Junit

Traits of JUnit

- JUnit is an open source framework, which is used for writing and running tests.
- Provides annotations to identify test methods.
- Provides assertions for testing expected results.
- Provides test runners for running tests.
- JUnit tests can be run automatically and they check their own results and provide immediate feedback.
- JUnit tests can be organized into test suites containing test cases and even other test suites

Local Environment Setup

- Step 1: JDK 1.5 or above
- Step 2: Download Junit jar files
- Step 3: Create Junit library (eclipse) / Maven pom.xml / build.gradle

Features of JUnit

- Fixtures
- Test suites
- Test runners
- JUnit Classes

Fixtures

- Fixtures is a fixed state of a set of objects used as a baseline for running tests.
- The purpose of a test fixture is to ensure that there is a well-known and fixed environment in which tests are run so that results are repeatable.
- setUp() method, which runs before every test invocation
- tearDown() method, which runs after every test method

Test Suites

- A test suite bundles a few unit test cases and runs them together.
- In JUnit, both @RunWith and @Suite annotation are used to run the suite test.

Test Runners

- Test runner are used for executing the test cases.

JUnit Classes

- Built-in classes used for writing tes cases
- Examples:
    - Assert
    - TestCase
    - TestResult

Assertions

- All the assertions are in the Assert class
- This class provides the assertion methods for writing test.
- Only failed assertions are recorded
- The most importan assertions:
    - assertEquals
    - assertTrue
    - assertFalse
    - assertNotNull
    - assertNull
    - assertSame
    - assertNotSame
    - assertArrayEquals

Annotations

- Annotations are like meta-tags that you can add to your code, and apply them to methods or in class.
- @Test: the testcase to run (public void)
- @Before: A method that runs before the testcase (public void). Used for preparations and test data creation
- @After: A method that runs after the testcase (public void). Used for releasing and de-allocating external resources
- @BeforeClass: The first method to run before doing any tests (public static void). Can be used to start up external resources
- @AfterClass: The very last method to run after everything finishes (public static void). Used for cleaning up
- @Ignore: Method or Class won't be run during a test run

Executing Tests

- The test cases are executed using JUnitCore class.
- JUnitCore is a facade for running tests.
- It supports running JUnit 4 tests, JUnit 3.8.x tests, and mixtures.
- See example

Suite Testing

- Test suite is used to bundle a few unit test cases and run them together.
- In JUnit, both @RunWith and @Suite annotations are used to run the suite tests.
- See example

Timeout Testing

- JUnit provides a handy option of Timeout.
- If a test case takes more time than the specified number of milliseconds, then JUnit will automatically mark it as failed.
- The timeout parameter is used along with @Test annotation.
- See example

Exception Testing

- JUnit provides an option of tracing the exception handling of code.
- You can test whether the code throws a desired exception or not.
- The expected parameter is used along with @Test annotation. Let us see @Test(expected) in action.
- See example

Parameterized Testing

- JUnit 4 has introduced a new feature called parameterized tests.
- Parameterized tests allow a developer to run the same test over and over again using different values.
- There are five steps that you need to follow to create a parameterized test:
    - Annotate test class with @RunWith(Parameterized.class).
    - Create a public static method annotated with @Parameters that returns a Collection of Objects (as Array) as test data set.
    - Create a public constructor that takes in what is equivalent to one "row" of test data.
    - Create an instance variable for each "column" of test data.
    - Create your test case(s) using the instance variables as the source of the test data.
- The test case will be invoked once for each row of data.