



Unit testing

JUnit 4
JUnit 5

Sunil Ghargaonkar

What's wrong with JUnit 4



Quiet Old



Java Updates



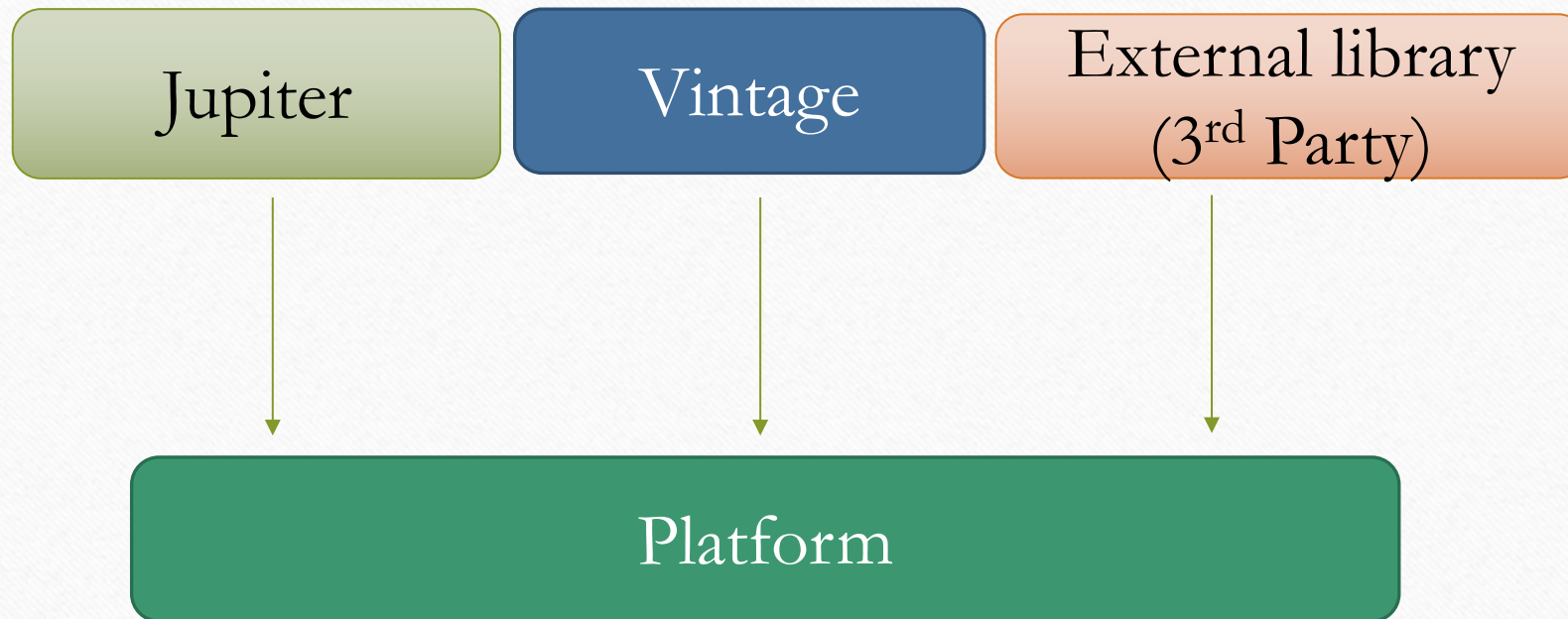
Architecture



Bugs & Feature
requests

Sunil Ghargaonkar

Unit 5 Architecture



Sunil Ghargaonkar

JUnit 5

Vs

JUnit 4

Sunil Ghargaonkar

- | | |
|----------------|--------------------|
| • @Before | @BeforeEach |
| • @After | @AfterEach |
| • @BeforeClass | @BeforeAll |
| • @AfterClass | @AfterAll |
| • @Ignore | @Disabled |
| • @Category | @Tag |
| • @RunWith | @ExtendWith |
| • @Rule | @ExtendWith |
| • @ClassRule | @RegisterExtension |

