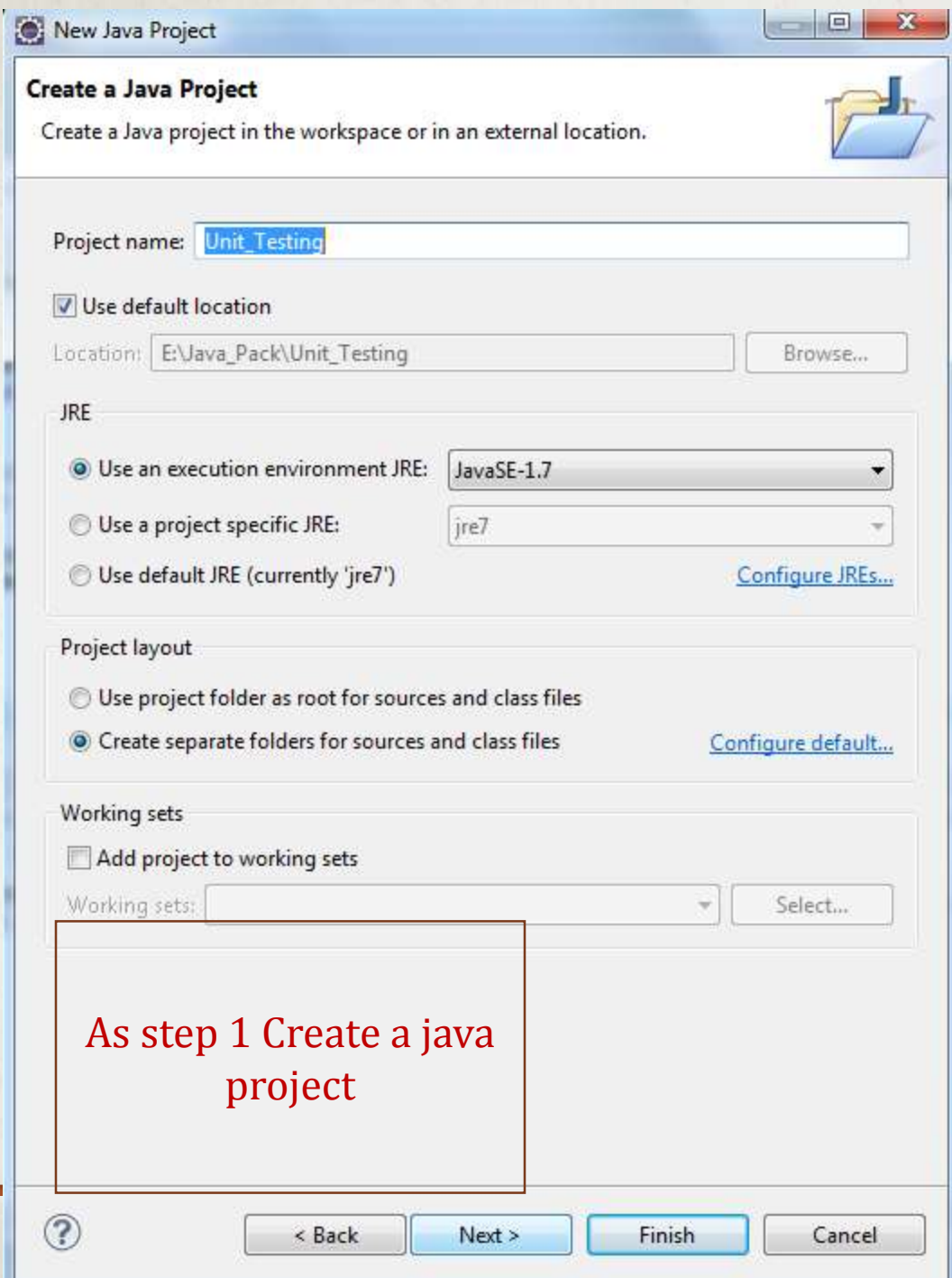
Brought to you by: dsaff, egamma, emeade, kbeck

Summary Files Reviews Support Wiki Mailing Lists Code Cvs News

Looking for the latest version? [Download junit-4.10.jar \(253.2 kB\)](#)

Home / junit /junit/4.10/junit-4.10.jar: released on 2011-09-29 19:21:39 UTC

Name	Modified	Size	Downloads / Week
Parent folder			
4.10	2011-09-29		1,235
4.9	2011-08-22		6
4.8.1	2009-12-08		139
4.8	2009-12-01		3
4.7	2009-08-04		12
4.6	2009-04-14		2
4.5	2008-08-19		9



JAR FILES USED IN THE MENTIONED SAMPLES



hamcrest-core-1.3.jar



junit-4.11.jar

API'S OF JUNIT

Class Name	Functionality
Assert	A set of assert methods.
TestCase	A test case defines the fixture to run multiple tests.
TestResult	A TestResult collects the results of executing a test case.
TestSuite	A TestSuite is a Composite of Tests.

