TestNG

Test Next Generation

Lecture Objectives:

junit

1) Explain the differences between Junit and TestNG
2) Explain the concept of a data provider
3) Construct a set of test cases using TestNG and data providers
4) Compare and contrasts the differences in philosophy between Junit and testNG.
5) Explain how to use different folders to organize test code separate from source code.

Discussion: What is Junit?

Unit testing tool

⇒ framework

⇒ test methods

⇒ Assert things

⇒ Results reporting (minimal)

⇒ Test Execution

⇒ Automated

Discussion: What is Junit?

Problems:

⇒ Not ideal w/ coverage

⇒ Verbose to write tests

⇒ Not good @ running part of tests.

MSOE

What is TestNG

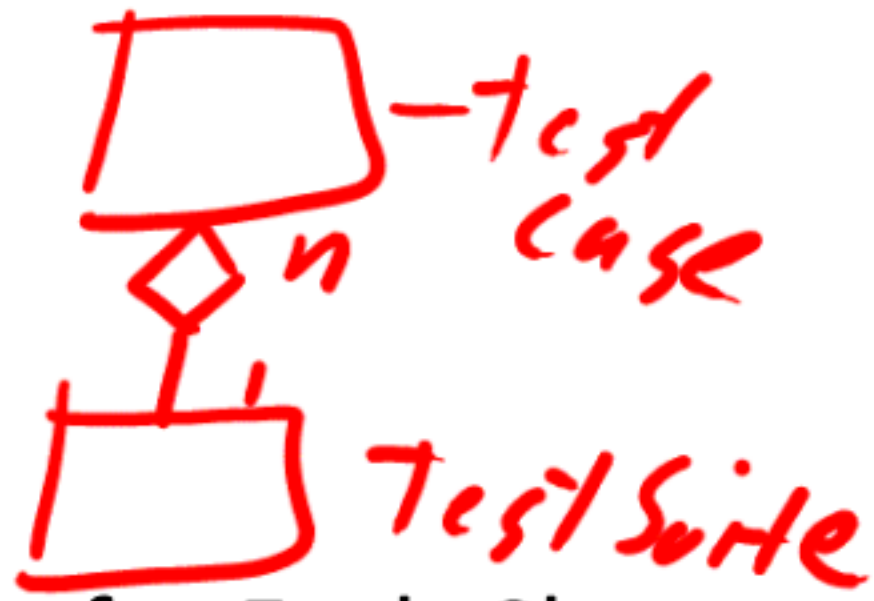
- Automated testing framework
- NG = Next Generation
- Similar to JUnit (especially JUnit 4)
- Not a JUnit extension
  - inspired by JUnit
- Designed to be better than JUnit, especially when testing integrated classes
- Created by Dr. Cédric Beust (of Google)
- Open source (http://testng.org)

A Bit of History

- **sUnit** for Smalltalk (~1998, Kent Beck)
- **JUnit** (~2000, Kent Beck & Erich Gamma)
  - Latest 3.x: 3.8.2 (March 2006)
- **TestNG** (~2004)
  - Latest: 5.8 (March 2008)
- **JUnit 4.x** (2006)
  - Latest: 4.5 (August 2008)

Junit Philosophy

- Human Judgment Should not be Needed to "Test"
- Two Basic Classes:
  - TestCase
  - TestSuite
- Write a Simple Test Class for Each Class to be Tested
- Not Necessary, but More Manageable
  - YourTestCase Extends JUnit TestCase
  - TestSuite comprises of TestCases and other TestSuites
- Minimal "Framework Code"



More code for your framework.

TestNG Philosophy

- (Try to) Provide What's not Present in JUnit
  - The So-called JUnit Frustrations
- Unit Testing -> Integration Testing
- Pioneered Annotation Based Testing
  - Lots of 'em
  - Superset of What JUnit Has
- Talk of Suite, Test, Class
  - <suite><test><classes>...
- Novel Way of Looking at Testing Java Programs? - Maybe !

