JUnit

Authored by : Sangeeta Joshi ***Presented by: Sangeeta Joshi***

Agenda

- What is Unit Testing
- Benefits of Unit Testing
- Limitations of Unit Testing
- Annotations

Unit Testing

- In computer programming, **unit testing** is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use.
- one can view a unit as the smallest testable part of an application
- Ideally, each test case is independent from the others
- Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended

Benefits

- **Find problems early**

  Unit tests find problems early in the development cycle.

- **Simplifies integration**

  Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

- **Documentation**

  Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit and how to use it can look at the unit tests to gain a basic understanding of the unit's API

Limitations

Unit testing limitations

- Testing cannot be expected to catch every error in the program:
- it is impossible to evaluate every execution path in all but the most trivial programs. The same is true for unit testing.
- Unit testing by definition only tests the functionality of the units themselves. Therefore, it will not catch integration errors or broader system-level errors (such as functions performed across multiple units, or non-functional test areas such as performance).
- Unit testing should be done in conjunction with other software testing activities.
- Like all forms of software testing, unit tests can only show the presence of errors; they cannot show the absence of errors.

JUnit4 – @Before vs @BeforeClass / @After vs @AfterClass

- ***@Before:*** is used to execute set of preconditions before executing a test.. Method that is marked with @Before will be executed before executing every test in the class.

- ***@After:*** gets executed after execution of every test. If we need to reset some variable after execution of every test then this annotation can be used with a method that has the needed code.

- ***@BeforeClass :***If a JUnit test case class contains lot of tests which all together need a method which sets up a precondition and that needs to be executed before executing the Test Case class then we can utilize this annotation.

- ***@AfterClass:*** This annotation can be used to execute a method that needs to be executed after executing all the tests in a JUnit Test Case class.

Any Questions?