Documenting Your Code with Tests

Billy Korando
SENIOR SOFTWARE CONSULTANT - KEYHOLE SOFTWARE
@BIllyKorando

Agenda

Tests as documentation

Introduction to JUnit 5

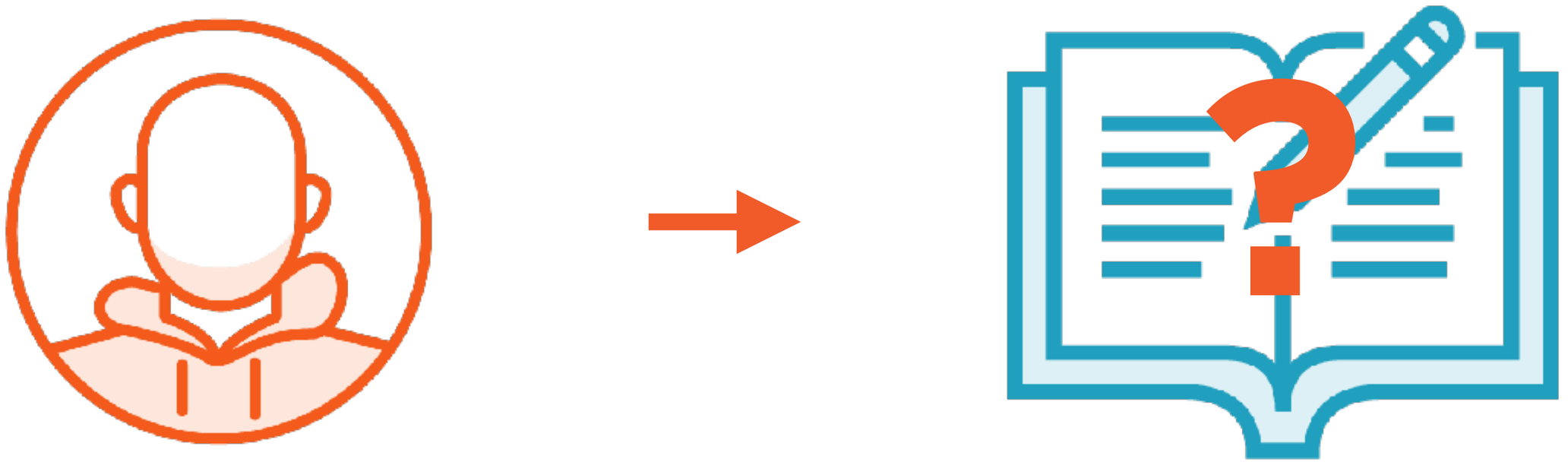
Acceptance testing and behavior driven development

