OCI | WE ARE SOFTWARE ENGINEERS.

APACHECON North America

Make Your Testing Groovy

Dr Paul King
OCI Groovy Lead
@paulk_asert
https://speakerdeck.com/paulk/make-your-testing-groovy
https://github.com/paulk-asert/MakeTestingGroovy

objectcomputing.com

WE ARE SOFTWARE ENGINEERS.

We deliver mission-critical software solutions that accelerate innovation within your organization and stand up to the evolving demands of your business.

OCI | WE ARE SOFTWARE ENGINEERS.

- 160+ engineers
- Home of Grails & Micronaut
- Friend of Groovy
- Global Footprint

What is Groovy?

"Groovy is like a super version of Java. It leverages Java features but adds productivity features and provides great flexibility and extensibility."

```
Groovy = Java – boiler plate code
              + better functional programming
              + dynamic nature
              + extensible type system
              + runtime & compile-time metaprogramming
              + flexible language grammar (DSLs)
              + scripting
              + GDK library
```

Groovy as a Java superset
It's so easy to learn!
Groovy
Groovy
Groovy

Java code for list manipulation

```java
import java.util.List;
import java.util.ArrayList;

class Main {
    private List keepShorterThan(List strings, int length) {
        List result = new ArrayList();
        for (int i = 0; i < strings.size(); i++) {
            String s = (String) strings.get(i);
            if (s.length() < length) {
                result.add(s);
            }
        }
        return result;
    }
    public static void main(String[] args) {
        List names = new ArrayList();
        names.add("Ted"); names.add("Fred");
        names.add("Jed"); names.add("Ned");
        System.out.println(names);
        Main m = new Main();
        List shortNames = m.keepShorterThan(names, 4);
        System.out.println(shortNames.size());
        for (int i = 0; i < shortNames.size(); i++) {
            String s = (String) shortNames.get(i);
            System.out.println(s);
        }
    }
}
```

Groovy code for list manipulation

```
import java.util.List;
import java.util.ArrayList;

class Main {
    private List keepShorterThan(List strings, int length) {
        List result = new ArrayList();
        for (int i = 0; i < strings.size(); i++) {
            String s = (String) strings.get(i);
            if (s.length() < length) {
                result.add(s);
            }
        }
        return result;
    }
    public static void main(String[] args) {
        List names = new ArrayList();
        names.add("Ted"); names.add("Fred");
        names.add("Jed"); names.add("Ned");
        System.out.println(names);
        Main m = new Main();
        List shortNames = m.keepShorterThan(names, 4);
        System.out.println(shortNames.size());
        for (int i = 0; i < shortNames.size(); i++) {
            String s = (String) shortNames.get(i);
            System.out.println(s);
        }
    }
}
```

Rename `Main.java` *to* `Main.groovy`

Some Java Boilerplate identified

```
import java.util.List;
import java.util.ArrayList;

class Main {
    private List keepShorterThan(List strings, int length) {
        List result = new ArrayList();
        for (int i = 0; i < strings.size(); i++) {
            String s = (String) strings.get(i);
            if (s.length() < length) {
                result.add(s);
            }
        }
        return result;
    }
    public static void main(String[] args) {
        List names = new ArrayList();
        names.add("Ted"); names.add("Fred");
        names.add("Jed"); names.add("Ned");
        System.out.println(names);
        Main m = new Main();
        List shortNames = m.keepShorterThan(names, 4);
        System.out.println(shortNames.size());
        for (int i = 0; i < shortNames.size(); i++) {
            String s = (String) shortNames.get(i);
            System.out.println(s);
        }
    }
}
```

Are the semicolons needed?
And shouldn't we us more modern list notation?
Why not import common libraries?
Do we need the static types?
Must we always have a main method and class definition?
How about improved consistency?

Java Boilerplate removed

```
def keepShorterThan(strings, length) {
    def result = new ArrayList()
    for (s in strings) {
        if (s.size() < length) {
            result.add(s)
        }
    }
    return result
}

names = new ArrayList()
names.add("Ted"); names.add("Fred")
names.add("Jed"); names.add("Ned")
System.out.println(names)
shortNames = keepShorterThan(names, 4)
System.out.println(shortNames.size())
for (s in shortNames) {
    System.out.println(s)
}
```

More Java Boilerplate identified

```
def keepShorterThan(strings, length) {
    def result = new ArrayList()
    for (s in strings) {
        if (s.size() < length) {
            result.add(s)
        }
    }
    return result
}

names = new ArrayList()
names.add("Ted"); names.add("Fred")
names.add("Jed"); names.add("Ned")
System.out.println(names)
shortNames = keepShorterThan(names, 4)
System.out.println(shortNames.size())
for (s in shortNames) {
    System.out.println(s)
}
```

Shouldn’t we have special notation for lists?

And special facilities for list processing?

Is ‘return’ needed at end?

Is the method now needed?

Simplify common methods?

Remove unambiguous brackets?

Boilerplate removed = nicer Groovy version

```
names = ["Ted", "Fred", "Jed", "Ned"]
println names
shortNames = names.findAll{ it.size() < 4 }
println shortNames.size()
shortNames.each{ println it }
```

Output:

```
["Ted", "Fred", "Jed", "Ned"]
3
Ted
Jed
Ned
```

Or Groovy DSL version if required

```
given the names "Ted", "Fred", "Jed" and "Ned"
display all the names
display the number of names having size less than 4
display the names having size less than 4
```

Or Groovy DSL version if required

```
given the names "Ted", "Fred", "Jed" and "Ned"
display all the names
display the number of names having size less than 4
display the names having size less than 4
```

```
given(the).names("Ted", "Fred", "Jed").and("Ned")
display(all).the(names)
display(the).number(of).names(having).size(less).than(4)
display(the).names(having).size(less).than(4)
```

Or Groovy DSL version if required

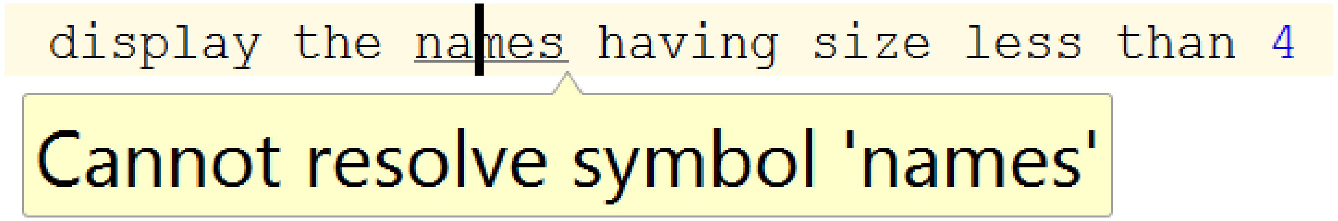
```
given the names "Ted", "Fred", "Jed" and "Ned"
display all the names
display the number of names having size less than 4
display the names having size less than 4
```

```
given(the).names("Ted", "Fred", "Jed").and("Ned")
display(all).the(names)
display(the).number(of).names(having).size(less).than(4)
display(the).names(having).size(less).than(4)
```

```
names = []
def of, having, less
def given(_the) { [names:{ Object[] ns -> names.addAll(ns)
  [and: { n -> names += n }] }] }
def the = [
  number: { _of -> [names: { _having -> [size: { _less -> [than: { size ->
    println names.findAll{ it.size() < size }.size() }]}] }] },
  names: { _having -> [size: { _less -> [than: { size ->
    names.findAll{ it.size() < size }.each{ println it } }]}] }
]
def all = [the: { println it }]
def display(arg) { arg }
```

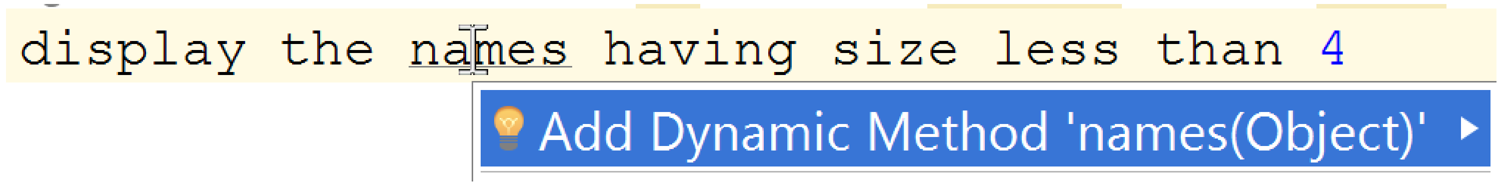
Or Groovy DSL version if required

```
given the names "Ted", "Fred", "Jed" and "Ned"
display all the names
display the number of names having size less than 4
display the names having size less than 4
```



```
display the names having size less than 4
```

Cannot resolve symbol 'names'



```
display the names having size less than 4
```

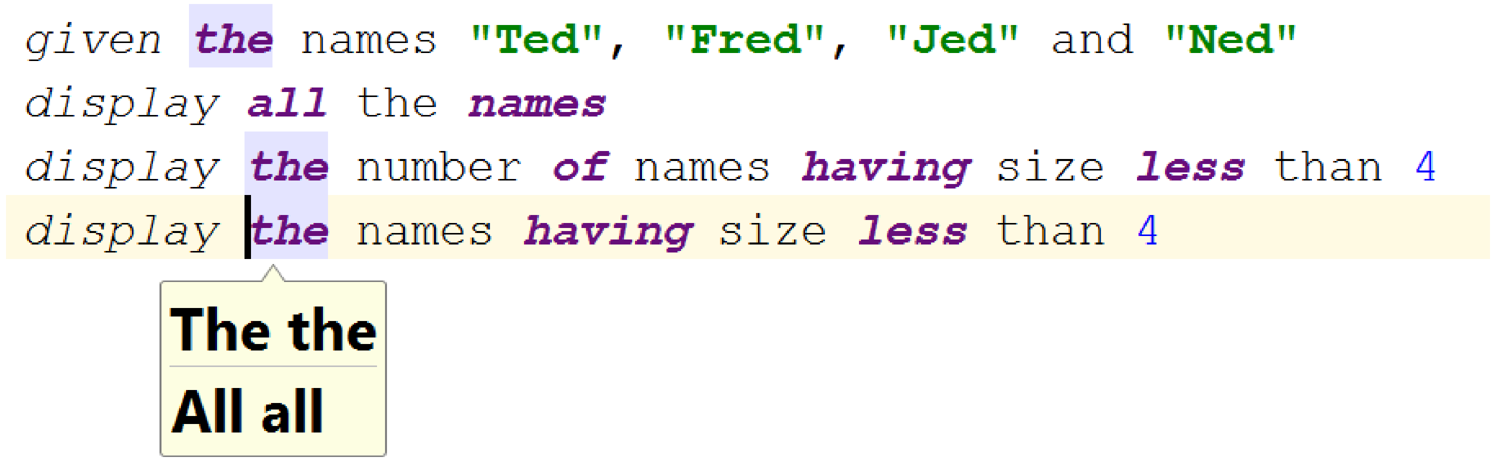
Or use GDSL (IntelliJ IDEA) or DSLD (Eclipse)

Or typed Groovy DSL version if required

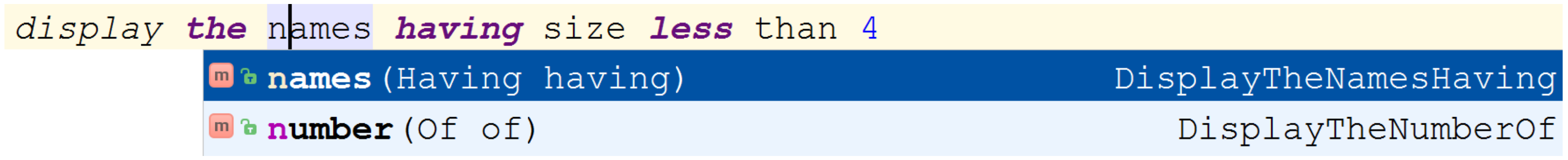
```
given the names "Ted", "Fred", "Jed" and "Ned"
display all the names
display the number of names having size less than 4
display the names having size less than 4
```

```
…
enum The { the }
enum Having { having }
enum Of { of }
…
class DisplayThe {
    DisplayTheNamesHaving names(Having having) {
        new DisplayTheNamesHaving()
    }
    DisplayTheNumberOf number(Of of) {
        new DisplayTheNumberOf()
    }
}
…
// plus 50 lines
```

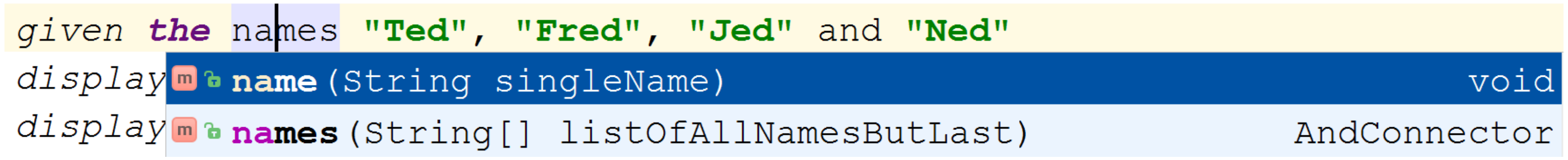
Or typed Groovy DSL version if required



```
given the names "Ted", "Fred", "Jed" and "Ned"
display all the names
display the number of names having size less than 4
display the names having size less than 4
```

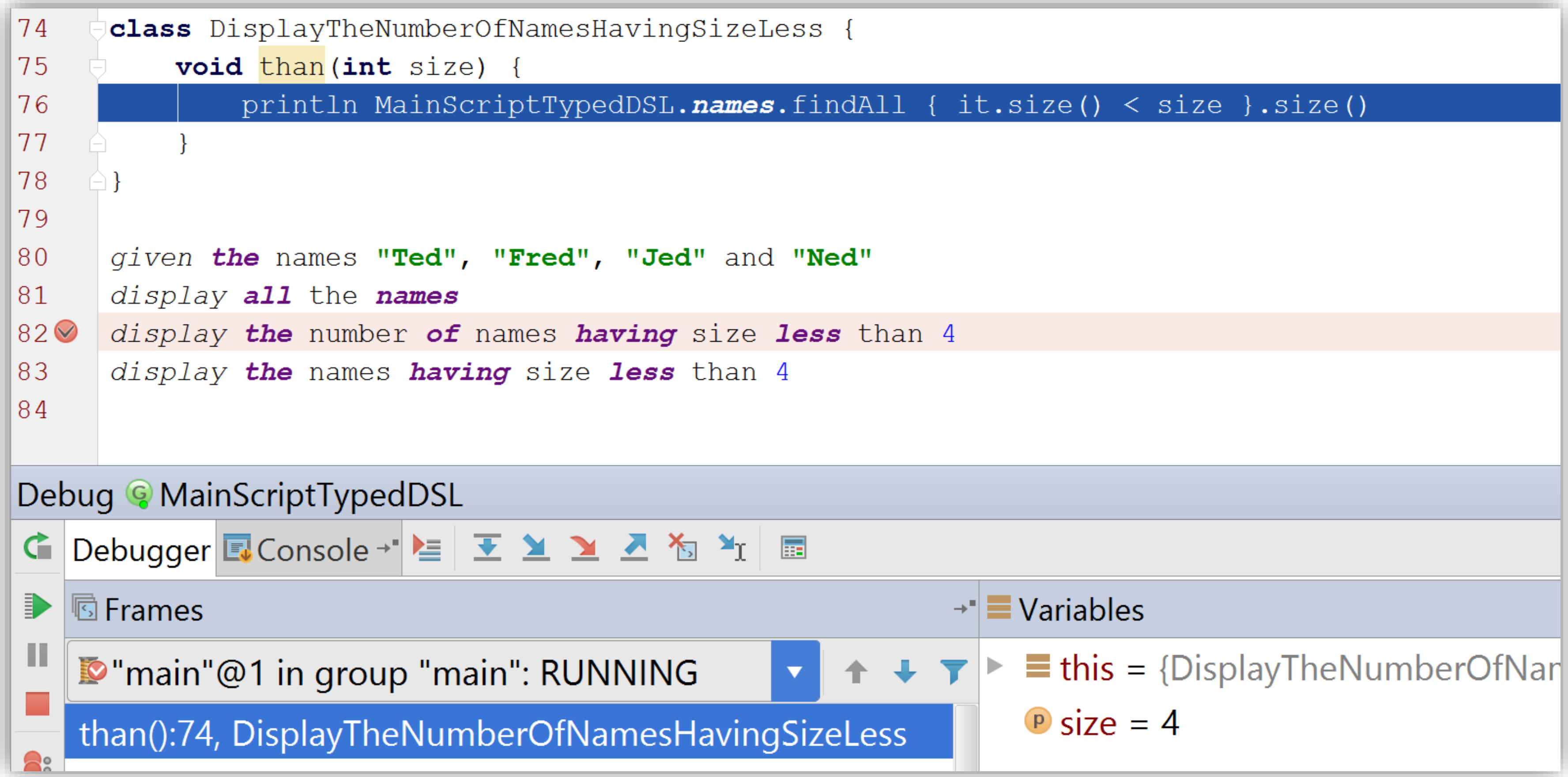


```
display the names having size less than 4
names(Having having)                    DisplayTheNamesHaving
number(Of of)                           DisplayTheNumberOf
```



```
given the names "Ted", "Fred", "Jed" and "Ned"
display name(String singleName)                 void
display names(String[] listOfAllNamesButLast)   AndConnector
```

Groovy DSL being debugged



```
class DisplayTheNumberOfNamesHavingSizeLess {
    void than(int size) {
        println MainScriptTypedDSL.names.findAll { it.size() < size }.size()
    }
}

given the names "Ted", "Fred", "Jed" and "Ned"
display all the names
display the number of names having size less than 4
display the names having size less than 4
```

Debug MainScriptTypedDSL

"main"@1 in group "main": RUNNING

than():74, DisplayTheNumberOfNamesHavingSizeLess

this = {DisplayTheNumberOfNar

size = 4

Or typed Groovy DSL version if required

```
@TypeChecked
def method() {
    given the names "Ted", "Fred", "Jed" and "Ned"
    display all the names
    display the number of names having size less than 4
    display the names having size less than 4
}
```

☑

Or typed Groovy DSL version if required

```
@TypeChecked
def method() {
    given the names "Ted", "Fred", "Jed" and "Ned"
    display all the names
    display the number of names having size less than 4
    display the names having size less than 4
}
```

✓

```
@TypeChecked
def method() {
    given the names "Ted", "Fred", 42 and "Ned"
    display all the names
    display the number of names having size less than 4
    display the names having size less than 4
}
```

✗

```
[Static type checking] - Cannot find matching method
GivenThe#names(java.lang.String, java.lang.String, int).
```

Or typed Groovy DSL version if required

```
@TypeChecked(extensions='EdChecker.groovy')
def method() {
    given the names "Ted", "Fred", "Jed" and "Ned"
    display all the names
    display the number of names having size less than 4
    display the names having size less than 4
}
```

Or extensible typed Groovy DSL version if required

```groovy
@TypeChecked(extensions='EdChecker.groovy')
def method() {
    given the names "Ted", "Fred", "Jed" and "Ned"
    display all the names
    display the number of names having size less than 4
    display the names having size less than 4
}
```

```groovy
afterMethodCall { mc ->
  mc.arguments.each {
    if (isConstantExpression(it)) {
      if (it.value instanceof String && !it.value.endsWith('ed')) {
        addStaticTypeError("I don't like the name '${it.value}'", mc)
      }
    }
  }
}
```

☑

Or typed Groovy DSL version if required

```
@TypeChecked(extensions='EdChecker.groovy')
def method() {
    given the names "Ted", "Fred", "Jed" and "Ned"
    display all the names
    display the number of names having size less than 4
    display the names having size less than 4
}
```

```
afterMethodCall { mc ->
  mc.arguments.each {
    if (isConstantExpression(it)) {
      if (it.value instanceof String && !it.value.endsWith('ed')) {
        addStaticTypeError("I don't like the name '${it.value}'", mc)
      }
    }
  }
}
```

Or typed Groovy DSL version if required

```
@TypeChecked(extensions='EdChecker.groovy')
def method() {
    given the names "Ted", "Mary", "Jed" and "Pete"
    display all the names
    display the number of names having size less than 4
    display the names having size less than 4
}
```

```
afterMethodCall { mc ->
  mc.arguments.each {
    if (isConstantExpression(it)) {
      if (it.value instanceof String && !it.value.endsWith('ed')) {
        addStaticTypeError("I don't like the name '${it.value}'", mc)
      }
    }
  }
}
```

Or typed Groovy DSL version if required

```groovy
@TypeChecked(extensions='EdChecker.groovy')
def method() {
    given the names "Ted", "Mary", "Jed" and "Pete"
    display all the names
    display the number of names having size less than 4
    display the names having size less than 4
}
```

```groovy
afterMethodCall { mc ->
  mc.arguments.each {
    if (isConstantExpression(it)) {
      if (it.value instanceof String && !it.value.endsWith('ed')) {
        addStaticTypeError("I don't like the name '${it.value}'", mc)
      }
    }
  }
}
```

```
[Static type checking] - I don't like the name 'Mary'
 at line: 83, column: 21

[Static type checking] - I don't like the name 'Pete'
 at line: 83, column: 5
```

