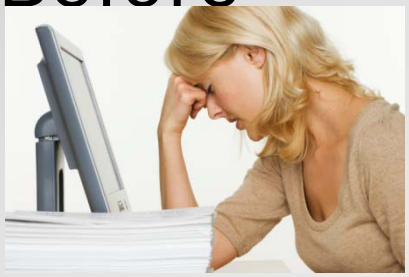
Junit Overview

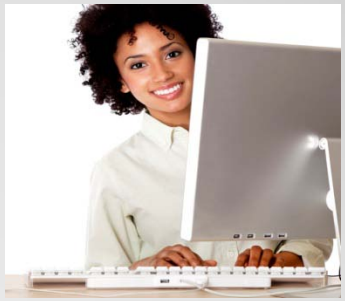
By Ana I. Duncan

Agenda

- What Is Junit
- Why Junit, Why test?
- Junit Lifecycle
- Junit Examples from CM
- Other Testing frameworks
- Resources
- Before



After



What is JUNIT

- JUnit is a member of the xUnit testing framework family and now the de facto standard testing framework for Java development. JUnit, originally created by Kent Beck and Erich Gamma, is an API that enables developers to easily create Java test cases. It provides a comprehensive assertion facility to verify expected versus actual results.
- For those interested in design patterns, JUnit is also a great case study because it is very pattern-dense.

Why Junit, why testing?

- Using a testing framework is beneficial because it forces you to explicitly declare the expected results of specific program execution routes.
- When debugging it is possible to write a test which expresses the result you are trying to achieve and then debug until the test comes out positive. By having a set of tests that test all the core components of the project it is possible to modify specific areas of the project and immediately see the effect the modifications have on the other areas by the results of the test, hence, side-effects can be quickly realized.
- JUnit promotes the idea of “first testing then coding”, in that it is possible to setup test data for a unit which defines what the expected output is and then code until the tests pass. It is believed by some that this practice of "test a little, code a little, test a little, code a little..." increases programmer productivity and stability of program code whilst reducing programmer stress and the time spent debugging.
- It also integrates with Ant, Maven, Spring

Other Motivations

- It looks good on a developer's resume
- Less phone calls from customers to the Help Desk
- More defects are caught on Development before going higher on SDLC, so it is easier and cheaper to fix them now than later
- Emma Reports (Code Coverage) will look 'greener'. This will make management team very happy

Testcase Lifecycle

- The lifecycle of a TestCase used by the JUnit framework is as follows:

