Writing the First Test

Harit Himanshu

FOUNDER, BONSAIILABS.COM

@harittweets   www.bonsaiilabs.com

Agenda

- Understanding Test and Suite
- Writing the first Test
- Running the test using IntelliJ IDEA
- Running the test using Scala command-line
- Running the test using *sbt* command-line
- Running the test using *sbt* console
- Understanding Testing Styles
- Understanding FlatSpec Style
- Understanding the Test Lifecycle

What Is a Test?

Name

Start

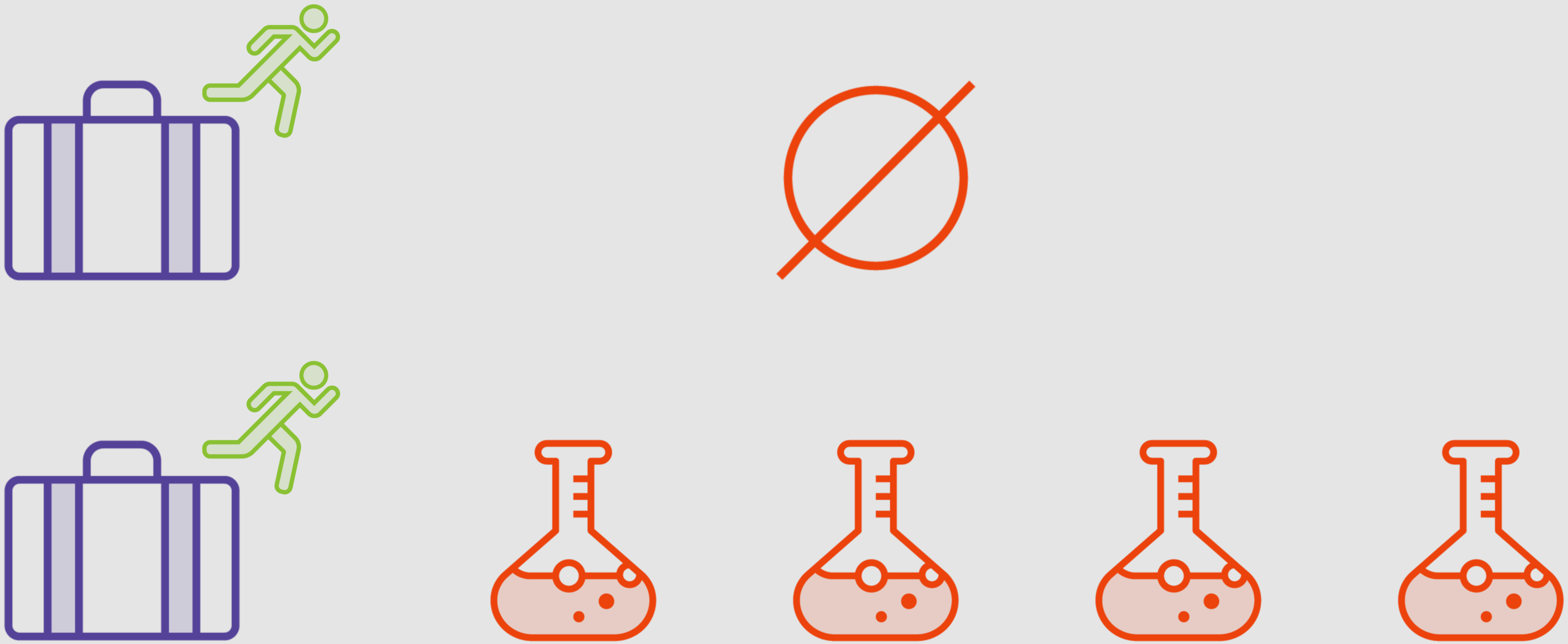
Succeed

Fail

Pending

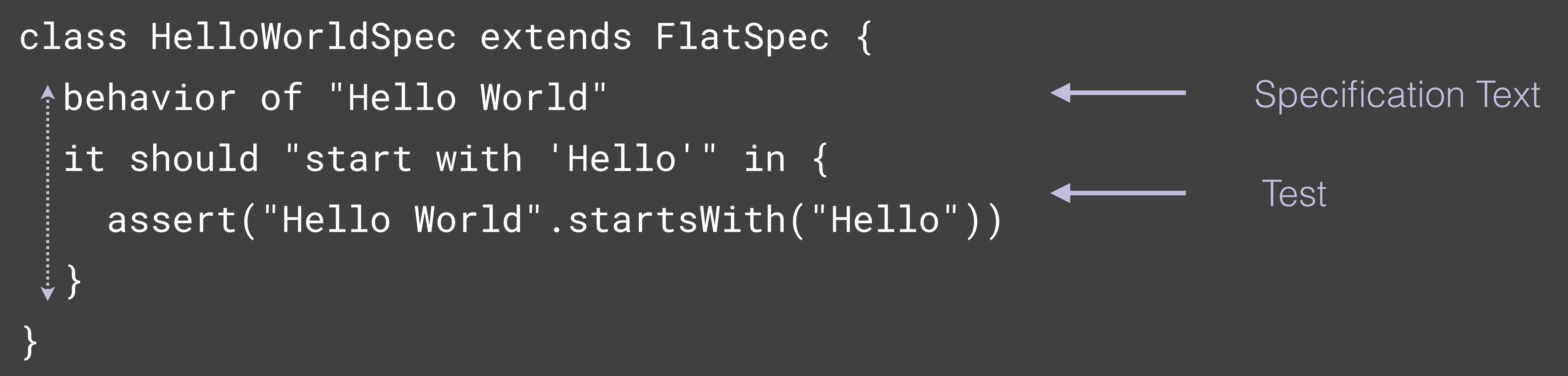
Cancel

What Is a Suite?



Writing the First Test

Understanding the Testing Styles



```
class HelloWorldSpec extends FlatSpec {
  behavior of "Hello World"
  it should "start with 'Hello'" in {
    assert("Hello World".startsWith("Hello"))
  }
}
```

Specification Text

Test

1. FlatSpec

Teams wishing to move from xUnit to BDD style tests

```
class HelloWorldFunSuiteSpec extends FunSuite {
  test("A String 'Hello World' should start with 'Hello'") {