Some common languages when Groovy was born

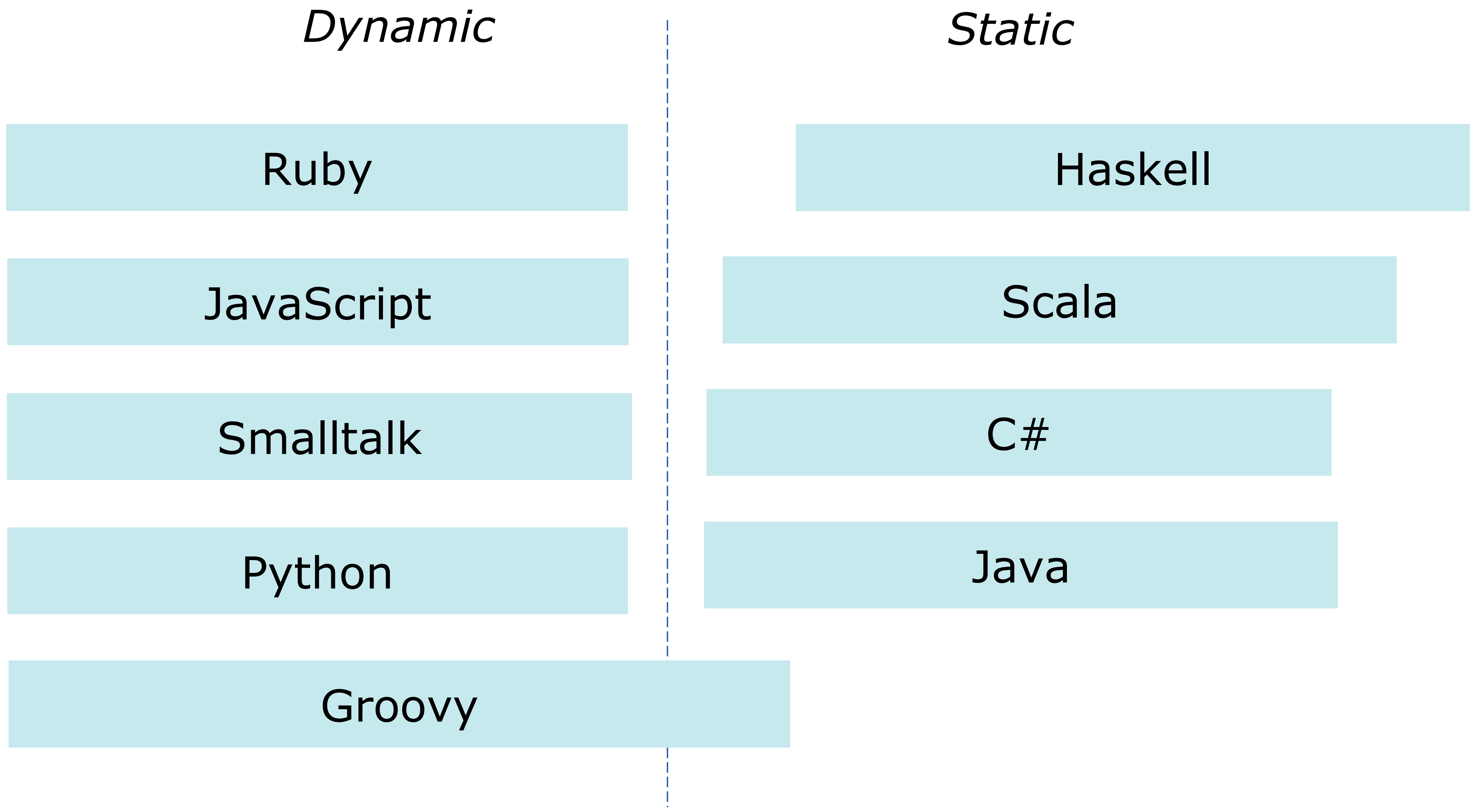
Dynamic

- Ruby
- JavaScript
- Smalltalk
- Python

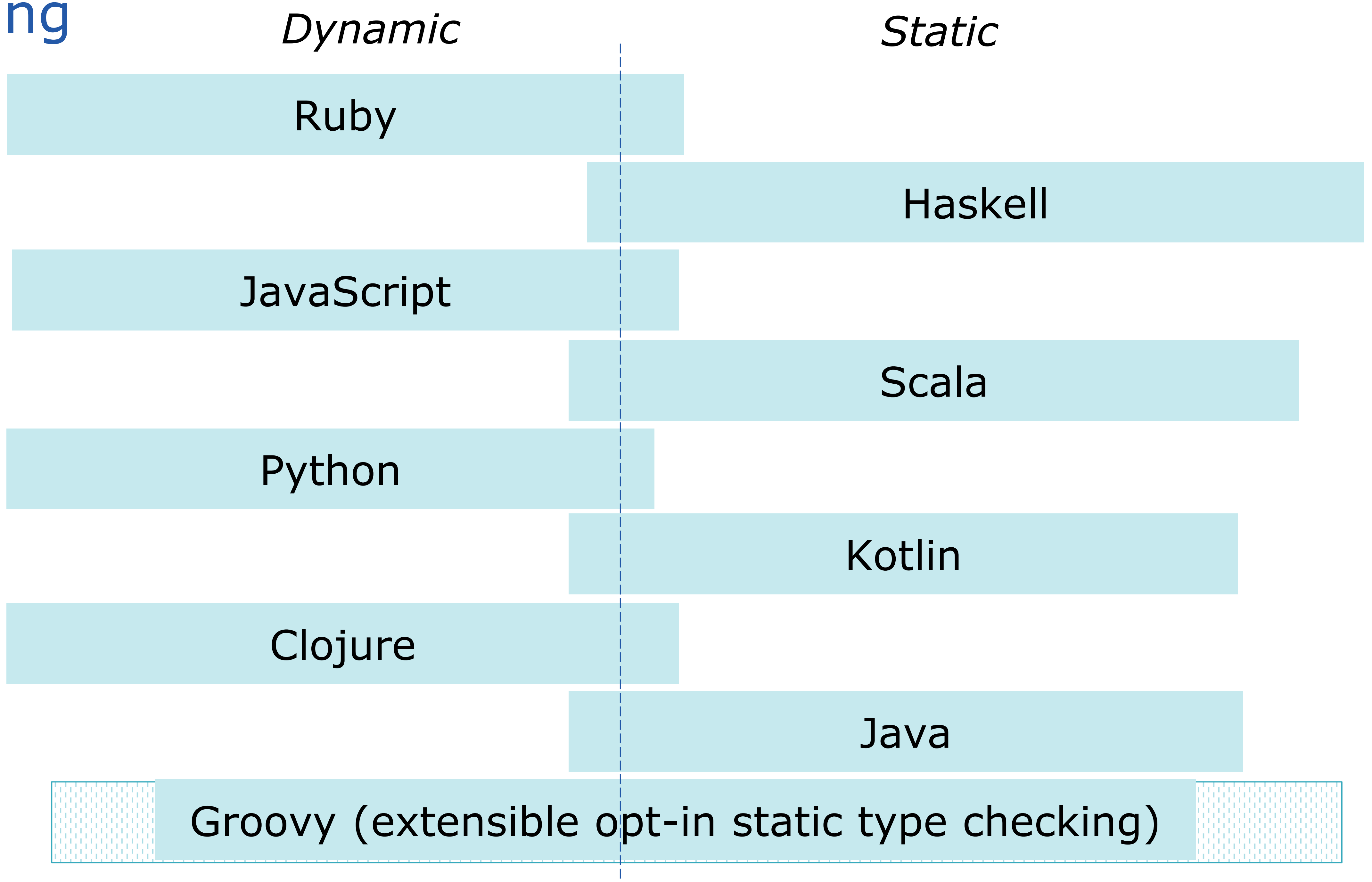
Static

- Haskell
- Scala
- C#
- Java

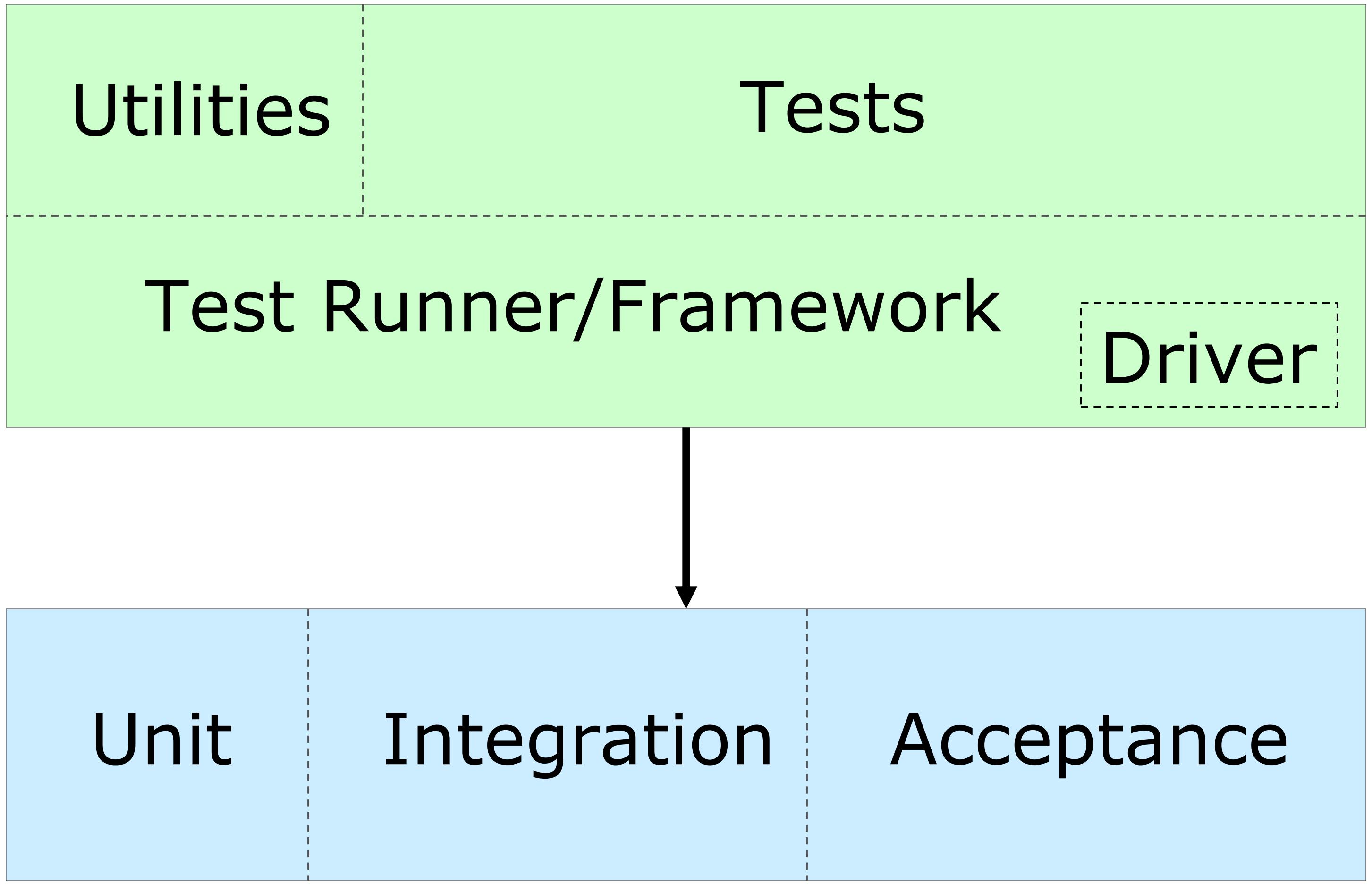
Some common languages when Groovy was born



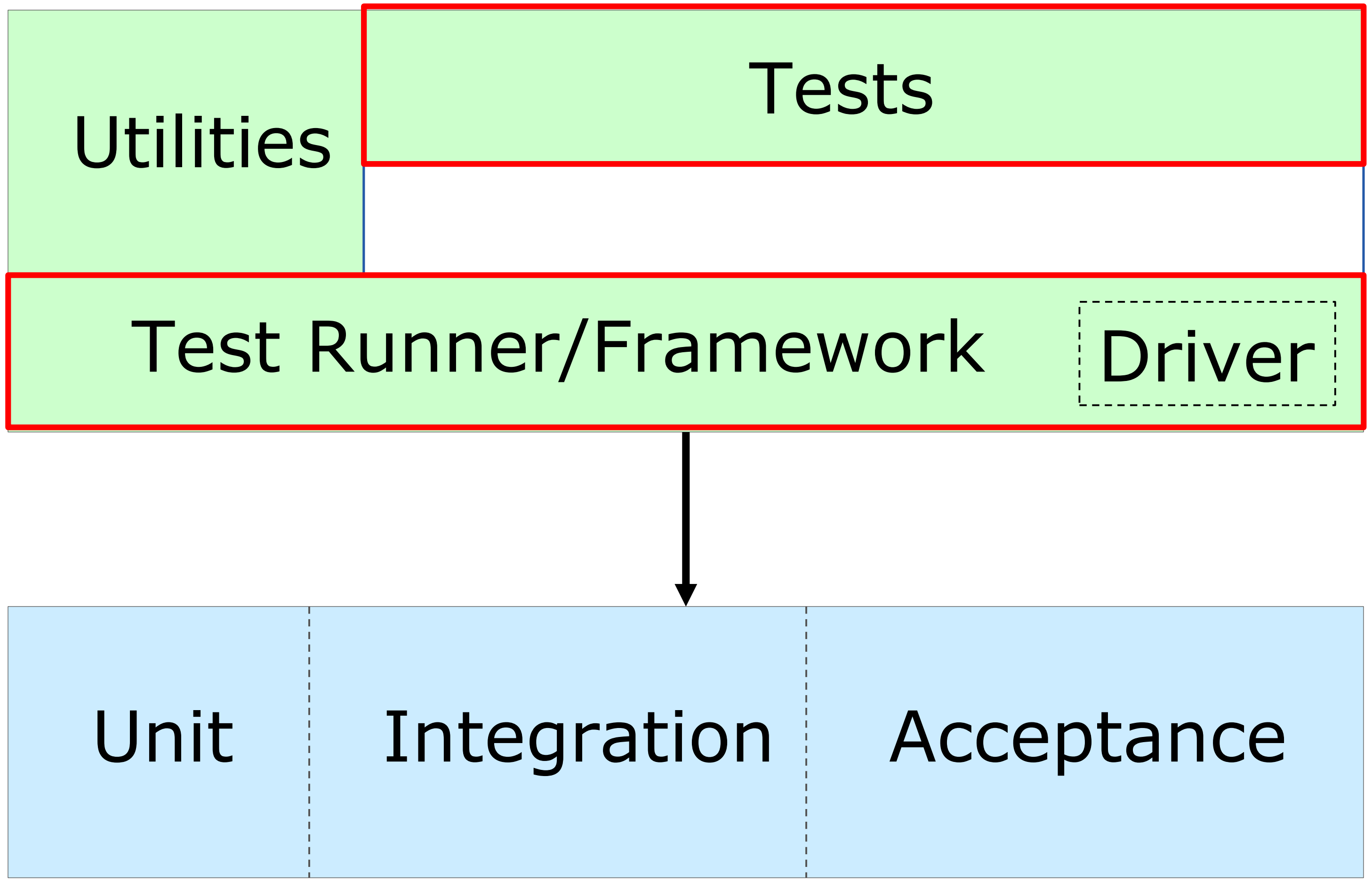
Typing


Dynamic
Static
Ruby
Haskell
JavaScript
Scala
Python
Kotlin
Clojure
Java
Groovy (extensible opt-in static type checking)

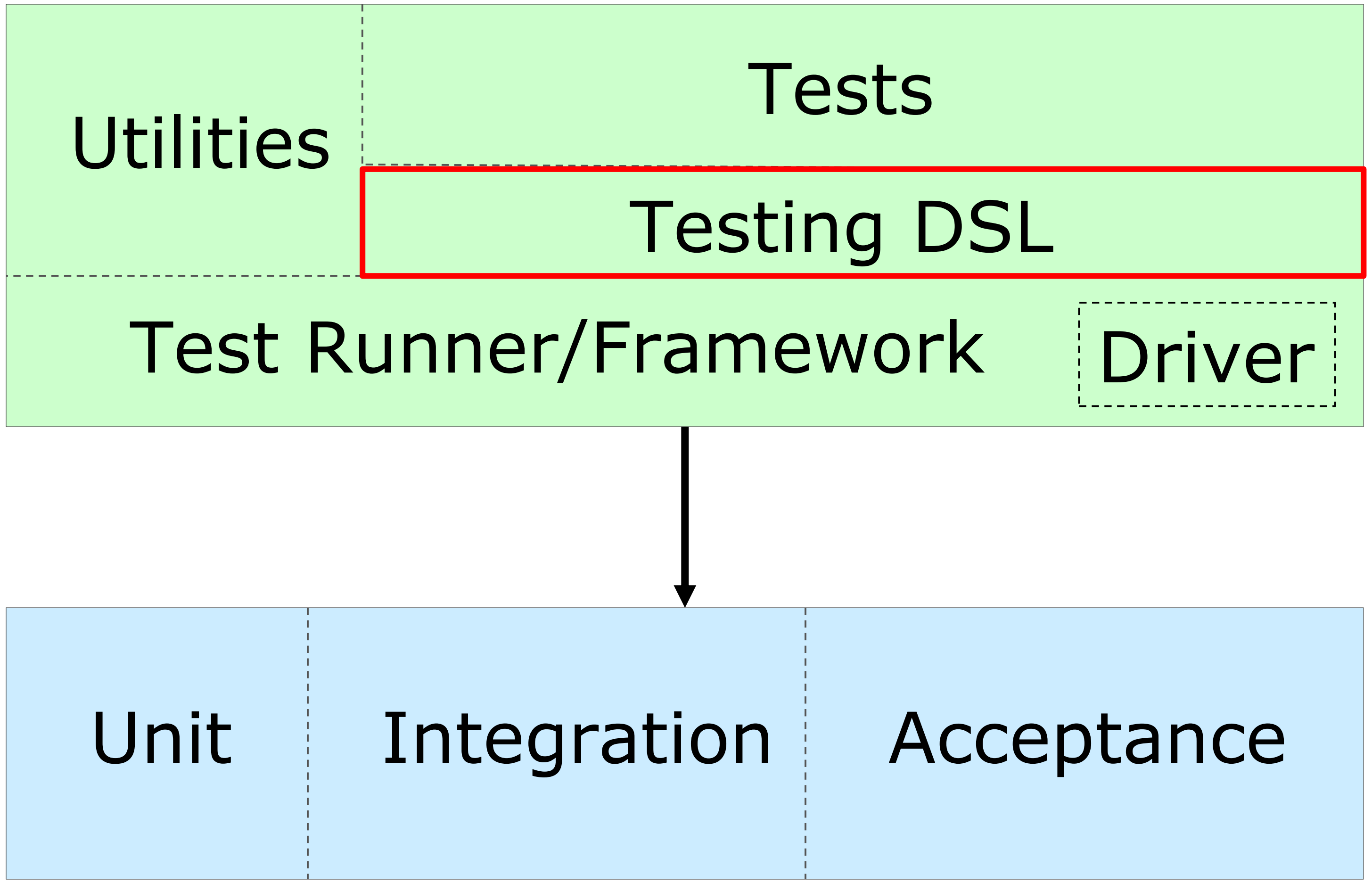
Understanding testing



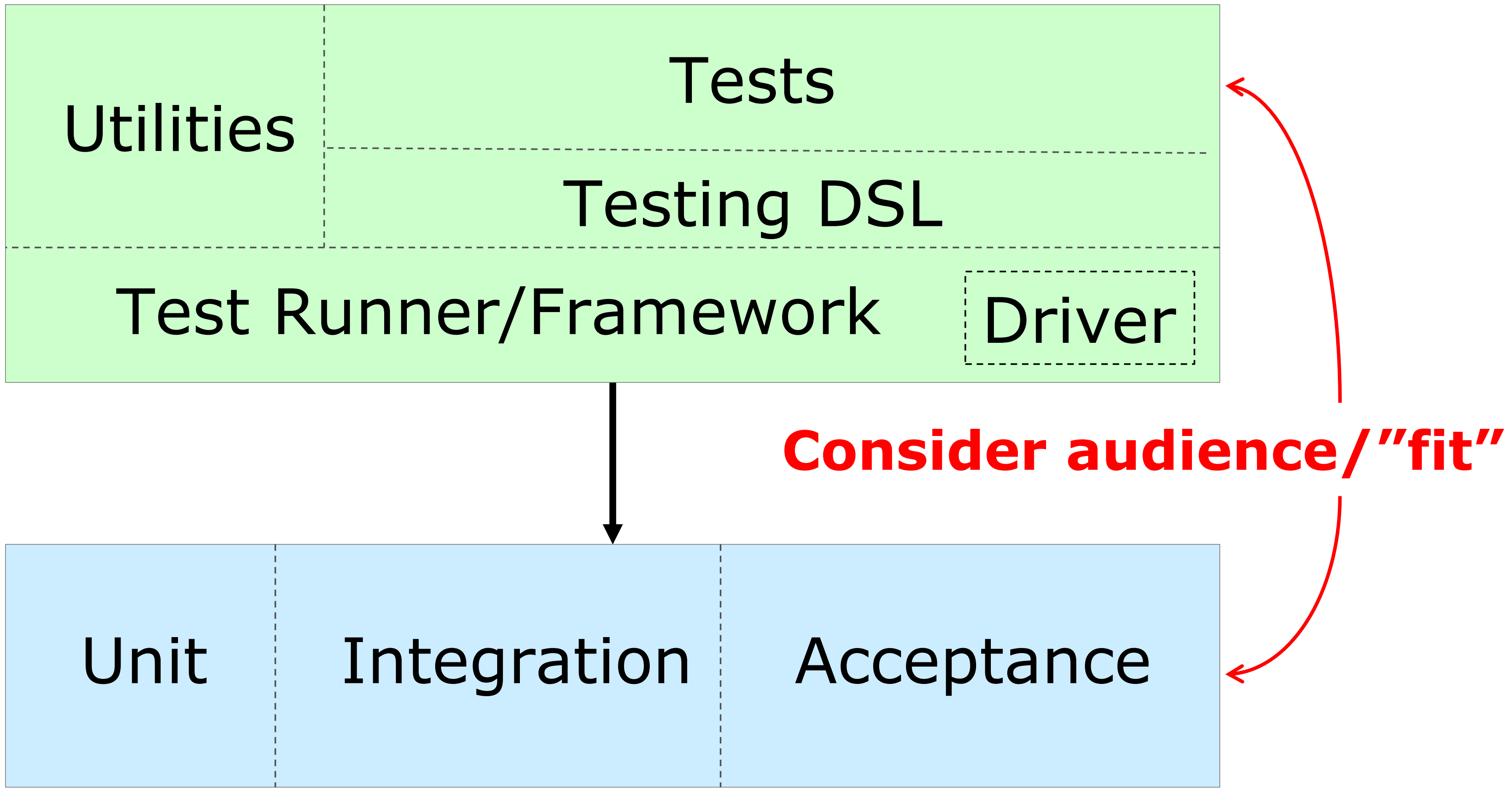
Understanding testing with a DSL



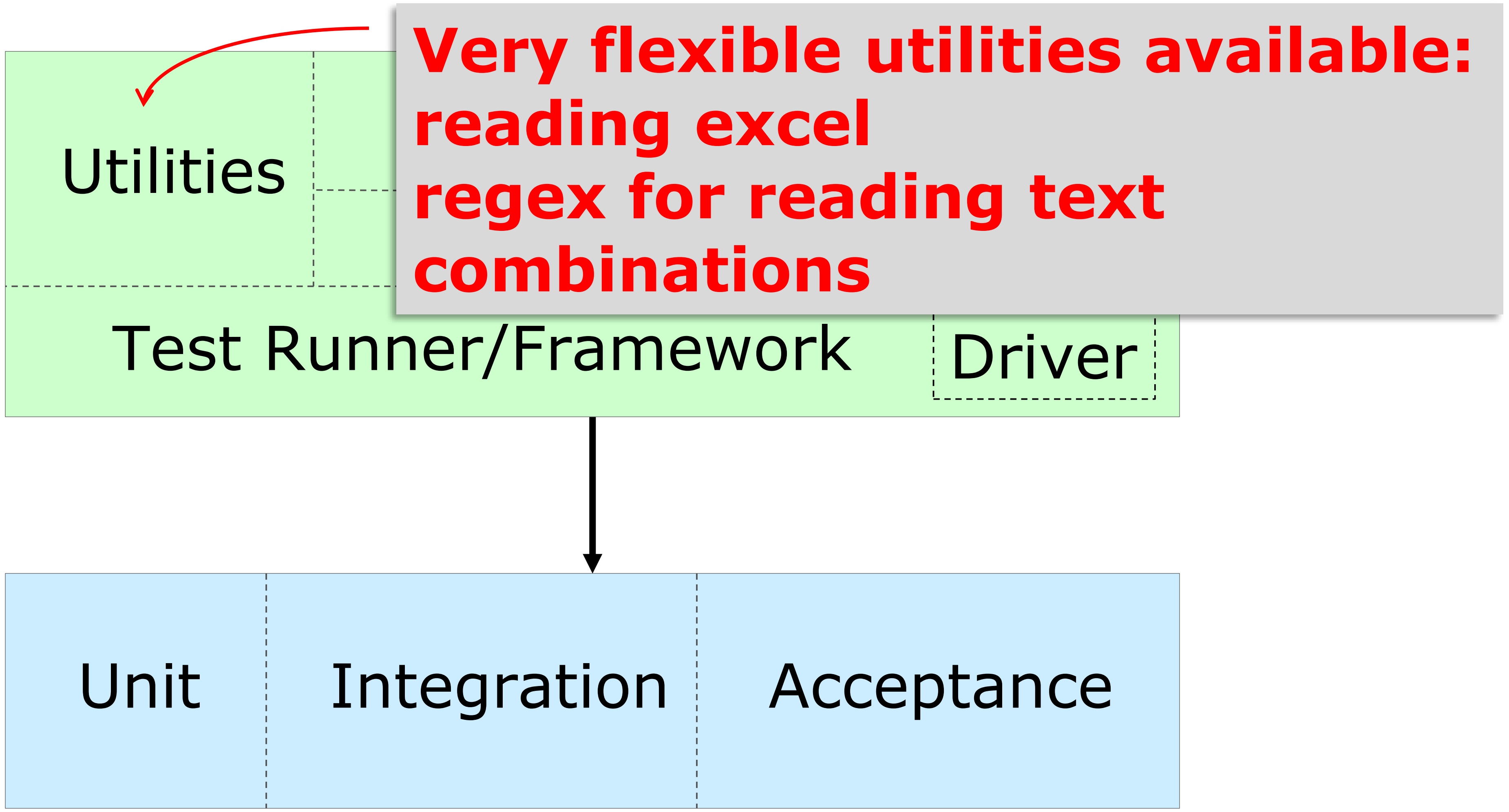
Understanding testing with a DSL



Why Groovy?

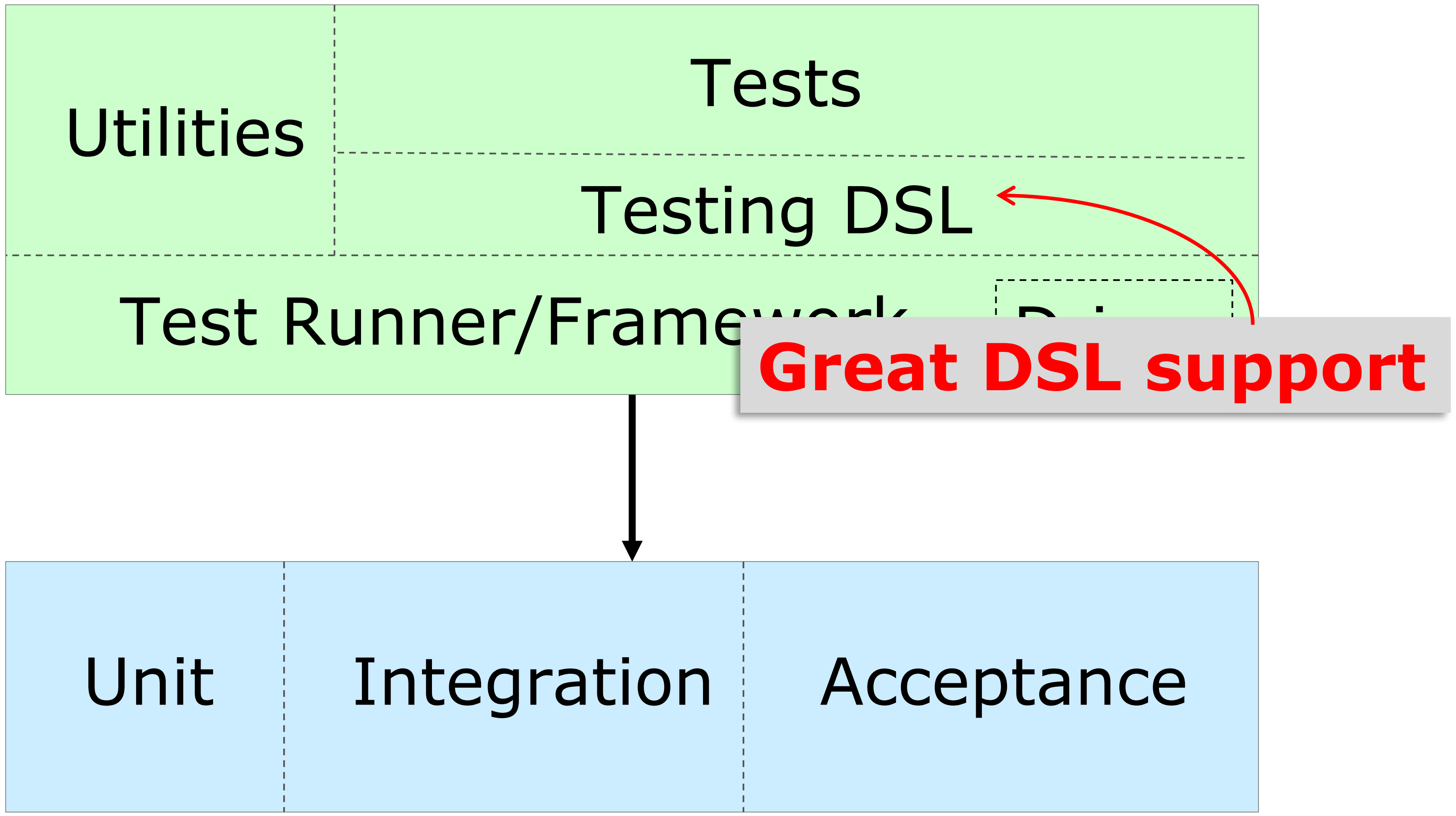

Utilities
Tests
Testing DSL
Test Runner/Framework
Driver
Consider audience/"fit"
Unit
Integration
Acceptance

Why Groovy?

