Java JUnit

for Unit Testing

JUNIT API (JUNIT 4.X)

ERIC Y. CHOU, PH.D.

IEEE SENIOR MEMBER

What is JUnit

JUnit is a 3rd-Party API from junit.org

JUnit	
Developer(s)	Kent Beck, Erich Gamma, David Saff, Mike Clark (University of Calgary)
Stable release	4.12 / December 4, 2014
Written in	Java
Operating system	Cross-platform
Type	Unit testing tool
License	Eclipse Public License
Website	junit.org

JUnit

Compatible with Other Languages

Ports

Actionscript (FlexUnit)
Ada (AUnit)
C (CUnit)
C# (NUnit)
C++ (CPPUnit, CxxTest)
Coldfusion (MXUnit)
Erlang (EUnit)
Eiffel (Auto-Test)
Fortran (fUnit, pFUnit)
Delphi (DUnit)
Free Pascal (FPCUnit)
Haskell (HUnit)
JavaScript (JSUnit)
Microsoft .NET (NUnit)
Objective-C (OCUnit)
OCaml (OUnit)
Perl (Test::Class and Test::Unit)
PHP (PHPUnit)
Python (PyUnit)
Qt (QTestLib)
R (RUnit)
Ruby (Test::Unit)

JUnit

Automating Tests

Nearly every programmer tests his code. Testing with **JUnit** isn't a totally different activity from what you're doing right now. It's a different way of doing what you're already doing. The difference is between testing, that is checking that your program behaves as expected, and having a battery of tests, little programs that automatically check to ensure that your program behaves as expected. In this chapter we'll go from typical **println( )-based** testing code to a fully automated test.

JUnit's Goals

Every framework has to resolve a set of constraints, some of which seem always to conflict with each other. JUnit is no exception; simultaneously tests should be:

- **Easy to write.** Test code should contain no extraneous overhead.
- **Easy to learn to write.** Because our target audience for JUnit is programmers who are not usually professional testers, we would like to minimize the barriers to test writing.
- **Quick to execute.** Tests should run fast so we can run them hundreds or thousands of times a day.
- **Easy to execute.** The tests should run at the touch of a button and present their results in a clear and unambiguous format.
- **Isolated.** Tests should not affect each other. If the order in which the tests are run changes, the results shouldn't change.
- **Composable.** We should be able to run any number or combination of tests together. This is a corollary of isolation.

JUnit API

For most uses, JUnit has a simple API: subclass TestCase for your test cases and call assertTrue( ) or assertEquals( ) from time to time.

Most of the time you will encounter five **classes** or **interfaces** when you are using **JUnit**:

Assert (an utility class)

A collection of static methods for checking actual values against expected values

Test (a test handler interface)