@TEST SAMPLE

```
package Test;
```

```
public class Pet {  
    public Pet() {}
```

```
    public String meow() {  
        return "Meow";  
    }  
}
```

```
package Test;
```

```
import static org.junit.Assert.*;  
import org.junit.Test;  
import Test.Pet;
```

```
public class PetTest {  
    @Test  
    public void testPet() {  
    }
```

```
    @Test  
    public void testMeow()  
    {  
        Pet testPet = new Pet();  
        assertTrue("Meow".equals(testPet.meow()));  
    }  
}
```



Package Explorer

Java_Samples
Unit_Testing
src
Test
Message
Pet.java
PetTest.java
TestC
TestC
TestC
TestC
TestR
TestS
Unit.ja
UnitT
JRE System L
Referenced L

- New
- Open F3
- Open With
- Open Type Hierarchy F4
- Show In Alt+Shift+W
- Copy Ctrl+C
- Copy Qualified Name
- Paste Ctrl+V
- Delete Delete
- Remove from Context Ctrl+Alt+Shift+Down
- Build Path
- Source Alt+Shift+S
- Refactor Alt+Shift+T
- Import...
- Export...
- References
- Declarations
- Refresh F5
- Assign Working Sets...
- Run As
 - 1 Run on Server Alt+Shift+X, R
 - 2 JUnit Test Alt+Shift+X, T
 - Run Configurations...
- Debug As
- Profile As
- Validate
- Team
- Compare With
- Replace With
- Restore from Local History...
- Web Services
- Properties Alt+Enter



Quick Access

Java EE Java

Task List

Find All Activate...

Connect Mylyn

Connect to your task and ALM tools or
create a local task.

Outline

Test
Pet
Pet()
meow() : String

Declaration Console

Program Files\Java\jre7\bin\javaw.exe (Jul 28, 2016 6:22:48 PM)



Quick Access

Java EE Java

Package Explorer JUnit



Finished after 0.015 seconds

Runs: 2/2 Errors: 0 Failures: 0

Test.PetTest [Runner: JUnit 4] (0.000 s)

- testPet (0.000 s)
- testMeaow (0.000 s)

Pet.java PetTest.java

```
package Test;

import static org.junit.Assert.*;
import org.junit.Test;
import Test.Pet;

public class PetTest {
    @Test
    public void testPet() {
    }

    @Test
    public void testMeaow()
    {
        Pet testPet = new Pet();
        assertTrue("Meaow".equals(testPet.meow()));
    }
}
```

Task List



Find

All Activate...

Connect Mylyn

Connect to your task and ALM tools
or create a local task.

Outline



- Test
 - PetTest
 - testPet() : void
 - testMeaow() : void

Failure Trace



Problems @ Javadoc Declaration Console



<terminated> PetTest [JUnit] C:\Program Files\Java\jre7\bin\javaw.exe (Jul 28, 2016 6:28:06 PM)

CREATE A JAVA CLASS & TEST CLASS

- Create a class TestSamples.java

```
package TestSamples;
```

```
public class TestSample {  
    public int multiply(int x, int y) {  
        return x * y;  
    }  
}
```

- Right click on TestSamples.java, select New ->others-->(Inside JUnit folder)JUnit Test case and select Next button, the following window appears

New JUnit Test Case

JUnit Test Case

⚠ This package name is discouraged. By convention, package names usually start with a lowercase letter

☐ New JUnit 3 test ☒ New JUnit 4 test

Source folder:

Package:

Name:

Superclass:

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:

Click Next

?

New JUnit Test Case

Test Methods

Select methods for which test method stubs should be created.

Available methods:

- ✓ ☒ **TestSample**
- ✓ ☒ multiply(int, int)
- ✗ ☐ Object
 - ✗ ☐ Object()
 - ✗ ☐ getClass()
 - ✗ ☐ hashCode()
 - ✗ ☐ equals(Object)
 - ✗ ☐ clone()
 - ✗ ☐ toString()
 - ✗ ☐ notify()
 - ✗ ☐ notifyAll()
 - ✗ ☐ wait(long)
 - ✗ ☐ wait(long, int)
 - ✗ ☐ wait()
 - ✗ ☐ finalize()

1 method selected.

☐ Create final method stubs
☐ Create tasks for generated test methods

Select the check box
TestSample and click
Finish

?

- On clicking finish a new file named TestSampleTest.java gets created automatically and is as follows

```
package TestSamples;

import static org.junit.Assert.*;

import org.junit.Test;

public class TestSampleTest {

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

}
```

- Now edit the stub so as to make it assertive.