Test class/method (JUnit 3 vs. 4)

```
// JUnit 3
import junit.framework.TestCase;
public class MyTestClass extends TestCase {
  public void testSomething1() throws ... {
   ... }
  public void testSomething2() throws ... {
   ... }
}
```

methods must begin w/ test.

```
// JUnit 4
import org.junit.Test;
public class MyTestClass {
  @Test public void aTest1() throws ... { ...
   }
  @Test public void aTest2() throws ... { ...
   }
}
```

@Test annotation

MSOE

Test class/method (JUnit 4 vs. TestNG)

```
import org.junit.Test;                    // JUnit 4
... or
import org.testng.annotations.Test;       // TestNG

public class MyTestClass {
   @Test
   public void aTestMethod() throws ... { ...
   }
}
```

Same

Assertions: TestNG versus Junit 3/4

- JUnit 3 assertions inherited by base TestCase
- JUnit 4: similar to TestNG
- TestNG assertEquals(...) assertion signatures are different than JUnit's:
- JUnit: [msg,] expected, actual
- TestNG: actual, expected [, msg]
- Can use JUnit 3/4 assertions (library issues)
- Can use TestNG's copy of JUnit 3 assertions (org.testng.AssertJUnit): migration aid.
- JUnit4/TestNG assertions richer than JUnit 3; each contains features that the other doesn't.

More assertions

Annotations

- @Test
  - Identify a test method
- @BeforeClass, @AfterClass, @BeforeMethod, …
  - Identify a method used to configure your test
- @DataProvider
  - Create parameters to pass to your test methods
- @Factory
  - Create your own test objects at runtime

TestNG Terminology

Suites. Each suite contains…

Tests. Each test contains…

Classes. Each class contains…

Methods

Each of these @Configuration can wrap any of the locations listed above.

```
@BeforeTest
public void initTest() { … }

@AfterSuite
public void cleanUp() { … }
```

@BeforeSuite

@AfterSuite

etc.

Data Providers

```
@DataProvider(name = "user-agents")
public Object[][] createUserAgents() {
   return new Object[][] {
     new Object[] { "MSIE", 200},
     new Object[] { "FireFox", 200};
     new Object[] { "Safari", 301};
     new Object[] { "WAP", -1};
   }
}
```

```
@Test(dataProvider = "user-agents")
public void verifyUserAgent(String s, int
   code) {
   assert getReturnCodeFor(s) == code;
}
```

The High Blood Pressure Problem Again

Blood Pressure Category	Systolic mm Hg (upper #)		Diastolic mm Hg (lower #)
Normal	less than **120**	and	less than **80**
Prehypertension	**120 – 139**	or	**80 – 89**
High Blood Pressure (Hypertension) **Stage 1**	**140 – 159**	or	**90 – 99**
High Blood Pressure (Hypertension) **Stage 2**	**160** or higher	or	**100** or higher
Hypertensive Crisis (Emergency care needed)	Higher than **180**	or	Higher than **110**

UML of system under test

class Class Model

«enumeration»
BloodPressureDiagnosisSystem::
BloodPressureDiagnosis

- NORMAL
- PREHYPERTENSION
- STAGE_1_HBP
- STAGE_2_HBP
- HYPERTENSIVE_CRISIS

Enumeration of BP types

BloodPressureDiagnosisSystem

- \- age: int
- \- patientName: String

- \+ BloodPressureDiagnosisSystem(String, int)
- \+ diagnoseBP(int, int) : BloodPressureDiagnosis
- \+ getAge() : int
- \+ getPatientName() : String

Constructor

Diagnosis

The class under test

```
/**
 * This will instantiate a new instance of the Blood Pressure Diagnosis system.
 * @param patientName This is the name of the patient. Must consist of a first and a last name
separated by a space, and each of the first and last names must be 2 or more characters in
length.
 * @param patientAge This is the age of the patient in years. Must be greater than or equal to
0.
 * @throws Exception An Exception will be thrown if the age is out of range or if the name is
incorrectly formatted.
 */
public BloodPressureDiagnosisSystem(String patientName, int patientAge) throws Exception {
/**
 * This method will diagnose a persons blood pressure condition based upon the American Heart
Association clinical definitions.
(http://www.heart.org/HEARTORG/Conditions/HighBloodPressure/AboutHighBloodPressure
 * /Understanding-Blood-Pressure-Readings_UCM_301764_Article.jsp)
* @param systolic This is the systolic blood pressure, the upper number, whichis the pressure in
the arteries when the heart is beating.
 * @param diastolic This is the diastolic blood pressure, which is the pressure between heart
beats.
 * @return The return will be a diagnosis of the given blood pressure.
 * @throws ArithmeticException
 *          An Arithmetic exception will be thrown if the pressure is negative, or if the systolic
pressure is less than or equal to the diastolic pressure. */
public BloodPressureDiagnosis diagnoseBP(int systolic, int diastolic)
```

Robust Worst Case Boundary Value