MEDICAPS UNIVERSITY

Introduction to JUnit

Laxmikant Soni

JUnit

- A tool for test-driven development

History

- Kent Beck developed the first xUnit automated test tool for Smalltalk in mid-90’s
- Beck and Gamma (of GoF) developed JUnit on a flight from Zurich to Washington, D.C.
- Martin Fowler: “Never in the field of software development was so much owed by so many to so few lines of code.”

History

- JUnit became the standard for TDD in Java
- IDE support: Eclipse, BlueJ, JBuilder, DrJava
- Other xUnit tools: Perl, C++, Python, VB, C#, ...

Why create a test suite?

- You need to test your code
- Ad hoc testing vs. building a reusable test suite

Why create a test suite?

Ad hoc testing
- Quick, unplanned checks
- Relies on developer memory and intuition
- Risk of missing edge cases
- Difficult to repeat consistently

Why create a test suite?

Test suite
- Organized, reusable set of test cases
- Can be automated and run anytime
- Ensures consistency across versions
- Facilitates regression testing
- Provides documentation of intended behavior
- Increases confidence before deployment

Why create a test suite - Advantages

Advantages:

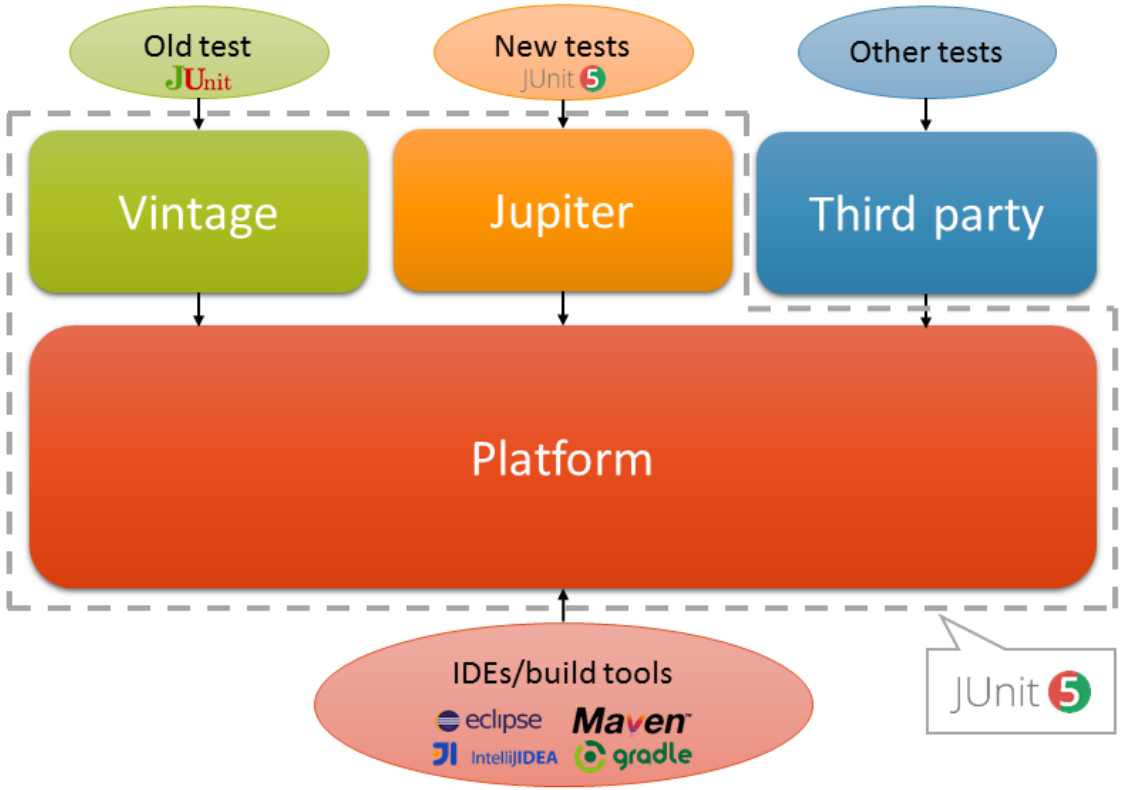
- Fewer bugs in delivered code
- Easier maintenance & refactoring
- Faster debugging: failures point directly to the issue
- Provides safety net when adding new features
- Encourages modular, cleaner code design

Why create a test suite - Advantages

Advantages:

- Improves developer confidence in making changes
- Enables continuous integration (CI) and automated testing pipelines
- Acts as executable documentation of system behavior
- Reduces overall development cost by catching errors early

Architectural Overview



JUnit 5 Architecture

Writing a TestCase

```
public class CounterTest extends
junit.framework.TestCase {
    Counter counter1;

    protected void setUp() {
        counter1 = new Counter();
    }

    public void testIncrement() {
        assertTrue(counter1.increment() == 1);
        assertTrue(counter1.increment() == 2);
    }

    public void testDecrement() {
        assertTrue(counter1.decrement() == -1);
    }
}
```

Assert Methods

Method	Description
`assertTrue(String msg, Boolean test)`	Verifies that the condition is `true`
`assertFalse(String msg, Boolean test)`	Verifies that the condition is `false`
`assertNull(String msg, Object obj)`	Verifies that the object is `null`
`assertNotNull(String msg, Object obj)`	Verifies that the object is not `null`
`assertEquals(String msg, Object expected, Object actual)`	Verifies that two objects are equal (uses `equals`)
`assertSame(String msg, Object expected, Object actual)`	Verifies that two objects reference the same instance (==)
`assertNotSame(String msg, Object expected, Object actual)`	Verifies that two objects do **not** reference the same instance

TestSuites

- Collect a selection of tests into a unit
- Automatic in most IDEs
- Can be written manually if needed

```
public static Test suite() {
    TestSuite suite = new TestSuite();
    suite.addTestSuite(TestBowl.class);
    suite.addTestSuite(TestFruit.class);
    suite.addTestSuite(CounterTest.class);
    return suite;
}
```

Summary

- JUnit is a widely-used testing framework for Java that enables **unit testing** and **test-driven development (TDD)**.
- Provides annotations, assertions, and test suites for **organized and automated testing**.
- Supports integration with IDEs and build tools like **Gradle** and **Maven** for seamless workflow.
- Promotes **code quality, reliability, and maintainability** by detecting issues early.
- Understanding JUnit is essential for **robust Java application development**.