- After editing the stub, now the class file and its test case looks something like this

```
package TestSamples;

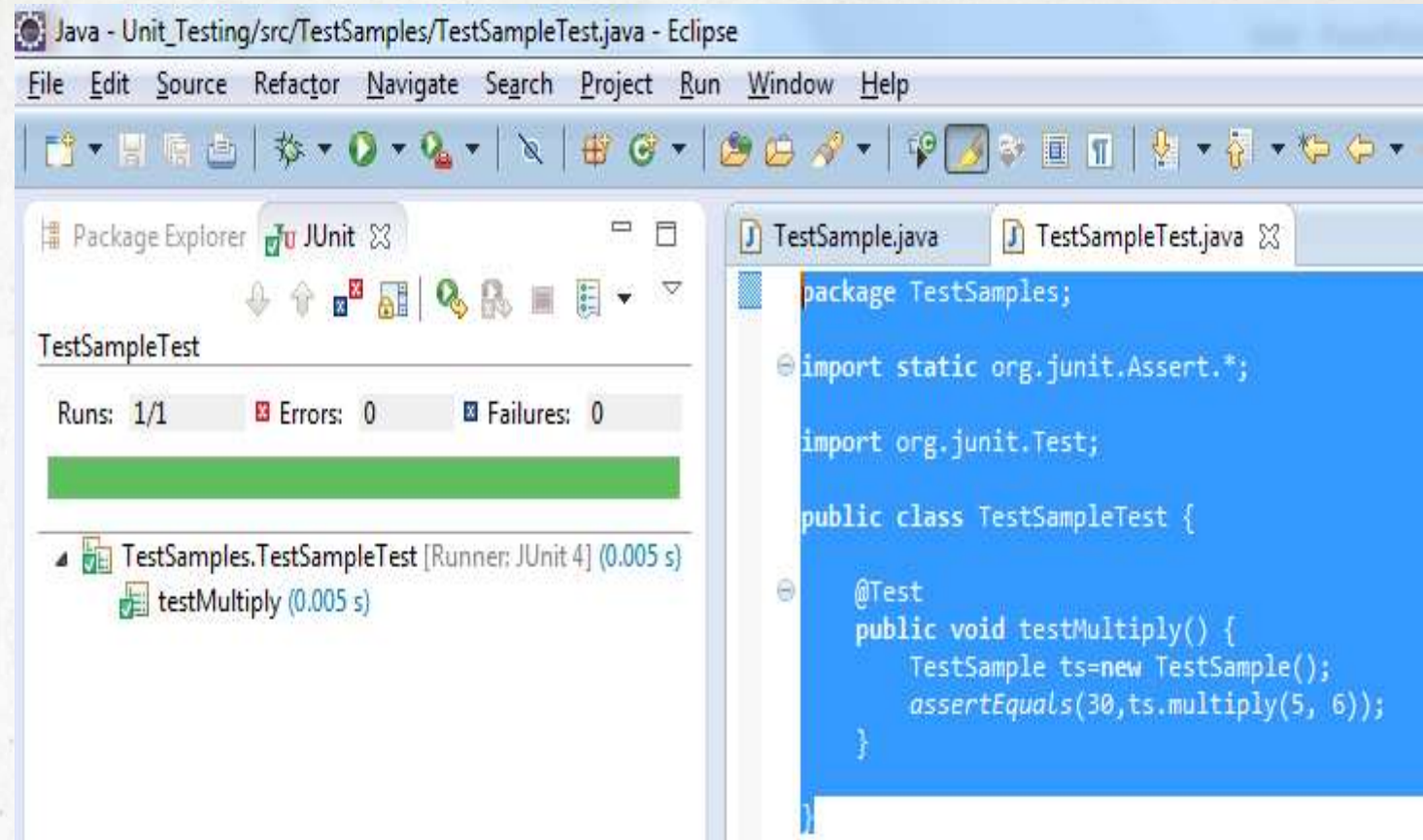
import static org.junit.Assert.*;

import org.junit.Test;

public class TestSampleTest {