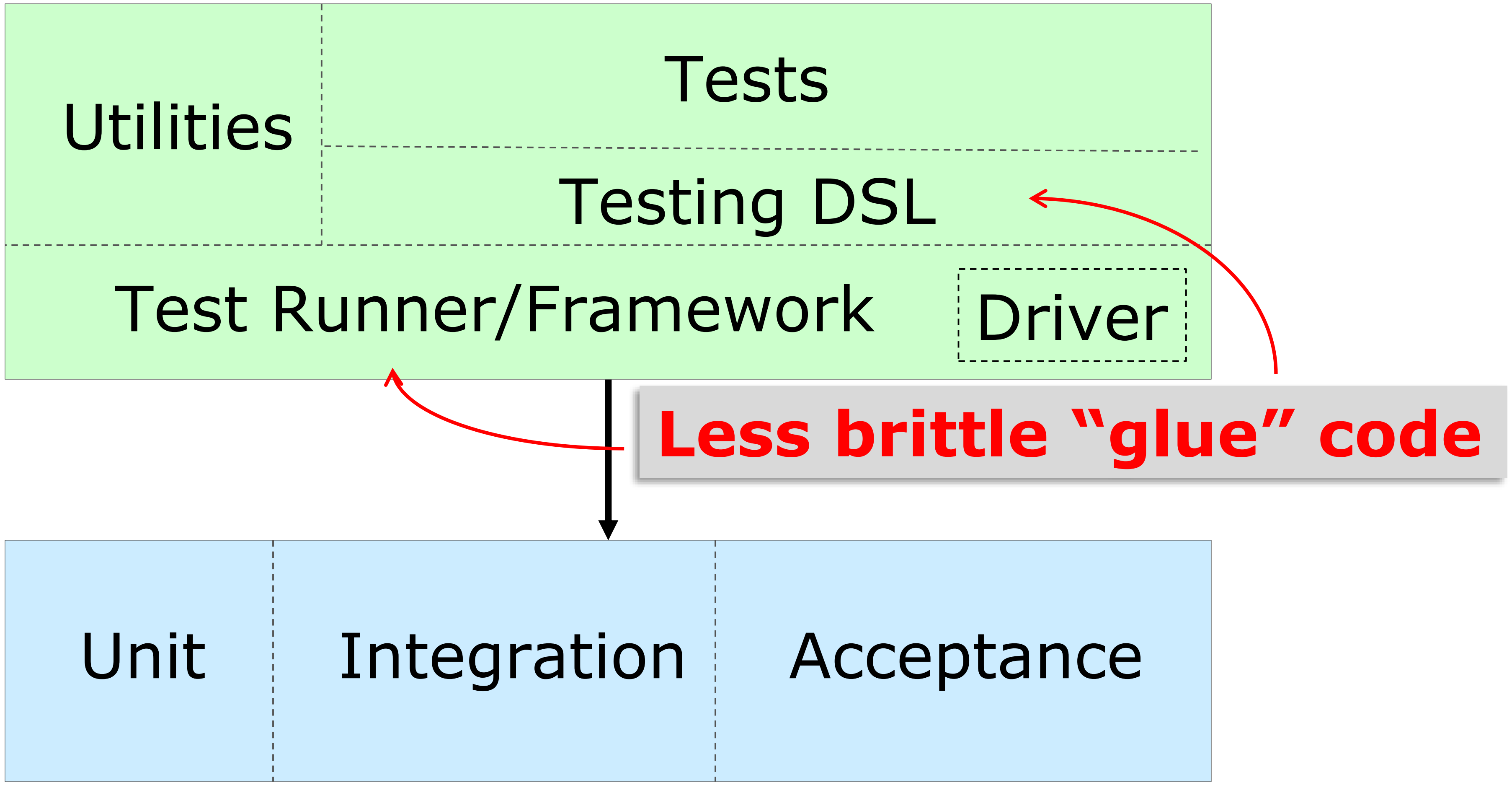


Very flexible utilities available:
reading excel
regex for reading text
combinations

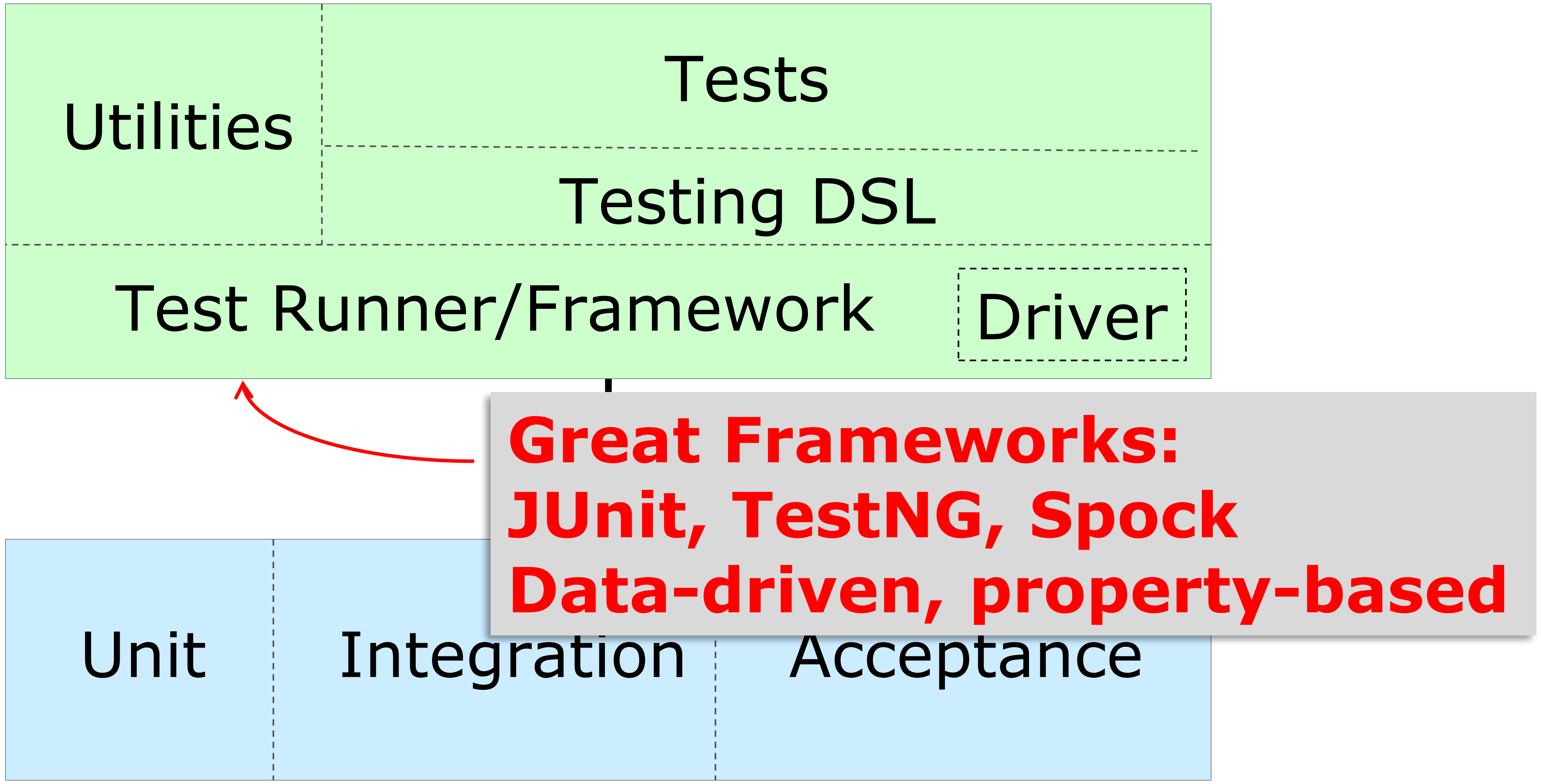
Why Groovy?


Utilities
Tests
Testing DSL
Test Runner/Framework
Great DSL support
Unit
Integration
Acceptance

Why Groovy?

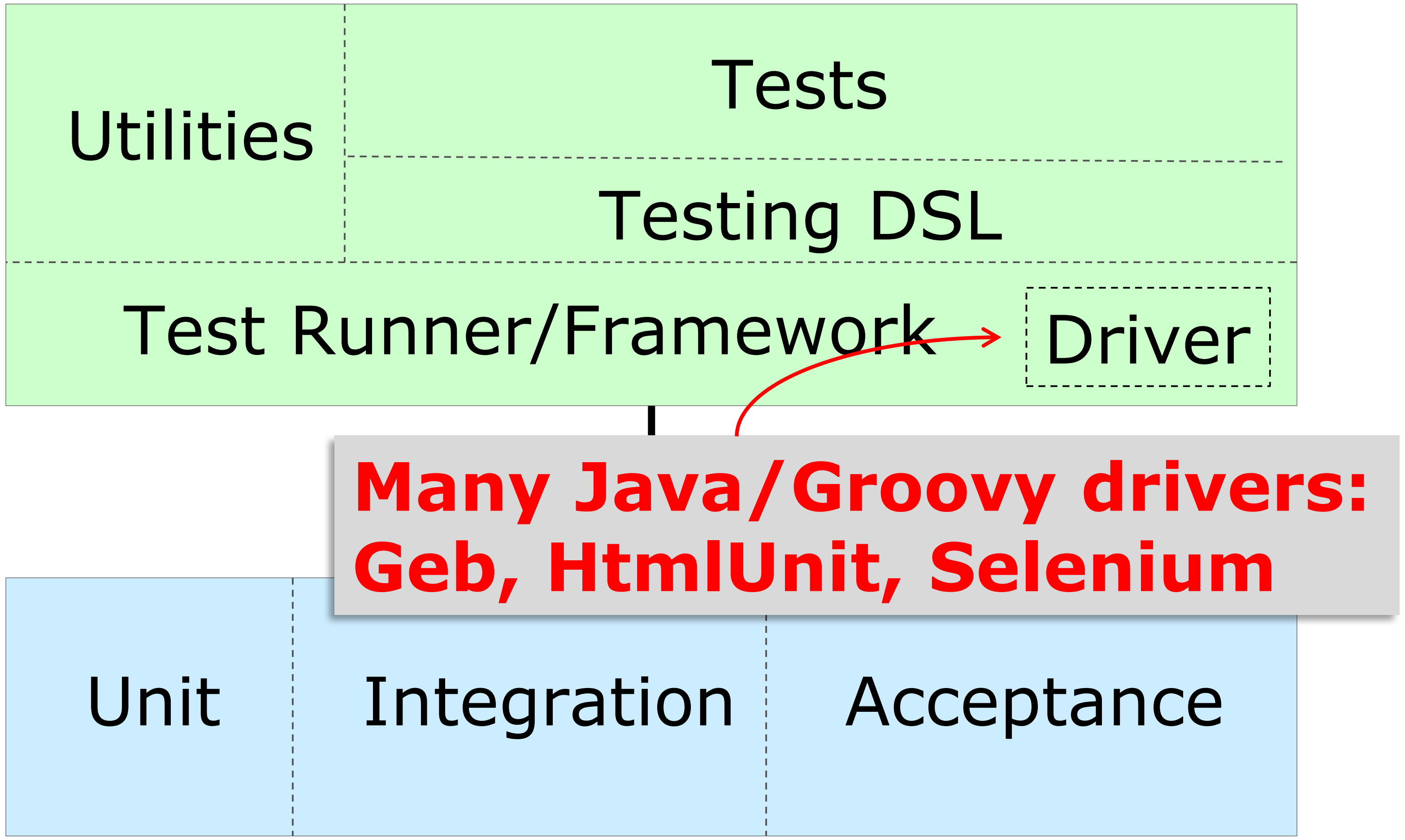

Utilities
Tests
Testing DSL
Test Runner/Framework
Driver
Less brittle "glue" code
Unit
Integration
Acceptance

Why Groovy?



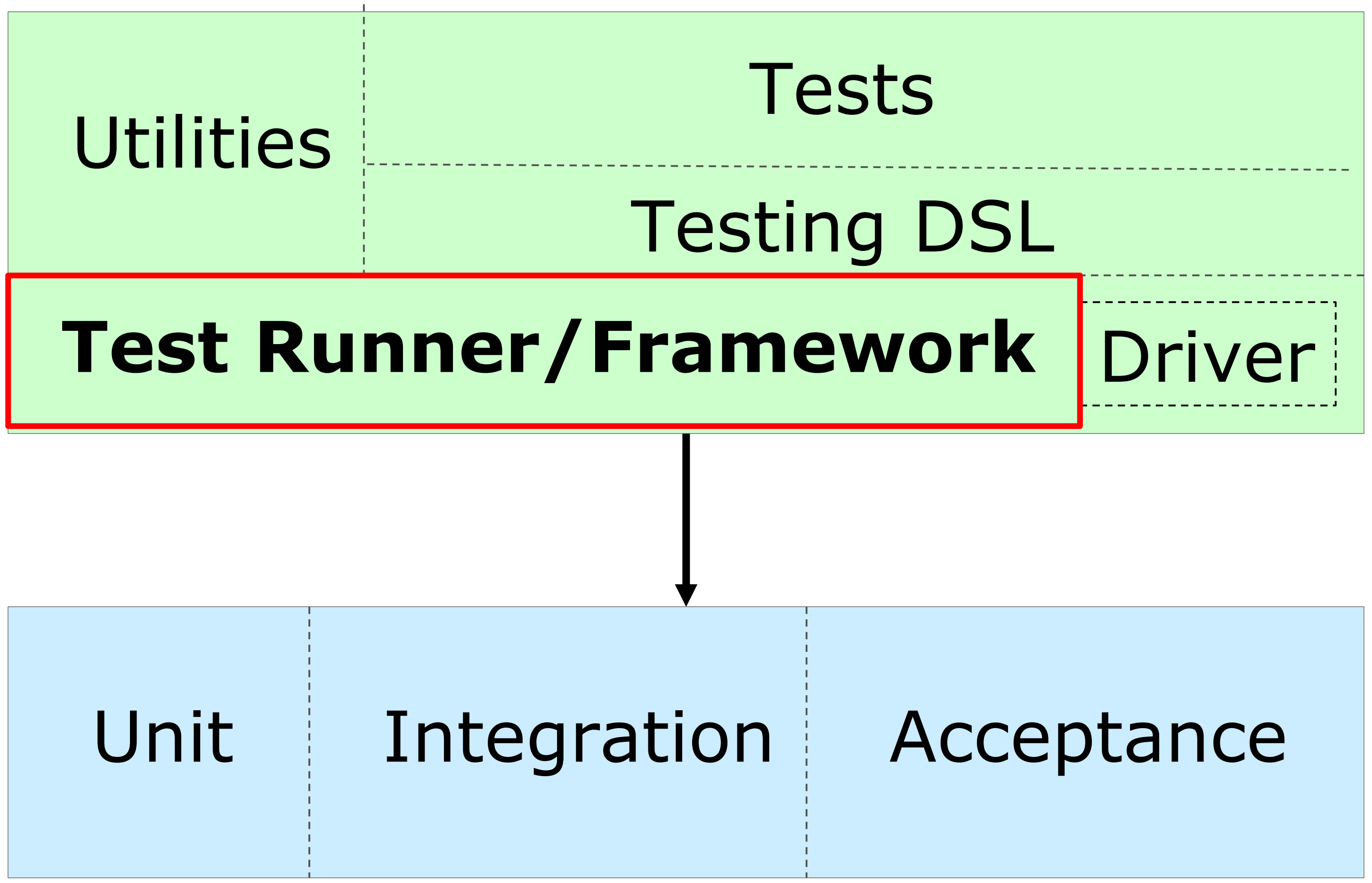
Great Frameworks:
JUnit, TestNG, Spock
Data-driven, property-based

Why Groovy?



Many Java/Groovy drivers: Geb, HtmlUnit, Selenium

Looking at testing frameworks



Testing Frameworks

None

JUnit 3

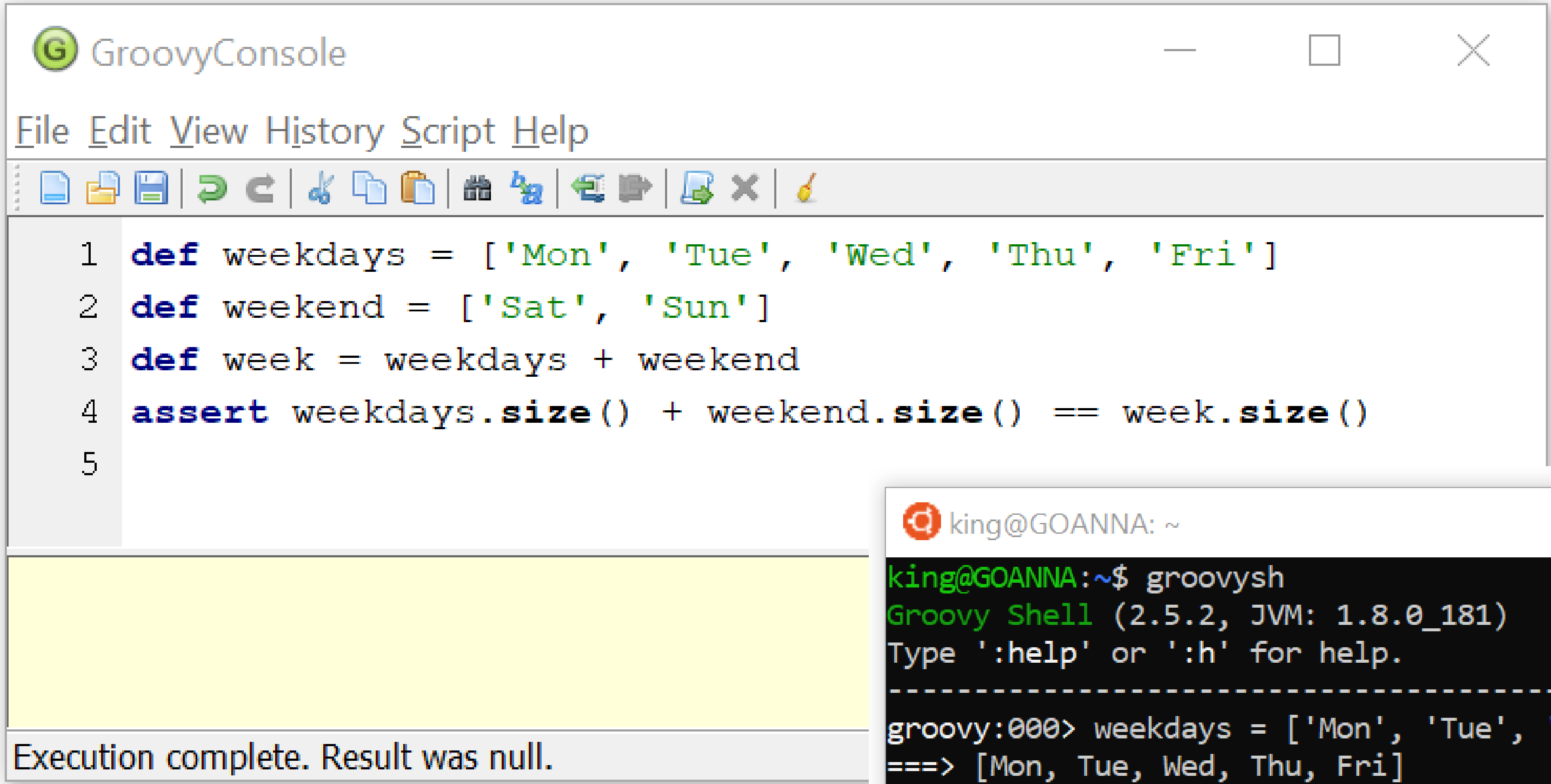
JUnit 4

JUnit 5

TestNG

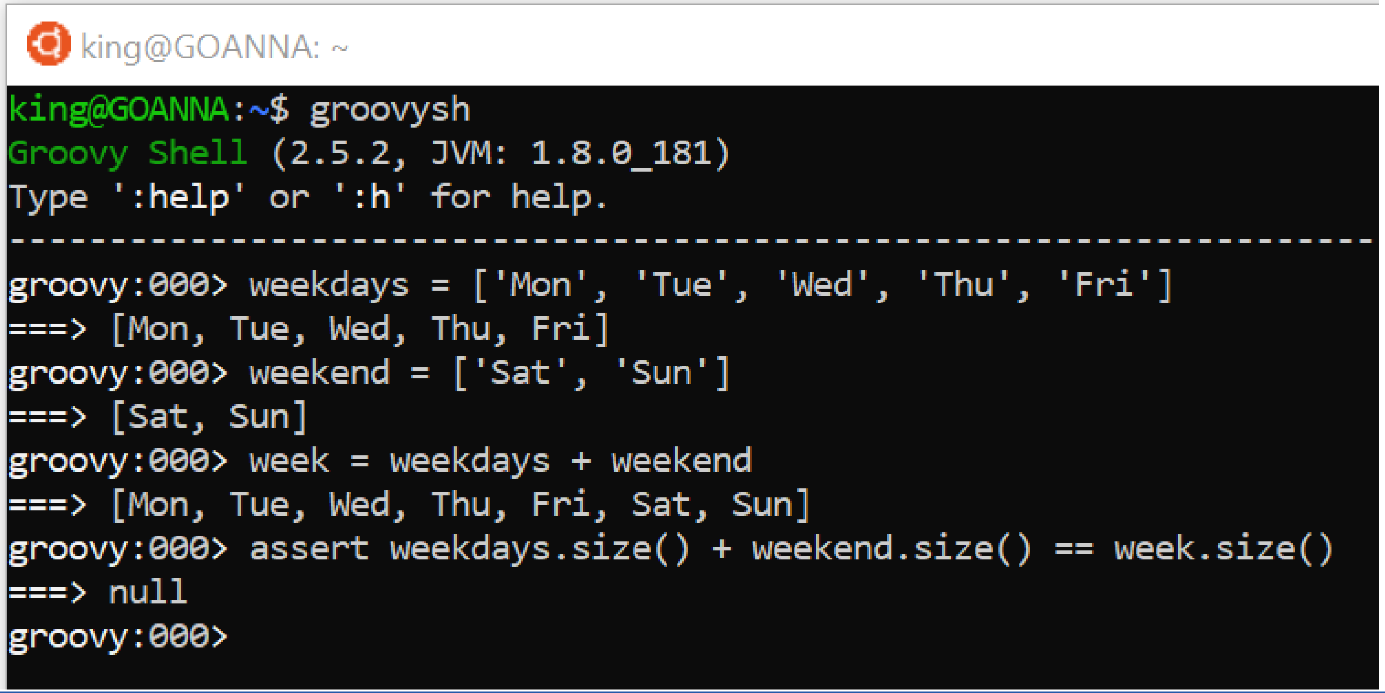
Spock

No framework



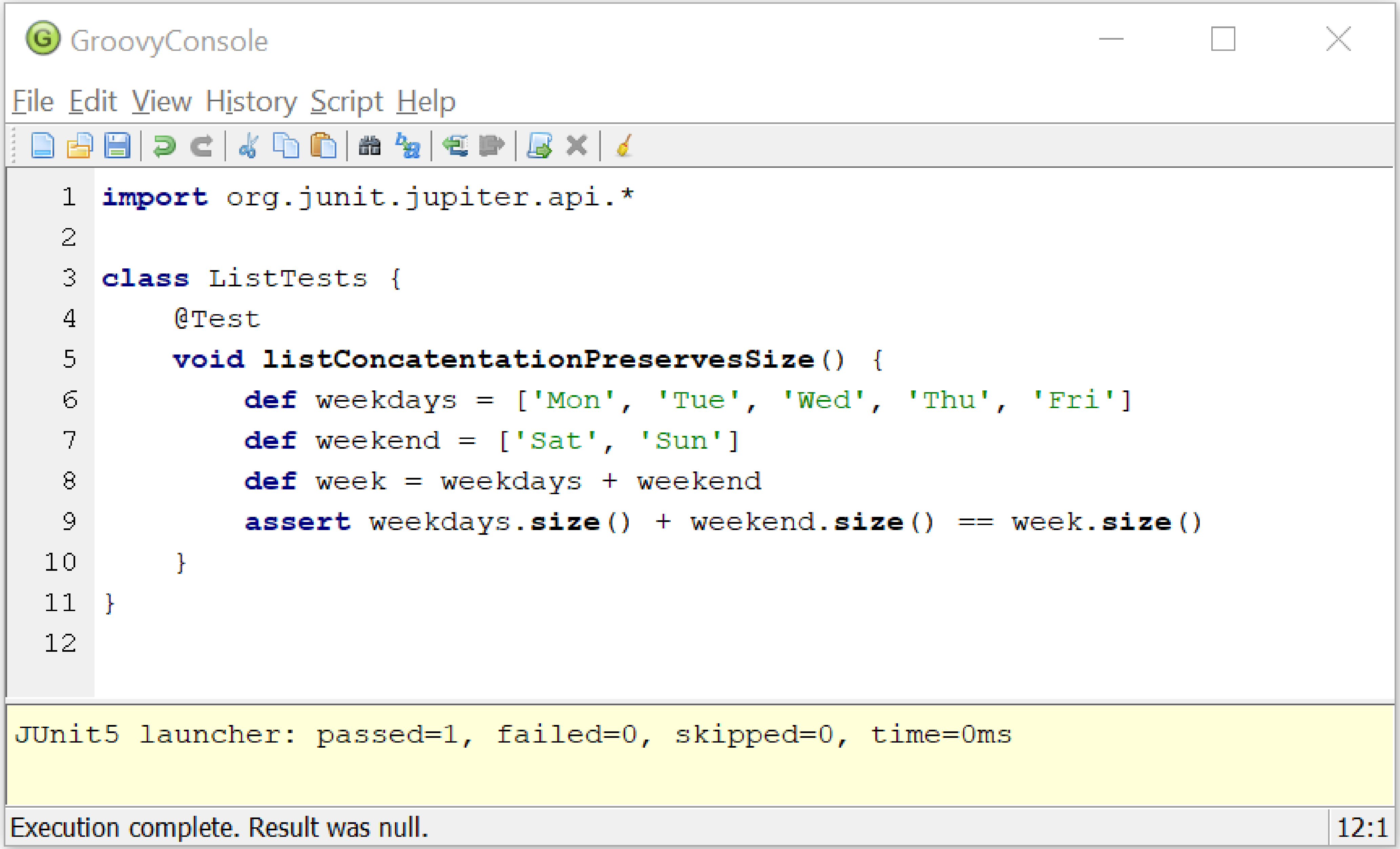
```
def weekdays = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']
def weekend = ['Sat', 'Sun']
def week = weekdays + weekend
assert weekdays.size() + weekend.size() == week.size()

```

Execution complete. Result was null.



```
king@GOANNA:~$ groovysh
Groovy Shell (2.5.2, JVM: 1.8.0_181)
Type ':help' or ':h' for help.
-------------------------------------------------------------------
groovy:000> weekdays = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']
===> [Mon, Tue, Wed, Thu, Fri]
groovy:000> weekend = ['Sat', 'Sun']
===> [Sat, Sun]
groovy:000> week = weekdays + weekend
===> [Mon, Tue, Wed, Thu, Fri, Sat, Sun]
groovy:000> assert weekdays.size() + weekend.size() == week.size()
===> null
groovy:000>
```

JUnit5

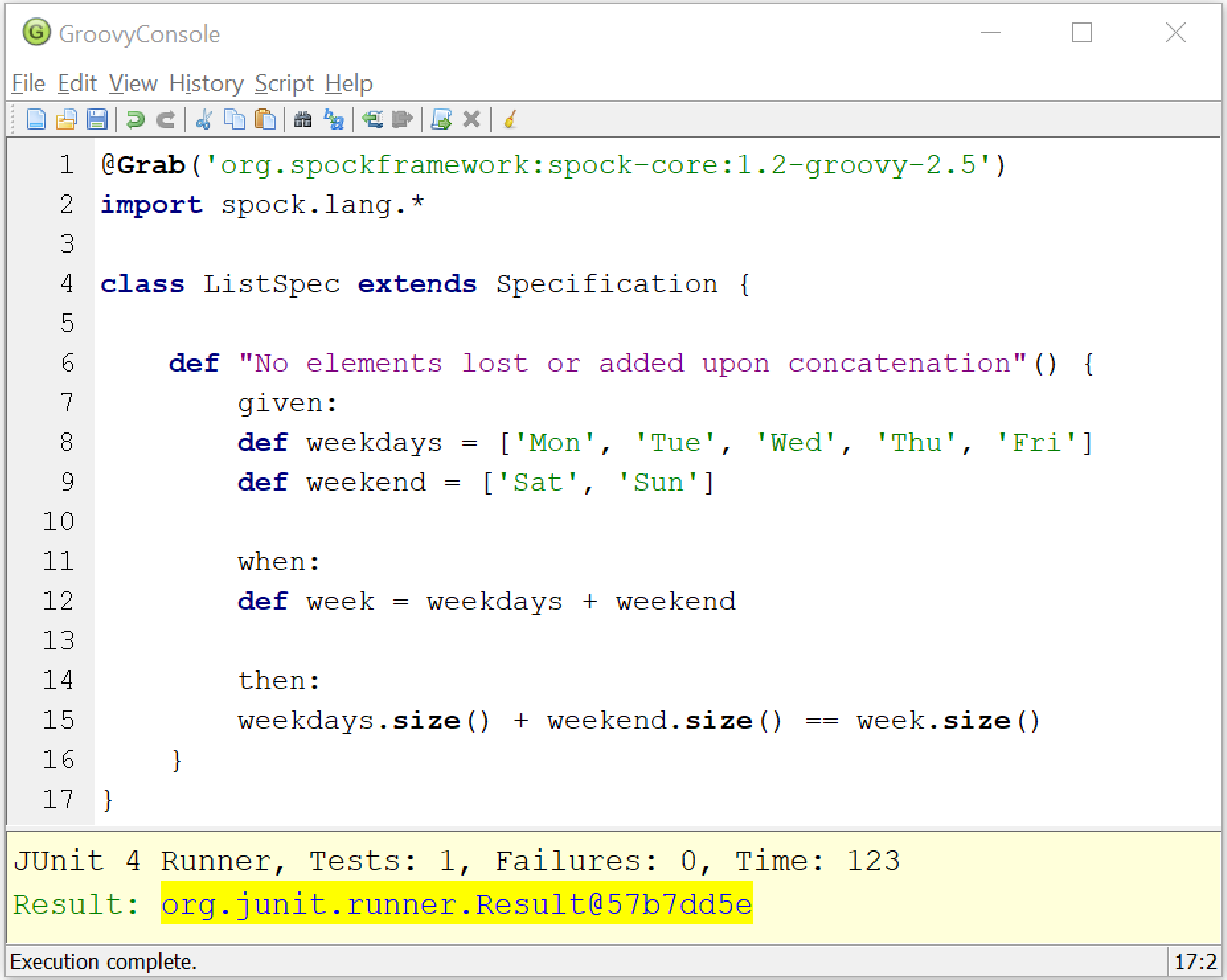


```
import org.junit.jupiter.api.*

class ListTests {
    @Test
    void listConcatententationPreservesSize() {
        def weekdays = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']
        def weekend = ['Sat', 'Sun']
        def week = weekdays + weekend
        assert weekdays.size() + weekend.size() == week.size()
    }
}