Tools and practices for writing readable tests

Tools for generating documentation from tests

Tests as Documentation

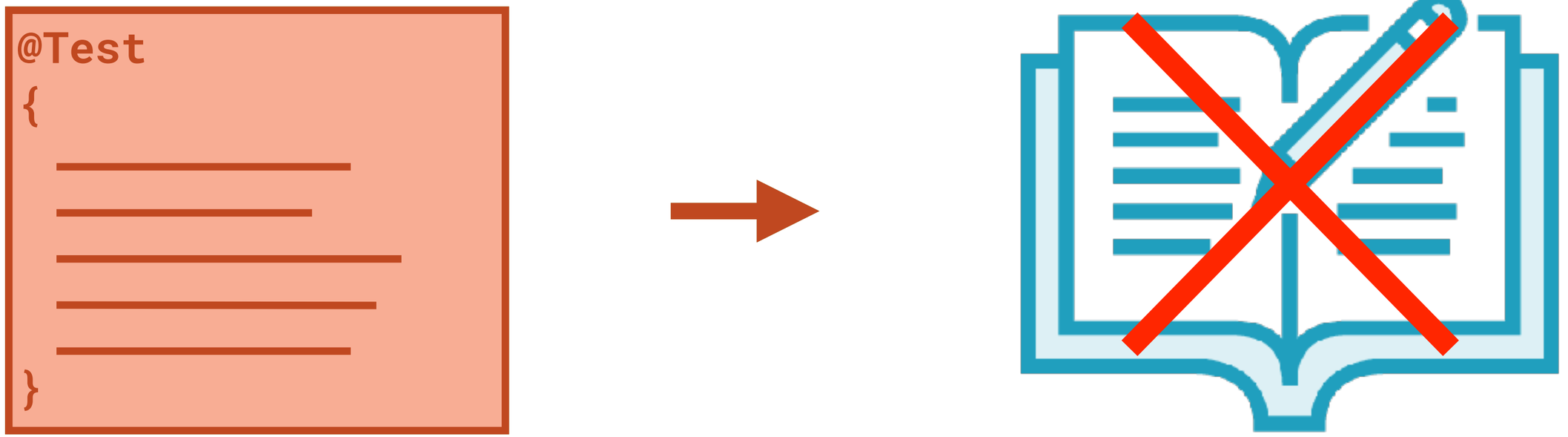
Tests as Documentation



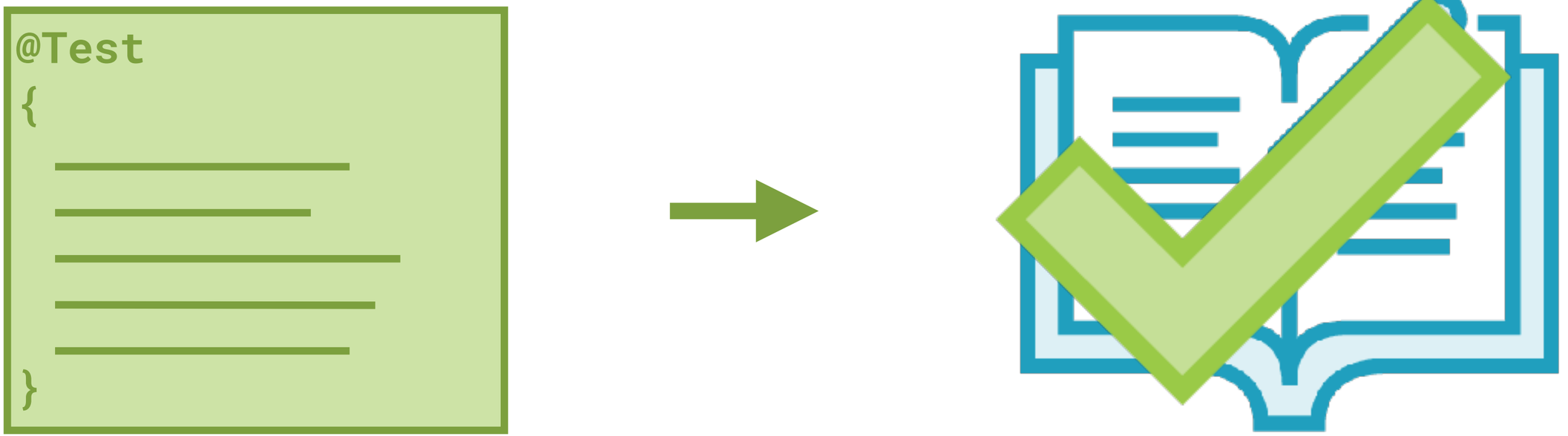
Tests as Documentation

```
@Test
executeThisScenario
{
 //given
 thesePreconditions()
 //when
 thisActionOccurs()
 //then
 thisShouldHappen()
}
```

Tests as Documentation



Tests as Documentation



Tests as Documentation

- Won't cover all documentation requirements
- Documentation remains in sync
- Further increases value of writing automated tests

Welcome to JUnit 5

JUnit 4

- Released in 2006
  - Java 5 was current Java release
- Used annotations to mark tests

JUnit 5

- Java 8 baseline
- Native support for lambdas
- Provide extension points rather than features
- Readability enhancements
- Modularized