    @Test
    public void testMultiply() {
        TestSample ts=new TestSample();
        assertEquals(30,ts.multiply(5, 6));
    }

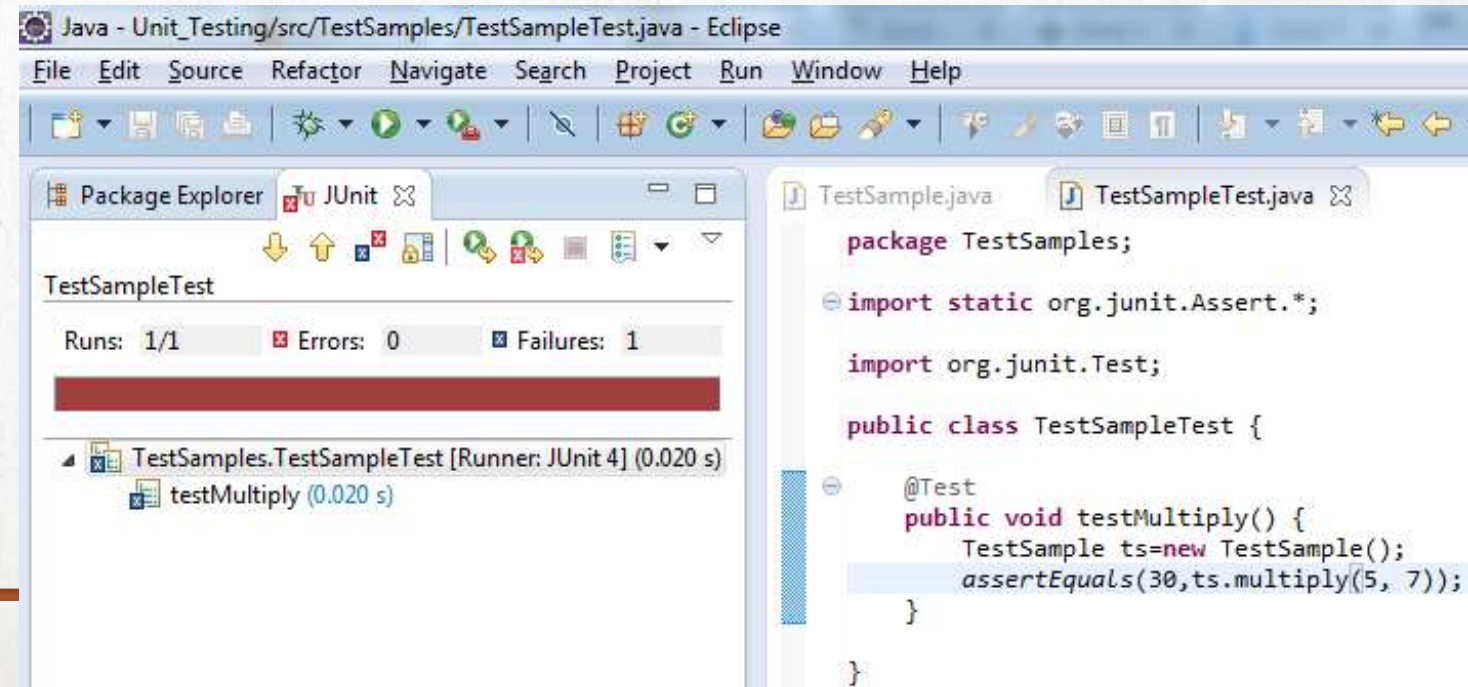
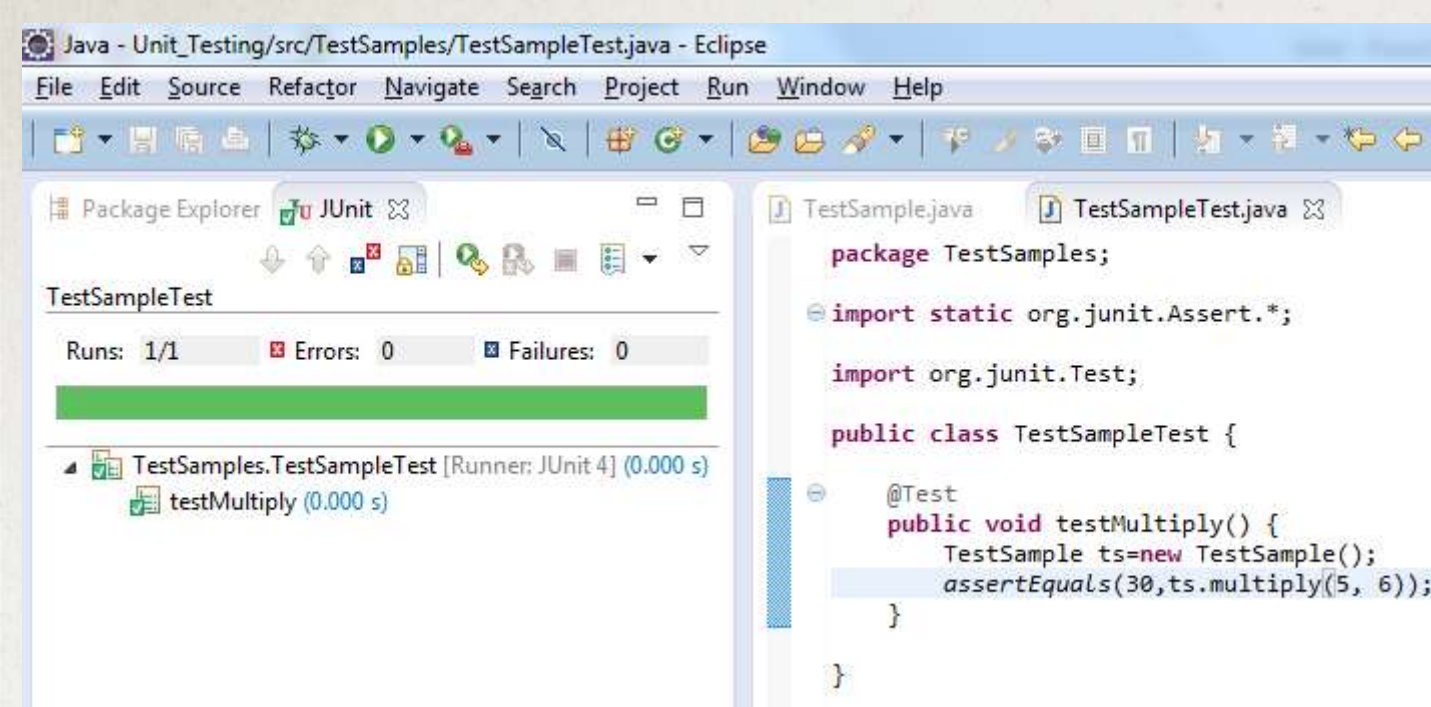
}
```