```

```
JUnit5 launcher: passed=1, failed=0, skipped=0, time=0ms
```

Execution complete. Result was null. 12:1

Spock



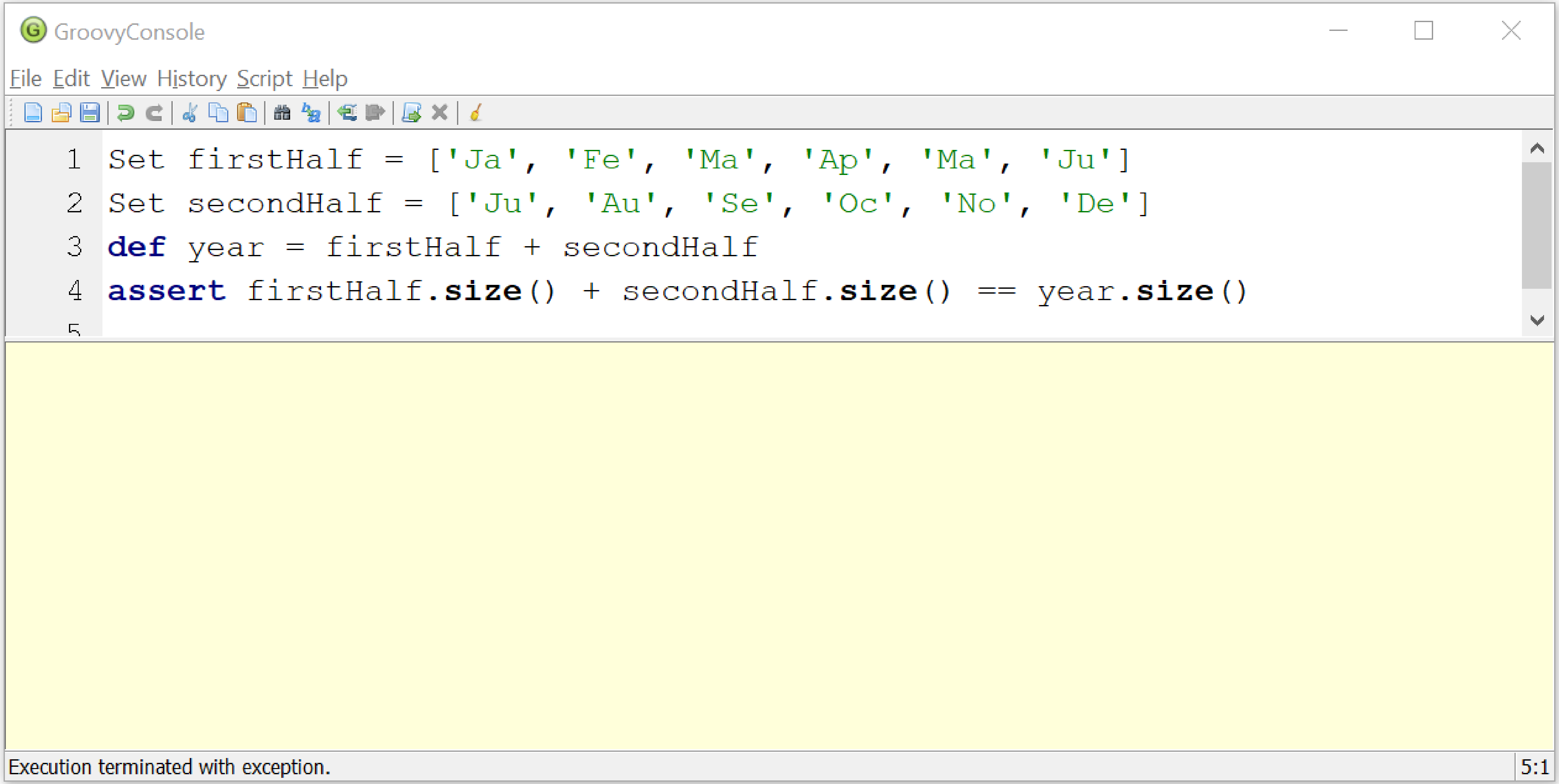
```
@Grab('org.spockframework:spock-core:1.2-groovy-2.5')
import spock.lang.*

class ListSpec extends Specification {

    def "No elements lost or added upon concatenation"() {
        given:
        def weekdays = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']
        def weekend = ['Sat', 'Sun']

        when:
        def week = weekdays + weekend

        then:
        weekdays.size() + weekend.size() == week.size()
    }
}
```

```
JUnit 4 Runner, Tests: 1, Failures: 0, Time: 123
Result: org.junit.runner.Result@57b7dd5e
```

Execution complete. 17:2

BDD: Given [initial context], when [event occurs], then [ensure some outcomes]

Power Assert

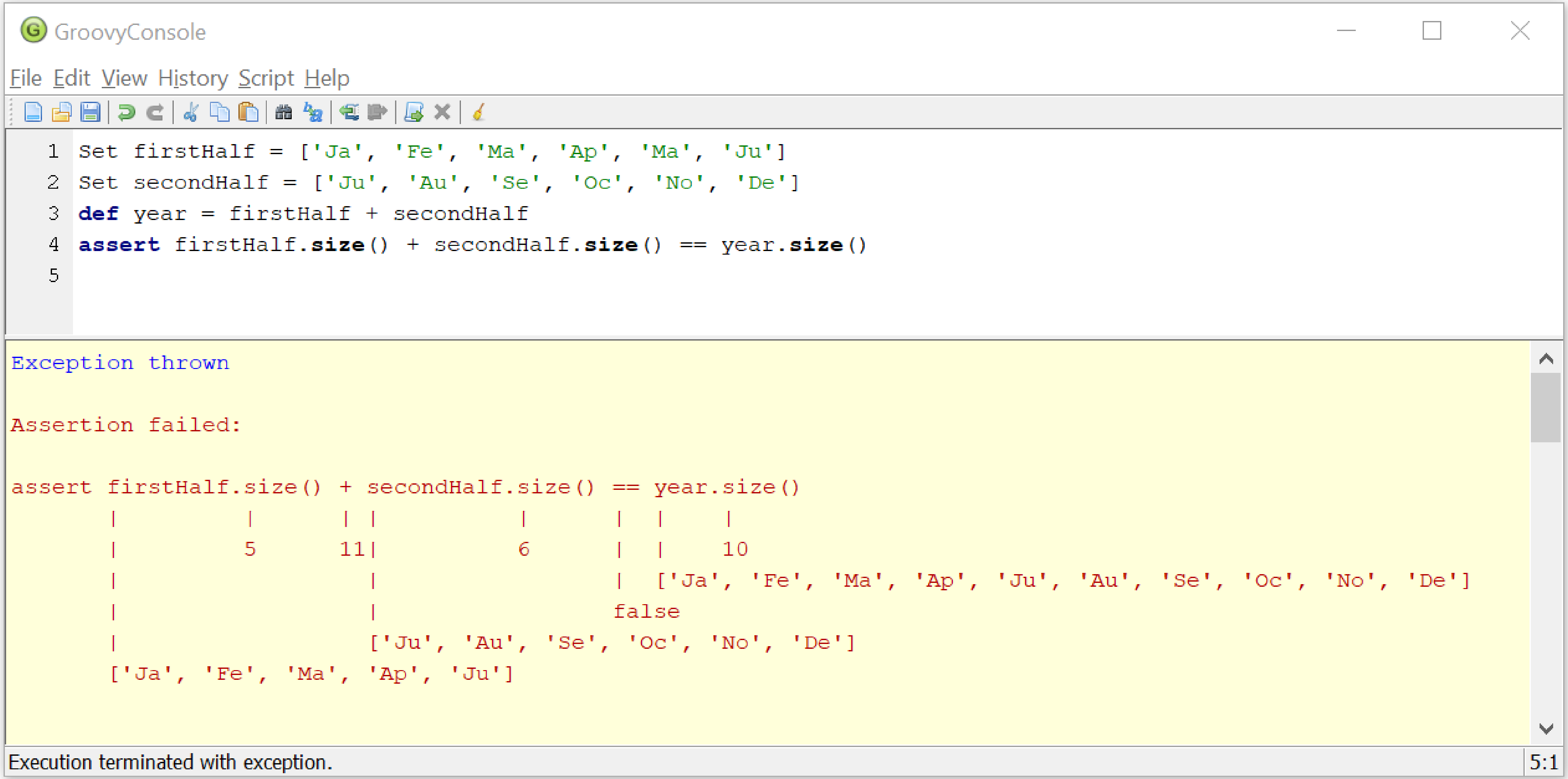


```
Set firstHalf = ['Ja', 'Fe', 'Ma', 'Ap', 'Ma', 'Ju']
Set secondHalf = ['Ju', 'Au', 'Se', 'Oc', 'No', 'De']
def year = firstHalf + secondHalf
assert firstHalf.size() + secondHalf.size() == year.size()
```

Execution terminated with exception.

5:1

Power Assert



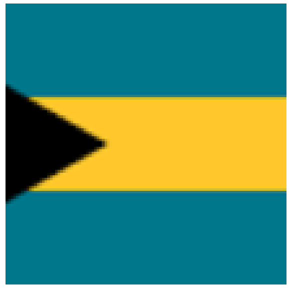
```
Set firstHalf = ['Ja', 'Fe', 'Ma', 'Ap', 'Ma', 'Ju']
Set secondHalf = ['Ju', 'Au', 'Se', 'Oc', 'No', 'De']
def year = firstHalf + secondHalf
assert firstHalf.size() + secondHalf.size() == year.size()
```

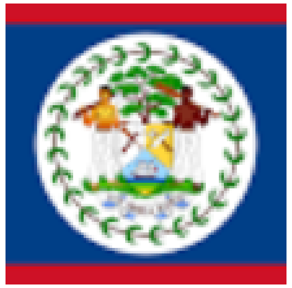
```
Exception thrown

Assertion failed:

assert firstHalf.size() + secondHalf.size() == year.size()
       |         |      | |          |      |  |    |
       |         5      11|          6      |  |    10
       |                  |                 |  ['Ja', 'Fe', 'Ma', 'Ap', 'Ju', 'Au', 'Se', 'Oc', 'No', 'De']
       |                  |                 false
       |                  ['Ju', 'Au', 'Se', 'Oc', 'No', 'De']
       ['Ja', 'Fe', 'Ma', 'Ap', 'Ju']
```

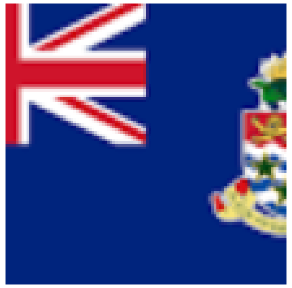
Execution terminated with exception.

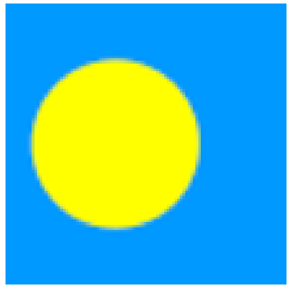
5:1

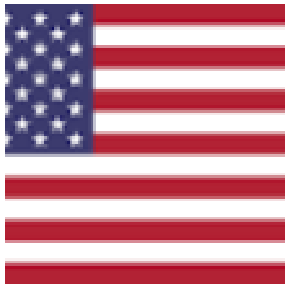
What is the distinguishing characteristic?


Bahamas


Belize


Cayman Islands

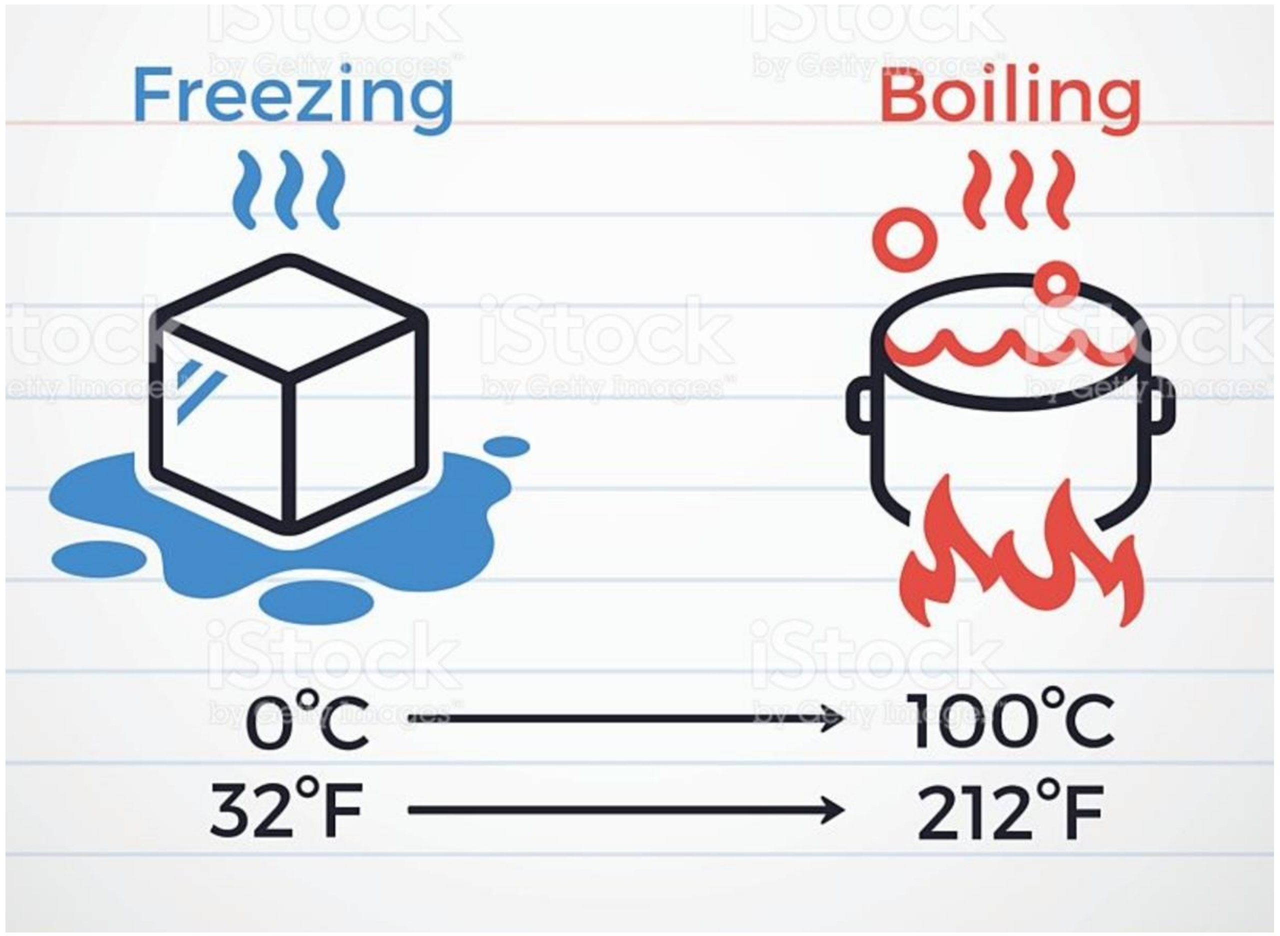

Palau


United States of America

Built-in assertions

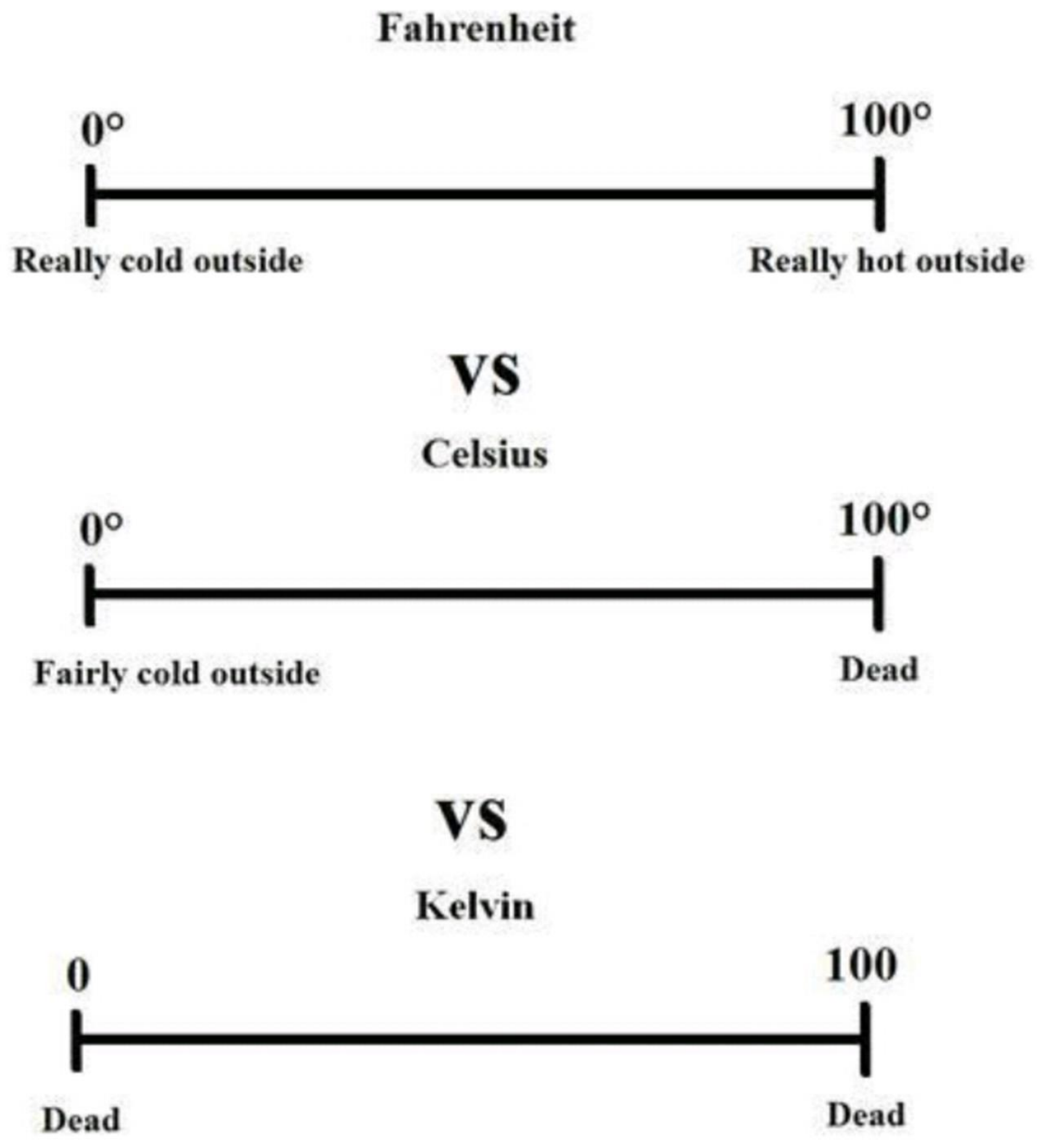
```
class Converter {
    static celsius (fahrenheit) {
        (fahrenheit - 32) * 5 / 9
    }
}
```

Temperature scales



http://www.istockphoto.com/dk/vector/freezing-and-boiling-points-in-celsius-and-fahrenheit-gm533454646-94496887

Temperature scales



http://imgur.com/gallery/KB3nyXX

Built-in assertions

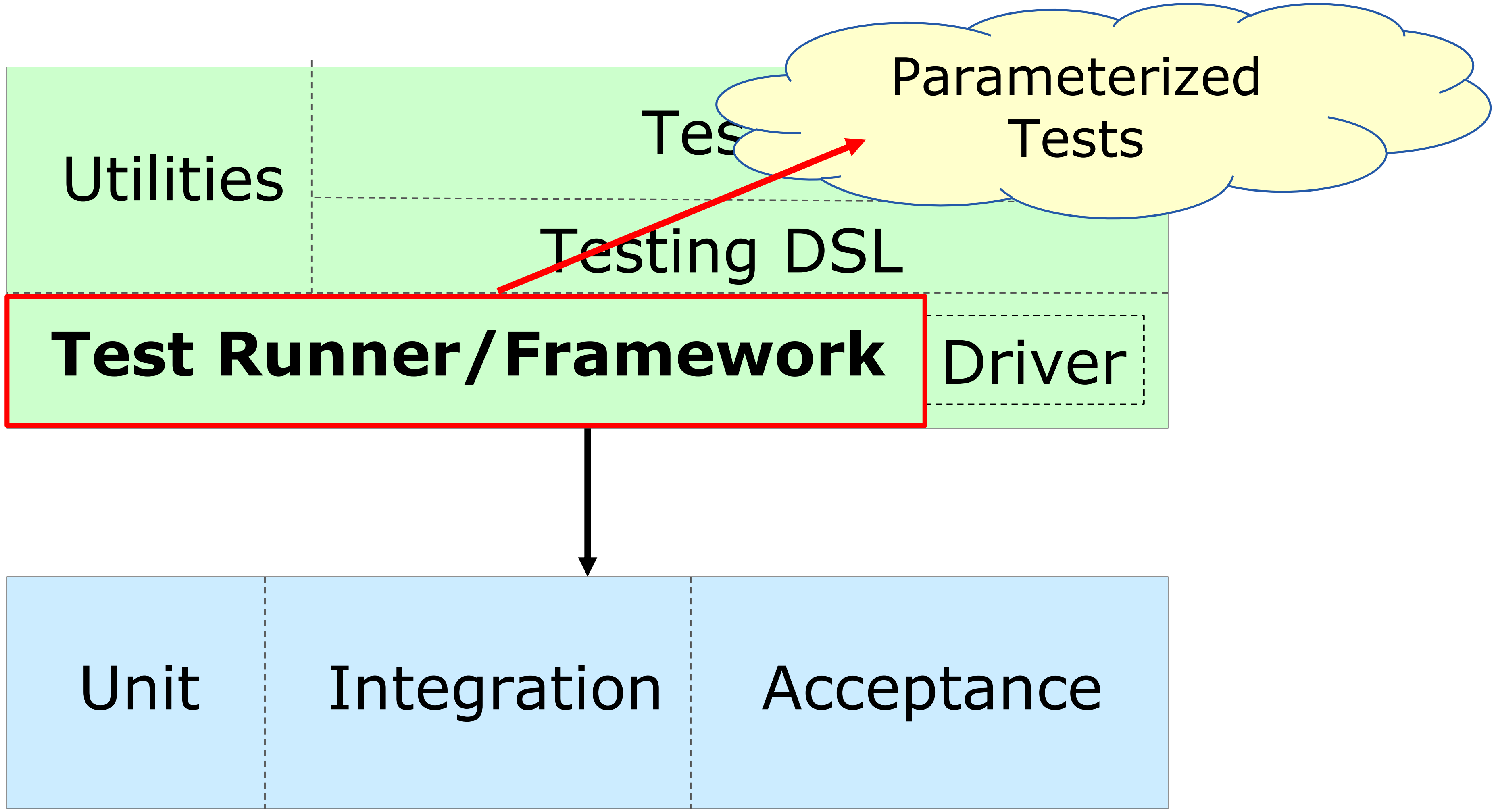
```
class Converter {
    static celsius (fahrenheit) {
        (fahrenheit - 32) * 5 / 9
    }
}
```

```
import static Converter.celsius

assert  20 == celsius(68)
assert  35 == celsius(95)
assert -17 == celsius(0).toInteger()
assert   0 == celsius(32)
```

JUnit5

```
import org.junit.jupiter.api.*
import static Converter.celsius

class ConverterJUnit5Tests {
    @Test
    void freezing() {
        assert celsius(32) == 0
    }

    @Test
    void boiling() {
        assert celsius(212) == 100
    }
}
```

Looking at testing frameworks – other features

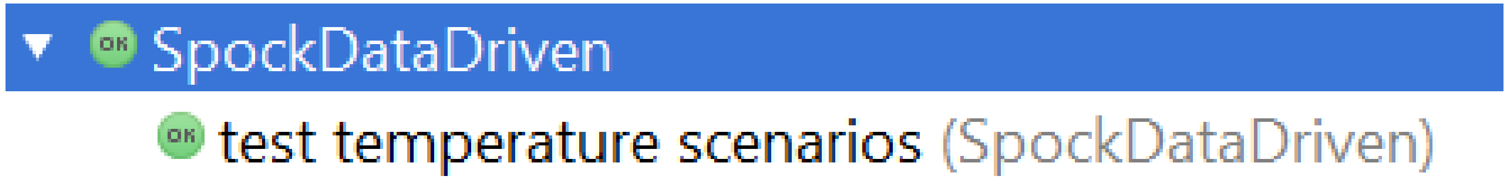


Parameterized

```
import org.junit.Test
import org.junit.runner.RunWith
import org.junit.runners.Parameterized
import org.junit.runners.Parameterized.Parameters
import static Converter.celsius

@RunWith(Parameterized)
class DataDrivenJUnitTest {
  private c, f, scenario

  @Parameters static scenarios() {[
      [0,   32,  'Freezing'],
      [20,  68,  'Garden party conditions'],
      [35,  95,  'Beach conditions'],
      [100, 212, 'Boiling']
  ]*.toArray()}

  DataDrivenJUnitTest(c, f, scenario)
    this.c = c
    this.f = f
    this.scenario = scenario
  }

  @Test void convert() {
    def actual = celsius(f)
    def msg = "$scenario: ${f}°F should convert into ${c}°C"
    assert c == actual, msg
  }
}
```

Spock

```
import spock.lang.*
import static Converter.celsius

class SpockDataDriven extends Specification {
  def "test temperature scenarios"() {
    expect:
    celsius(tempF) == tempC

    where:
    scenario                  | tempF || tempC
    'Freezing'                |    32 ||     0
    'Garden party conditions' |    68 ||    20
    'Beach conditions'        |    95 ||    35
    'Boiling'                 |   212 ||   100
  }
}
```