Welcome to JUnit 5

Vintage

Supports legacy JUnit programming model

Platform

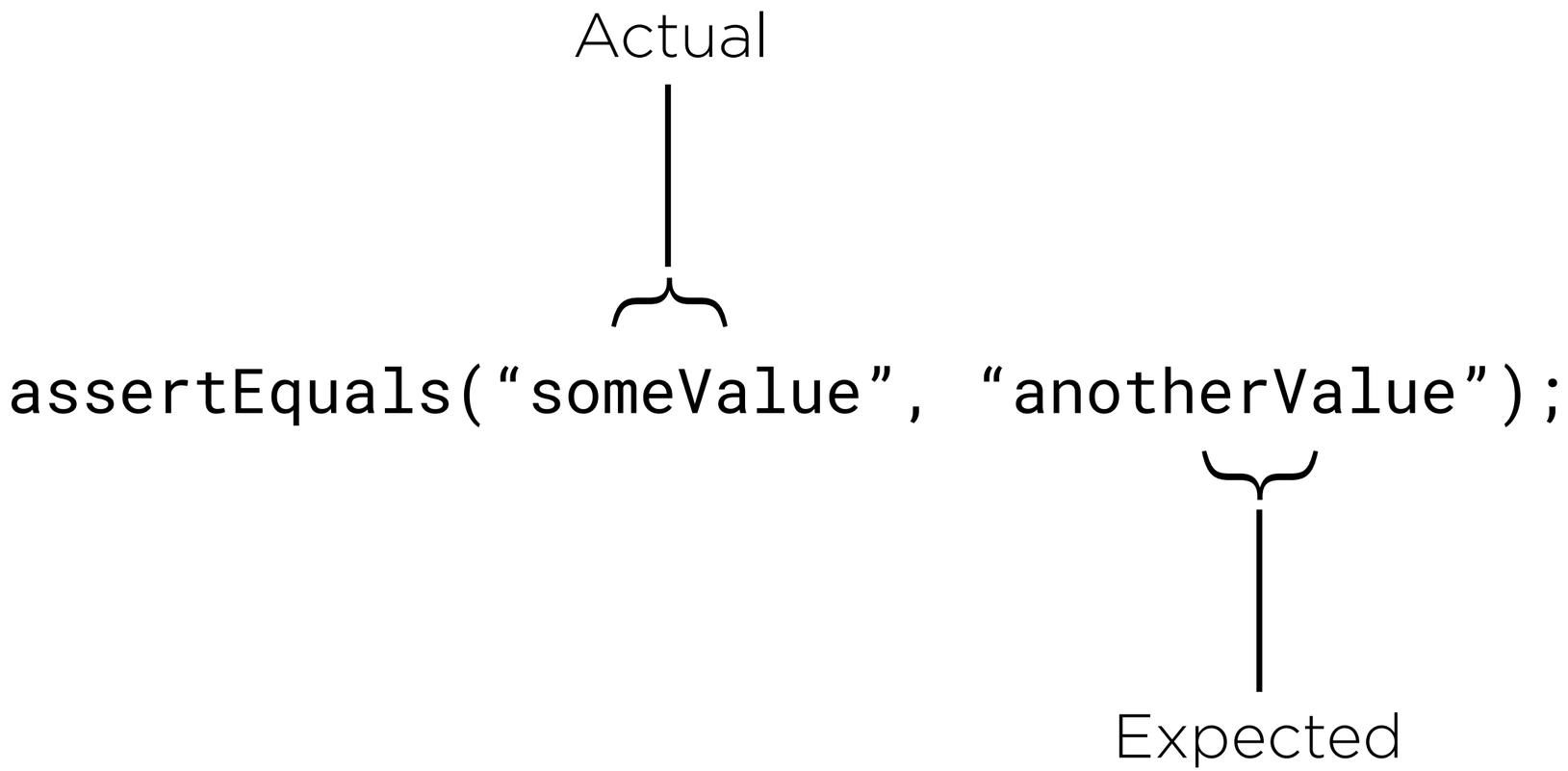
Module for supporting of build tools

Jupiter

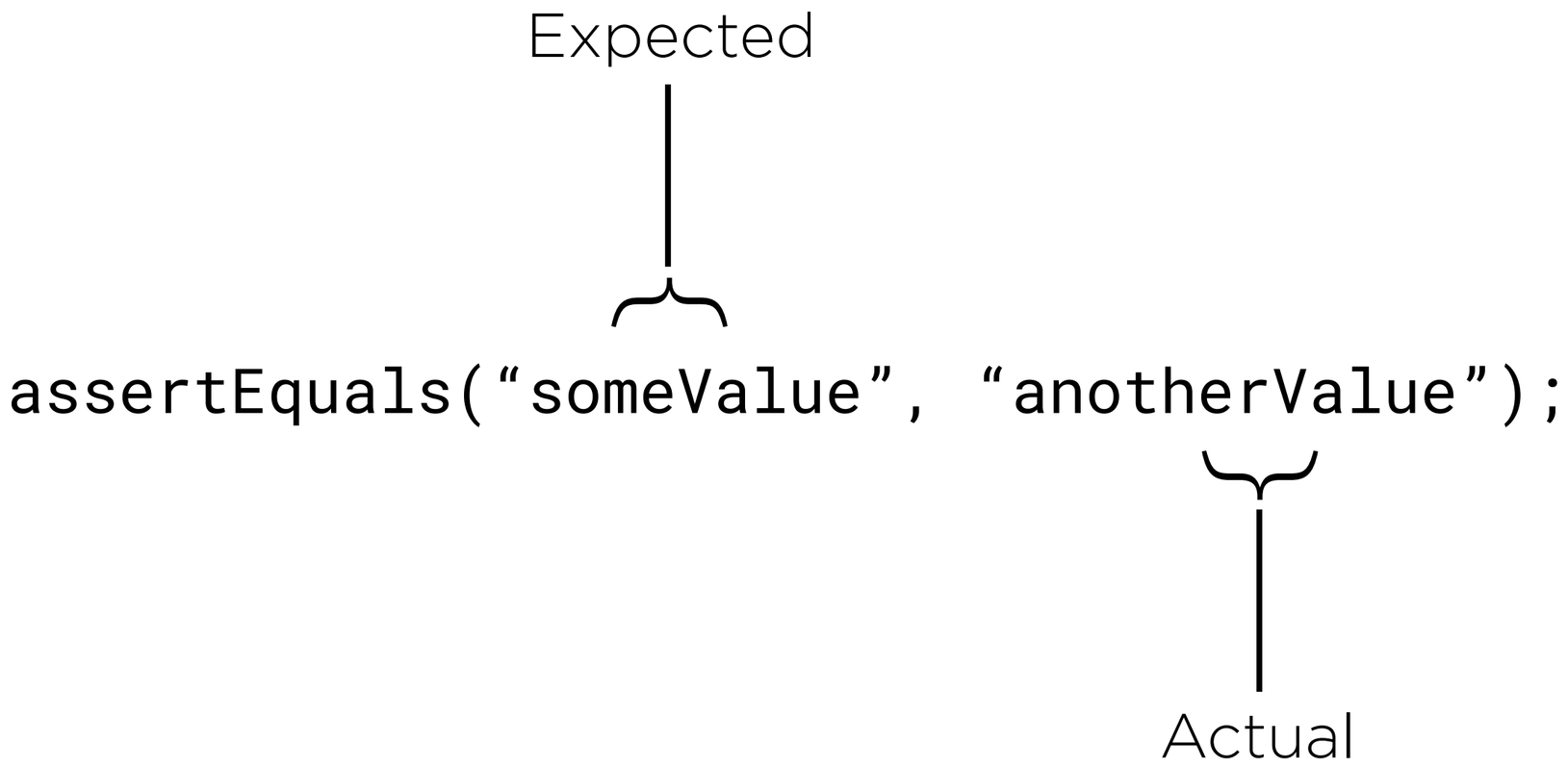
New Unit programming model

AssertJ and Readable Tests

How I Think assertEquals Works



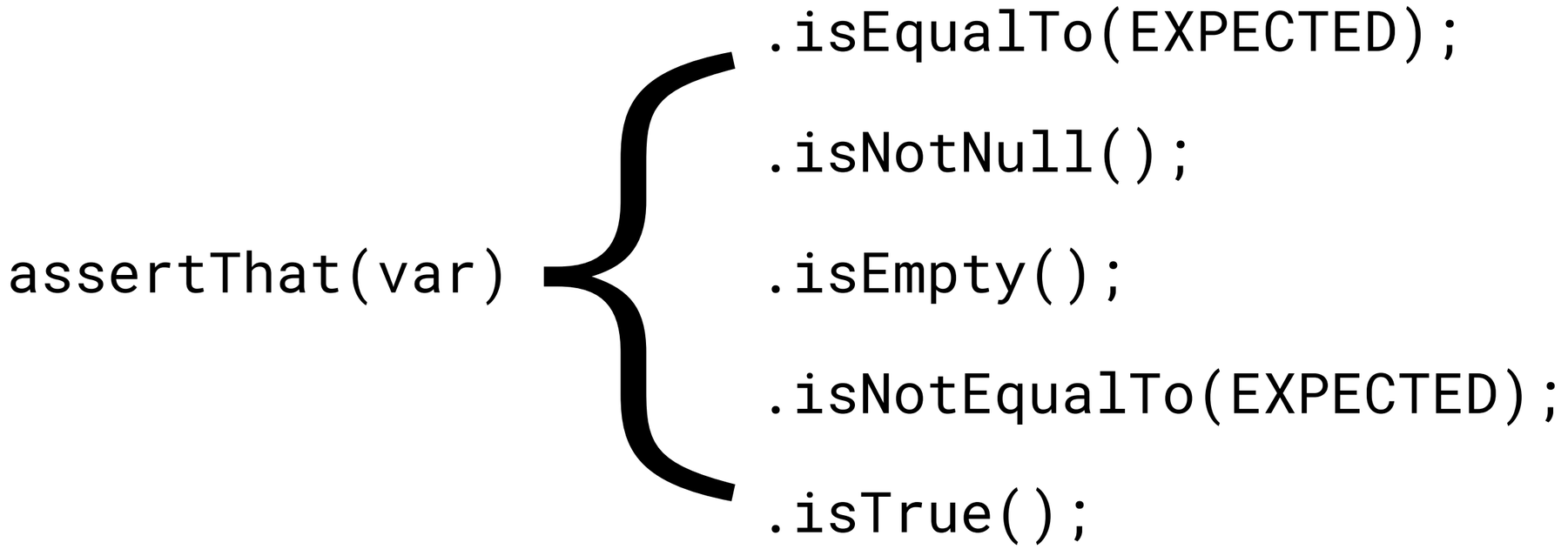
How assertEquals Actually Works



JUnits Many Assert “Roots”

```
assertEquals(“someValue”, “anotherValue”);
assertNull(var);
assertNotNull(var);
assertTrue(var);
```

Declarative Assertions With AssertJ



Demo

Using AssertJ to write more readable tests

Parameterized Tests

```
@Test
public void testFooNotNull(){
   assertThat(“foo”).isNotNull();
}

@Test
public void testBarNotNull(){
   assertThat(“bar”).isNotNull();
}

@Test
public void testFooBarNotNull(){
   assertThat(“fooBar”).isNotNull();
}
```

Testing Multiple Similar Scenarios

Very repetitive

Can become difficult to read if testing a lot of scenarios

If code is refactored might mean updating or fixing a lot of tests

```java
@ParamterizedTest
@MethodSource(“values”)
public void testIsNotNull(String argument){
  assertThat(argument).isNotNull();
}

public Stream<String> values(argument){
  return Stream.of(“foo”, ”bar”, “fooBar”);
}
```

Parameterized Tests in JUnit 5

Reuse the same method for test multiple scenarios

Easier to read and change

Other options for sources including; ValueSource, EnumSource, CsvSource, et al.