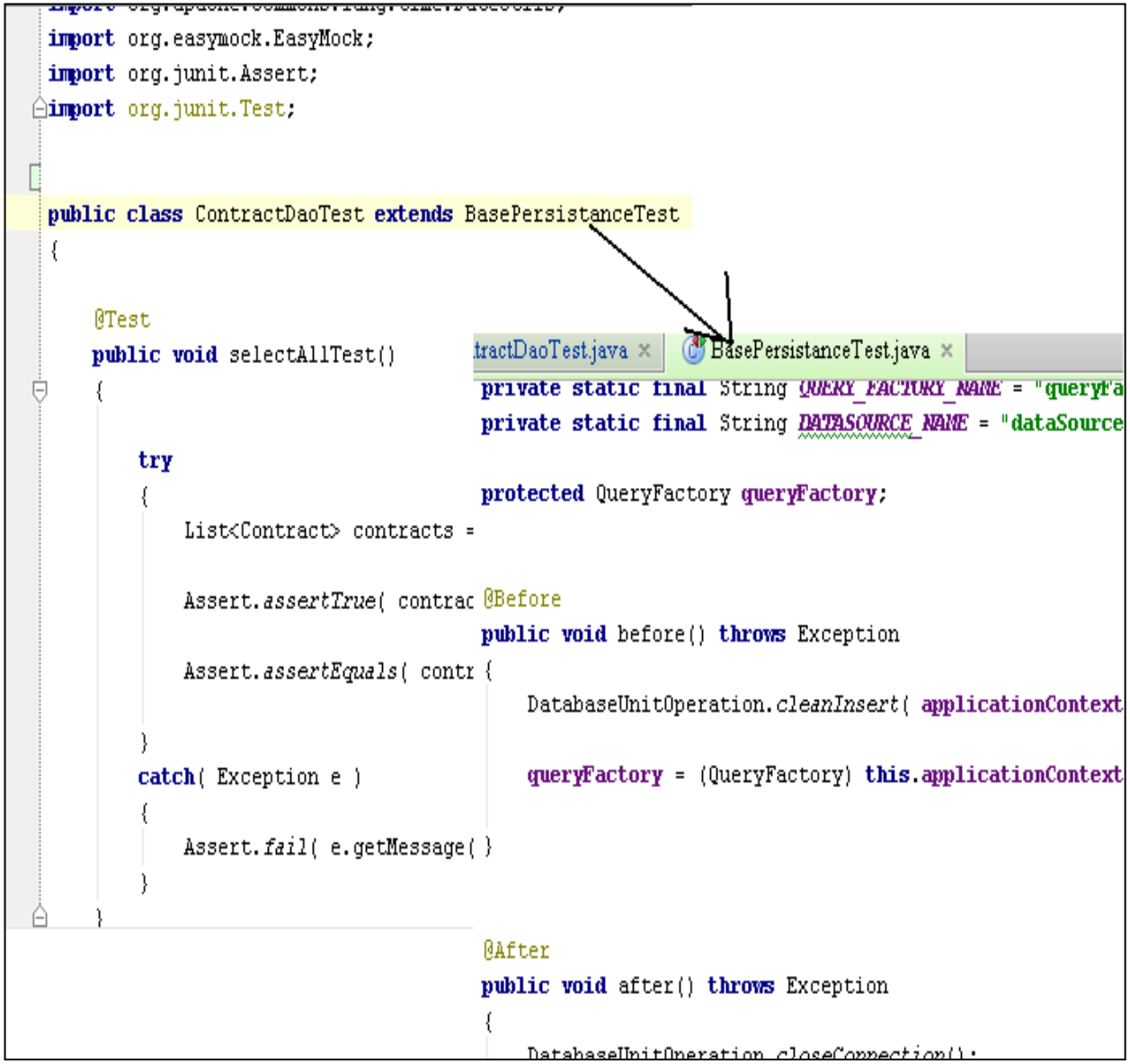
- Execute public void setUp().
- Call a test-prefixed method.
- Execute public void tearDown().
- Repeat these steps for each test method.

```
import org.junit.*; PROMOTES USE OF ANNOTATIONS

public class TestFoobar{
    @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
    }
@After
    public void tearDown() throws Exception {
        // Code executed after each test
    }
```

Lifecycle of a testcase

Good testing practices

- If it is a calculation, test valid values (happy path), invalid values, invalid data type, null values. Test if throwing the correct exception
- If it is an object, test that it returns the number of expected object, that the fields of the objects matches the content
- If testing an external service, use a another open source framework such as Mockito to create a mock service
- Each unit test should be independent of other test. If you write one unit test for multiple methods then the result may be confusing. Test behavior is more important than methods. Each test case has to be in the same package as the code to be tested
- Use naming conventions methods such as xxxxTestCases()
- Follow the Maven convention, Testcases should be located under Test package



Example of Test Case 1: This is an example of how the test cases are implemented in CM using a parent test Methods setup and teardown are implemented on the BasePersistenceTest class

```
        Assert.assertNotNull( result );
        Assert.assertEquals(new Integer(3), result.getId());
        Assert.assertEquals(new BigDecimal("30.00"), result.getRate());
        Assert.assertEquals(new Integer(2), result.getCode());


        result.setCode(new Integer(2));
        result.setRate(new BigDecimal("70.00"));


        AgencyToken token = new AgencyToken();
        token.setUserIdentifier( "UnitTestUserID" );

        int updates = this.contractDetailExtensionDAO.update(result, token);
        Assert.assertEquals(1, updates);


        ContractDetailExtension updated = this.contractDetailExtensionDAO.select( 3 );

        Assert.assertEquals(new Integer(3), updated.getId());
        Assert.assertEquals(new BigDecimal("70.00"), updated.getRate());
        Assert.assertEquals(new Integer(2), updated.getCode());
        Assert.assertEquals("UnitTestUserID", updated.getMaintenance().getLastChangedUserName());

    }
    catch( Exception e )
    -
```