EXECUTING A JUNIT TEST

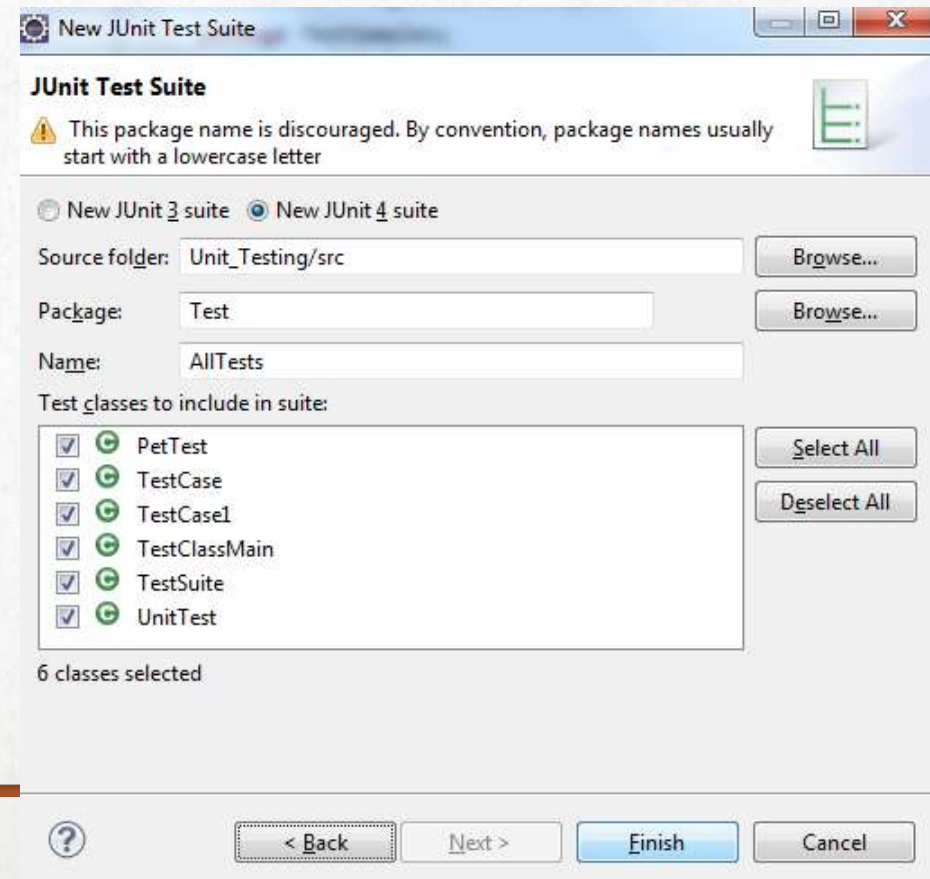
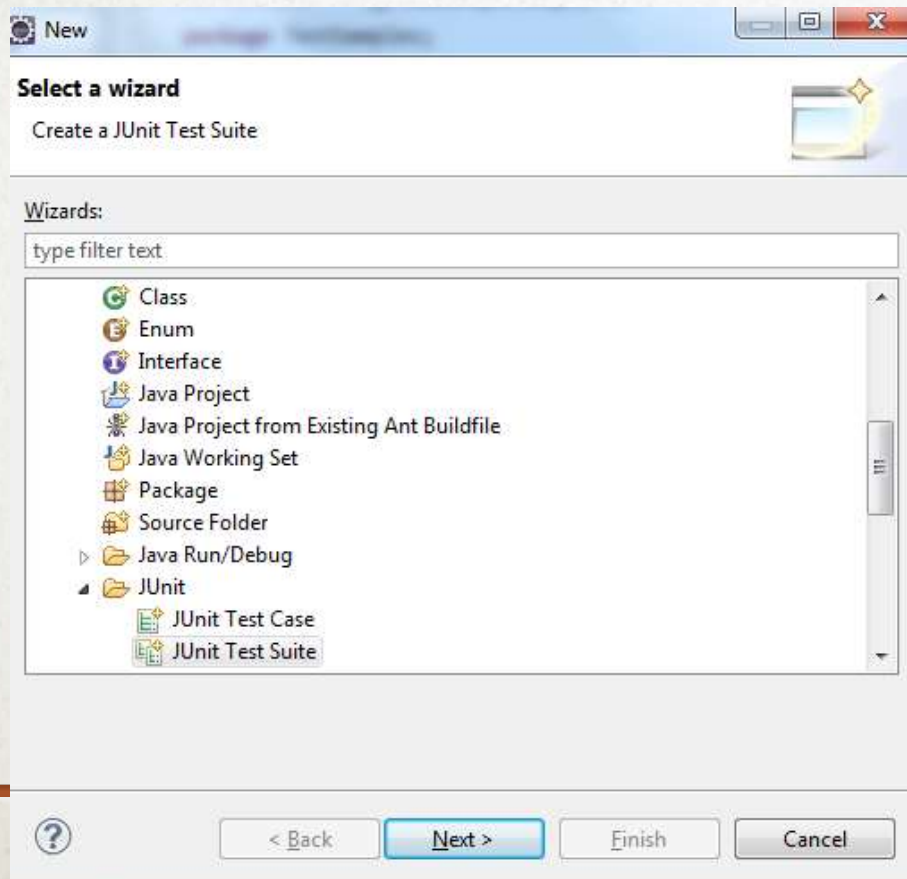
- Right click on your new test class and select Run-As-> Junit Test.
 - If the output shows green flag, your test is successful.
 - If it shows red, it means class have some failures or errors.
 - It will also show you the methods which has errors and the Failure Trace
-

- Both failure and successful scenarios are displayed here for the same



CREATING A TEST SUITE

- Multiple tests can be combined into a test suite. All test in this test suite will then be executed if you run the test suite.
- To create a new test suite, select your package, right mouse click->, select New->Others->JUnit ->JUnit Test Suite->next->Finish.



CREATING A TEST SUITE(CONTD...)

