[AT Ralph] TestNG Practice Part1

Target: Setup TestNG in you local environment, familiar with the TestNG grammer

For Challenge: Complete 1 Calculator TestNG test

Part 1: What Is Testing Framework

A testing framework or more specifically a testing automation framework is an execution environment for automated tests. It is the overall system in which the tests will be automated.

How many testing framework we have: https://en.wikipedia.org/wiki/List_of_unit_testing_frameworks

How many testing framework java have:

Name	xUnit	Source	Remarks
Arquillian	Yes	[205]	Open source framework for writing Integration and funcional tests. It includes Arquillian graphene, Drone and Selenium to write tests to the visual layer too.
beanSpec		[206]	Behavior-driven development
BeanTest	No	[207]	A tiny Java web test framework built to use WebDriver/HTMLUnit within BeanShell scripts
Cactus			A JUnit extension for testing Java EE and web applications. Cactus tests are executed inside the Java EE/web container.
Concordion		[208]	Acceptance test-driven development, Behavior-driven development, Specification by example
Concutest		[209]	A framework for testing concurrent programs
Cucumber-JVM		[210]	Behavior-driven development replaces deprecated JRuby-based Cuke4Duke
Cuppa		[211]	Behavior-driven development framework for Java 8
DbUnit		[212]	A JUnit extension to perform unit testing with database-driven programs
EasyMock		[213]	A mock framework [214]
EtlUnit	Yes	[215]	A unit testing framework for Extract-Transform-Load processes, written in Java. Capable of testing Oracle, Informatica, SqlServer, PostGreSQL, MySQL, etc.[216]
EvoSuite		[217]	A test case generation tool that can automatically generate JUnit tests.
GrandTestAuto		[218]	GrandTestAuto (GTA) is a platform for the complete automated testing of Java software. Tests can be distributed across a number of machines on a network.
GroboUtils		[219]	A JUnit extension providing automated documentation, class hierarchy unit testing, code coverage, and multi-threaded tests.
HavaRunner	Yes	[220]	A JUnit runner with built-in concurrency support, suites and scenarios.
Instinct		[221]	Behavior-driven development
Java Server-Side Testing framework (JSST)		[222]	Java Server-Side Testing framework which is based on the similar idea to the one of Apache CACTUS, but unlike CACTUS it's not coupled to JUnit 3.x and can be used in conjunction with any testing framework.
JBehave		[223]	Behavior-driven development
JDave		[224]	Behavior-driven development
JExample	Yes	[225]	A JUnit extension that uses dependencies between test cases to reduce code duplication and improves defect localization.
JGiven		[226]	Behavior-driven development
JMock		[227]	A mock framework
JMockit		[228]	Open source framework. Tests can easily be written that will mock final classes, static methods, constructors, and so on. There are no limitations.
Jnario	Yes	[229]	Behavior-driven development like Cucumber
Jtest	Yes	[230]	Commercial. Automated unit/component test generation and execution with code coverage and runtime error detection. Also provides static analysis and peer code review.
Jukito		[231]	Combines Mockito and Guice to allow automatic mocking and instantiation of dependencies
JUnit	Yes	[232]	
JUnitEE		[233]	A JUnit extension for testing Java EE applications
JWalk		[234]	Fast, semi-automatic creation of exhaustive unit test-sets
Mockito		[235]	A mock framework, using the [12] library
Mockrunner		[236]	A JUnit extension for testing testing servlets, filters, tag classes and Struts actions and forms.
Needle		[237]	Open source framework for testing Java EE components outside of the container in isolation.
NUTester		[238]	Testing framework developed at Northeastern University to aid in teaching introductory computer science courses in Java
OpenPojo		[239]	Open source framework used to validate and enforce POJO behavior as well as manage identity - equals, hashCode & toString.
PowerMock		[240]	An extension to both Mockito and EasyMock that allows mocking of static methods, constructors, final classes and methods, private methods, removal of static initializers and more.

Randoop	Yes	[241]	Automatically finds bugs and generates unit tests for Java, via feedback-directed random testing (a variant of Fuzzing).
SpryTest	Yes	[242]	Commercial. Automated Unit Testing Framework for Java
SureAssert		[243]	An integrated Java unit testing solution for Eclipse. Contract-First Design and test-driven development
TestNG	Yes	[244]	Tests can include unit tests, functional tests, and integration tests. Has facilities to create even non-functional tests (as loading tests, timed tests).[245][246]
Unitils		[247]	Offers general utilities and features for helping with persistence layer testing and testing with mock objects. Offers specific support for testing application code that makes use of JPA, hibernate and spring. Unitils integrates with the test frameworks JUnit and TestNG.
XMLUnit		[248]	JUnit and NUnit testing for XML