Example of Test Case 2: happy path scenario

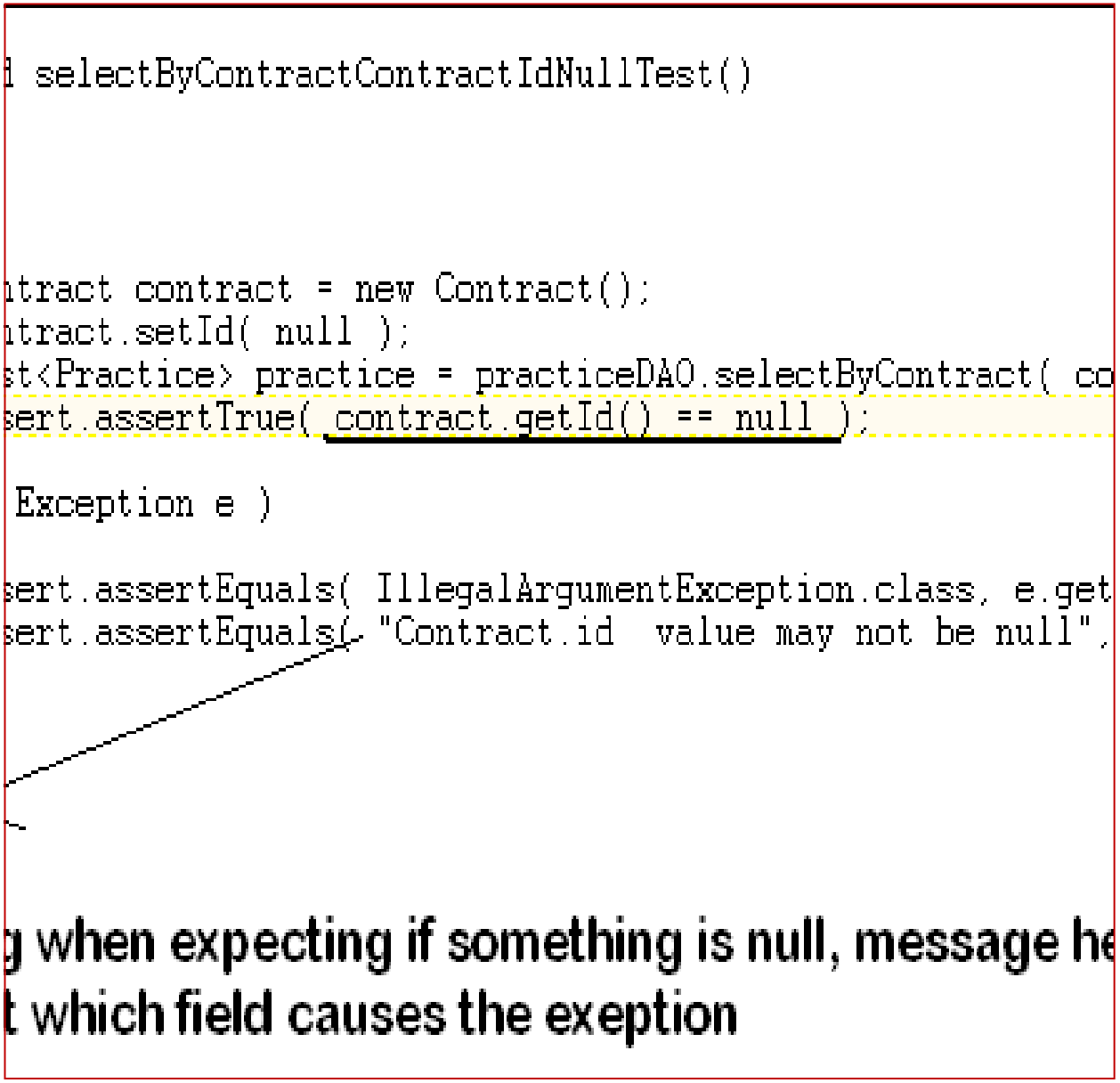
```
// start in current fiscal year, prior calendar year
int contractStartYear = this.currentFiscalYear -
String start = "12/01/" + contractStartYear;
// end after current fiscal year
String end = "01/31/" + this.currentFiscalYear;
BigDecimal rental = new BigDecimal( "857.2" );
BigDecimal acres = new BigDecimal( "946.1" );
Contract contract = this.buildTestContract( start, end, rental, acres );
ObligationCalculator calc = new CrpObligationCalculator();
BigDecimal result = calc.calculateObligation( contract );

// All of December and January = 62 days
// ( 62 days / 365 days ) * 857.2 * 946.1 = 137758.380...
// on leap year (366 days) = 137381.991...
if( isLeapYear( this.currentFiscalYear ) )
{
    Assert.assertEquals( new BigDecimal( "137381.99"), result );
}
else
{
    Assert.assertEquals( new BigDecimal( "137758.38"), result );
}

////////////////////////////Additional Test Cases when calculation is performed
//Non-happy path
//Contract IS NULL
BigDecimal result1 = calc.calculateObligation( null );

//Any of the relevant fields (used on the calculation) on Contract obj is NULL
Contract contract1 = this.buildTestContract( start, end, rental, acres );
contract1.setPayableAcreage( null );
BigDecimal result2 = calc.calculateObligation( contract1 );


Contract contract2 = this.buildTestContract( start, end, rental, acres );
contract2.setContractRentalRate( null );
BigDecimal result3 = calc.calculateObligation( contract2 );

Contract contract3 = this.buildTestContract( null, end, rental, null );
contract3.setContractRentalRate( null );
```