- As a result the following TestSuite is created

```
package Test;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({ PetTest.class, TestCase.class, TestCase1.class,
TestClassMain.class, TestSuite.class, UnitTest.class })
public class AllTests {

}
```




Quick Access

Java EE Java

Package Explorer JUnit



Finished after 0.025 seconds

Runs: 8/8 Errors: 0 Failures: 0

Test.AllTests [Runner: JUnit 4] (0.005 s)

- Test.PetTest (0.000 s)
- Test.TestCase
- Test.TestCase1
- Test.TestClassMain (0.000 s)
- Test.TestSuite (0.000 s)
- Test.UnitTest (0.005 s)

AllTests.java

```
package Test;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({ PetTest.class, TestCase.class, TestCase1.class,
    TestClassMain.class, TestSuite.class, UnitTest.class })
public class AllTests {

}
```

Task List

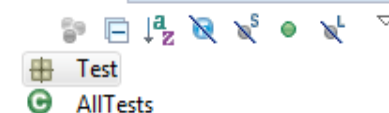


Find All Activate...

Connect Mylyn

Connect to your task and ALM tools
or create a local task.

Outline



Failure Trace



Problems Javadoc Declaration Console



<terminated> AllTests [JUnit] C:\Program Files\Java\jre7\bin\javaw.exe (Jul 28, 2016 8:02:24 PM)

```
Inside testPrintMessage()
Robert
Inside testSalutationMessage()
Hi!Robert
Inside testPrintMessage()
Robert
Inside testSalutationMessage()
Hi!Robert
```

TESTCASE CLASS

- org.junit.TestCase class is declared as follows.

```
public abstract class TestCase extends Assert implements Test
```

- A test case defines the fixture to run multiple tests.

FREQUENTLY USED METHODS FROM TESTCASE CLASS

- **int countTestCases()** Counts the number of test cases.
 - **TestResult createResult()** Creates a default TestResult object.
 - **String getName()** Gets the name of a TestCase.
 - **TestResult run()** A convenience method to run this test.
 - **void run(TestResult result)** Runs the test case and collects the results in TestResult.
 - **void setName(String name)** Sets the name of a TestCase.
 - **void setUp()** Sets up the fixture, for example, open a database connection.
 - **void tearDown()** Tears down the fixture, for example, close a database connection.
 - **String toString()** Returns a string representation of the test case.
-

```
package TestSamples;
```

TESTCASE CLASS EXAMPLE

```
import junit.framework.TestCase;
```

```
import org.junit.Before;
```

```
import org.junit.Test;
```

```
public class TestJunit extends TestCase {
```

```
    protected double fValue1;
```

```
    protected double fValue2;
```

```
    @Before
```

```
    public void setUp() {
```

```
        System.out.println("setup");
```

```
        fValue1= 8.0;
```

```
        fValue2= 9.0;
```

```
    }
```

```
    @Test
```

```
    public void testCaseAdd() {
```

```
        //count the number of test cases
```

```
        System.out.println("No of Test Case = "+ this.countTestCases());
```

```
        //test getName
```

```
        String name= this.getName();
```

```
        System.out.println("Test Case Name = "+ name);
```

```
        //test setName
```

```
        this.setName("testCaseAddNew");
```

```
        String newName= this.getName();
```

```
        System.out.println("Updated Test Case Name = "+ newName);
```

```
        assertEquals(5.0, fValue1+fValue2);
```

```
    }}
```



```
package TestSamples;

import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class Main {
    public static void main(String[] args) {
        Result result = JUnitCore.runClasses(TestJUnit.class);
        for (Failure failure : result.getFailures()) {
            System.out.println(failure.toString());
        }
        System.out.println(result.wasSuccessful());
    }
}
```

Output:

setup

No of Test Case = 1

Test Case Name = testCaseAdd

Updated Test Case Name = testCaseAddNew

testCaseAddNew(TestSamples.TestJUnit): expected:<5.0> but was:<17.0>

false

JUNIT TESTRESULT

- TestResult collects the results of executing a test case.