Prominent Features



DISPLAY NAME



ASSERT ALL



NESTED
ANNOTATION



PARAMETERIZATION



REPEAT TESTS

Sunil Ghargaonkar

Display Name

Name your tests with
@DisplayName to maintain
readability

Sunil Ghargaonkar

```
class CustomerVerificationTest {  
    final long accountNumber = 1111;  
    final long pin = 1234;  
    final long accountBalance = 1000;  
  
    @Test  
    @DisplayName("Given Account number exist in DB login is successful")  
    void whenAccountNumberAndPinMatchesThenCustomerShouldBeVerifiedSuccessfully() {...}  
  
    @Test  
    @DisplayName("Given Account number exist in DB but pin doesn't match Then login is not allowed")  
    void whenAccountNumberExistsInDBButPinDoesntMatchThenCustomerShouldNotVerified() {...}  
  
    @Test  
    @DisplayName("Given Account number doesn't exist in DB Then login is not allowed")  
    void whenAccountNumberDoesntExistsInDBThenCustomerShouldNotVerified() {...}  
    @Test  
    @DisplayName("Given DB doesn't have record Then login is not allowed")  
    void whenAccountNumberDoesNotProvidedThenCustomerShouldNotVerified() {  
        final CustomerVerification customerVerification = new CustomerVerification();  
  
        assertFalse(customerVerification.isValidCustomer( accountNumber: accountNumber + 1, pin));  
    }  
}
```

in: CustomerVerificationTest x	
✓ Test Results	238 ms
✓ CustomerVerificationTest	238 ms
✓ Given Account number exist in DB login is successful	48 ms
✓ Given DB doesn't have record Then login is not allowed	183 ms
✓ Given Account number doesn't exist in DB Then login is not allowed	5 ms
✓ Given Account number exist in DB but pin doesn't match Then login is not allowed	2 ms


```

@Test
@DisplayName("Verify Assert All Functionality")
void assertAllFunctionalityVerify() {
    assertAll(
        () → assertTrue( condition: 3 < 4, message: "Seems like 3 is > 4 :p"),
        () → assertEquals( expected: 4, actual: 2, message: "of course 4 ≠ 2, what else you expect"),
        () → assertFalse("4".equals(2*1), message: "Buggy calculator saying 2*1 = 4 :p"),
        () → assertEquals( expected: 1, actual: 2, message: "of course 1 ≠ 2, what else you expect")
    )
}

```

Run: AssertAllExample x

Tests failed: 1 of 1 test - 15 ms

Test Results

- AssertAllExample
 - Verify Assert All Functionality
 - of course 4 ≠ 2, what else you expect ⇒ expected: <4> but was: <2>
Comparison Failure:
Expected :4
Actual :2
[Click to see difference](#)
 - of course 1 ≠ 2, what else you expect ⇒ expected: <1> but was: <2>
Comparison Failure:
Expected :1
Actual :2
[Click to see difference](#)

Assert All

Run set of assertions
together with `@assertAll`

Sunil Ghargaonkar

```

public class NestedExample {
    @BeforeEach
    @DisplayName("BeforeEach→ Outside Nested Group")
    void beforeEachTest(){
        System.out.println("BeforeEach→ Outside Nested Group");
    }

    @Test
    @DisplayName("Inside Single Test ")
    void acceptDelivery() {
        System.out.println("Inside Single Test ");
    }

    @Nested
    @DisplayName("Nested Test Group")
    class NonSPFTests {
        @BeforeEach
        @DisplayName("BeforeEach→ Inside Nested class")
        void beforeallNestedClassTest(){
            System.out.println("BeforeEach→ Inside Nested class");
        }

        @Test
        @DisplayName("Inside Nested Group Test -1")
        void nonSpfAuthSpoofQA() {
            System.out.println("Inside Nested Test-1 ");
        }

        @Test
        @DisplayName("Inside Nested Group Test -2")
        void nonSpfAuthSpoofNotAuthorisedQA() {
            System.out.println("Inside Nested Test-2 ");
        }
    }
}