Part 2: Why We Learn TestNG

TestNG is a testing framework inspired from JUnit and NUnit but introducing some new functionalities that make it more powerful and easier to use, such as:

- Annotations.
- Run your tests in arbitrarily big thread pools with various policies available (all methods in their own thread, one thread per test class, etc...).
- Test that your code is multithread safe.
- Flexible test configuration.
- Support for data-driven testing (with @DataProvider).
- Support for parameters.
- Powerful execution model (no more TestSuite).
- Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...).
- Embeds BeanShell for further flexibility.
- Default JDK functions for runtime and logging (no dependencies).
- Dependent methods for application server testing.

TestNG is designed to cover all categories of tests: unit, functional, end-to-end, integration, etc…

If you interest: **Junit vs TestNG** https://www.mkyong.com/unittest/junit-4-vs-testng-comparison/

Part 3: Setup TestNG Environment

Install TestNG plugin in your IDE

Use eclipse as example(some IDE already contains testNG)

Follow manual: http://toolsqa.com/selenium-webdriver/install-testng/

Part 4: HelloTestNG

Hello TestNG Example:

```
import org.testng.Assert;
import org.testng.annotations.Test;
public class HelloTestNG {
 @Test
 public void helloTestNG() {
  Assert.fail("Sorry, you fail the hello testNG");
 }
}
```

practice 1: Create the your first helloTestNG, instead of fail, try to pass it.

TestNG grammer:

http://testng.org/javadocs/org/testng/Assert.html

Basic Assert: assertEquals, assetTrue, assertNull, assertFalse, assertSame.....

SoftAssert and HardAssert ?

```
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
import org.testng.asserts.Assertion;
import org.testng.asserts.SoftAssert;
public class SoftHardAsserts {
  @BeforeTest
  public void beforeTest() {
  }
  @AfterTest
  public void afterTest() {
  }
  @Test
  public void softAssertExample() {
   // testNG 6.9.10 API
   SoftAssert softAssert = new SoftAssert();

   softAssert.assertTrue(true);
   softAssert.assertFalse(false);
   softAssert.assertFalse(true);
   softAssert.assertFalse(false);
   softAssert.assertTrue(false);
   softAssert.assertAll();
  }

  @Test
  public void hardAssertExample() {
   // testNG 6.9.10 API
   Assertion hardAssert = new Assertion();

   hardAssert.assertTrue(true);
   hardAssert.assertFalse(false);
   hardAssert.assertFalse(true);
   hardAssert.assertFalse(false);
   hardAssert.assertTrue(false);
  }

}
```

practice 2: Play with the Assert, filled every asserts in your HelloTestNG class with softAssert, such as assert.equarls("Trump", "Make the American great Again "):

Part 5: Create TestNG From Java Class

Test.org offical example: http://testng.org/doc/eclipse.html

practice 3: convert your "areYouPassExam" method test to testNG test

original code:

```
public class PracticeFour {
 public static boolean areYouPassExam(int score){
  if(score<60){
   return false;
  } else if(score<=100){
   return true;
  } else {
   return false;
  }
 }

 public static void main(String[] args) {
  // TODO Auto-generated method stub
  int score1 = 61;
  int score2 = -1;
  int score3 = 100;
  int score4 = 99999;
  int score5 = 44;
  System.out.println(areYouPassExam(score1));
  System.out.println(areYouPassExam(score2));
  System.out.println(areYouPassExam(score3));
  System.out.println(areYouPassExam(score4));
  System.out.println(areYouPassExam(score5));
 }
}
```

Please use testNG format to test it.

Part 6: Test A Complete Class

practice 4: Test the Bad Calculator

Bad Calculator

```
import java.util.Random;
public class CalculatorBad {

 public static int add(int number1, int number2) {
  return Math.abs(number1 + number2);
 }
 public static int sub(int number1, int number2) {
  Random rand = new Random();
  int bad = rand.nextInt(2);
  return number1 + bad - number2;
 }
 public static int mul(int number1, int number2) {
  try {
   Thread.sleep(10000);
  } catch (InterruptedException e) {
   e.printStackTrace();
  }
  return number1 * number2;
 }
 // Integer divide. Return a truncated int.
 public static int divInt(int number1, int number2) {
  return number1 / number2;
 }

}
```

please use testNG to test all the methods of the calculator, test how bad it is. Find as much bug as you can by testNG.

Good Resource:

- ***TestNG Tutorial:*** **http://www.mkyong.com/tutorials/testng-tutorials/**
- ***TestNG org:*** **http://testng.org/doc/documentation-main.html**