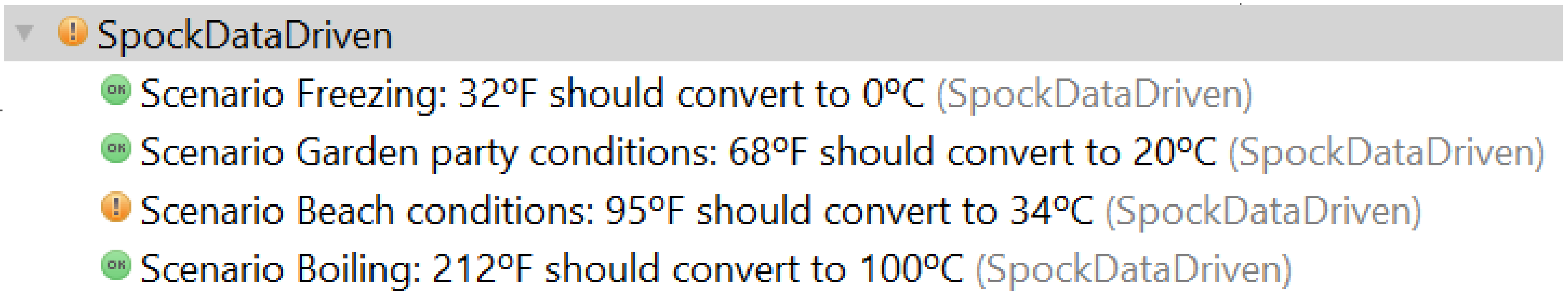
```
▼ SpockDataDriven
    test temperature scenarios (SpockDataDriven)
```

Spock - Celsius

```
import spock.lang.*
import static Converter.celsius

class SpockDataDriven extends Specification {
  @Unroll
  def "Scenario #scenario: #tempF°F should convert to #tempC°C"() {
    expect:
    celsius(tempF) == tempC

    where:
    scenario                    | tempF || tempC
    'Freezing'                  |    32 ||     0
    'Garden party conditions'   |    68 ||    20
    'Beach conditions'          |    95 ||    34
    'Boiling'                   |   212 ||   100
  }
}
```

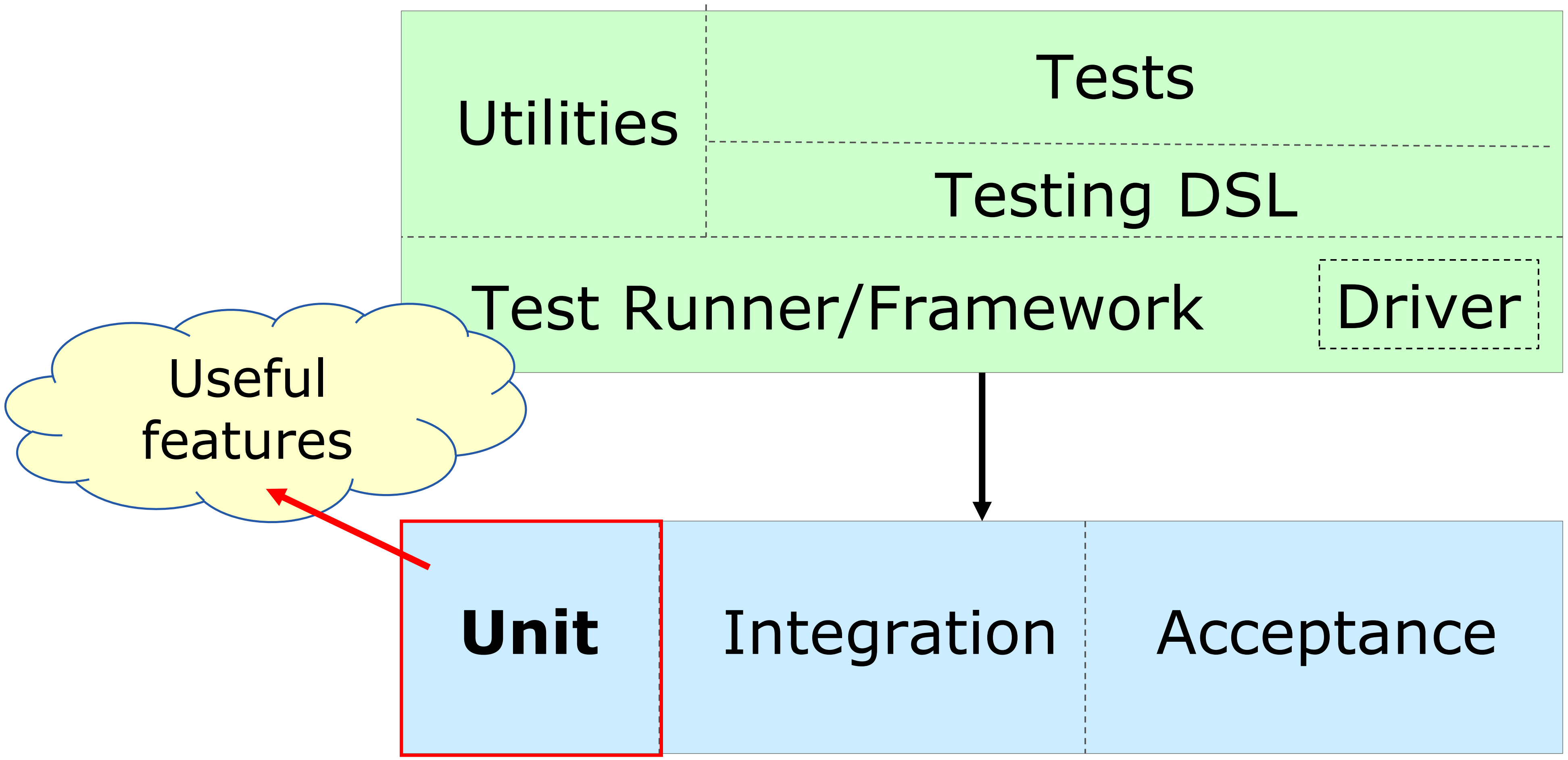
Spock - Celsius

```
import spock.lang.*
import static Converter.celsius

class SpockDataDriven extends Specification {
  @Unroll
  def "Scenario #scenario: #tempF°F should convert to #tempC°C"() {
    expect:
    celsius(tempF) == tempC

    where:
    scenario                      | tempF || tempC
    'Freezing'                    |    32 ||     0
    'Garden party conditions'     |    68 ||    20
    'Beach conditions'            |    95 ||    34
    'Boiling'                     |   212 ||   100
  }
}
```



```
▼ SpockDataDriven
    Scenario Freezing: 32ºF should convert to 0ºC (SpockDataDriven)
    Scenario Garden party conditions: 68ºF should convert to 20ºC (SpockDataDriven)
    Scenario Beach conditions: 95ºF should convert to 34ºC (SpockDataDriven)
    Scenario Boiling: 212ºF should convert to 100ºC (SpockDataDriven)
```

Other Groovy features



Other useful features for Unit testing

Mocking

- Metaprogramming options
- Spocks Mock/Spy

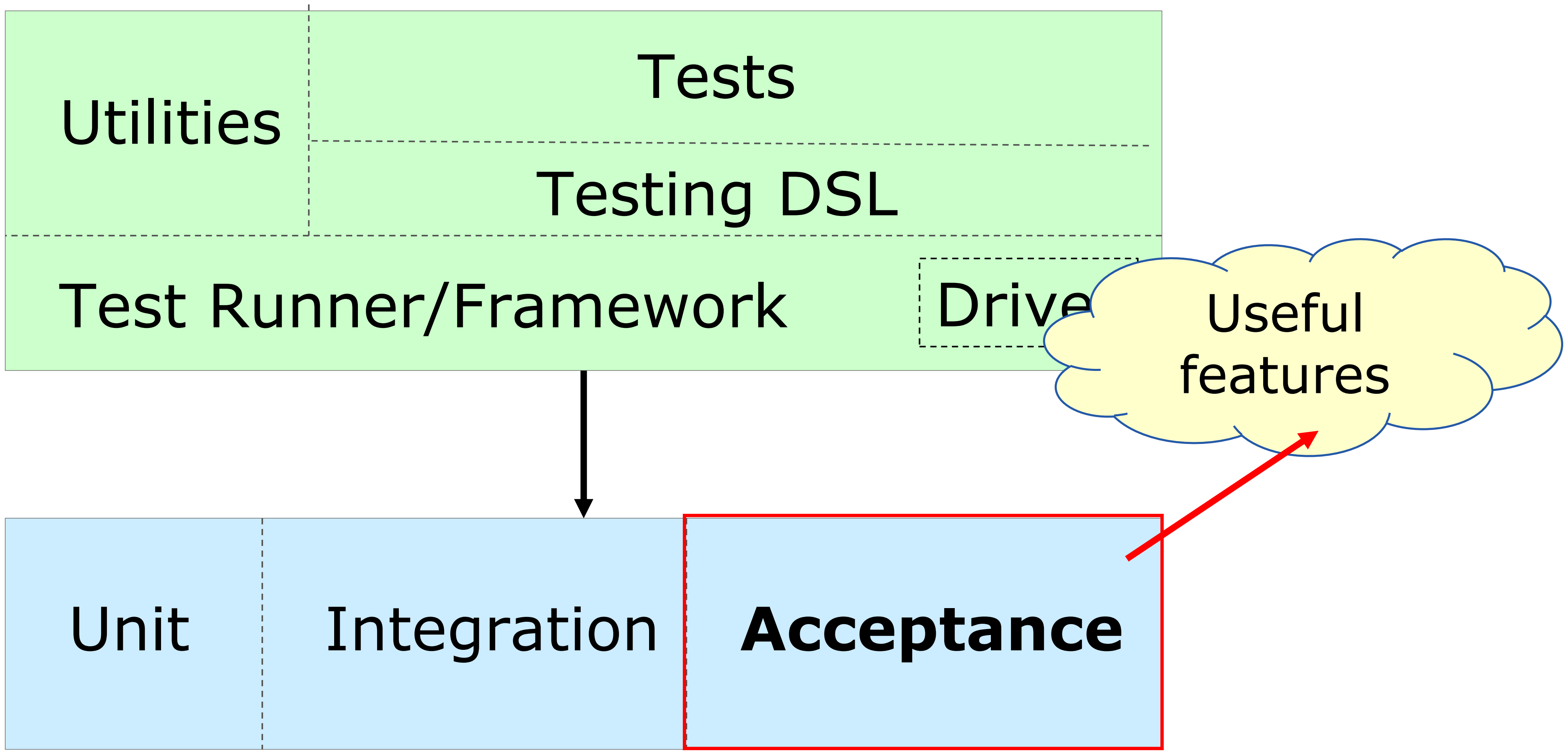
Tricks

- Peeking into private fields/calling private methods

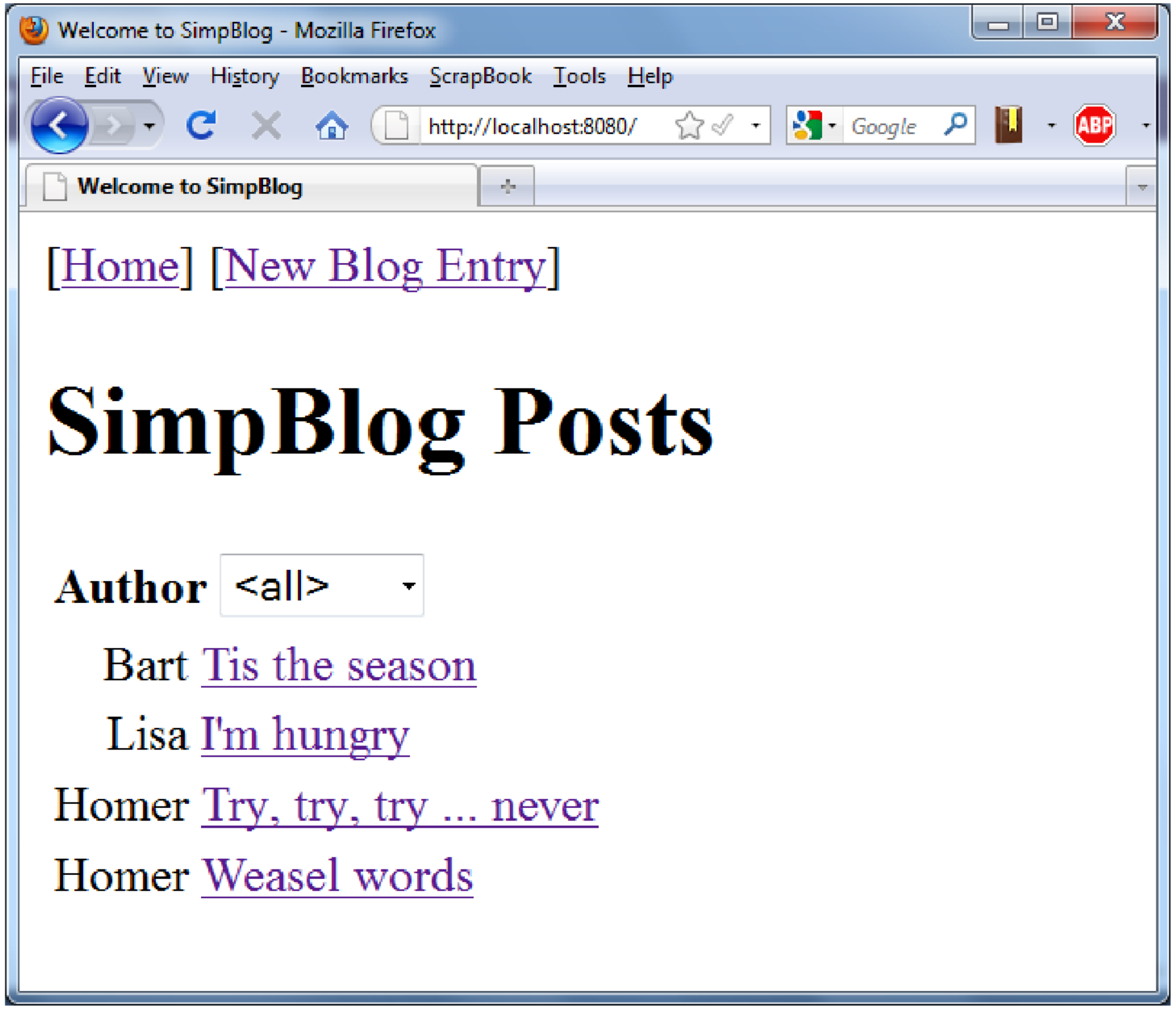
Generally no need for an assertion framework

- Hamcrest, FEST, AssertJ, Google truth

Other Groovy features



Case Study


Welcome to SimpBlog - Mozilla Firefox
File Edit View History Bookmarks ScrapBook Tools Help
http://localhost:8080/
Google
Welcome to SimpBlog

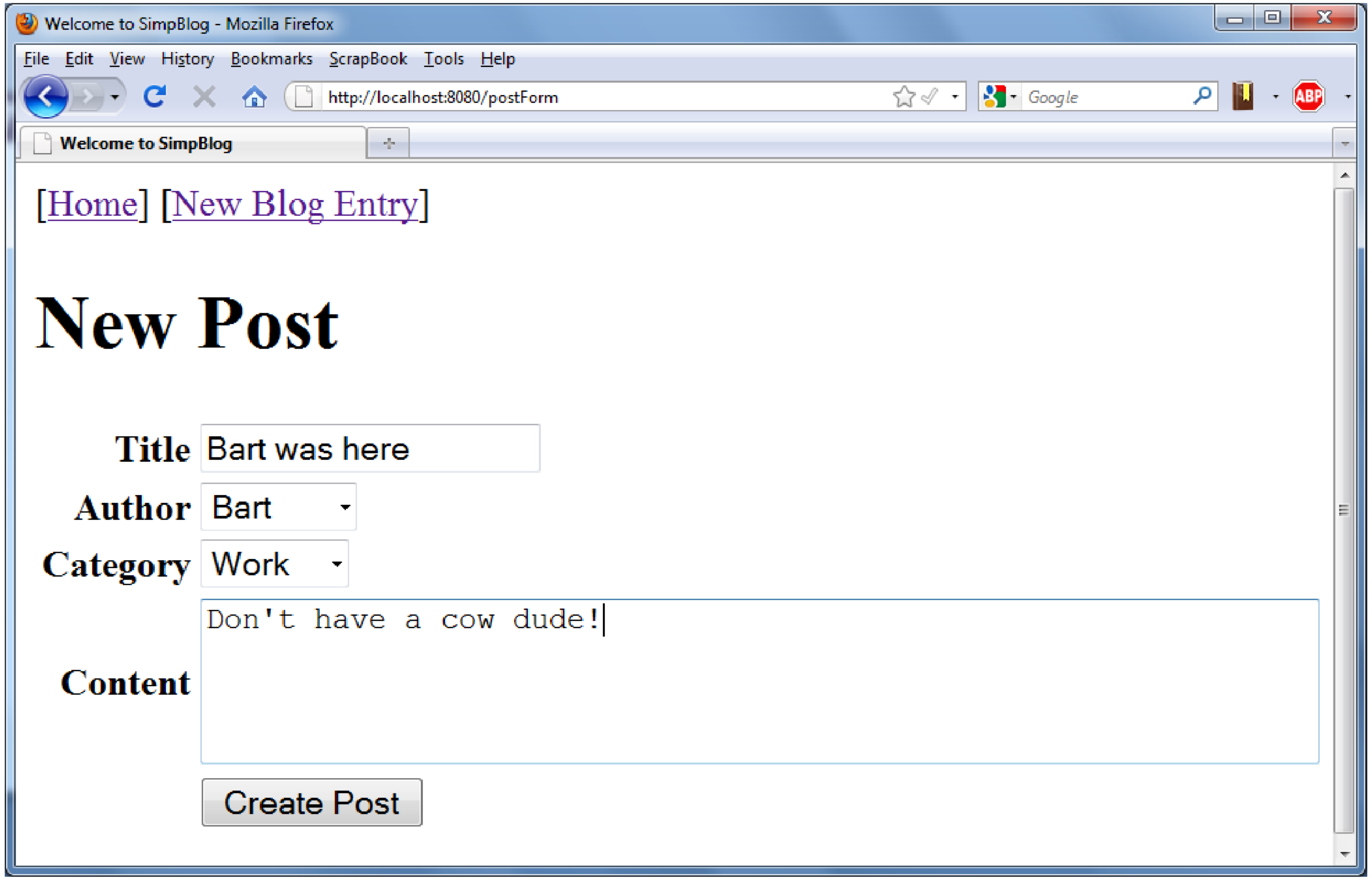
[Home] [New Blog Entry]

SimpBlog Posts

Author <all>

Bart Tis the season
Lisa I'm hungry
Homer Try, try, try ... never
Homer Weasel words

Case Study


Welcome to SimpBlog - Mozilla Firefox

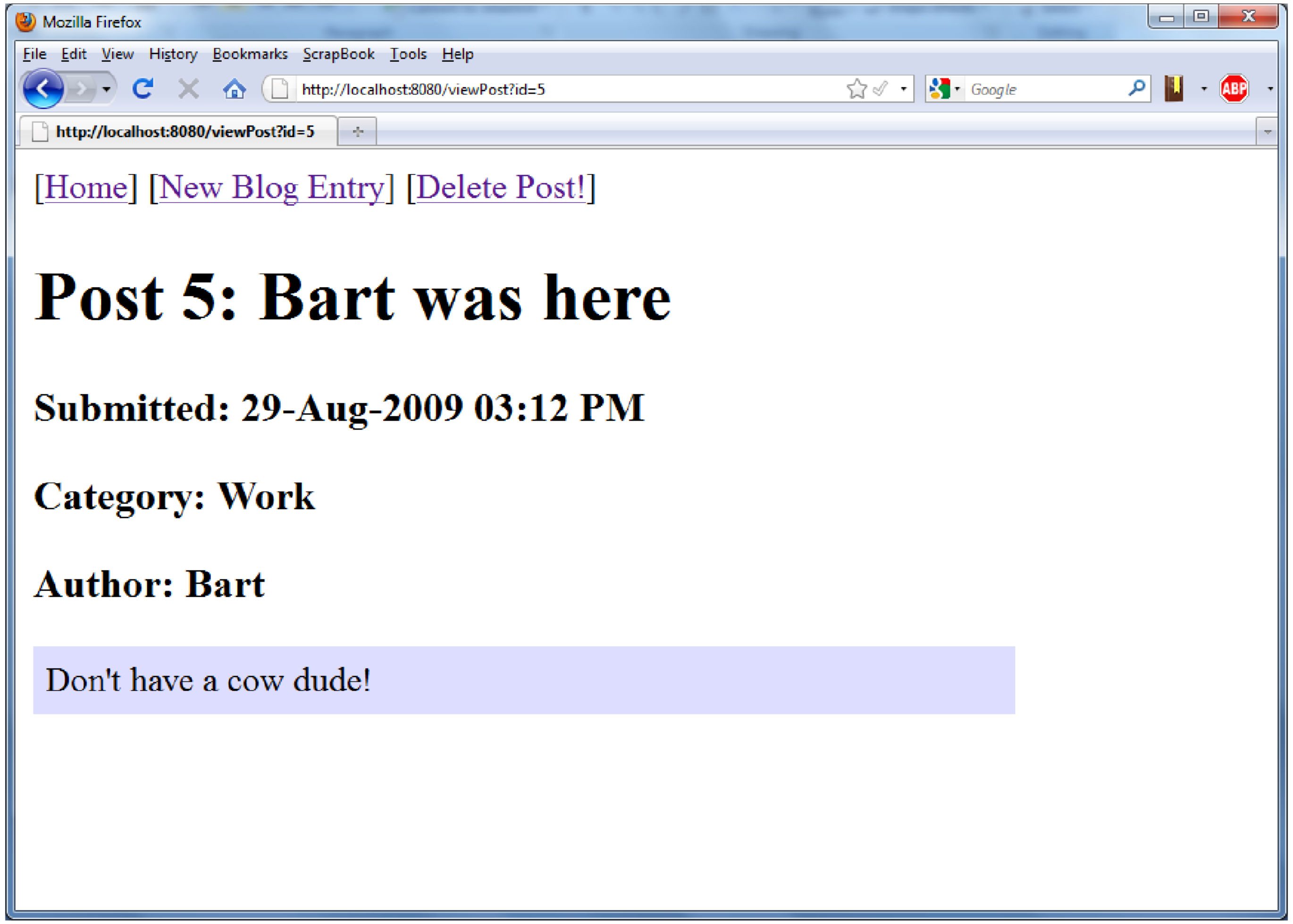
http://localhost:8080/postForm

[Home] [New Blog Entry]

New Post

Title	Bart was here
Author	Bart
Category	Work
Content	Don't have a cow dude!

Case Study


Mozilla Firefox
File Edit View History Bookmarks ScrapBook Tools Help
http://localhost:8080/viewPost?id=5
Google
http://localhost:8080/viewPost?id=5

[Home] [New Blog Entry] [Delete Post!]

Post 5: Bart was here

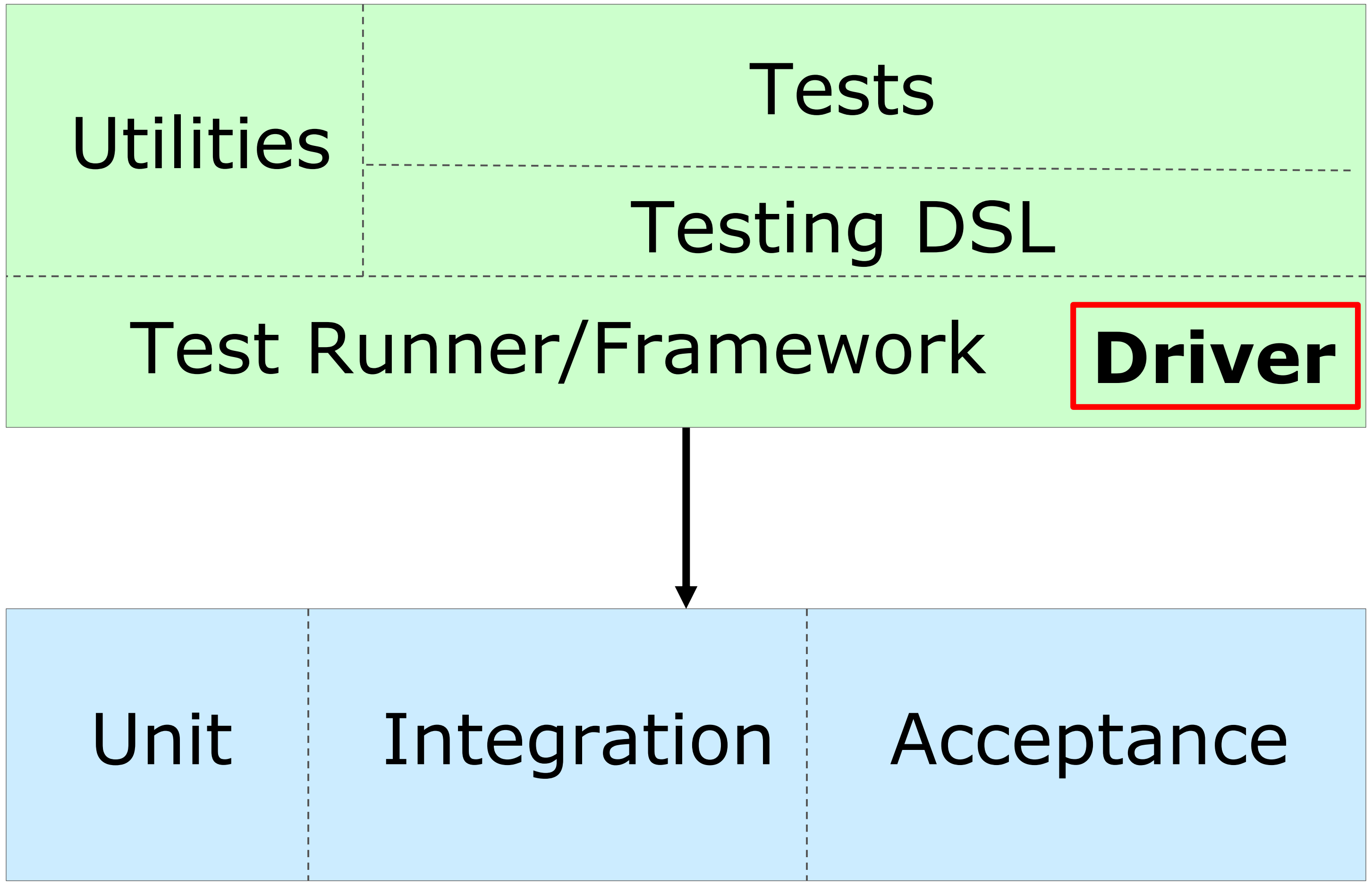
Submitted: 29-Aug-2009 03:12 PM

Category: Work

Author: Bart

Don't have a cow dude!