    assert("Hello World".startsWith("Hello"))
  }
}
```

2. FunSuite

For teams coming from xUnit frameworks

```
class HelloWorldFunSpec extends FunSpec {
  describe("A 'Hello World' String") {
    it("should start with 'Hello'") {
      assert("Hello World".startsWith("Hello"))
    }
  }
}
```

3. FunSpec

Team coming from Ruby’s RSpec

```
class HelloWorldFeatureSpec extends FeatureSpec with GivenWhenThen {
  info("As a developer”)
  info("I want to test Strings")
  feature("A String") {
    scenario("When a user provides 'Hello World' as input") {
      Then("The input must start with 'Hello'")
      assert("Hello World".startsWith("Hello"))
    }
  }
}
```

4. FeatureSpec

Intended for acceptance testing

WordSpec

Teams coming from spec or spec2 library

Team wanting to enforce high degree of discipline in their test specification

FreeSpec

Team experienced in BDD style

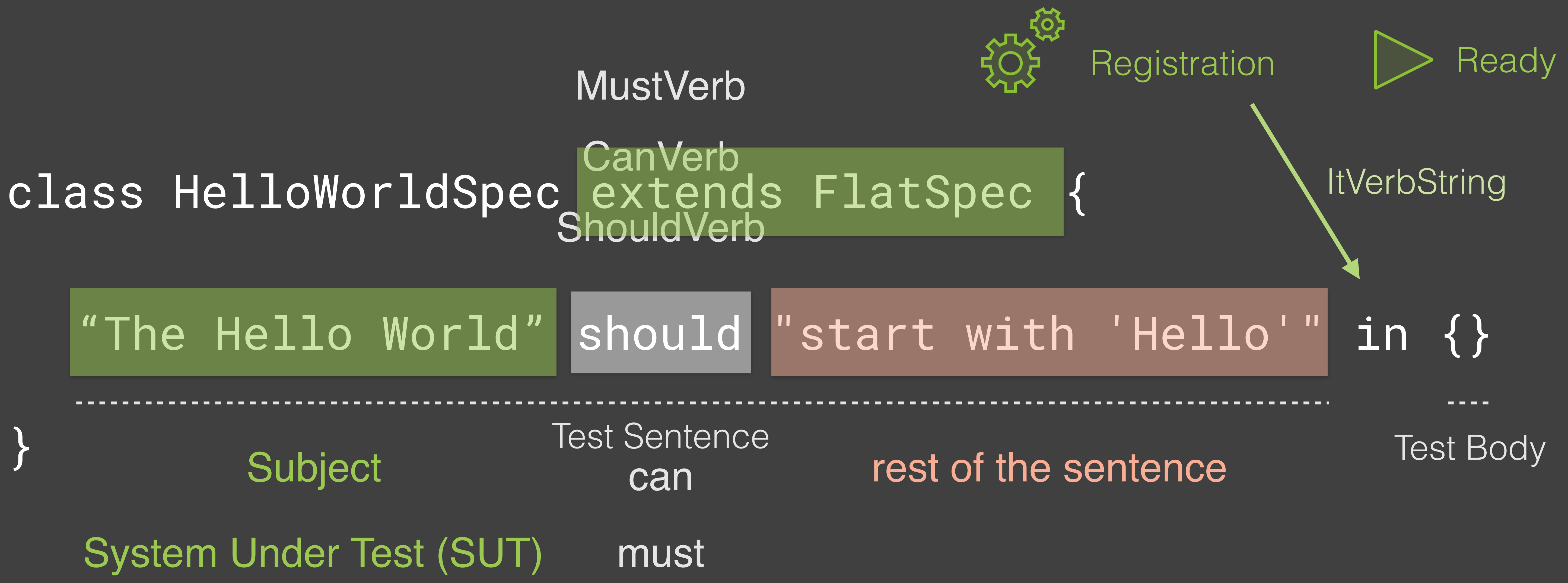
Team wanting describe their own specification