Demo

1. Verify both dates are in correct format

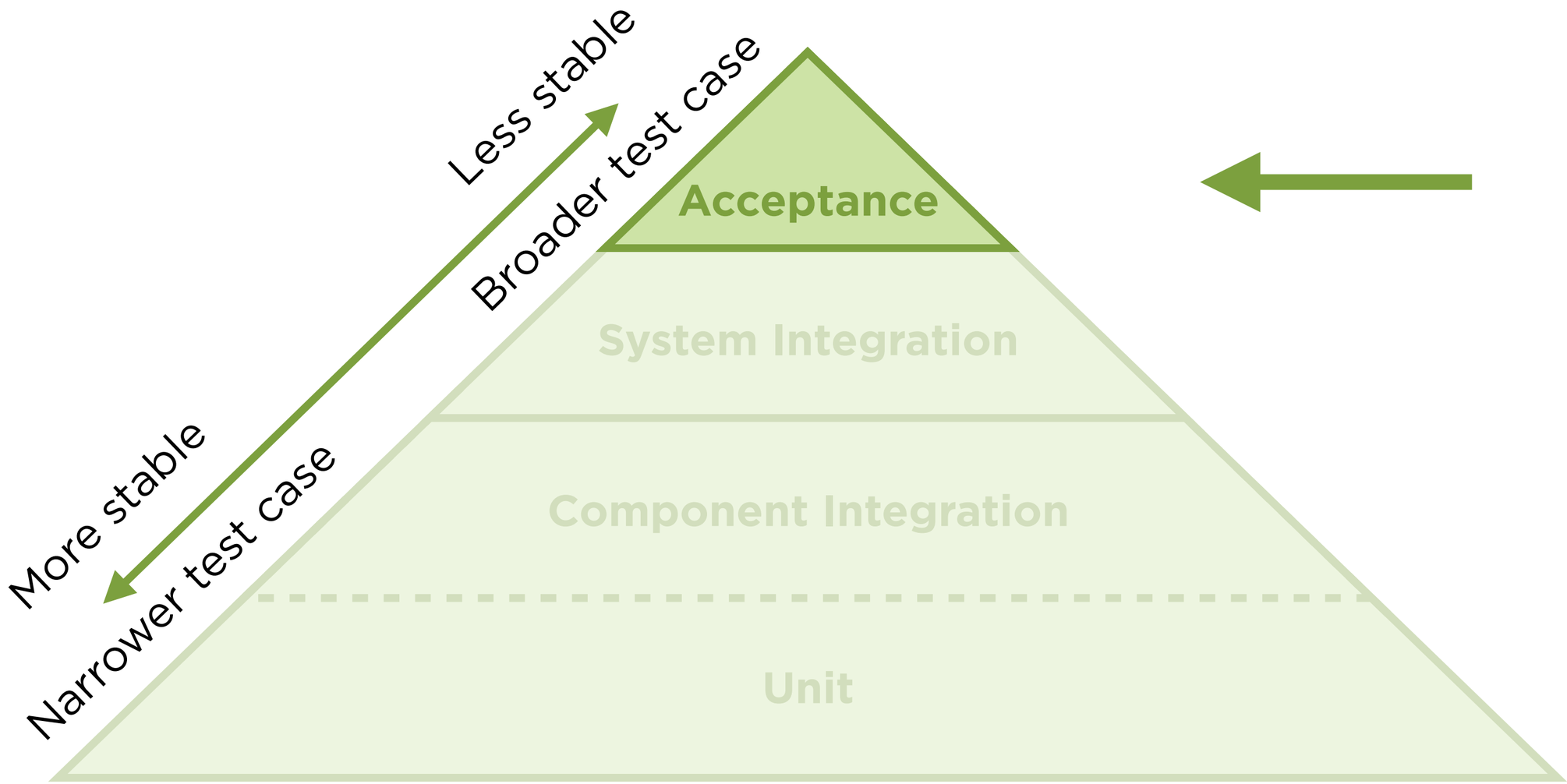
2. Verify both dates are valid dates

