

Intro to Testing with JUnit 5

Jeanne Boyarsky
Sept 17, 2021
KCDC

speakerdeck.com/boyarsky

Change
this report to add a field?
No problem! Wait. What is this
supposed to do?



This
is taking so long. Ok got it.
My customers will be happy
now.



The
numbers on the report are
wrong. How could you let this
happen!!!?



Uh oh.
I wish I had used JUnit!



- JUnit 5?
- JUnit 4?
- None

About Me



- Java Champion
- Author
- Developer at NYC bank for 19+ years
- FIRST Robotics Mentor

@jeanneboyarsky

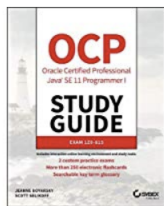
Pause for a Commercial

Amazon Best Sellers

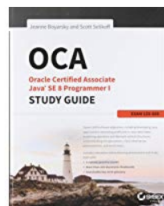
Our most popular products based on sales. Updated hourly.

Best Sellers in Oracle Certification

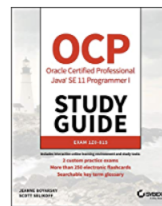
#1



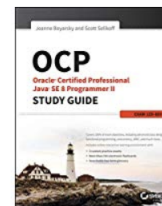
#2



#3



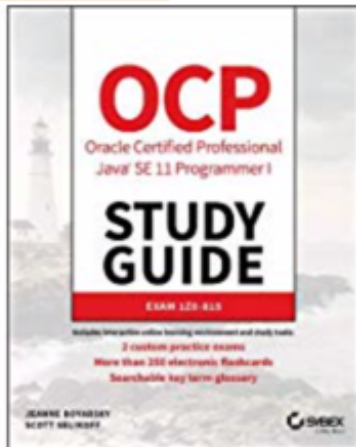
#4



#5



#1 New Release



Java certs

- Java 8
- Java 11
- Java 17 next

Book giveaway at end!

@jeanneboyarsky

Another Commercial

Titanium Sponsors



Platinum Sponsors



Gold Sponsors



After this presentation

- Presentation:

<http://speakerdeck.com/boyarsky>

- Samples and code we write together:

<https://github.com/boyarsky/2021-kcdc-junit5>

Agenda

★ Intro to JUnit 5

- Common testing patterns
- Brief look back at history (JUnit 3 and 4)
- JUnit 5 extensions
- Interactive TDD live coding

- Released Sept 10, 2017
- Requires Java 8
- Today sprinkling in Java 9+ features



Basic Flow

```
class ReportTest {
```

No need to
make public

```
    private Report report;
```

```
    @BeforeEach
```

Runs before
each test

```
    void setUp() {  
        report = new Report();  
    }
```

```
    @Test
```

Can have 1+ tests

```
    void createRow() {  
  
    }  
}
```

Common Class name patterns

Unit Tests	Integration Tests
Test*.java	IT*.java
*Test.java	*IT.java
*Tests.java	*ITCase.java
*TestCase.java	

Common test name examples

Method names

testCreateRow()

createRow()

createRow_forNullParams()

shouldHandleNullParams()

Assert Equals

```
@Test
void createRow() {
    String expected =
        "Jeanne Boyarsky,Intro to JUnit 5";
    String actual = report.createRow(
        "Jeanne Boyarsky", "Intro to JUnit 5");

    assertEquals(expected, actual, "row");
}
```

Order matters



Optional message



Message on failing assert

Message



```
row ==> expected: <Jeanne Boyarsky>  
      but was: <Jeanne Boyarsky,Intro to JUnit 5>  
Expected  :Jeanne Boyarsky  
Actual    :Jeanne Boyarsky,Intro to JUnit 5
```

Remember that order matters!