```

```

Run: NestedExample x
Test Results 8 ms
  ✓ NestedExample 8 ms
    ✓ Inside Single Test 6 ms
    ✓ Nested Test Group 3 ms
      ✓ Inside Nested Group Test -2 2 ms
      ✓ Inside Nested Group Test -1 1 ms
    BeforeEach→ Outside Nested Group
    Inside Single Test
    AfterEach→ Outside Nested Group

    BeforeEach→ Outside Nested Group
    BeforeEach→ Inside Nested class
    Inside Nested Test-2
    AfterEach→ Inside Nested class
    AfterEach→ Outside Nested Group

    BeforeEach→ Outside Nested Group
    BeforeEach→ Inside Nested class
    Inside Nested Test-1
    AfterEach→ Inside Nested class
    AfterEach→ Outside Nested Group

Process finished with exit code 0

```

Nested Annotation

Maintain the relationship among several groups of tests with `@Nested` annotation

Parameterized tests

- Run same tests multiple times with different arguments using [@ParameterizedTest](#) annotation.

```
@DisplayName("JUnit 5 Features")
public class JunitDemo {

    @ParameterizedTest
    @ValueSource(strings = { "racecar", "radar", "able was I ere I saw elba", "sunil" })
    @DisplayName("Parametrized Tests Example")
    void palindromes(String candidate) {
        assertTrue(isPalindrome(candidate));
    }

    private boolean isPalindrome(String candidate) {
        String rev = "";
        int length = candidate.length();

        for ( int i = length - 1; i >= 0; i-- )
            rev = rev + candidate.charAt(i);

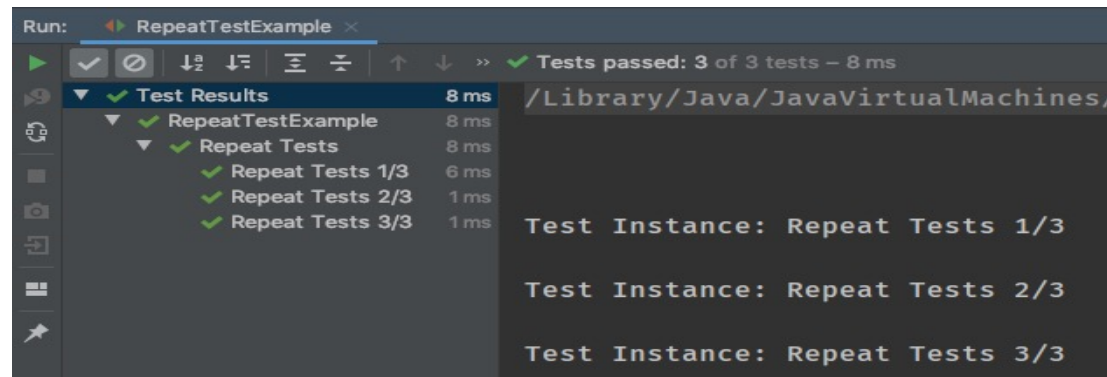
        if (candidate.equals(rev))
            return true;
        else
            return false;
    }
}
```

Run: JunitDemo.palindromes		
Test Results		
JUnit 5 Features		33 ms
palindromes(String)		33 ms
[1] racecar		29 ms
[2] radar		1 ms
[3] able was I ere I saw elba		
[4] sunil		3 ms

Repeat Tests

- Execute same tests multiple times using `@RepeatedTest` annotation

```
public class RepeatTestExample {  
    @RepeatedTest(value = 3, name = "{displayName} {currentRepetition}/{totalRepetitions}")  
    @DisplayName("Repeat!")  
    void customDisplayName(TestInfo testInfo) {  
        System.out.println("Test Instance: " + testInfo.getDisplayName());  
    }  
}
```



Run: RepeatTestExample	
✓	Tests passed: 3 of 3 tests – 8 ms
✓	Test Results 8 ms
✓	RepeatTestExample 8 ms
✓	Repeat Tests 8 ms
✓	Repeat Tests 1/3 6 ms
✓	Repeat Tests 2/3 1 ms
✓	Repeat Tests 3/3 1 ms
	Test Instance: Repeat Tests 1/3
	Test Instance: Repeat Tests 2/3
	Test Instance: Repeat Tests 3/3



