Introduction to JUnit

CS 3250
Software Testing

[Ammann and Offutt, “Introduction to Software Testing,” Ch. 3]
[https://junit.org/junit5/docs/current/user-guide/]
Today’s Objectives

• Understand JUnit test classes
• Understand structure of basic JUnit test methods
• JUnit assertions and other features
• Get started with JUnit
What is JUnit?

• An open source Java testing framework (junit.org) used to write and run repeatable automated tests

• JUnit is widely used in industry

• A structure for writing test drivers

• JUnit features include
  • Assertions to evaluate expected results
  • The ability to share common test data among tests
  • Test sets to easily organize and run tests
  • The ability to run tests from either a command line or a GUI
JUnit 5

- JUnit 4 – single jar file
- JUnit 5 – modular, flexible, robust, extensible
  (Not much changed between Junit 4 and Junit 5 in test writing styles)

Old tests (JUnit)

JUnit Vintage
Support JUnit3, Junit4

JUnit Platform
Define the TestEngine API for developing testing framework that runs on the platform

New tests (JUnit5)

JUnit Jupiter
Combine annotations and TestEngine

Other tests

Third party

Third party

IDEs/build tools

eclipse
Maven
Gradle
JUnit Tests

- For unit and integration testing
  - Entire object, part of an object (a method or some interacting methods), and interaction between several objects

- One test case in one test method

- A test class contains one or more test methods

- Test classes include
  - A collection of test methods
  - Method to set up the state before running each test (prefix)
  - Method to update the state after each test (postfix)
  - [Optional] Method to set up and update before and after all tests
Test Lifecycle

Setup
- Initialize the test fixture

Execute
- Interact with the subject under test

Verify
- Compare the actual (observed) result of running the test with the expected result – using assertion(s)

Teardown
- Release the test fixture to put the subject under test back into the initial state
Use the methods of the `org.junit.jupiter.api` class (Refer to Javadoc for a complete API)

<table>
<thead>
<tr>
<th>JUnit 5 annotation</th>
<th>Description</th>
<th>JUnit 4’s equivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>@BeforeEach</td>
<td>Method executed before each @Test in the current class</td>
<td>@Before</td>
</tr>
<tr>
<td>@AfterEach</td>
<td>Method executed after each @Test in the current class</td>
<td>@After</td>
</tr>
<tr>
<td>@BeforeAll</td>
<td>Method executed before all @Test in the current class</td>
<td>@BeforeClass</td>
</tr>
<tr>
<td>@AfterAll</td>
<td>Method executed after all @Test in the current class</td>
<td>@AfterClass</td>
</tr>
<tr>
<td>@Test</td>
<td>Define a test method</td>
<td>@Test</td>
</tr>
</tbody>
</table>
Lifecycle and Annotations

- @BeforeAll
- @BeforeEach
- @Test
- @AfterEach
- @AfterAll

```
initialize
```

Subject under test (SUT)
sometimes referred to as program under test (PUT)

```
execute
```

```
verify
```

```
teardown
```

```
test
```
Writing JUnit Tests (JUnit5)

- Download necessary jar files at `junit.org`
- Use the methods of the following classes
  - `org.junit.jupiter.api.AfterAll`
  - `org.junit.jupiter.api.AfterEach`
  - `org.junit.jupiter.api.BeforeAll`
  - `org.junit.jupiter.api.BeforeEach`
  - `org.junit.jupiter.api.Test`
  - `org.junit.jupiter.api.Assertions`

- Each test method
  - Checks a condition (assertion)
  - Reports to the test runner whether the test failed or succeeded
- The test runner uses the result to report to the user
- All of the methods return `void`
# Common Methods (JUnit 5)

<table>
<thead>
<tr>
<th>Assertions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>assertTrue(boolean condition)</code></td>
<td>Assert that a condition is true.</td>
</tr>
<tr>
<td><code>assertTrue(boolean condition, String message)</code></td>
<td>Assert that a condition is true. If the assertion is true, the string is ignored. Otherwise, the string is sent to the test engineer.</td>
</tr>
<tr>
<td><code>assertEquals(Object expected, Object actual)</code></td>
<td>Assert that two objects are equal.</td>
</tr>
<tr>
<td><code>fail(String message)</code></td>
<td>If a certain situation is expected when a certain section of code is reached, the string is sent to the test engineer. Often used to test exceptional behavior.</td>
</tr>
</tbody>
</table>

(Refer to Javadoc for a complete API)
package test;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface Test {
    String value() default;
}

import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

class LifecycleTest {
    @BeforeEach
    static void setUpBeforeClass() throws Exception {
        System.out.println("Setup all tests in the class");
    }

    @BeforeEach
    void setUp() throws Exception {
        System.out.println("Setup each test in the class");
    }

    @Test
    void testOne() {
        System.out.println("Test 1 -- be sure to use meaningful method name");
    }

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

    @AfterEach
    void tearDown() throws Exception {
        System.out.println("Teardown each test in the class");
    }

    @AfterAll
    static void tearDownAfterClass() throws Exception {
        System.out.println("Teardown all tests in the class");
    }
}
JUnit – Test Methods

1) Setup test case values

2) Execute program under test

3) Assert expected vs. actual test outputs

```java
@Test
g public void testNumZeroArrayWithNoZeros()
{
    int[] x = {1, 2, 3};
    int n = ArrayOperations.numZero(x);
    assertEquals(0, n);
}
```
Let’s do some exercises
Wrap-up

• Test automation
• Testability, Observability, Controllability
• Components of a test case
• JUnit
  • Test class, test method
  • Common methods: `assertTrue`, `assertEquals`, `fail`

What’s Next?

• More JUnit