BeforeAll

```
class ReportWithCacheTest {
```

```
    private static Map<String, String> CACHE;  
    private ReportWithCache report;
```

```
    @BeforeAll ←————— Runs once
```

```
    static void createCache() {  
        CACHE = Map.of("Intro to JUnit 5",  
                       "Jeanne Boyarsky");  
    }
```

static

```
    @BeforeEach  
    void setUp() {  
        report = new ReportWithCache(CACHE);  
    }
```

BeforeAll

```
@Test
void createRow_match() {
    String expected =
        "Jeanne Boyarsky,Intro to JUnit 5";
    String actual = report.createRow(
        "Intro to JUnit 5");
    assertEquals(expected, actual, "row");
}

@Test
void createRow_noMatch() {
    String expected = "Unknown,Lunch";
    String actual = report.createRow("Lunch");
    assertEquals(expected, actual, "default value");
}
```

@BeforeEach runs twice
@BeforeAll runs once

Review: fill in the blanks

```
class BlanksTest {  
    private String z;  
  
    @ BeforeAll  
    static void a() {  
        // lookup next id  
    }  
    @ AfterAll  
    static void b() { Target.cache.clear(); }  
    @ BeforeEach  
    void c() { z = "test data"; }  
    @ AfterEach  
    void d() { z = null; }  
    @ Test  
    void e() { assertNotNull(z); }  
}
```


Review: what is output?

```
class FlowTest {  
    private static String flow = "";  
    @BeforeAll  
    static void before() { flow+="x"; }  
    @AfterAll  
    static void after() { System.out.println(flow); }  
    @BeforeEach  
    void setUp() { flow+="y"; }  
    @AfterEach  
    void tearDown() { flow+="-"; }  
    @Test  
    void one() { flow+="z"; }  
    @Test  
    void two() { flow+="z"; }  
}
```

XYZ-VZ-

How about now?

```
class FlowTest {  
    private static String flow = "";  
    @BeforeAll  
    static void before() { flow+="x"; }  
    @AfterAll  
    static void after() { System.out.println(flow); }  
    @BeforeEach  
    void setUp() { flow+="y"; }  
    @AfterEach  
    void tearDown() { flow+="-"; }  
    @Test  
    void one() { flow+="1"; }  
    @Test  
    void two() { flow+="2"; }  
}
```

“deterministic but
non-obvious” order
do not rely on it

If you really need order

```
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
class OrderedFlowTest {

    private static String flow = "";

    @AfterAll
    static void after(){ System.out.println(flow); }

    @Test
    @Order(1)
    void one() { flow+="1"; }

    @Test
    @Order(2)
    void two() { flow+="2"; }
}
```

12

Are you *sure* you
need order?

Assertions without Lambdas

assertTrue	assertEquals	assertIterableEquals
assertFalse	assertNotEquals	assertLinesMatch
assertNull	assertArrayEquals	assertSame
assertNotNull		assertNotSame

Assert Examples

```
assertNotNull("");
```

```
assertTrue("").isEmpty());
```

```
assertNotSame("", new String(""));
```

```
assertLinesMatch(List.of("xyz"),  
    Arrays.asList("xyz"));
```

All have optional last
parameter with message

Review: fill in the blanks

```
@Test
void asserts() {
    String message = "sum";

    int sum = Adder.sum(1, 2, 3);

    assertTrue (sum == 6);

    assertEquals (6, sum, message );
}
```


assertTrue vs assertEquals

org.opentest4j.AssertionFailedError:
expected: <true> but was: <false>

org.opentest4j.AssertionFailedError:
sum ==> expected: <6> but was: <7>

Use most specific
assertion you can

Agenda



Intro to JUnit 5



Common testing patterns

- Brief look back at history (JUnit 3 and 4)
- JUnit 5 extensions
- Interactive TDD live coding

assertThrows

```
@Test
void throwsException() {
    assertThrows(IllegalArgumentException.class,
        () -> Exceptions.validate(null));
}
```

org.opentest4j.AssertionFailedError:
Expected java.lang.IllegalArgumentException to be thrown,
but nothing was thrown.