The interface of all objects that act like tests

TestCase

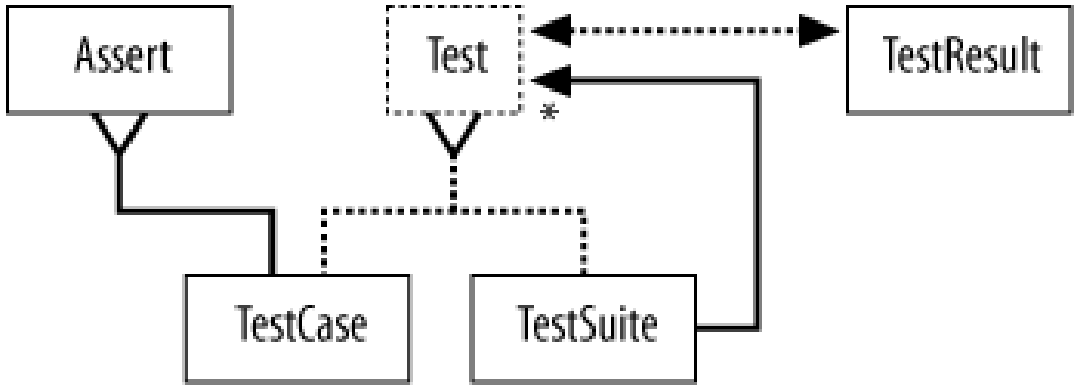
A single test

TestSuite

A collection of tests

TestResult

A summary of the results of running one or more tests



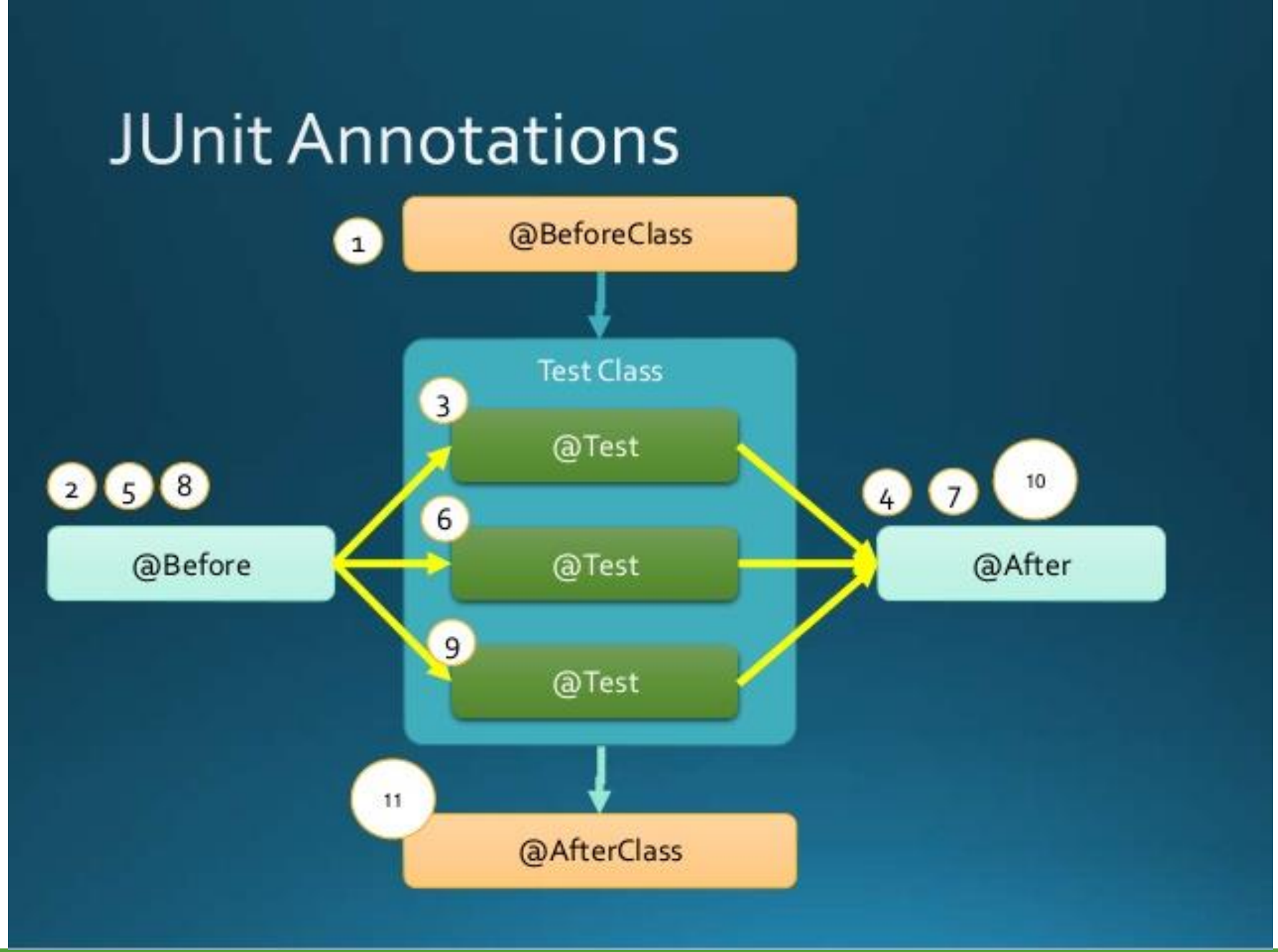
JUnit

Default Test Class by BlueJ

```
public class StudentTest
{
    /**
     * Default constructor for test class StudentTest
     */
    public StudentTest()
    {
    }

    /**
     * Sets up the test fixture.
     *
     * Called before every test case method.
     */
    @Before
    public void setUp()
    {
    }

    /**
     * Tears down the test fixture.
     *
     * Called after every test case method.
     */
    @After
    public void tearDown()
    {
    }
}
```


JUnit Annotations
1
@BeforeClass
Test Class
3
@Test
6
@Test
9
@Test
2
5
8
@Before
4
7
10
@After
11
@AfterClass

Complete Testing Setup with JUnit

```
public class FoobarTest
{
    @BeforeClass
    public static void setUpClass() throws Exception {    // Code executed before the first test method
    }
    @Before
    public void setUp() throws Exception {    // Code executed before each test
    }
    @Test
    public void testOneThing() {    // Code that tests one thing
    }
    @Test
    public void testAnotherThing() {    // Code that tests another thing
    }
    @Test
    public void testSomethingElse() {    // Code that tests something else
    }
    @After
    public void tearDown() throws Exception {    // Code executed after each test
    }
    @AfterClass
    public static void tearDownClass() throws Exception {    // Code executed after the last test method
    }
}
```

assert Expression$_1$;

JUnit

ad hoc Testing, If not Using JUnit

Go BlueJ!!!

