**Experiment 2 :  Test driven Development using JUnit**

Theory :

JUnit is a popular open-source unit testing framework for Java programming language. It is specifically designed to support and simplify the writing and running of unit tests—small, isolated tests that validate individual units (methods, functions, or classes) of source code.

JUnit provides a framework for creating test cases, defining test methods, organizing test suites, and running tests automatically. It is widely used in Java development to perform automated testing and ensure that individual units of code behave as expected.

Key features of JUnit include:

1. **Annotations:** JUnit uses annotations (such as **@Test**, **@Before**, **@After**, **@BeforeClass**, **@AfterClass**, etc.) to mark test methods and set up or tear down test fixtures.
2. **Assertions:** It offers a rich set of assertion methods (like **assertEquals**, **assertTrue**, **assertNotNull**, **assertThrows**, etc.) to verify expected outcomes and conditions in the code being tested.
3. **Test Runners:** JUnit provides test runners to execute tests and generate test results. For instance, JUnit 4 uses runners like **BlockJUnit4ClassRunner**, **Parameterized**, and **Suite** for different purposes.
4. **Test Suites:** Allows grouping of multiple test cases into test suites to run them collectively.
5. **Parameterized Tests:** Supports parameterized tests to run the same test with different input values.

JUnit follows the principles of Test-Driven Development (TDD), where tests are written before the actual code implementation. Developers create test cases to define the desired behavior of code units, then write the code to make those tests pass.

JUnit has been a fundamental tool in Java development, helping ensure the reliability and correctness of software by automating the testing process, identifying bugs early in development, and facilitating continuous integration and deployment practices. Additionally, it's widely supported by various Integrated Development Environments (IDEs) and build tools, making it a go-to choice for Java developers for unit testing.

Test-Driven Development (TDD) is a software development practice that revolves around writing tests before writing the actual code. JUnit is a popular unit testing framework for Java used in implementing TDD. Here's an overview of how to apply TDD using JUnit:

**1. Write a Failing Test:**

* Identify a small unit of functionality (e.g., a method or a specific behavior).
* Write a test case that describes the expected behavior. This test will initially fail as the functionality doesn't exist yet.

**2. Run the Test (and Watch it Fail):**

* Execute the test. As expected, it should fail because the functionality hasn't been implemented.

**3. Write the Minimum Code to Pass the Test:**

* Implement the minimal code required to make the failing test pass. Focus only on making the test pass without worrying about optimizations or additional features.

**Example (implementing Calculator class):**

**4. Run the Test (and Watch it Pass):**

* Execute the test again. This time, it should pass since the implemented code meets the test criteria.

**5. Refactor the Code (if needed):**

* Once the test passes, refactor the code to improve its structure, readability, or performance while ensuring the test still passes.

**6. Repeat the Cycle:**

* Identify the next piece of functionality or behavior to implement and write another test for it. Repeat the process by writing a failing test, writing code to pass the test, and then refactoring.

**Additional Tips:**

* Use assertions provided by JUnit (like **assertEquals**, **assertTrue**, **assertNotNull**, etc.) to verify expected behavior.
* Organize tests into different classes based on functionalities.
* Continuously run the tests to ensure new changes or additions don't break existing functionalities.

TDD with JUnit allows for the creation of comprehensive test suites that ensure code reliability and correctness while enabling developers to refactor and enhance code with confidence.