Looking at web drivers



Web testing drivers

None
Regex
XmlSlurper
Cyberneko
JSoup
HttpBuilder
HttpBuilderNG

JMeter
Ersatz

WebTest
HtmlUnit
Geb
WebDriver
Selenium
JWebUnit
Arquillian

Cucumber
JBehave
Serenity
RobotFramework
Concordion
EasyB
Tumbler
FitNesse/Slim

Case Study

```
def html = new URL('http://localhost:8080').text

assert html.contains('<title>Welcome to SimpBlog</title>')

html.find(~'<title>(.*)</title>') { all, title ->
    assert title == 'Welcome to SimpBlog'
}
```

Case Study

```
def page = new XmlSlurper().parse('http://localhost:8080/viewPost?id=1')
assert page.body.h1.text().contains('Christmas')
assert page.body.h3[1].text() == 'Category: Home'
assert page.body.h3[2].text() == 'Author: Bart'
assert page.body.table.tr.td.p.text() ==
        "Aren't we forgeting the true meaning of this day? You know,
the birth of Santa."
```

Case Study

```groovy
@Grab('net.sourceforge.nekohtml:nekohtml:1.9.22')
import org.cyberneko.html.parsers.SAXParser

def parser = new XmlSlurper(new SAXParser())
def page = parser.parse('http://localhost:8080/viewPost?id=1')
assert page.BODY.H1.text().contains('Christmas')
assert page.BODY.H3[1].text() == 'Category: Home'
assert page.BODY.H3[2].text() == 'Author: Bart'
assert page.BODY.TABLE.TBODY.TR.TD.P.text() ==
        "Aren't we forgeting the true meaning of this day? You
know, the birth of Santa."
```

Case Study

```
@Grab(group='org.codehaus.groovy.modules.http-builder',
    module='http-builder', version='0.5.0-RC1')
import groovyx.net.http.*
import static groovyx.net.http.ContentType.URLENC

def http = new HTTPBuilder('http://localhost:8080')
def postBody = [title:'Bart was here (and so was HttpBuilder)',
        content:'Cowabunga Dude!', author:'1', category:'3']
http.post(path:'/addPost', body: postBody,
        requestContentType: URLENC) { resp, html ->
    assert resp.contentType == 'text/html'
    assert resp.status == 200
    assert html.BODY.H1.text().matches('Post.*: Bart was here.*')
    assert html.BODY.H3[1].text() == 'Category: Home'
    assert html.BODY.H3[2].text() == 'Author: Bart'
    assert html.BODY.TABLE.TR.TD.P.text() == 'Cowabunga Dude!'
}
```

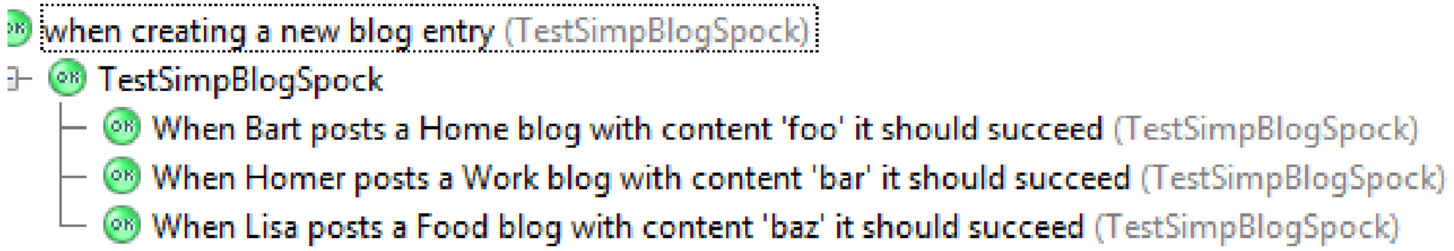
HtmlUnit

```
class TestSimpBlogJUnit4 {
    def page

    @Before
    void setUp() {
        page = new WebClient().getPage('http://localhost:8080/postForm')
        assert 'Welcome to SimpBlog' == page.titleText
    }

    @Test
    void bartWasHere() {
        // fill in blog entry and post it
        def form = page.getFormByName('post')
        form.getInputByName('title').setValueAttribute('Bart was here (HtmlUnit JUnit4)')
        form.getSelectByName('category').getOptions().find {
            it.text == 'School' }.setSelected(true)
        form.getTextAreaByName('content').setText('Cowabunga Dude!')
        def result = form.getInputByName('btnPost').click()

        // check blog post details
        assert result.getElementsByTagName('h1').item(0).
                textContent.matches('Post.*: Bart was here.*')
        def h3headings = result.getElementsByTagName('h3')
        assert h3headings.item(1).textContent == 'Category: School'
        assert h3headings.item(2).textContent == 'Author: Bart'

        // expecting: <table><tr><td><p>Cowabunga Dude!</p>...</table>
        def cell = result.getByXPath('//TABLE//TR/TD')[0]
        assert cell.textContent.trim() == 'Cowabunga Dude!'
    }
}
```

Case Study

```
...
  then:
    page.titleText == 'Welcome to SimpBlog'
    result.getElementsByTagName('h1').item(0).textContent.matches("Post.*: $aut
    subheadings.item(1).textContent == "Category: $category"
    subheadings.item(2).textContent == "Author: $author"

  and:
    para.textContent == content

  where:
    author   << ['Bart', 'Homer', 'Lisa']
    category << ['Home', 'Work', 'Food']
    content  << ['foo', 'bar', 'baz']
  }
}
```

// Optional use of 'and:'



```
when creating a new blog entry (TestSimpBlogSpock)
TestSimpBlogSpock
  When Bart posts a Home blog with content 'foo' it should succeed (TestSimpBlogSpock)
  When Homer posts a Work blog with content 'bar' it should succeed (TestSimpBlogSpock)
  When Lisa posts a Food blog with content 'baz' it should succeed (TestSimpBlogSpock)
```

Geb

```
@Grab("org.gebish:geb-core:1.1.1")
@Grab("org.seleniumhq.selenium:selenium-chrome-driver:3.4.0")
@Grab("org.seleniumhq.selenium:selenium-support:3.4.0")
import geb.Browser

Browser.drive {
    go 'http://localhost:8080/postForm'
    assert title == 'Welcome to SimpBlog'

    $("form").with {
        title = "Bart was here (Geb)"
        author = 'Bart'
        category = 'School'
        content = "Cowabunga Dude!"
        btnPost().click()
    }
    assert $("h1").text().matches("Post \\d+: Bart was here.*")
    assert $("h3")[1].text() == 'Category: School'
    assert $("h3")[2].text() == 'Author: Bart'
    assert $("p").text() == "Cowabunga Dude!"
}
```

Geb with pages

```
class NewPostPage extends Page {
    static url = "http://localhost:8080/postForm"
    static at = { title == 'Welcome to SimpBlog' }
    static content = {
        blogTitle { $("form").title() } // !title
        blogger { $("form").author() }
        label { $("form").category() }
        blogText { $("form").content() } // !content
        post(to: ViewPostPage) { btnPost() }
    }
}

class ViewPostPage extends Page {
    static at = { $("h1").text().contains('Post') }
    static content = {
        mainHeading { $("h1").text() }
        categoryHeading { $("h3")[1].text() }
        authorHeading { $("h3")[2].text() }
        blogText { $("p").text() }
    }
}
```

Geb with pages

```
class NewPostPage extends Page {
    static url = "http://localhost:8080/postForm"
    static at = { title == 'Welcome to SimpBlog' }
    static content = {





    }
}

class
    st
    st




    }
}
```

```
Browser.drive {
    to NewPostPage

    assert at(NewPostPage)
    blogTitle.value 'Bart was here (Geb pages)'
    blogger.value 'Bart'
    label.value 'School'
    blogText.value 'Cowabunga Dude!'
    post.click()

    assert at(ViewPostPage)
    assert mainHeading ==~ "Post \\d+: Bart was here.*"
    assert categoryHeading == 'Category: School'
    assert authorHeading == 'Author: Bart'
    assert blogText == "Cowabunga Dude!"
}
```

Case Study

```
ant.webtest(name: 'Test SimpBlog') {
  invoke url: "http://localhost:8080/",
      description: "Home Page"
  verifyTitle text: "Welcome to SimpBlog"
  group description: "Post New Blog Entry", {
    clickLink label: "New Blog Entry"
    setInputField name: "title",
        value: "Bart was here (and so was WebTest with Groovy)"
    setSelectField name: "category", text: "School"
    setInputField name: "content", value: "Cowabunga Dude!"
    clickButton name: "btnPost"
  }
  group description: "Check Blog Post", {
    verifyElementText type: "h1", regex: "true",
        text: "Post.*: Bart was here.*"
    verifyXPath xpath: "//h3[2]/text()", text: "Category: School"
    verifyXPath xpath: "//h3[3]/text()", text: "Author: Bart"
    verifyElementText type: "p", text: "Cowabunga Dude!"
  }
}
```

Case Study: Cucumber

```
//…

Given(~/^we are on the create blog entry page$/) { ->
    tester = new BlogTester('http://localhost:8080/postForm')
    tester.checkTitle 'Welcome to SimpBlog'
}

When(~/^I have entered '([^']*)' as the title$/) { String title ->
    formFields.title = title
}

When(~/^I have entered '([^']*)' into the content$/) { String content ->
    formFields.content = content
}

When(~/^I have selected '([^']*)' as the category$/) { String category ->
    formFields.category = category
}

When(~/^I click the 'Create Post' button$/) { ->
    tester.postBlog(formFields)
}

Then(~/^I expect the entry to be posted$/) { ->
    tester.checkHeadingMatches formFields.title
    tester.checkSubheading 'Category', formFields.category
    tester.checkPostText formFields.content
    tester.checkSubheading 'Author', formFields.author
}
```

Case Study: Cucumber

```
//…

Given(~/^we are on the create blog entry page$/) { ->
    tester = new BlogTester('http://localhost:8080/postForm')
    tester.checkTitle 'Welcome to SimpBlog'
}

When(~/^I have entered '([^']*)' as the title$/) { String title ->
    formFields.title = title
}

ng content ->

ng category ->

    tester.checkSubheading 'Category', formFields.category
    tester.checkPostText formFields.content
    tester.checkSubheading 'Author', formFields.author
}
```

```
Feature: Use the blogging system

  Scenario: Bart posts a new blog entry
    Given we are on the create blog entry page
    When I have entered 'Bart was here (Cuke Groovy)' as the title
      And I have entered 'Cowabunga Dude!' into the content
      And I have selected 'Home' as the category
      And I have selected 'Bart' as the author
      And I click the 'Create Post' button
    Then I expect the entry to be posted
```

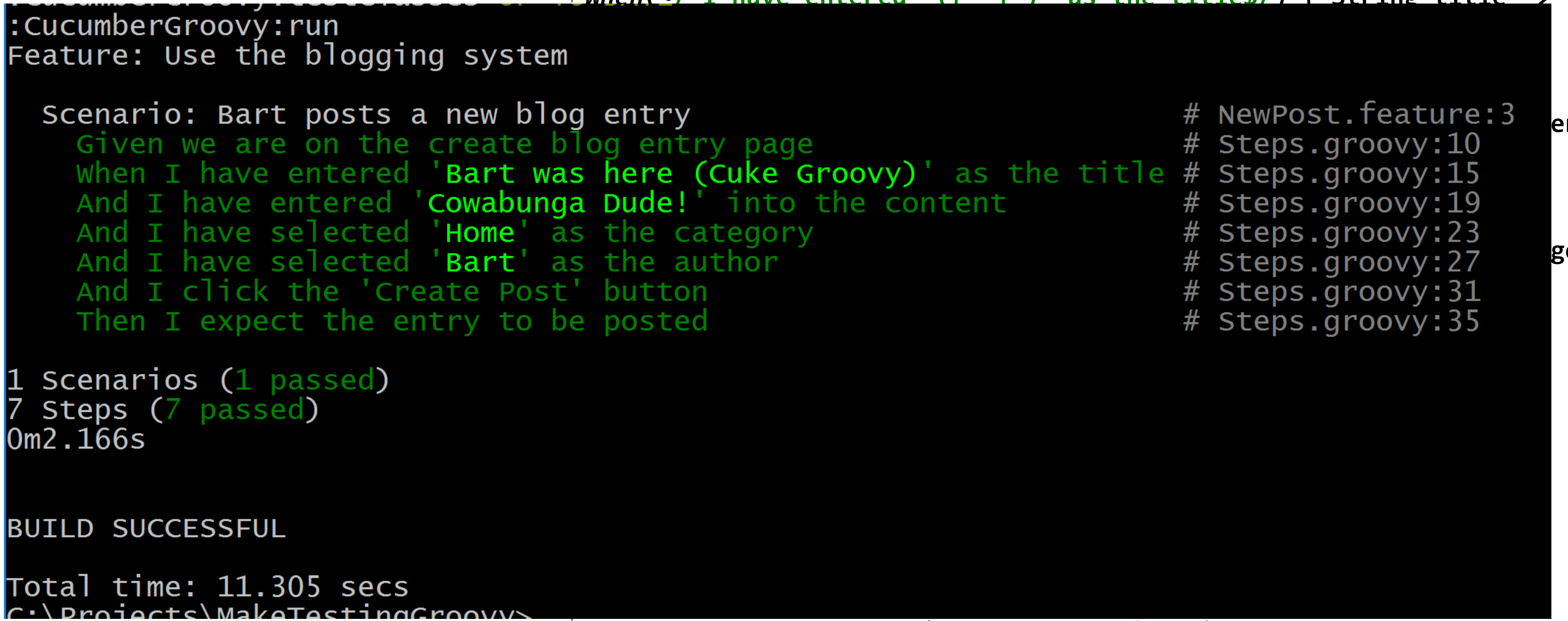
Case Study: Cucumber

```
//…

Given(~/^we are on the create blog entry page$/) { ->
    tester = new BlogTester('http://localhost:8080/postForm')
    tester.checkTitle 'Welcome to SimpBlog'
}

When(~/^I have entered '([^']*)' as the title$/) { String title ->
```

```
ent ->
```

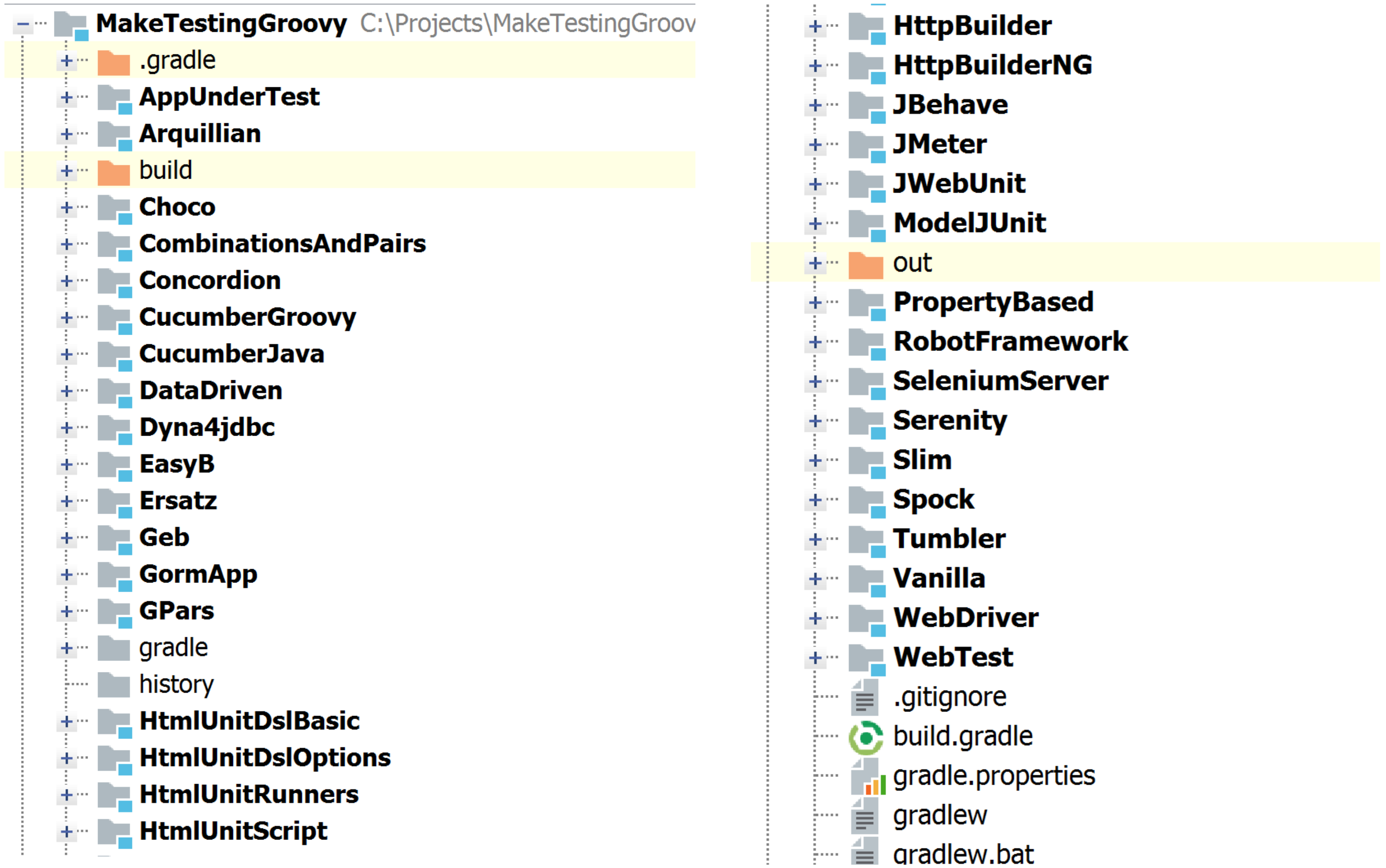
```
gory ->
```

```
    tester.checkSubheading 'Author', formFields.author
}
```



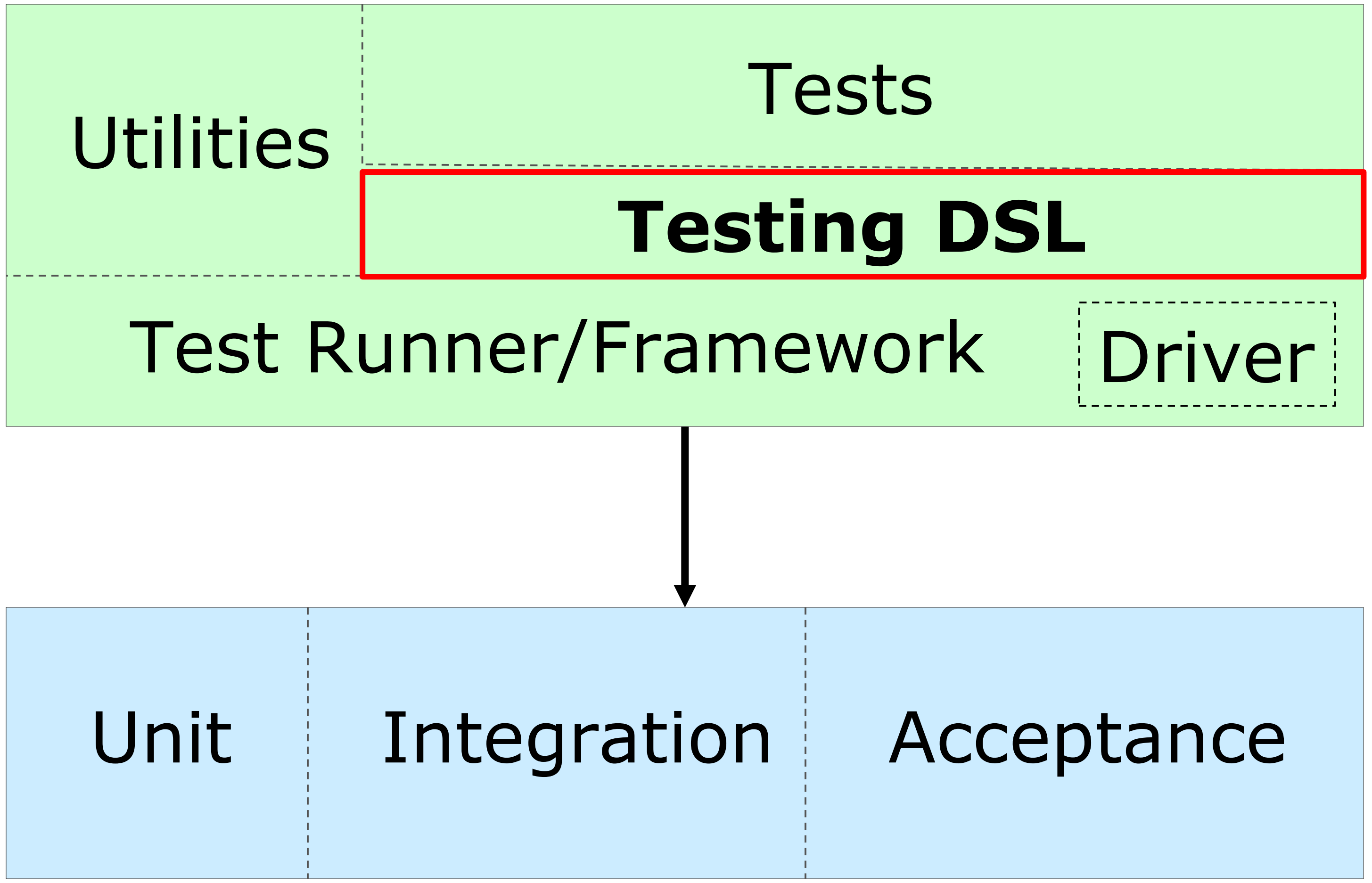
```
:CucumberGroovy:run
Feature: Use the blogging system

  Scenario: Bart posts a new blog entry                                # NewPost.feature:3
    Given we are on the create blog entry page                         # Steps.groovy:10
    When I have entered 'Bart was here (Cuke Groovy)' as the title     # Steps.groovy:15
    And I have entered 'Cowabunga Dude!' into the content              # Steps.groovy:19
    And I have selected 'Home' as the category                         # Steps.groovy:23
    And I have selected 'Bart' as the author                           # Steps.groovy:27
    And I click the 'Create Post' button                               # Steps.groovy:31
    Then I expect the entry to be posted                               # Steps.groovy:35

1 Scenarios (1 passed)
7 Steps (7 passed)
0m2.166s


BUILD SUCCESSFUL

Total time: 11.305 secs
C:\Projects\MakeTestingGroovy>
```

Other Web Drivers



MakeTestingGroovy C:\Projects\MakeTestingGroov
.gradle
AppUnderTest
Arquillian
build
Choco
CombinationsAndPairs
Concordion
CucumberGroovy
CucumberJava
DataDriven
Dyna4jdbc
EasyB
Ersatz
Geb
GormApp
GPars
gradle
history
HtmlUnitDslBasic
HtmlUnitDslOptions
HtmlUnitRunners
HtmlUnitScript
HttpBuilder
HttpBuilderNG
JBehave
JMeter
JWebUnit
ModelJUnit
out
PropertyBased
RobotFramework
SeleniumServer
Serenity
Slim
Spock
Tumbler
Vanilla
WebDriver
WebTest
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat

Looking at testing utilities



Testing DSLs

Low-level “DSL/fluent API”

```
def form = page.getFormByName('post')
form.getInputByName('title').setValueAttribute('Bart was here (HtmlUnit JUnit4)')
form.getSelectByName('category').getOptions().find {
        it.text == 'School' }.setSelected(true)
```

Medium-level DSL

```
clickLink label: "New Blog Entry"
setInputField name: "title",
    value: "Bart was here (and so was WebTest with Groovy)"
setSelectField name: "category", text: "School"
setInputField name: "content", value: "Cowabunga Dude!"
clickButton name: "btnPost"
```

Higher-level DSL

```
post blog from Bart with title "Bart rulz!" and category School and content "Cowabunga Dude!"
```

DSLs: Fluent API

```
class BlogTestCase extends GroovyTestCase {
    def page
    def lastResult

    void setUp() {
        page = new WebClient().getPage('http://localhost:8080/postForm')
    }

    def checkTitle(String title) {
        assert title == page.titleText
    }

    def prepareBlog() {
        new PrepareBlogEmpty()
    }

    // ...
}
```

DSLs: Fluent API

```
    // ...
    def postBlog(Map params) {
        def form = page.getFormByName('post')
        form.getInputByName('title').setValueAttribute(params.title)
        form.getSelectByName('category').options.find {
            it.text == params.category
        }.setSelected(true)
        form.getSelectByName('author').options.find {
            it.text == params.author
        }.setSelected(true)
        form.getTextAreaByName('content').setText(params.content)
        lastResult = form.getInputByName('btnPost').click()
    }

    def checkHeadingMatches(String regex) {
        def heading = lastResult.getElementsByTagName('h1').item(0)
        assert heading.textContent.matches(regex)
    }
    // ...
}
```

DSLs: Fluent API

```
class TestSimpBlogFluentApi extends BlogTestCase {

    void setUp() {
        super.setUp()
        checkTitle('Welcome to SimpBlog')
    }

    void testBartWasHere() {
        prepareBlog()
            .withTitle('Bart was here (HtmlUnit FluentApi)')
            .withAuthor('Bart')
            .withCategory('School')
            .withContent('Cowabunga Dude!')
            .post()

        checkHeadingMatches 'Post.*: Bart was here.*'
        checkSubheading 1, 'Category: School'
        checkSubheading 2, 'Author: Bart'
        checkPostText 'Cowabunga Dude!'
    }
```

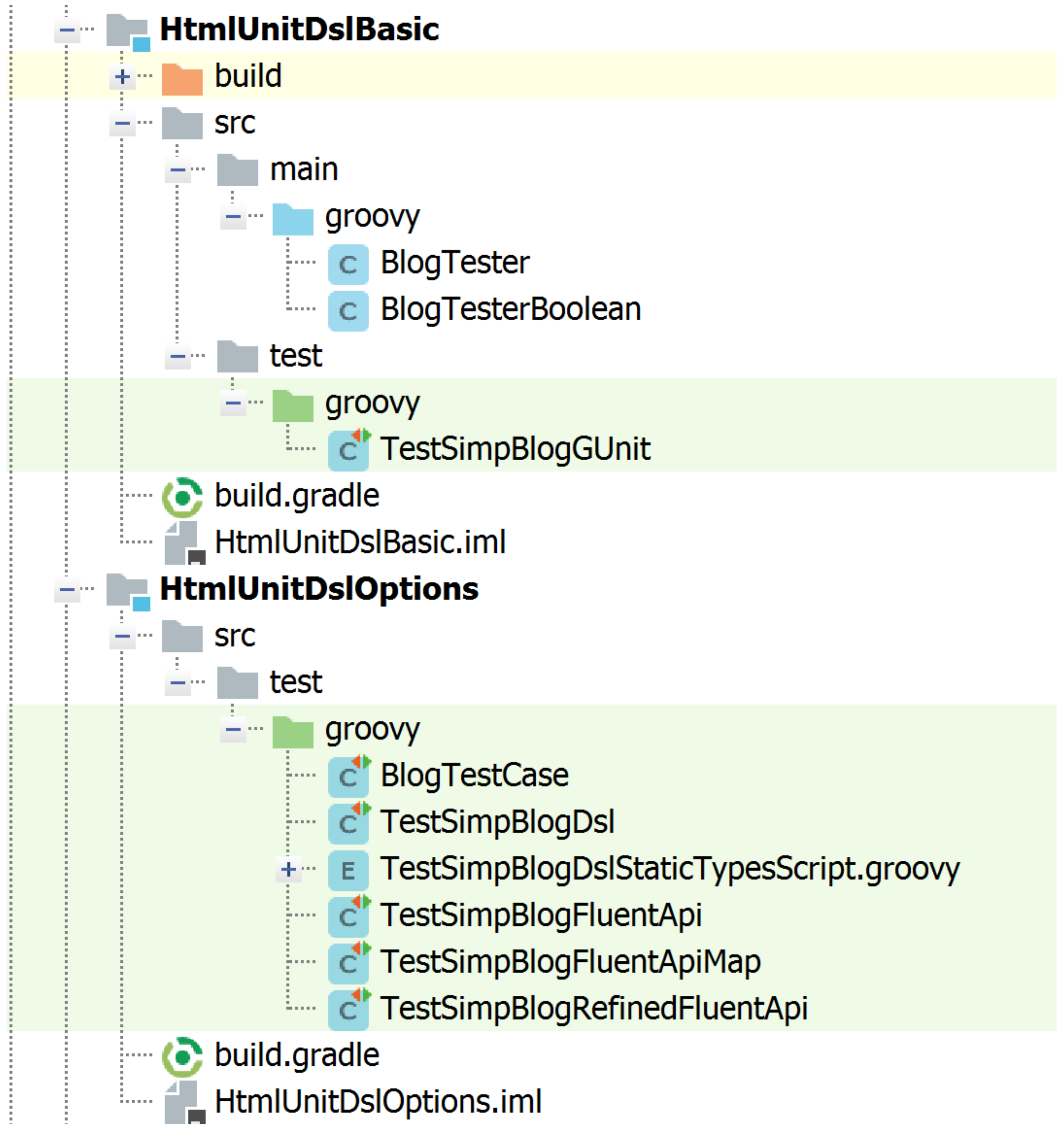
DSLs: command chains

```
class TestSimpBlogDsl extends BlogTestCase {

    private the, has, a, with, heading
    void setUp() {
        super.setUp()
    }

    def post(_a) {
        [
                blog: { _with ->
                    [title: { postTitle ->
                        [and: { __with ->
                            [author: { postAuthor ->
                                [and: { ___with ->
                                    [category: { postCategory ->
                                        [and: { ____with ->
                                            [content: { postContent ->
                                                postBlog(title: postTitle,
                                                        author:postAuthor,
                                                        content:postContent,
                                                        category: postCategory)
                }   ]}  ]}  ]}  ]}  ]}  ]}  ]}
        ]
    }

    def check(_the) {
        [
                browser: { _has -> [title: { checkTitle it }]},
                main: { _heading -> [matches: { checkHeadingMatches it }]},
                category: { _has -> [value: { checkSubheading 1, "Category: $it" }]},
                author: { _has -> [value: { checkSubheading 2, "Author: $it" }]},
                blog: { _has -> [text: { checkPostText it }]}
        ]
    }
    // ...
```