Java Keyword

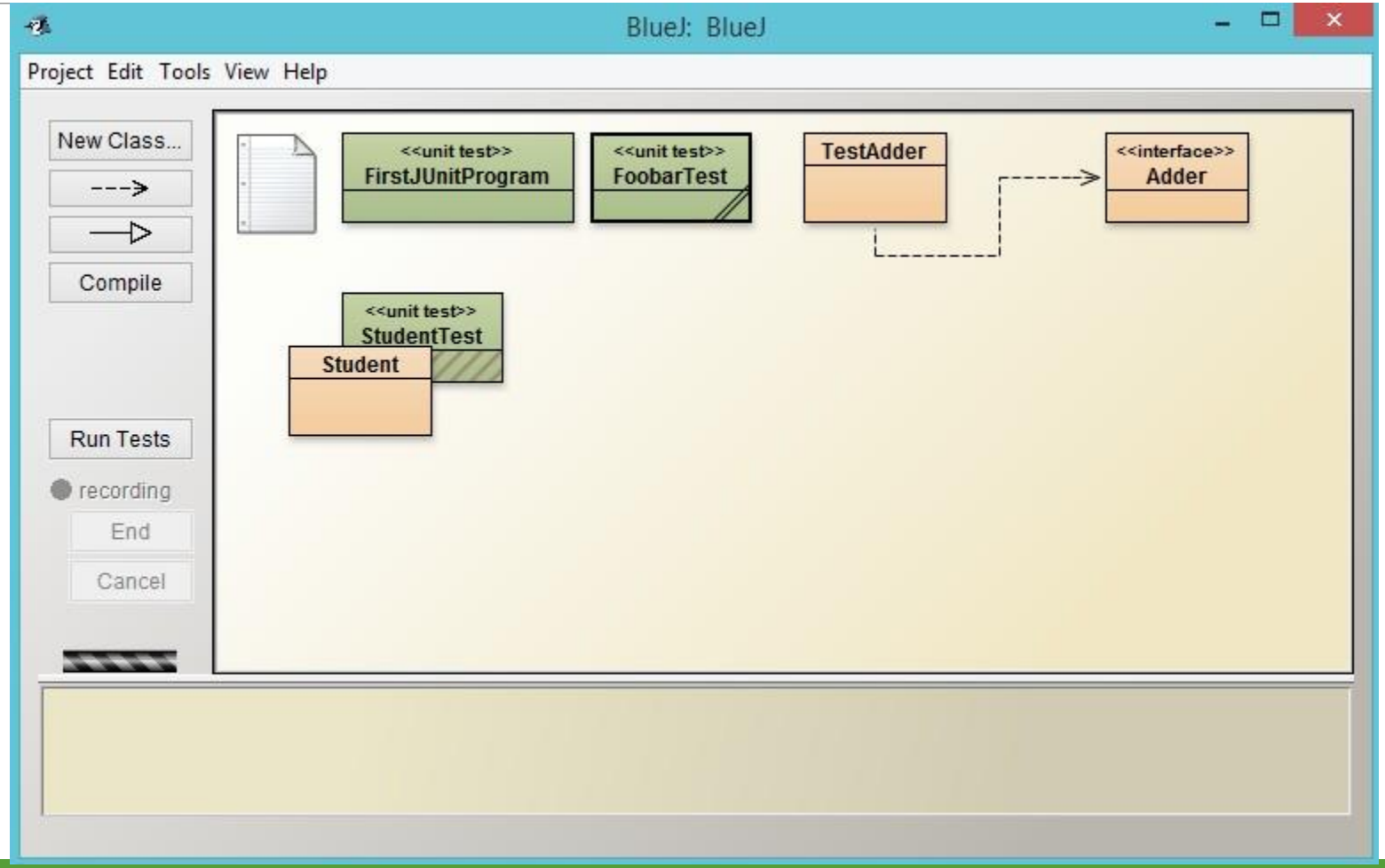
assert Expression1 ;

or

assert(Expression1);

TestAdder is a tester class.

Adder is a class.



Please refer to my Java Programming AP Edition or AP Computer Science Part 2: Chapter 12 for assertion and Exception.