Learning Consolidation

Unit Testing With JUnit

In this sprint, you have learned to:

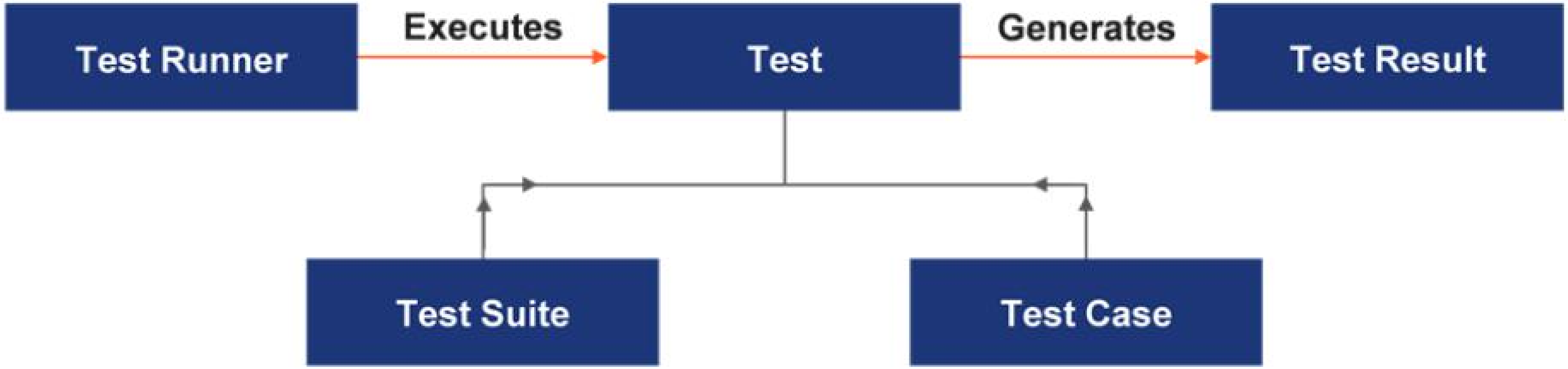
- Unit testing and its significance
- JUnit architecture
- JUnit test cases
- JUnit Assert Statements

Unit Testing and Its Significance

- Unit testing is one of the best development practices used to test smaller units of code.
- It tests the individual units of an application. A unit can be a
  - **method** of a class or
  - a complete **class**
- It ensures that even the smallest unit of code is bug-free and reusable.
- It ensures that the code functions efficiently on each unit of application.
- Unit tests help to fix bugs early in the development cycle and save time and money.

JUnit Architecture

- The architecture of JUnit refers to the process used by the JUnit framework to execute the tests and display the results.
- **Test case** : A single method that basically checks the code logic.
- **Test Suite** : Collection of multiple test cases.
- **Test Runner**: JUnit uses test runner to automatically run the test case or test suite.
- **Test Result** : Verifies the correctness of test cases and produces a test report.



JUnit Test Case

To write a JUnit test case, we have to create test classes in a different package so that the test code can be separated from the application code.

A test class can have multiple test cases.

Multiple test classes containing class-specific test cases can also be created.

JUnit Assert Statements

- **void assertEquals(boolean expected, boolean actual)** Checks that the two primitives/objects are equal.
- **void assertFalse(boolean condition)** Checks that a condition is false.
- **void assertNotNull(Object object)** Checks that an object isn't null.
- **void assertArrayEquals(expectedArray, resultArray)** The assertArrayEquals() method will test whether or not two arrays are equal to each other.
- **assertArrayEquals(String message,expected array,actual array)** Asserts that two byte arrays are equal; if they are not, an Assertion error is thrown with the given message.

JUnit Assert Statements (contd.)

- Test the functionality after a particular function has been annotated using the `@Test` annotation.
- Verify output of the function being tested with the expected output.
- **Methods in the Assert class:**
    - Verify the expected and actual results.
    - Compare the expected value with the actual value returned by the test.