Which Better?

```
@Test
void throwsException() {
    assertThrows(IllegalArgumentException.class,
        () -> Exceptions.validate(null));
}
```

```
@Test
void message() {
    IllegalArgumentException actual =
        assertThrows(IllegalArgumentException.class,
            () -> Exceptions.validate(null));
    assertEquals("str can't be null",
        actual.getMessage());
}
```

assertTimeout

```
@Test
void tooLong() {
    assertTimeout(Duration.ofSeconds(2),
        this::sketchyCode);
}

private void sketchyCode() {
    try {
        Thread.sleep(5_000);
    } catch (InterruptedException e) {
        // ignore
    }
}
```

org.opentest4j.AssertionFailedError:
execution exceeded timeout of 2000 ms by 3005 ms

assertAll - groups

```
@Test
void bean() {
    Bean bean = new Bean("J", "B");
    assertAll("name",
        () -> assertEquals("Jeanne",
            bean.getFirstName(), "first"),
        () -> assertEquals("Boyarsky",
            bean.getLastName(), "last"));
}
```

DefaultMultiCauseException: name (2 failures)
org.opentest4j.AssertionFailedError: first ==>
expected: <Jeanne> but was: <J>
org.opentest4j.AssertionFailedError: last ==>
expected: <Boyarsky> but was:

Side note - records

```
public record Record (String firstName,  
    String lastName) { }
```

```
@Test  
void record() {  
    Record record = new Record("J", "B");  
    Record expected = new Record(  
        "Jeanne", "Boyarsky");  
    assertEquals(expected, record, "name");  
}
```

org.opentest4j.AssertionFailedError: name ==> expected:
<Record[firstName=Jeanne, lastName=Boyarsky]> but was:
<Record[firstName=J, lastName=B]>

assertAll - dependencies

```
@Test
void bean() {
    Bean bean = null;
    assertAll("found name",
        () -> {
            assertNotNull(bean);
            assertAll("name values",
                () -> assertEquals("Jeanne",
                    bean.getFirstName(), "first"),
                () -> assertEquals("Boyarsky",
                    bean.getLastName(), "last"));
        }
    );
}
```

Doesn't run
inner assertAll

semicolons
because
lambda blocks

```
org.opentest4j.AssertionFailedError: expected: not <null>
org.opentest4j.MultipleFailuresError: found name (1 failure)
}
org.opentest4j.AssertionFailedError: expected: not <null>
```

Assuming

```
@Test
void bean() {
    Bean bean = null;
    assumeFalse(bean == null, "no name");

    assertEquals("Jeanne",
        bean.getFirstName(), "first");
    assertEquals("Boyarsky",
        bean.getLastName(), "last");
}
```

Reports test is ignored

Remember - simplest approach is best

Assumptions

Annotation	Parameters
<code>assumeTrue</code>	boolean or BooleanSupplier optional message
<code>assumeFalse</code>	boolean or BooleanSupplier optional message
<code>assumingThat</code>	boolean or BooleanSupplier Executable (runs the Executable only if the assumption is true)

Running only on Linux

```
@Test
void packagedOnServer() {
    assertTrue(System.getProperty("os.name")
        .toLowerCase().contains("linux"),
        "skip if not linux");

    // assertions here
}
```

Also reports as ignored

```
@Test
@EnabledOnOs(OS.LINUX)
void linuxAnnotation() {

    // assertions here
}
```

Simplest approach is still best

OS Annotations

Annotations	OS	
EnabledOnOS	WINDOWS	AIX
DisabledOnOS	MAC	SOLARIS
	LINUX	OTHER

```
@EnabledOnOs ( LINUX )
```

```
@EnabledOnOs ( { OS.LINUX, OS.MAC } )
```


Disabling Entirely

```
@Disabled
@Test
void uhOh() {

    // assertions here

}
```

Also reports as ignored

Display Names

```
@DisplayName("Requirement 123")
class DisplayNameTest {

    @Test
    @DisplayName("use case 1")
    void negativeNumber() {

    }

}
```

✓	✓ Test Results	22 ms
▼	✓ Requirement 123	22 ms
	✓ use case 1	22 ms

Tags & Build Tools

```
@Tag("slow")  
class TagTest {
```

```
    @Test  
    @Tag("full-suite")  
    void test() {  
    }  
}
```

```
<plugin>  
    <artifactId>  
        maven-surefire-plugin  
    </artifactId>  
    <configuration>  
        <groups>slow</groups>  
    </configuration>  
</plugin>
```

```
test {  
    useJUnitPlatform {  
        excludeTags 'slow', 'abc'  
    }  
}
```


Tags in the IDE

-ea

Tags ▼

Press ⌘ for field hints

All JForum Unit Tests

est Arguments JRE Classpath Source Environment Common

Run a single test

Project: JForum Browse...