TestResult Class

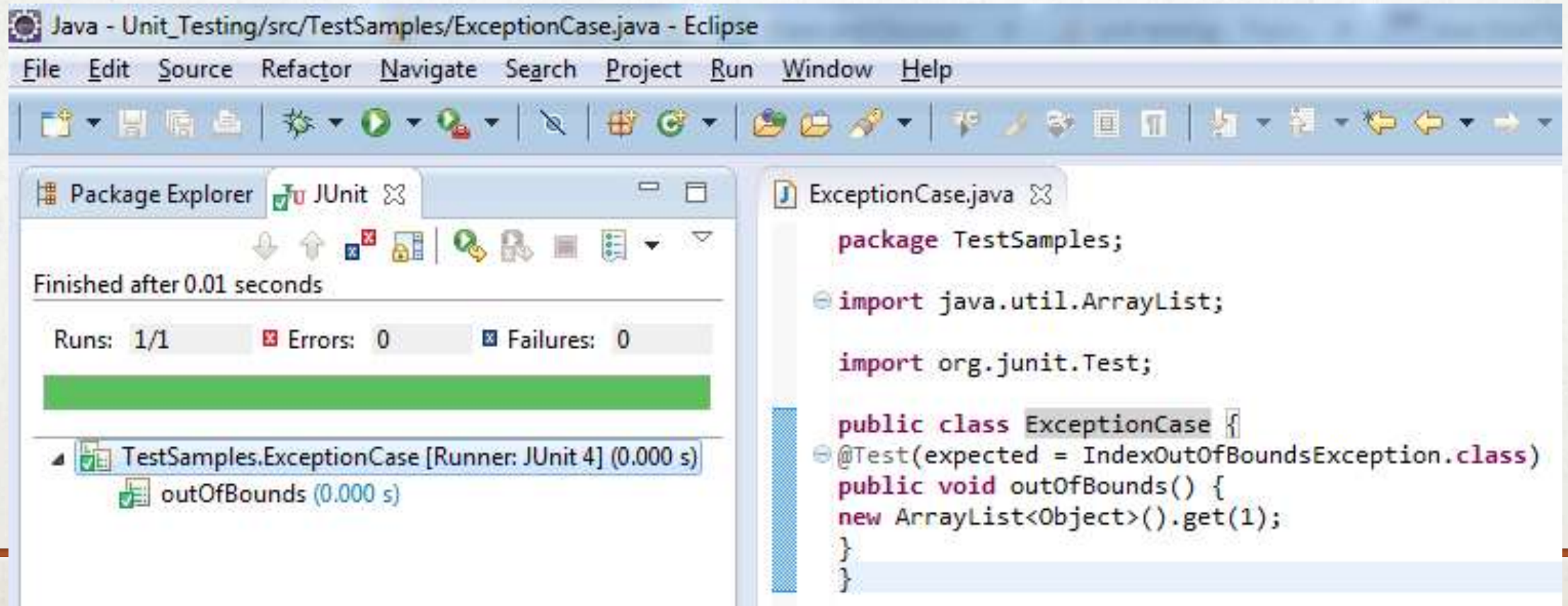
- The declaration for org.junit.TestResult class: `public class TestResult extends Object`
 - A failure is anticipated and checked for with assertions.
 - Errors are unanticipated problems such as `ArrayIndexOutOfBoundsException`.
-

FREQUENTLY USED METHODS FROM TESTRESULT CLASS.

- **void addError(Test test, Throwable t)** Adds an error to the list of errors.
 - **void addFailure(Test test, AssertionError t)** Adds a failure to the list of failures.
 - **void endTest(Test test)** Informs the result that a test was completed.
 - **int errorCount()** Gets the number of detected errors.
 - **Enumeration<TestFailure> errors()** Returns an Enumeration for the errors.
 - **int failureCount()** Gets the number of detected failures.
 - **void run(TestCase test)** Runs a TestCase.
 - **int runCount()** Gets the number of run tests.
 - **void startTest(Test test)** Informs the result that a test will be started.
 - **void stop()** Marks that the test run should stop.
-

TESTING FOR EXCEPTIONS IN JUNIT

- The Test annotation supports 'expected' parameters which declares that a test method should throw an exception. If it doesn't throw an exception or if it throws a different exception than the one declared, the test fails.
- For example, the following test succeeds :



The screenshot shows the Eclipse IDE interface. The top toolbar includes icons for file operations, running, and debugging. The Package Explorer on the left shows the project structure with a green checkmark next to the JUnit icon. The JUnit view displays the test results: 'Finished after 0.01 seconds', 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. A green progress bar indicates a successful run. The test details show 'TestSamples.ExceptionCase [Runner: JUnit 4] (0.000 s)' and 'outOfBounds (0.000 s)' with a green checkmark. The main editor window shows the source code of 'ExceptionCase.java'.

```
Java - Unit_Testing/src/TestSamples/ExceptionCase.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer JUnit
Finished after 0.01 seconds
Runs: 1/1 Errors: 0 Failures: 0
TestSamples.ExceptionCase [Runner: JUnit 4] (0.000 s)
  outOfBounds (0.000 s)

ExceptionCase.java
package TestSamples;

import java.util.ArrayList;

import org.junit.Test;

public class ExceptionCase {
    @Test(expected = IndexOutOfBoundsException.class)
    public void outOfBounds() {
        new ArrayList<Object>().get(1);
    }
}
```


JUNIT ANNOTATION IMPLEMENTATION COMPLETE EXAMPLE

- Junit Annotation Class : AnnotTest.java
- Test Runner Class : AnnotTestRunner.java

Source code and its out are displayed in the following slides

```
package Test;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Ignore;
import org.junit.Test;

public class AnnotTest {
    @BeforeClass
    public static void beforeClass() {
        System.out.println("in before class");
    }

    @AfterClass
    public static void afterClass() {
        System.out.println("in after class");
    }

    @Before
    public void before() {
        System.out.println("in before");
    }
}
```

```
@After
    public void after() {
        System.out.println("in after");
    }

@Test
    public void test() {
        System.out.println("in test");
    }

@Ignore
    public void ignoreTest() {
        System.out.println("in ignore test");
    }
}
```

```
package Test;

import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class AnnotTestRunner {
    public static void main(String[] args) {
        Result result = JUnitCore.runClasses(AnnotTest.class);
        for (Failure failure : result.getFailures()) {
            System.out.println(failure.toString());
        }
        System.out.println(result.wasSuccessful());
    }
}
```