DSLs: command chains

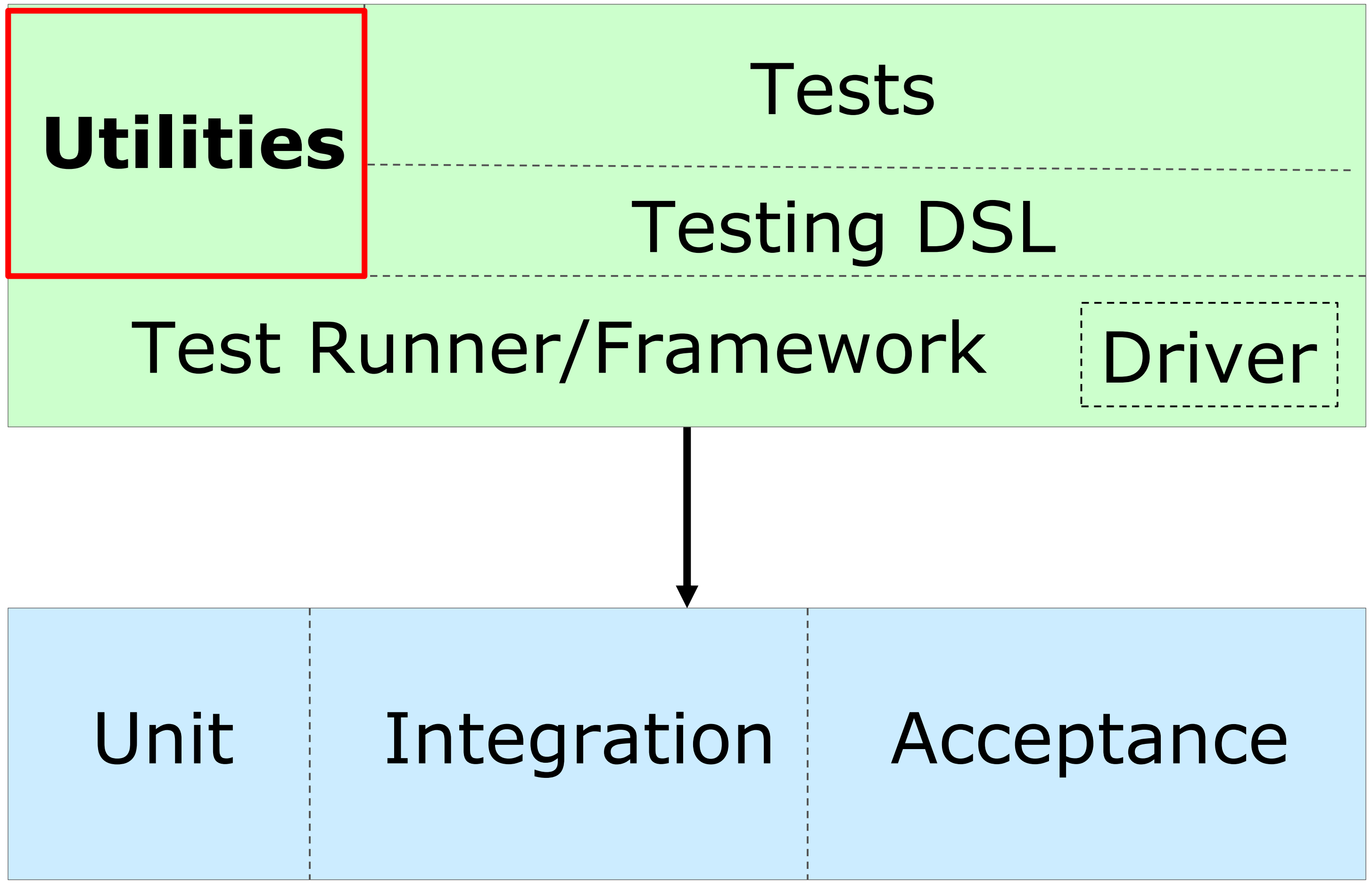
```
    // ...

    void testBartWasHere() {
        check the browser has title 'Welcome to SimpBlog'
        post a blog with title 'Bart was here (HtmlUnit DSL)' \
            and with author 'Bart' \
            and with category 'School' \
            and with content 'Cowabunga Dude!'
        check the main heading matches 'Post.*: Bart was here.*'
        check the category has value 'School'
        check the author has value 'Bart'
        check the blog has text 'Cowabunga Dude!'
    }
}
```

Testing DSLs



```
HtmlUnitDslBasic
  build
  src
    main
      groovy
        BlogTester
        BlogTesterBoolean
    test
      groovy
        TestSimpBlogGUnit
  build.gradle
  HtmlUnitDslBasic.iml
HtmlUnitDslOptions
  src
    test
      groovy
        BlogTestCase
        TestSimpBlogDsl
        TestSimpBlogDslStaticTypesScript.groovy
        TestSimpBlogFluentApi
        TestSimpBlogFluentApiMap
        TestSimpBlogRefinedFluentApi
  build.gradle
  HtmlUnitDslOptions.iml
```

Testing Utilities



Testing Utilities

All combinations

All pairs

Property-based testing

GPars

Constraint programming

ModelJUnit

All Combinations

- **Description**
  - Don't have a bunch of hard-coded, hard to maintain manual test data or even manually generated CSV file
  - Much better to generate test cases from succinct expressions of what you are trying to achieve

```
test('MacOS', '4G', '250G')
test('Linux', '4G', '250G')
test('Vista', '4G', '250G')
test('MacOS', '8G', '500G')
test('Linux', '8G', '500G')
test('Vista', '8G', '500G')
// 30 more rows
```

```
[
  ['MacOS', 'Linux', 'Vista'],
  ['2G', '4G', '6G', '8G'],
  ['250G', '350G', '500G']
].combinations().each{
  os, mem, disk ->
  test(os, mem, disk)
}
```

All Combinations Case Study

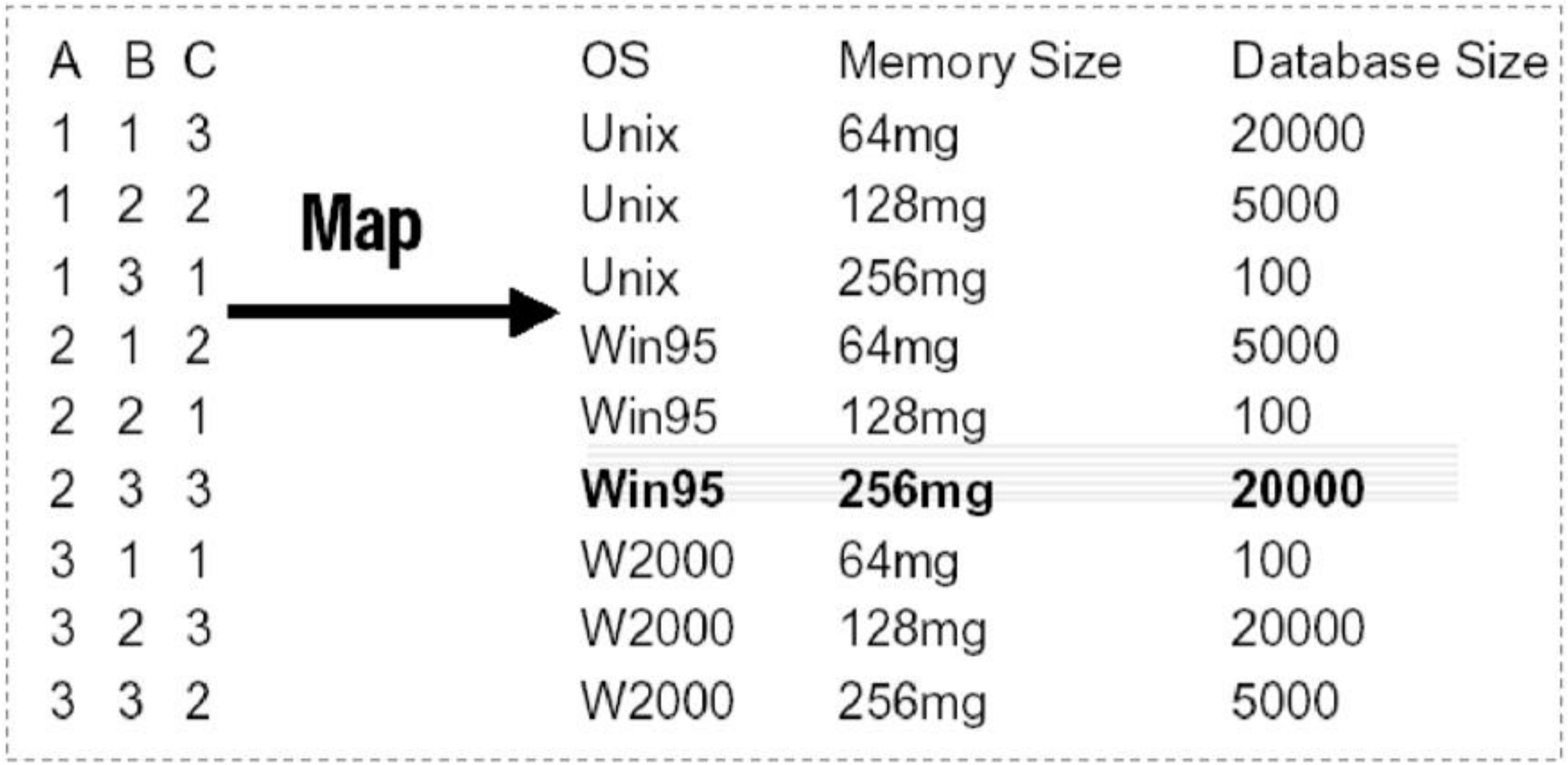
```
import com.gargoylesoftware.htmlunit.WebClient

def combos = [["Bart", "Homer", "Marge", "Lisa", "Maggie"],
              ["Work", "School", "Home", "Travel", "Food"],
              ["foo", "bar", "baz"]].combinations()
println "Found ${combos.size()} combos"
combos.each { author, category, content ->
    postAndCheck category, author, content
}

def postAndCheck(String category, String author, String content) {
    // ...
    // details not shown (ran with HtmlUnit)
    // ...
}
```

```
Found 75 combos
```

All Pairs

AKA:

- Pairwise testing
- Orthogonal array testing



A	B	C
1	1	3
1	2	2
1	3	1
2	1	2
2	2	1
2	3	3
3	1	1
3	2	3
3	3	2

Map $\longrightarrow$

OS	Memory Size	Database Size
Unix	64mg	20000
Unix	128mg	5000
Unix	256mg	100
Win95	64mg	5000
Win95	128mg	100
Win95	**256mg**	**20000**
W2000	64mg	100
W2000	128mg	20000
W2000	256mg	5000

Technique to limit the explosion of test cases

- Identify equivalence classes
- Many faults result from adverse two-way interactions

All Pairs motivation

```
def myAdder(String one, String two) {
    if (!one.isInteger()) one = '0'
    else if (!two.isInteger()) two = '0'
    one.toInteger() + two.toInteger()
}

assert myAdder('40', '2') == 42
assert myAdder('40', '') == 40
assert myAdder('', '2') == 2
```

☑

All Pairs motivation

```
def myAdder(String one, String two) {
    if (!one.isInteger()) one = '0'
    else if (!two.isInteger()) two = '0'
    one.toInteger() + two.toInteger()
}

assert myAdder('40', '2') == 42
assert myAdder('40', '') == 40
assert myAdder('', '2') == 2
assert myAdder('', '') == 0             ✗
```

All Pairs motivation

```
def myAdder(String one, String two) {
    if (!one.isInteger()) one = '0'
    else if (!two.isInteger()) two = '0'
    one.toInteger() + two.toInteger()
}

assert myAdder('40', '2') == 42
assert myAdder('40', '') == 40
assert myAdder('', '2') == 2
assert myAdder('', '') == 0            ✗
```

All Pairs motivation

```
def myAdder(String one, String two) {
    if (!one.isInteger()) one = '0'
    if (!two.isInteger()) two = '0'
    one.toInteger() + two.toInteger()
}

assert myAdder('40', '2') == 42
assert myAdder('40', '') == 40
assert myAdder('', '2') == 2
assert myAdder('', '') == 0
```

☑

All Pairs Case Study

```
Found 18 pairs
[content:bar, category:Food, author:Bart]
[content:bar, category:School, author:Homer]
[content:foo, category:Work, author:Bart]
[content:baz, category:School, author:Homer]
[content:bar, category:Home, author:Maggie]
[content:foo, category:School, author:Marge]
[content:bar, category:Work, author:Bart]
[content:baz, category:Travel, author:Bart]
[content:foo, category:Home, author:Homer]
[content:bar, category:Travel, author:Marge]
[content:baz, category:Work, author:Homer]
[content:bar, category:Travel, author:Lisa]
[content:baz, category:Travel, author:Maggie]
[content:baz, category:Home, author:Marge]
[content:baz, category:Food, author:Homer]
[content:baz, category:Travel, author:Lisa]
[content:foo, category:Food, author:Maggie]
[content:foo, category:Travel, author:Lisa]
```

Property-based testing

Agile testing game (TDD)

- Minimum test code to steer design of minimal production code with desired business functionality but 100% code coverage
- “Grey box” testing
- Rarely used with functional programming

Property-based testing

Agile testing game (TDD)

- Minimum test code to steer design of minimal production code with desired business functionality but 100% code coverage
- “Grey box” testing
- Rarely used with functional programming
- Instead validate certain properties

```
for (words in someNonEmptyLists(strings())) {
    assert words*.size().sum() == words.sum().size()
}
```

Property-based testing

```
@Grab('net.java.quickcheck:quickcheck:0.6')
import static
net.java.quickcheck.generator.PrimitiveGenerators.*
import static java.lang.Math.round
import static Converter.celsius

def gen = integers(-40, 240)
def liquidC =  0..100
def liquidF = 32..212
100.times {
  int f = gen.next()
  int c = round(celsius(f))
  assert c <= f
  assert c in liquidC == f in liquidF
}
```

Property-based testing with Spock

```
@Unroll
def 'test reverse #string'() {
    when:
        def reversed = string.reverse()

    then:
        reversed.size() == string.size()
        if (string) {
            string.eachWithIndex { letter, i ->
                letter == reversed[-(i + 1)]
            }
        }
        reversed.reverse() == string

    where:
        string << Gen.these('', 'foo').then(Gen.string).take(10000)
}
```

```
Gen.type(Person,
    id: Gen.integer(200..10000),
    name: Gen.string(~/[A-Z][a-z]+( [A-Z][a-z]+)?/),
    birthDate: Gen.date(Date.parse('MM/dd/yyyy','01/01/1940'), new Date()),
    title: Gen.these('', null).then(Gen.any('Dr.', 'Mr.', 'Ms.', 'Mrs.')),
    gender: Gen.character('MFTU'))
```

https://github.com/Bijnagte/spock-genesis

Property-based testing: spock genesis

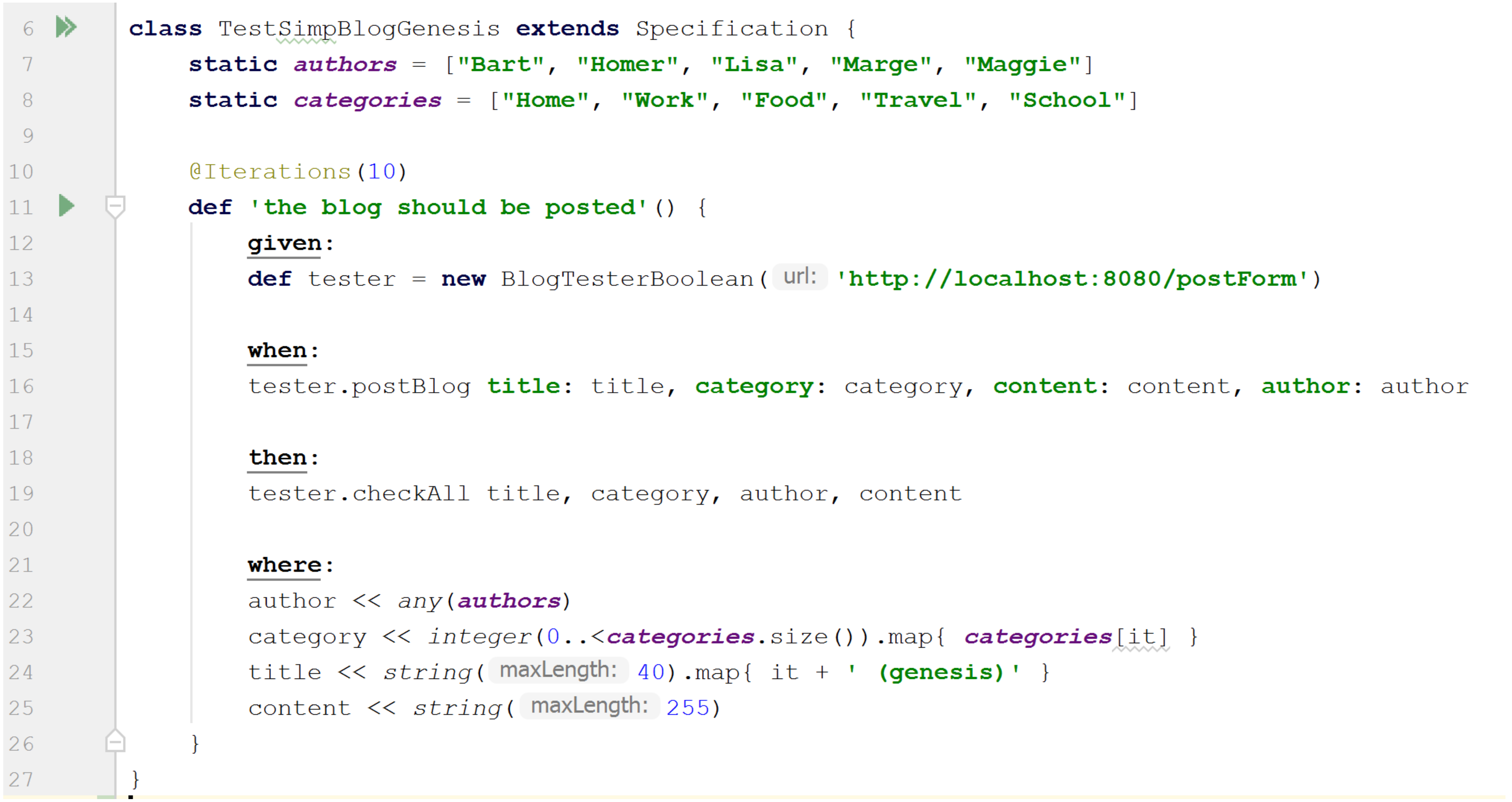
```groovy
@Grab('com.nagternal:spock-genesis:0.6.0')
@GrabExclude('org.codehaus.groovy:groovy-all')
import spock.genesis.transform.Iterations
import spock.lang.Specification
import static Converter.celsius
import static java.lang.Math.round
import static spock.genesis.Gen.integer

class ConverterSpec extends Specification {
    def liquidC = 0..100
    def liquidF = 32..212

    @Iterations(100)
    def "test phase maintained"() {
        given:
        int tempF = integer(-40..240).iterator().next()

        when:
        int tempC = round(celsius(tempF))

        then:
        tempC <= tempF
        tempC in liquidC == tempF in liquidF
    }
…
```

Property-based testing: spock genesis

```
@Grab('com.nagternal:spock-genesis:0.6.0')
@GrabExclude('org.codehaus.groovy:groovy-all')
import spock.genesis.transform.Iterations
import spock.lang.Specification
import static Converter.celsius
import static java.lang.Math.round
import static spock.genesis.Gen.integer

class ConverterSpec extends Specification {
    def liquidC =
    def liquidF =

    @Iterations(1
    def "test pha
        given:
        int tempF

        when:
        int tempC

        then:
        tempC <=
        tempC in
    }
…
```

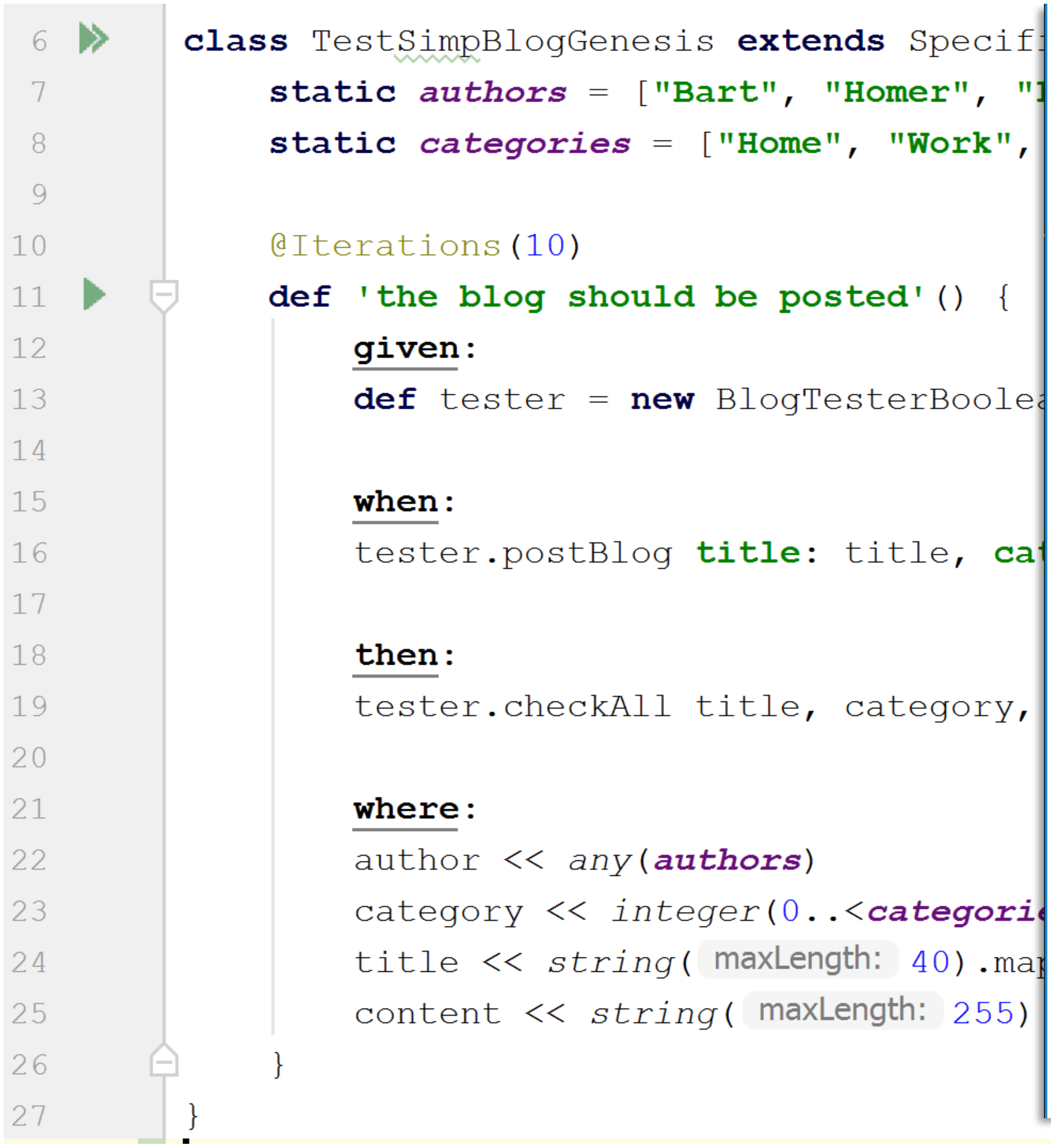
```
…

    @Iterations(100)
    def "test order maintained"() {
        given:
        int tempF1 = integer(-273..999).iterator().next()
        int tempF2 = integer(-273..999).iterator().next()

        when:
        int tempC1 = round(celsius(tempF1))
        int tempC2 = round(celsius(tempF2))

        then:
        (tempF1 <=> tempF2) == (tempC1 <=> tempC2)
    }
}
```