Test class: Search...

Test method: (all methods) Search...

Run all tests in the selected project, package or source file

src/test/java

Include and exclude tags: Configure...

runner: JUnit 5

Keep JUnit running after a test run when debugging

Show Command Line Revert Apply

Configure Tags

☐ Include Tags
Newline separated tags to be included in the test run:

☐ Exclude Tags
Newline separated tags to be excluded from the test run:

Cancel OK

Repeated Test

```
@RepeatedTest(100)  
void threadsDoNotFail() {  
  
}
```

Runs 100 times.
Fails test if any fail

Testing System.out.println

```
class HelloWorldTest {  
    private PrintStream originalSystemOut;  
    private ByteArrayOutputStream mockSystemOut;  
    @BeforeEach  
    void setUp() {  
        mockSystemOut = new ByteArrayOutputStream();  
        System.setOut(new PrintStream(mockSystemOut));  
    }  
    @AfterEach  
    void restoreSystemOut() {  
        System.setOut(originalSystemOut);  
    }  
    @Test  
    void prints() {  
        HelloWorld.main();  
        String actual = mockSystemOut.toString();  
        assertEquals("Hello World", actual.strip());  
    }  
}
```

Warning: be
careful if running
multi-threaded

Agenda

- ✓ Intro to JUnit 5
- ✓ Common testing patterns
- ★ Brief look back at history
(JUnit 3 and 4)
 - JUnit 5 extensions
 - Interactive TDD live coding

	JUnit 3.X	JUnit 4.X	JUnit 5.X
Released	?	2006	2017
Superclass	TestCase	No	No
Package	junit .framework	org.junit	org.junit .jupiter....
Minimum Java version	?	5	8

	JUnit 3.X	JUnit 4.X	JUnit 5.X
assertEquals	[message,] expected, actual	[message,] expected, actual	expected, actual [,message]
Setup method name	setUp()	any	any
Access control	public	public	package private or higher

	JUnit 3.X	JUnit 4.X	JUnit 5.X
Test names	test*	Any	Any
Skipping a test	Comment it out	@Ignore	@Disabled

Agenda

- ✓ Intro to JUnit 5
- ✓ Common testing patterns
- ✓ Brief look back at history (JUnit 3 and 4)
- ★ JUnit 5 extensions
 - Interactive TDD live coding

Parameterized Tests

Dependency

testImplementation

```
'org.junit.jupiter:junit-jupiter-params:5.7.2'
```


```
enum Env { DEV, QA, PROD};
```

```
@ParameterizedTest
```

```
@EnumSource(Env.class)
```

```
void byEnum(Env env) {
```

```
}
```



✓	Test Results	50 ms
✓	ParamsTest	50 ms
✓	byEnum(Env)	39 ms
✓	[1] DEV	36 ms
✓	[2] QA	1 ms
✓	[3] PROD	2 ms
✓	byMethodElsewhere(String)	1 ms
✓	[1] Conf	1 ms
✓	byValue(String)	2 ms
✓	[1] SpaceX	1 ms
✓	[2] Boeing	1 ms
✓	byMethod(Double)	8 ms
✓	[1] 0.295782212817267	1 ms
✓	[2] 0.45260921881877	1 ms
✓	[3] 0.38916748710577	0 ms
✓	[4] 0.04555551952654	2 ms
✓	[5] 0.89066711451509	2 ms
✓	[6] 0.76341466052043	0 ms
✓	[7] 0.159716317168808	1 ms
✓	[8] 0.54929210906793	1 ms
✓	[9] 0.21516974673209	0 ms
✓	[10] 0.4265391463718	0 ms

Parameterized Tests

```
@ParameterizedTest  
@ValueSource(strings = { "SpaceX", "Boeing" })  
void byValue(String company) {  
  
}
```

Supports all primitives, strings, and classes

Parameterized Tests

```
static Stream<Double> data() {  
    return Stream.generate(  
        () -> Math.random()).limit(10);  
}
```

```
@ParameterizedTest  
@MethodSource("data")  
void byMethod(Double number) {  
}
```