Output:

```
in before class
in before
in test
in after
in after class
true
```


ORGANIZING TEST FIXTURES WITH SETUP/TEARDOWN AND @BEFORE/ @AFTER

- JUnit test runners automatically invoke the setUp() method before running each test. This method typically initializes fields, turns on logging, resets environment variables, and so forth.
 - JUnit test runners automatically invoke the tearDown() method after running each test which releases above mention resources after test.
 - Annotating a public void method with @Before causes that method to be run before the Test method.
 - Annotating a public void method with @After causes that method to be run after the Test method.
 - Annotating a public static void method with @BeforeClass causes it to be run once before any of the test methods in the class.
 - Annotating a public static void method with @AfterClass causes it to be run after all tests have finished.
-



Package Explorer JUnit

Finished after 0.011 seconds

Runs: 2/2 Errors: 0 Failures: 0

TestSamples.SetTearTest [Runner: JUnit 4] (0.000 s)

Failure Trace

The methods will execute in the following order:

- oneTimeSetUp()
- setUp()
- testEmptyCollection()
- tearDown()
- setUp()
- testOneItemCollection()
- tearDown()
- OneTimeTearDown()

```
SetTearTest.java
package TestSamples;

import org.junit.*;
import static org.junit.Assert.*;
import java.util.*;

public class SetTearTest {
    private Collection collection;

    @BeforeClass
    public static void oneTimeSetUp() { // one-time initialization code
    }

    @AfterClass
    public static void oneTimeTearDown() { // one-time cleanup code
    }

    @Before
    public void setUp() { collection = new ArrayList();
    }

    @After
    public void tearDown() {collection.clear();
    }

    @Test
    public void testEmptyCollection() {assertTrue(collection.isEmpty());
    }

    @Test
    public void testOneItemCollection() {collection.add("itemA");
    assertEquals(1, collection.size());
    } }
```

A COMPLETE EXAMPLE OF JUNIT TESTING

- Pojo class : Student.java
- Business Logic : StudentDetails.java
- Test Case class : StudentDetailsTest.java
- Main Class : StudentMain.java

```
package Testing;
public class StudentDetails {
    public String calculatePercentage(Student student){
        String status;
        if(student.getPercentage() < 50){
            status = "Fail";
        }else{
            status = "Pass";
        }
        return status;
    }
}
```

```
package Testing;
```

```
public class Student {
    private String name;
    private String city;
    private int age;
    private int percentage;
}
```

```
package Testing;
```

```
import org.junit.Test;
```

```
import static org.junit.Assert.assertEquals;
```

```
public class StudentDetailsTest {
```

```
    StudentDetails studentDetails = new StudentDetails();
```

```
    Student student = new Student();
```

```
    @Test
```

```
    public void CalculatePercentage() {
```

```
        student.setName("Derick");
```

```
        student.setAge(25);
```

```
        student.setCity("Agra");
```

```
        student.setPercentage(79);
```

```
        String status=studentDetails.calculatePercentage(student);
```

```
        assertEquals("Fail", status, "Fail");
```

```
    }
```

```
}
```



```
package Testing;
```

```
import org.junit.runner.JUnitCore;
```

```
import org.junit.runner.Result;
```

```
import org.junit.runner.notification.Failure;
```

```
public class StudentMain {
```

```
    public static void main(String[] args) {
```

```
        Result result = JUnitCore.runClasses(StudentDetailsTest.class);
```

```
        for (Failure failure : result.getFailures()) {
```

```
            System.out.println(failure.toString());
```

```
        }
```

```
        System.out.println(result.wasSuccessful());
```

```
    }
```

```
}
```




Package Explorer JUnit



Finished after 0.011 seconds

Runs: 2/2 Errors: 0 Failures: 0

TestSamples.SetTearTest [Runner: JUnit 4] (0.000 s)

testOneItemCollection (0.000 s)

testEmptyCollection (0.000 s)

Failure Trace

Student.java StudentDetails.java *StudentDetailsTest.java StudentMain.java

```
package Testing;

import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class StudentMain {
    public static void main(String[] args) {
        Result result = JUnitCore.runClasses(StudentDetailsTest.class);
        for (Failure failure : result.getFailures()) {
            System.out.println(failure.toString());
        }
        System.out.println(result.wasSuccessful());
    }
}
```

Problems Javadoc Declaration Console

```
<terminated> StudentMain [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Jul 29, 2016 12:07:29 PM)
testCalculateAppriasal(Testing.StudentDetailsTest): Fail expected:<[Pass]> but was:<[Fail]>
false
```

JUNIT TIME TEST

- Junit has a handy option of Timeout.
 - If a test case takes more time than specified milliseconds then Junit will automatically mark it as failed.
 - The timeout parameter is used along with @Test annotation.
-

```
package Test;

import static org.junit.Assert.*;
import org.junit.Test;
import Test.Pet;

public class PetTest {
    @Test
    public void testPet() {

    }

    @Test (timeout=5000)
    public void testMeaow()
    {
        Pet testPet = new Pet();
        assertTrue("Meaow".equals(testPet.meow()));
    }
}
```

```
package Test;

import org.junit.runner.JUnit4;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class AnnotTestRunner {
    public static void main(String[] args) {
        Result result =
            JUnit4.runClasses(PetTest.class);
        for (Failure failure :
            result.getFailures()) {

            System.out.println(failure.toString());
        }

        System.out.println(result.wasSuccessful());
    }
}
```

Output: true

THANK YOU...