PropSpec

Teams wanting to perform Property-Based Testing.

Allows teams to create test matrix

Understanding FlatSpec Style



MustVerb

Registration

Ready

CanVerb

```
class HelloWorldSpec extends FlatSpec {
  “The Hello World” should "start with 'Hello'" in {}
}
```

ShouldVerb

ItVerbString

Test Sentence

Test Body

Subject

can

rest of the sentence

System Under Test (SUT)

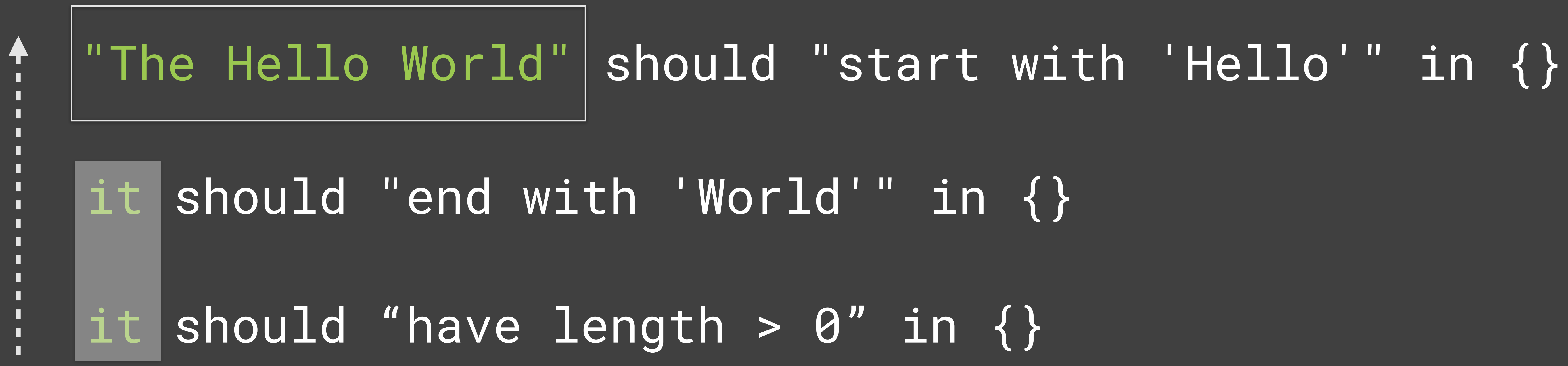
must

Multiple Tests in FlatSpec

```
"The Hello World" should "start with 'Hello'" in {}

“The Hello World" should "end with 'World'" in {}
```

Multiple Tests in FlatSpec



```
"The Hello World" should "start with 'Hello'" in {}

it should "end with 'World'" in {}

it should “have length > 0” in {}
```