3. Verify both dates are in the future

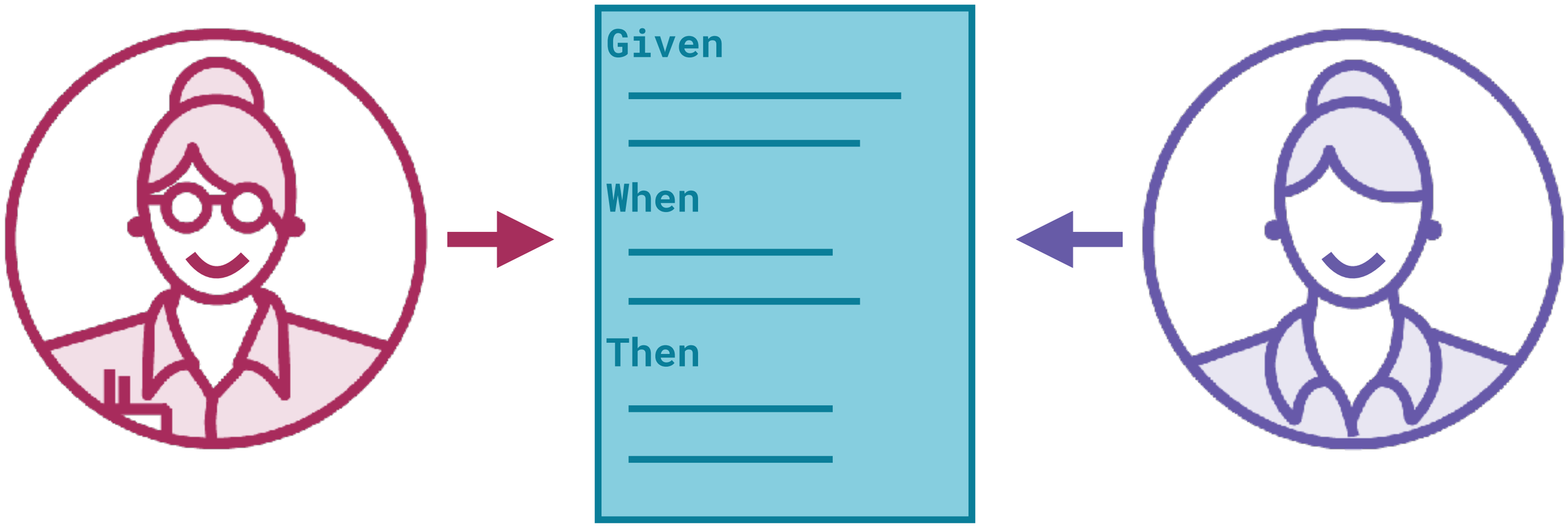
4. Verify end date is at least one day after start date