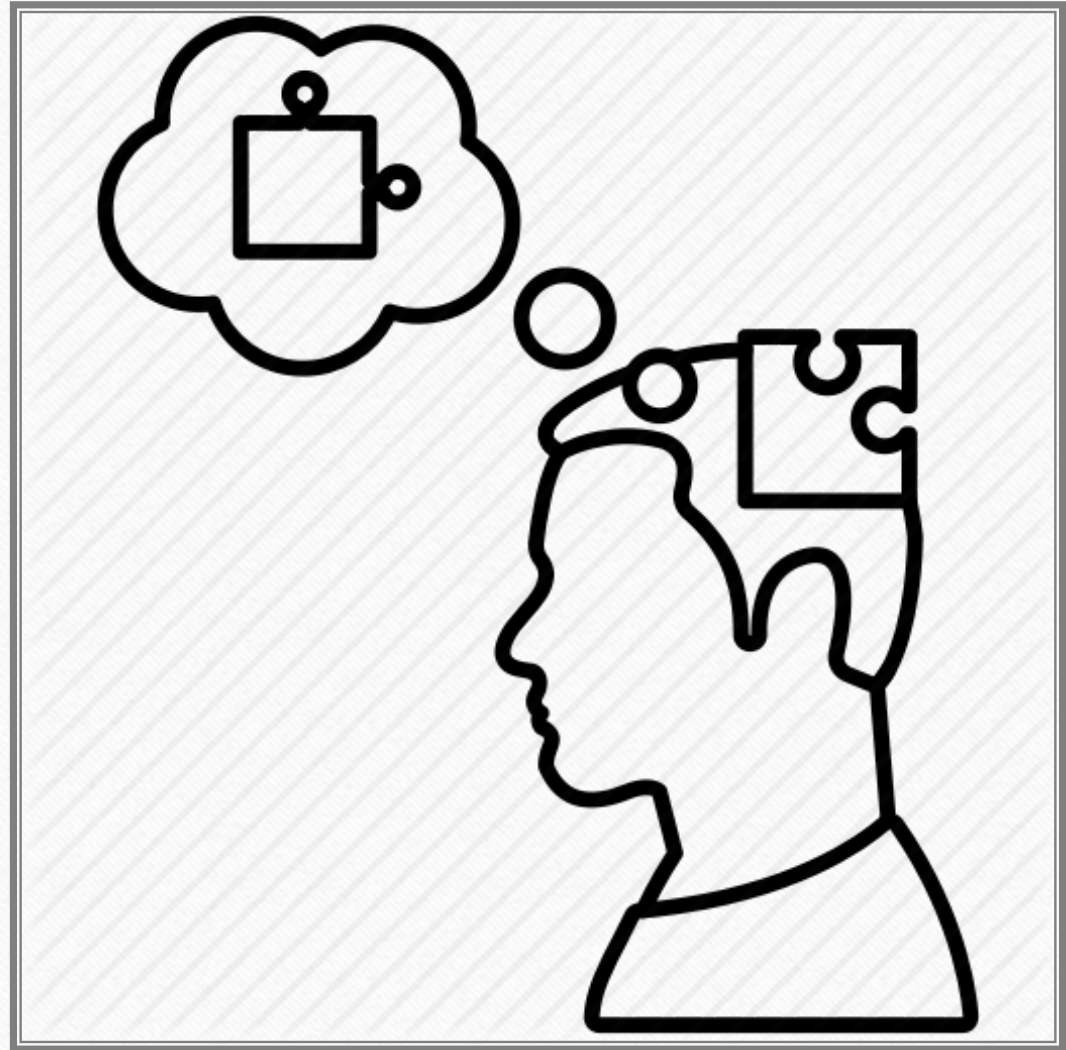
More cool stuff

- Conditional Test Execution
- ExtendWith (Allows multiple runners)
- TestFactory
- Test Execution Order

Sunil Ghargaonkar

JUnit 4 or JUnit 5

What's your thought?





Thank You