Add aditional testcases to cover non-happy path

Example of Test Case 3: Adding 3 more non-happy path scenarios

```
    contractClu.getCluAcres();
    contractClu.getCluIdentifier();
    contractClu.getContractId();
    contractClu.getFarmTract();
    contractClu.getMaintenance();
    contractClu.getDirtyFlag();

}


@Test
public void toStringTest()
{
    contractClu.toString();
}


@Test
public void hashCodeTest()
{
    contractClu.hashCode();
}
@Test
public void testEquals()
{
    Assert.assertTrue(contractClu.equals(contractClu));
```

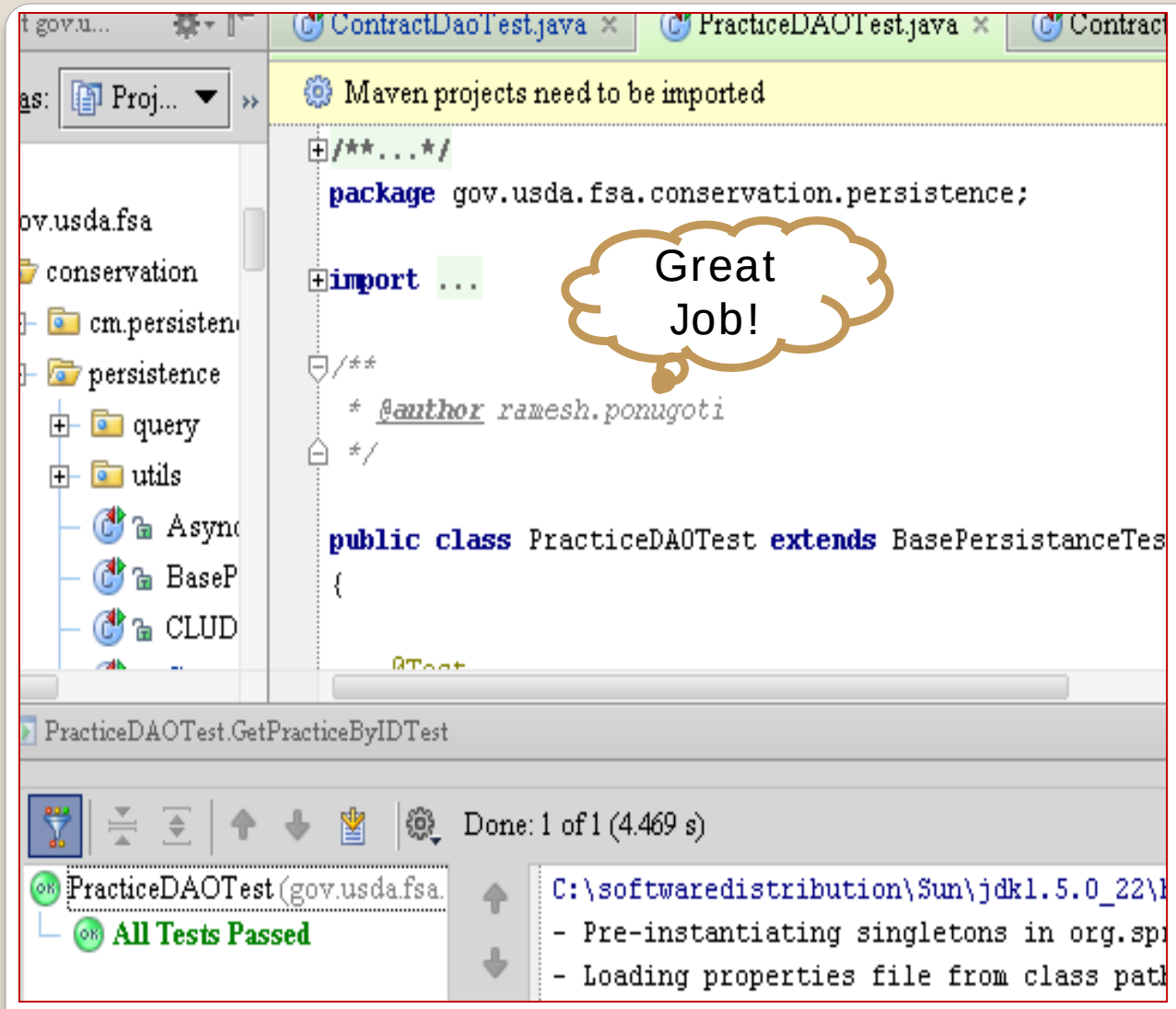
Example of Test Case 4: test cases of getters, toString(), hashCode(), and equals()



Example of Test Case 5: test cases when expecting a null. Make sure it throws the right exception

```
c void selectDAOExceptionTest()

ry


   BasicDataSource mockDataSource = EasyMock.createMock( BasicDataSourc
   EasyMock.expect( mockDataSource.getConnection() ).andThrow( new SQLE
   EasyMock.replay( mockDataSource );

   queryFactory.setDataSource( mockDataSource );
   PracticeDAOimpl localPracticeDAO = new PracticeDAOimpl( Practice.cla
   localPracticeDAO.setQueryFactory( queryFactory );

   practiceDAO.selectAll();
   EasyMock.verify( mockDataSource );

   Assert
           .fail( "DaoException  Named Query [ selectAll ] - Could not


atch( Exception e )

   Assert.assertEquals( DaoException.class, e.getClass() );
   Assert.assertEquals(
           "Named Query [ selectAll ] - Could not get JDBC Connection;
           e.getMessage() );
```

Example of Test Case 6: test cases Exceptions



This is what you want to see...GREEN!!

Resources

- http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks#Java
-
- http://htmlunit.sourceforge.net/
-
- http://jakarta.apache.org/cactus/
-
- http://junit.sourceforge.net/junit3.8.1/index.html
-
- http://java.sun.com/developer/Books/javaprogramming/ant/ant_chap04.pdf
-
- http://en.wikipedia.org/wiki/JUnit
- http://junit.org/apidocs/junit/framework/Assert.html