5. Verify all known errors returned in single response

Acceptance Testing



Acceptance Testing



Acceptance Testing

Given

Preconditions to a scenario

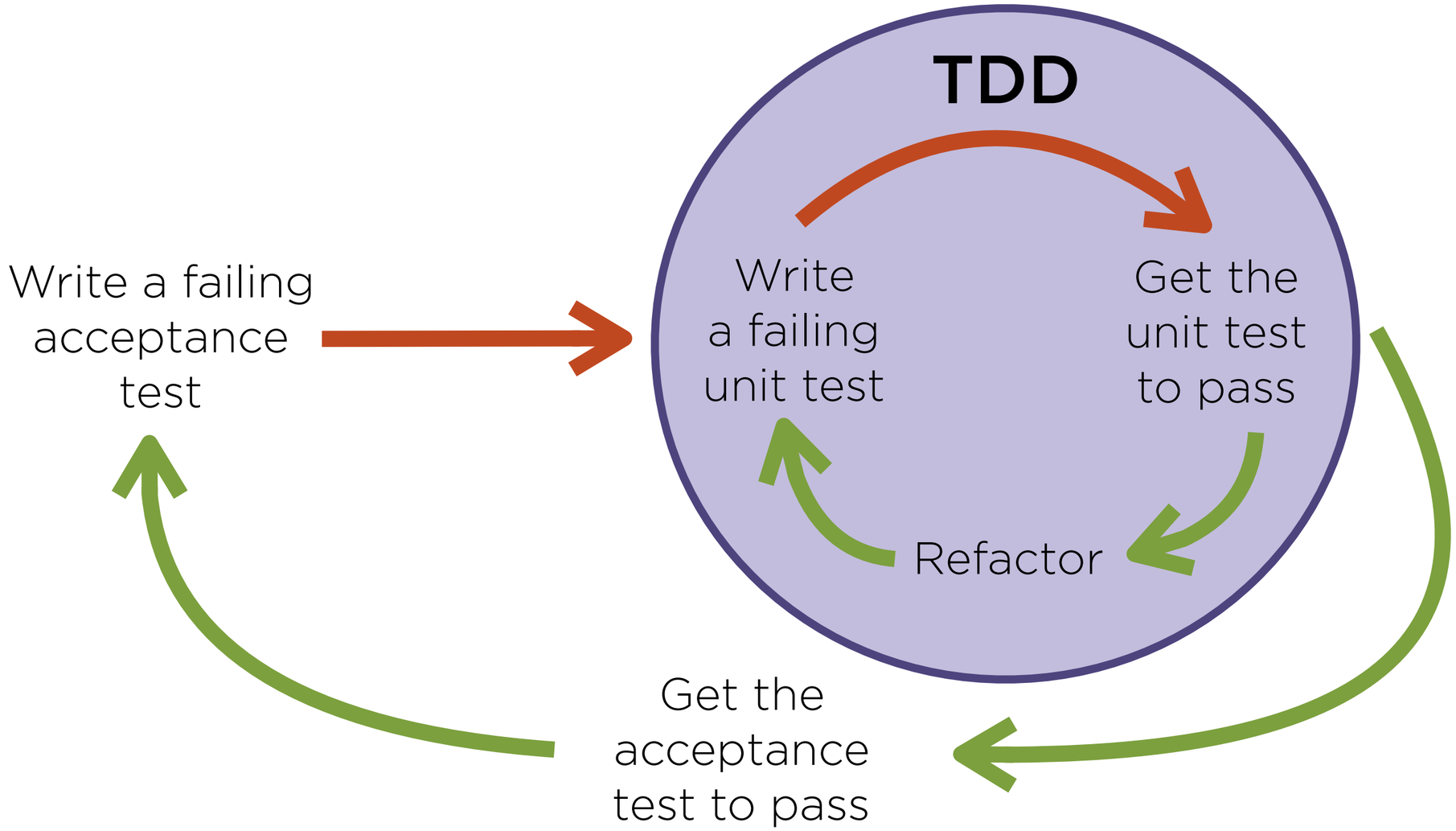
When

Actions that will be performed during the scenario

Then

The system should produce this result

Behavior-driven Development



Behavior-driven Development

Given: The user is an existing customer “Joe User”

When: The user books room 123 for 5 nights

Then: The user should receive message confirming booking

Test: lookUpCustomerByUsername

Test: lookUpRoomByRoomNumber
Test: createABooking

Test: lookUpBookingById

Things to Keep in Mind with BDD

- Writing acceptance tests is a skill
- Both developer and business analyst must learn it
- Requires upfront understanding of feature being written
- You're not going to get it right the first time

Behavior Driven Development in Action

Spring REST Docs

Demo

Using Spring REST Docs with automated tests

Resources

AsciiDoc homepage: http://asciidoc.org/

Summary

Learned how to leverage tests as documentation

Introduced to JUnit 5

Learned about BDD