Multiple Tests in FlatSpec

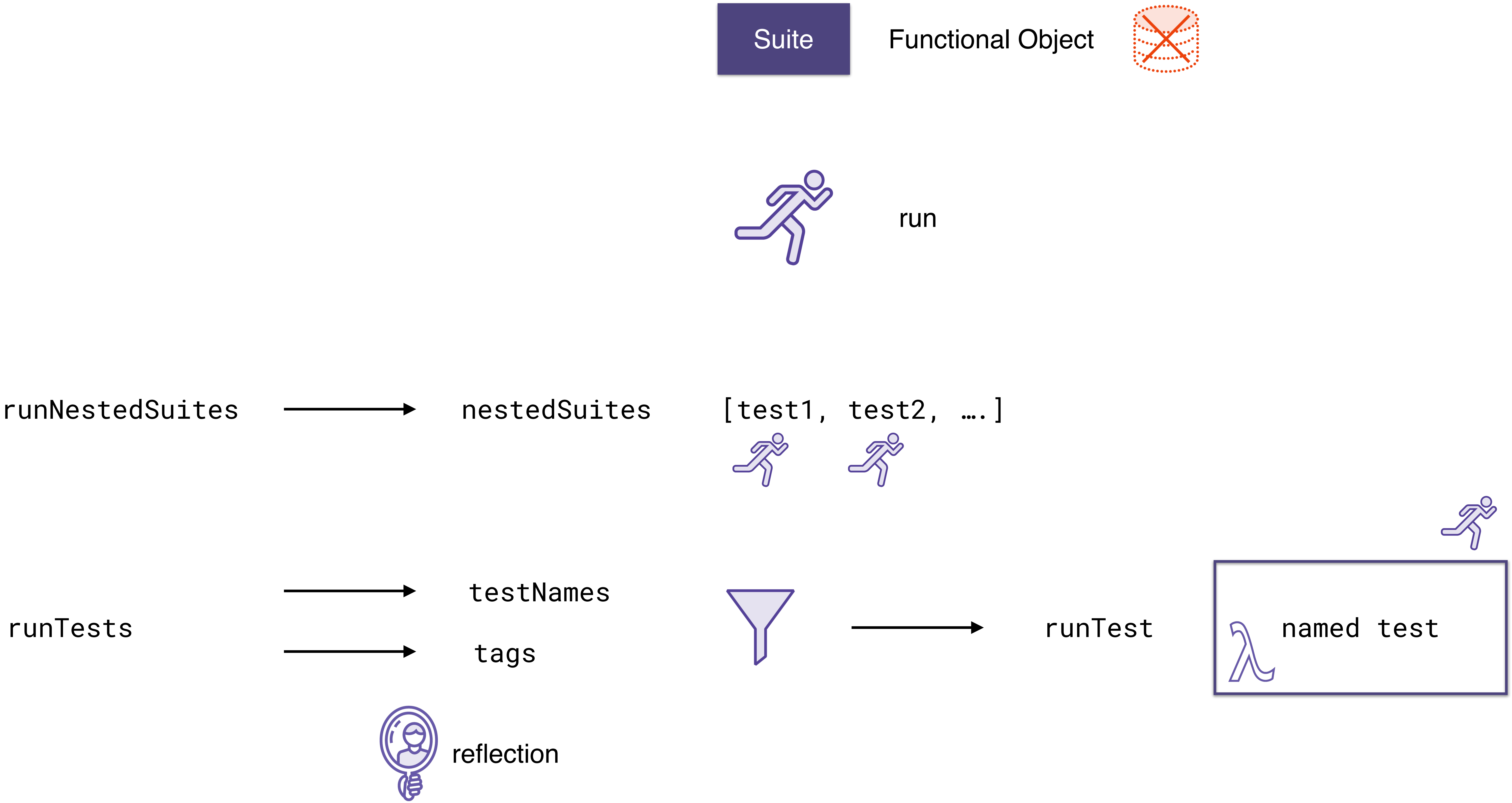
```
behavior of "Hello World"

it should "start with 'Hello'" in {}

it should "end with 'World'" in {}

it should “have length > 0” in {}
```

Understanding Test Lifecycle



Suite

Functional Object

run

runNestedSuites → nestedSuites [test1, test2, ….]

runTests → testNames

→ tags

reflection

→ runTest

λ named test

Agenda

- Understanding Test and Suite
- Writing the first Test
- Running the test using IntelliJ IDEA
- Running the test using Scala command-line
- Running the test using *sbt* command-line
- Running the test using *sbt* console
- Understanding Testing Styles
- Understanding FlatSpec Style
- Understanding the Test Lifecycle