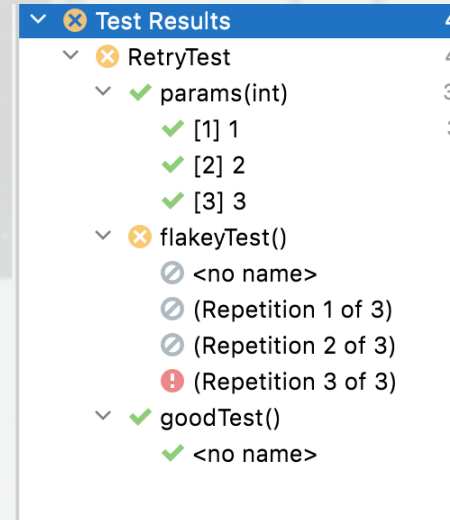
```
@ParameterizedTest  
@MethodSource("com.jeanneboyarsky.kcdc.junit5." +  
    "extensions.Enabled#getEnabled")  
void byMethodElsewhere(String name) {  
}
```

Retry Failed Tests

Dependency

```
testImplementation  
'io.github.artsok:rerunner-jupiter:2.1.6'
```

```
@RepeatedIfExceptionsTest(repeats = 3)  
void flakeyTest() {  
    throw new RuntimeException("Uh oh");  
}  
  
@RepeatedIfExceptionsTest(repeats = 3)  
void goodTest() { }  
  
@ParameterizedRepeatedIfExceptionsTest  
@ValueSource(ints = {1,2,3})  
void params(int num) { }
```



Hamcrest Matchers

Dependency

```
testImplementation 'org.hamcrest:hamcrest-library:2.2'
```

```
import org.junit.jupiter.api.Test;

import static org.hamcrest.CoreMatchers.containsString;
import static org.hamcrest.MatcherAssert.assertThat;

public class HamcrestTest {

    @Test
    void matcher() {
        assertThat("kc", containsString("kc"));
    }
}
```

Common Matchers

Type	Examples
Combiners	anyOf, allOf, both, either
Lists	hasItem, is
By type	any
Equality	equalTo, nullValue, notNull, containsString, startsWith, endsWith

Mockito

Dependency

```
testImplementation 'org.mockito:mockito-junit-jupiter:3.12.1'  
testImplementation 'org.mockito:mockito-core:3.12.1'
```

```
@ExtendWith(MockitoExtension.class)  
public class MockingTest {  
    private Processor processor;  
  
    @Mock  
    Function functionMock;  
  
    @BeforeEach  
    void setUp() {  
        processor = new Processor(functionMock);  
    }  
}
```


Mockito

```
@Test
void process() {
    String text = "abc";
    int expected = 42;
    when(functionMock.apply(text))
        .thenReturn(expected);

    int actual = processor.process(text);
    assertEquals(expected, actual, "result");
    verify(functionMock, times(1)).apply(text);
}
}
```

Common Verifiers

Examples

`atLeastOnce()`, `atLeast(int)`

`atMostOnce()`, `atMost()`

`times()`

`never()`

Also useful:
`@Mock(lenient=true)`

JUnit Pioneer

Dependency

```
testImplementation 'org.junit-pioneer:junit-pioneer:1.4.2'
```

- Retry on failure (no param version)
- Threadsafe scheduler for System.in/out
- And so much more

JUnit Pioneer

Dependency

```
testImplementation 'org.junit-pioneer:junit-pioneer:1.4.2'
```

- Retry on failure (no param version)
- Threadsafe scheduler for System.in/out
- And so much more

Converting from JUnit 4

- IntelliJ does this well
- Before that, I wrote a program which was adopted:

[https://github.com/junit-pioneer/
convert-junit4-to-junit5](https://github.com/junit-pioneer/convert-junit4-to-junit5)

Agenda

- ✓ Intro to JUnit 5
- ✓ Common testing patterns
- ✓ Brief look back at history
(JUnit 3 and 4)
- ✓ JUnit 5 extensions
- ★ Interactive TDD live coding

Live TDD

- Let's write some code!
- Then book giveaway