Property-based testing: Case Study

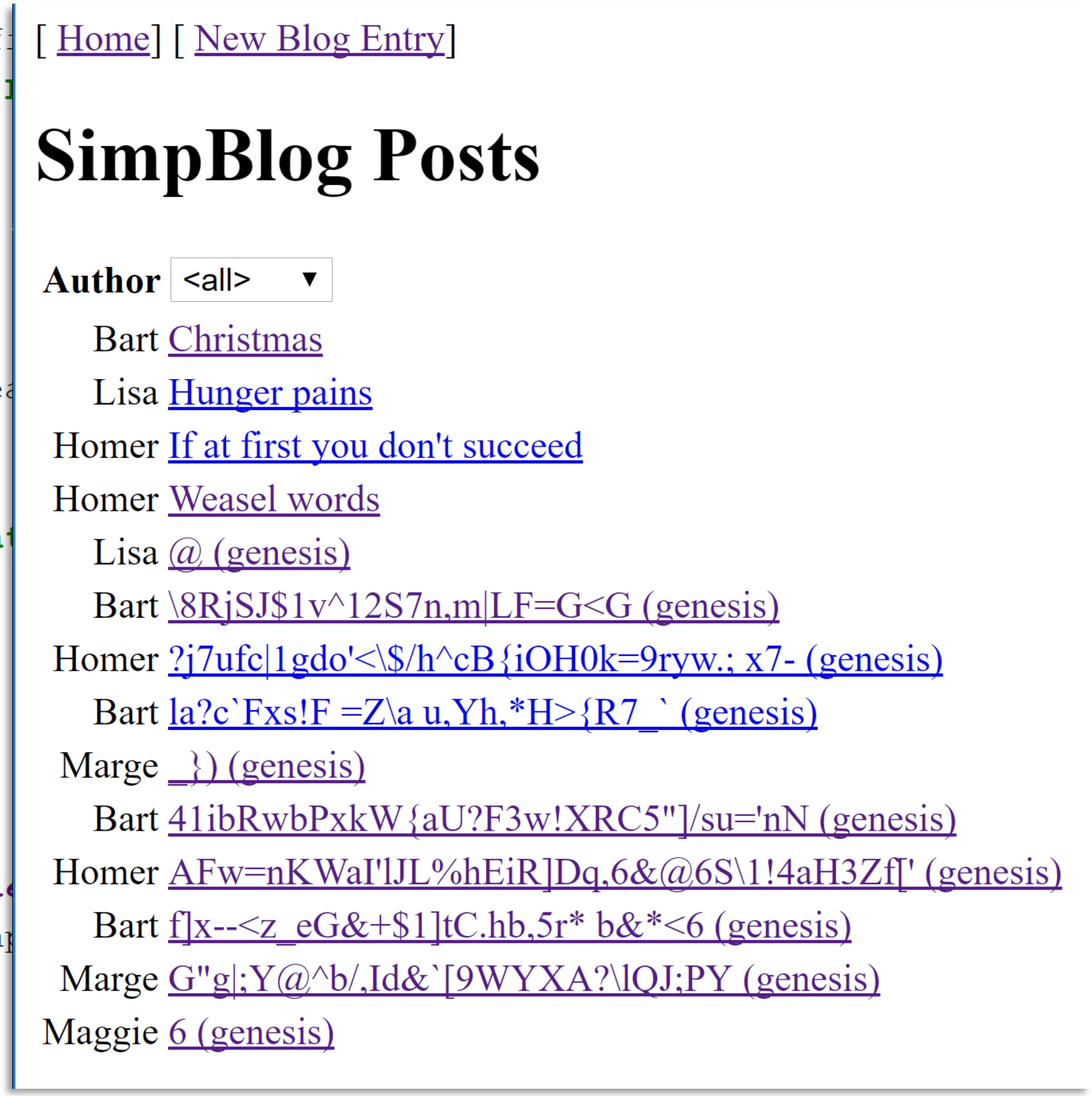


```
class TestSimpBlogGenesis extends Specification {
    static authors = ["Bart", "Homer", "Lisa", "Marge", "Maggie"]
    static categories = ["Home", "Work", "Food", "Travel", "School"]

    @Iterations(10)
    def 'the blog should be posted'() {
        given:
        def tester = new BlogTesterBoolean( url: 'http://localhost:8080/postForm')

        when:
        tester.postBlog title: title, category: category, content: content, author: author

        then:
        tester.checkAll title, category, author, content

        where:
        author << any(authors)
        category << integer(0..<categories.size()).map{ categories[it] }
        title << string( maxLength: 40).map{ it + ' (genesis)' }
        content << string( maxLength: 255)
    }
}
```

Property-based testing: Case Study



```
class TestSimpBlogGenesis extends Specifi
    static authors = ["Bart", "Homer", "L
    static categories = ["Home", "Work",

    @Iterations(10)
    def 'the blog should be posted'() {
        given:
        def tester = new BlogTesterBoolea

        when:
        tester.postBlog title: title, cat

        then:
        tester.checkAll title, category,

        where:
        author << any(authors)
        category << integer(0..<categorie
        title << string( maxLength: 40).ma
        content << string( maxLength: 255)
    }
}
```



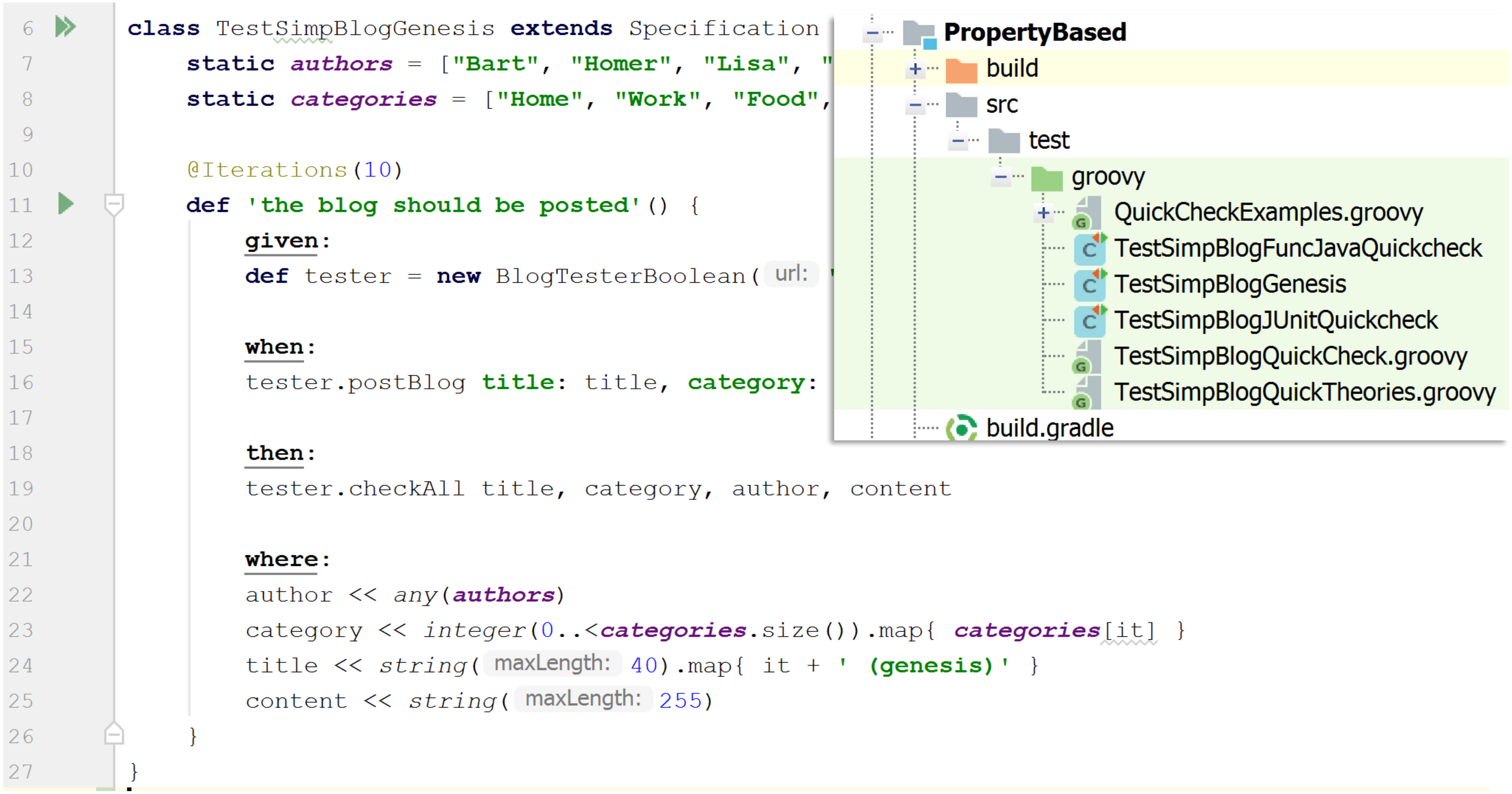
[Home] [New Blog Entry]

SimpBlog Posts

Author <all> ▼

Bart Christmas
Lisa Hunger pains
Homer If at first you don't succeed
Homer Weasel words
Lisa @ (genesis)
Bart \8RjSJ$1v^12S7n,m|LF=G<G (genesis)
Homer ?j7ufc|1gdo'<\$/h^cB{iOH0k=9ryw.; x7- (genesis)
Bart la?c`Fxs!F =Z\a u,Yh,*H>{R7_` (genesis)
Marge _}) (genesis)
Bart 41ibRwbPxkW{aU?F3w!XRC5"]/su='nN (genesis)
Homer AFw=nKWaI'lJL%hEiR]Dq,6&@6S\1!4aH3Zf[' (genesis)
Bart f]x--<z_eG&+$1]tC.hb,5r* b&*<6 (genesis)
Marge G"g|;Y@^b/,Id&`[9WYXA?\lQJ;PY (genesis)
Maggie 6 (genesis)

Property-based testing: Case Study



```
class TestSimpBlogGenesis extends Specification
    static authors = ["Bart", "Homer", "Lisa", "
    static categories = ["Home", "Work", "Food",

    @Iterations(10)
    def 'the blog should be posted'() {
        given:
        def tester = new BlogTesterBoolean( url: '

        when:
        tester.postBlog title: title, category:

        then:
        tester.checkAll title, category, author, content

        where:
        author << any(authors)
        category << integer(0..<categories.size()).map{ categories[it] }
        title << string( maxLength: 40).map{ it + ' (genesis)' }
        content << string( maxLength: 255)
    }
}
```

```
PropertyBased
    build
    src
        test
            groovy
                QuickCheckExamples.groovy
                TestSimpBlogFuncJavaQuickcheck
                TestSimpBlogGenesis
                TestSimpBlogJUnitQuickcheck
                TestSimpBlogQuickCheck.groovy
                TestSimpBlogQuickTheories.groovy
    build.gradle
```

GPars

Library classes and DSL allowing you to handle tasks concurrently:

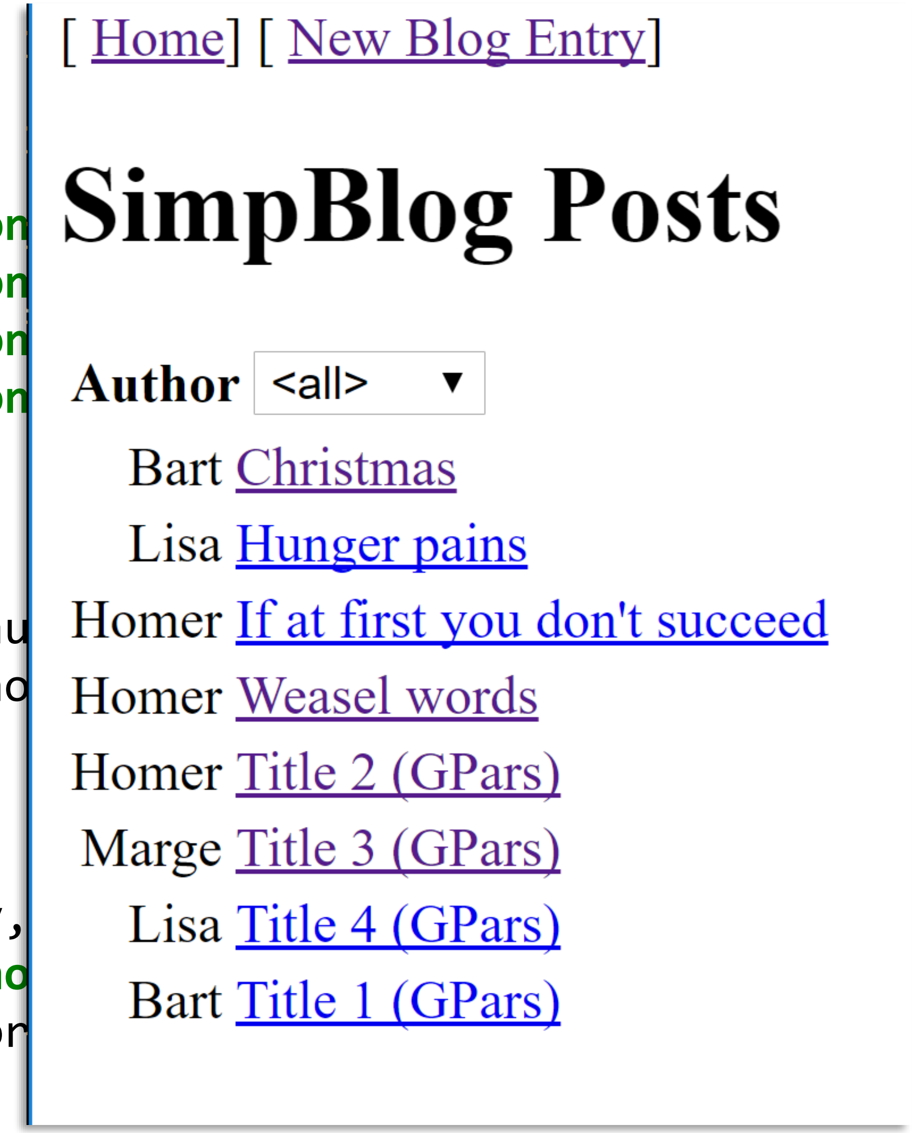
- ***Data Parallelism*** map, filter, reduce functionality in parallel with parallel array support
- *Asynchronous functions* extend the Java executor services to enable multi-threaded closure processing
- *Dataflow Concurrency* supports natural shared-memory concurrency model, using single-assignment variables
- *Actors* provide Erlang/Scala-like actors including "remote" actors on other machines
- Safe *Agents* provide a non-blocking mt-safe reference to mutable state; like "agents" in Clojure

Case Study with GPars

```
//@Grab('org.codehaus.gpars:gpars:0.12')
import groovyx.gpars.GParsPool

def testCases = [
    ['Title 1 (GPars)', 'Home',   'Bart',  'Content 1'],
    ['Title 2 (GPars)', 'Work',   'Homer', 'Content 2'],
    ['Title 3 (GPars)', 'Travel', 'Marge', 'Content 3'],
    ['Title 4 (GPars)', 'Food',   'Lisa',  'Content 4']
]

GParsPool.withPool {
    testCases.eachParallel{ title, category, author, content ->
            postAndCheck title, category, author, content
    }
}

def postAndCheck(String title, String category, String author, String content) {
    def tester = new BlogTester('http://localhost:8080/postForm')
    tester.postAndCheck title, category, author, content
}
```

Case Study with GPars

```
//@Grab('org.codehaus.gpars:gpars:0.12')
import groovyx.gpars.GParsPool

def testCases = [
    ['Title 1 (GPars)', 'Home',   'Bart',  'Con
    ['Title 2 (GPars)', 'Work',   'Homer', 'Con
    ['Title 3 (GPars)', 'Travel', 'Marge', 'Con
    ['Title 4 (GPars)', 'Food',   'Lisa',  'Con
]

GParsPool.withPool {
    testCases.eachParallel{ title, category, au
            postAndCheck title, category, autho
    }
}

def postAndCheck(String title, String category,
    def tester = new BlogTester('http://localho
    tester.postAndCheck title, category, author
}
```



[Home] [New Blog Entry]

SimpBlog Posts

Author <all>

Bart Christmas
Lisa Hunger pains
Homer If at first you don't succeed
Homer Weasel words
Homer Title 2 (GPars)
Marge Title 3 (GPars)
Lisa Title 4 (GPars)
Bart Title 1 (GPars)

Constraint/Logic Programming

Description

- Style of programming where relations between variables are stated in the form of constraints
- First made popular by logic programming languages such as Prolog but the style is now also used outside logic programming specific languages
- Constraints differ from the common primitives of other programming languages in that they do not specify one or more steps to execute but rather the properties of a solution to be found
- Popular libraries used with Groovy supporting constraint programming include Gecode/J, Choco and tuProlog
- We'll look at Choco as an example

Case Study with Constraint Programming

You have been asked to set up some test cases representing the Simpsons' weekly blogging habits

After some careful study you observe the following behavior

- They never blog on the same day
- Marge blogs only on a Saturday or Sunday
- Maggie blogs only on a Tuesday or Thursday
- Lisa blogs only on a Monday, Wednesday or Friday
- Bart blogs only on the day after Lisa
- Homer only blogs if noone else blogged the previous day and doesn't allow anyone to blog the next day

Case Study with Constraint Programming

```groovy
//@Grab('org.choco-solver:choco-solver:4.0.4')
import org.chocosolver.solver.Model

def m = new Model()

daysOfWeek = ["Sunday", "Monday", "Tuesday", "Wednesday",
              "Thursday", "Friday", "Saturday"]
def (SUN, MON, TUE, WED, THU, FRI, SAT) = 0..6

def bart = m.intVar('Bart', 0, 6)
def homer = m.intVar('Homer', 0, 6)
def marge = m.intVar('Marge', 0, 6)
def lisa = m.intVar('Lisa', 0, 6)
def maggie = m.intVar('Maggie', 0, 6)
def authors = [bart, homer, marge, lisa, maggie]
//…
```

Case Study with Constraint Programming

```
// They never blog on the same day
m.allDifferent(*authors).post()

// Marge blogs only on a Saturday or Sunday
m.or(m.arithm(marge, "=", SAT), m.arithm(marge, "=", SUN)).post()

// Maggie blogs only on a Tuesday or Thursday
m.or(m.arithm(maggie, "=", TUE), m.arithm(maggie, "=", THU)).post()

// Lisa blogs only on a Monday, Wednesday or Friday
m.or(m.arithm(lisa, "=", MON), m.arithm(lisa, "=", WED), m.arithm(lisa, "=", FRI)).post()

// Bart blogs only on the day after Lisa
m.arithm(bart, "-", lisa, "=", 1).post()

// Homer only blogs if noone else blogged the previous
// day and doesn't allow anyone to blog the next day
m.and(m.distance(homer, marge, "!=", 1),
        m.distance(homer, bart, "!=", 1),
        m.distance(homer, maggie, "!=", 1),
        m.distance(homer, lisa, "!=", 1)).post()
//…
```

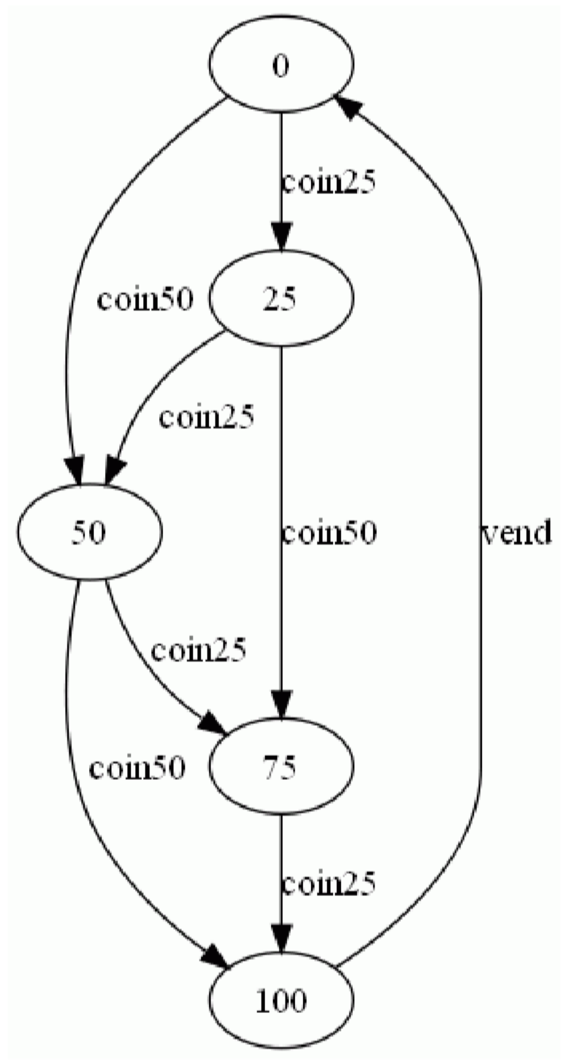
Case Study with Constraint Programming

```
def solutions = []
while (m.solver.solve()) {
    solutions << pad('') + authors.collect {
        pad(daysOfWeek[it.value])
    }.join()
}
if (solutions) {
    println pad("Solutions:") + authors.collect {
        pad(it.name)
    }.join()
    println solutions.join('\n')
} else {
    println "No Solutions"
}

def pa
```

```
Solutions:     Bart           Homer          Marge          Lisa           Maggie
               Thursday       Sunday         Saturday       Wednesday      Tuesday
               Thursday       Saturday       Sunday         Wednesday      Tuesday
               Saturday       Tuesday        Sunday         Friday         Thursday
               Tuesday        Saturday       Sunday         Monday         Thursday
```

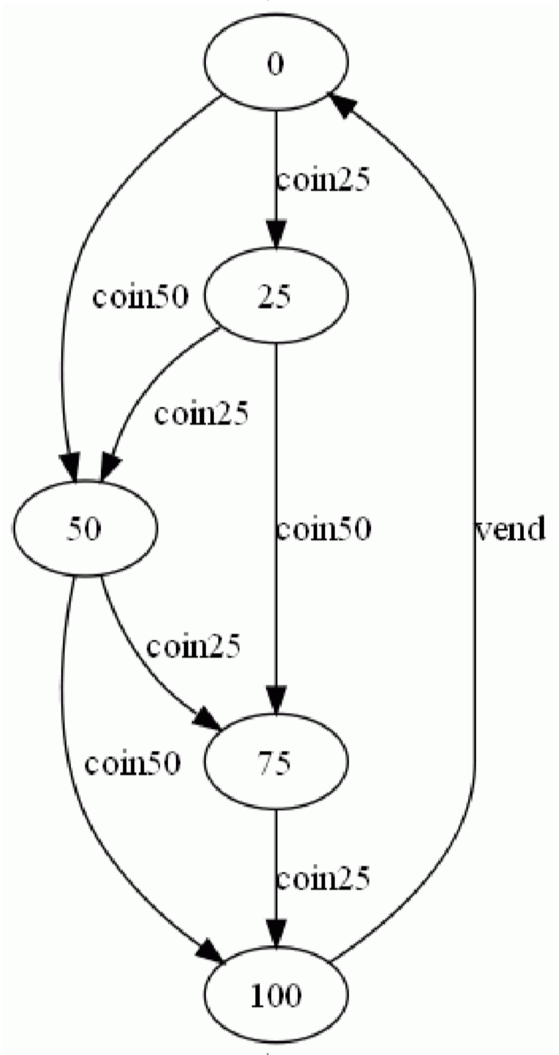
ModelJUnit

- **Description**
  - **Supports model-based testing**
  - **Allows you to write simple finite state machine (FSM) models or extended finite state machine (EFSM) models in Java or Groovy**
  - **You can then generate tests from those models and measure various model coverage metrics**



ModelJUnit

```
// require modeljunit.jar
import nz.ac.waikato.modeljunit.coverage.*
import nz.ac.waikato.modeljunit.*

class VendingMachineModel implements FsmModel {
    def state = 0 // 0,25,50,75,100
    void reset(boolean testing) {state = 0}

    boolean vendGuard() {state == 100}
    @Action void vend() {state = 0}

    boolean coin25Guard() {state <= 75}
    @Action void coin25() {state += 25}

    boolean coin50Guard() {state <= 50}
    @Action void coin50() {state += 50}
}

def tester = new RandomTester(new VendingMachineModel())
tester.buildGraph()
def metrics = [new ActionCoverage(), new StateCoverage(),
   new TransitionCoverage(), new TransitionPairCoverage()]
metrics.each { tester.addCoverageMetric it }

tester.addListener "verbose"
tester.generate 20

println '\nMetrics Summary:'
tester.printCoverage()
```



ModelJUnit

```
...
done (0, coin50, 50)
done (50, coin25, 75)
done (75, coin25, 100)
done Random reset(true)
done (0, coin50, 50)
done (50, coin25, 75)
done (75, coin25, 100)
done (100, vend, 0)
done (0, coin50, 50)
done (50, coin50, 100)
done (100, vend, 0)
done (0, coin25, 25)
done (25, coin25, 50)
done Random reset(true)
done (0, coin50, 50)
done (50, coin25, 75)
done (75, coin25, 100)
done (100, vend, 0)
done (0, coin50, 50)
done (50, coin25, 75)
...
```

```
...
Metrics Summary:
action coverage: 3/3
state coverage: 5/5
transition coverage: 7/8
transition-pair coverage: 8/12
...
```

Case Study with ModelJUnit

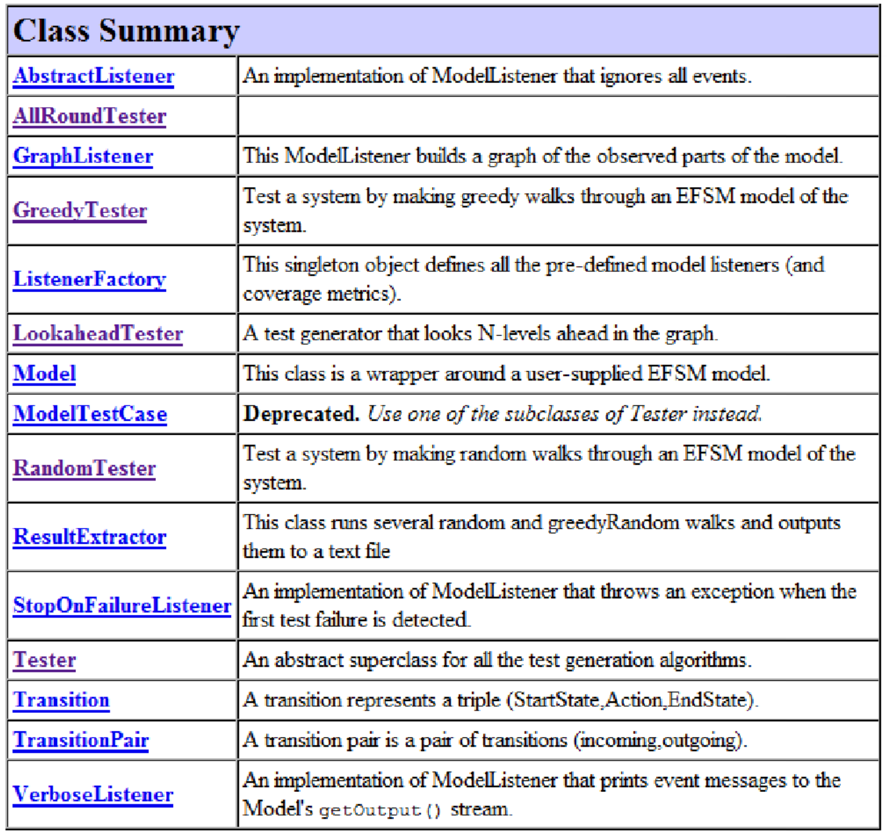
- **Does the order in which form information is entered affect the application?**
  - **Could AJAX effects be causing unexpected results?**

```
...
  void reset(boolean testing) {
    authorSelected = false
    categorySelected = false
    titleEntered = false
    contentEntered = false
    client = new WebClient()
    page = client.getPage('http://localhost:8080/postForm')
    assert 'Welcome to SimpBlog' == page.titleText
    form = page.getFormByName('post')
  }
...
```

```
// require modeljunit.jar, htmlunit.jar
import nz.ac.waikato.modeljunit.coverage.*
import nz.ac.waikato.modeljunit.*
import com.gargoylesoftware.htmlunit.WebCl

class SimpBlogModel implements FsmModel {
  boolean authorSelected = false
  boolean categorySelected = false
  boolean titleEntered = false
  boolean contentEntered = false
  int count = 0
  def client, page, form

  // Special known method, allows equivalence class definition
  // example states: __ __ __ __, AU __ __ __, AU CA TI CO
  def getState() {
    "${authorSelected ? ' AU ' : ' __ '}${categorySelected ? ' CA ' : ' __ '}" +
    "${titleEntered ? ' TI ' : ' __ '}${contentEntered ? ' CO ' : ' __ '}"
  }
...
```

Case Study with ModelJUnit

```
def tester = new RandomTester(new SimpBlogModel())
tester.buildGraph()
def metrics = [
    new ActionCoverage(),
    new StateCoverage(),
    new TransitionCoverage(),
    new TransitionPairCoverage()
]
metrics.each {
    tester.addCoverageMetric it
}

tester.addListener "verbose"
tester.generate 50

println '\nMetrics Summary:'
tester.printCoverage()

def graphListener = tester.model.getListener("graph")
graphListener.printGraphDot "simpblog.dot"
println "\nGraph contains " + graphListener.graph.numVertices() +
        " states and " + graphListener.graph.numEdges() + " transitions."
```



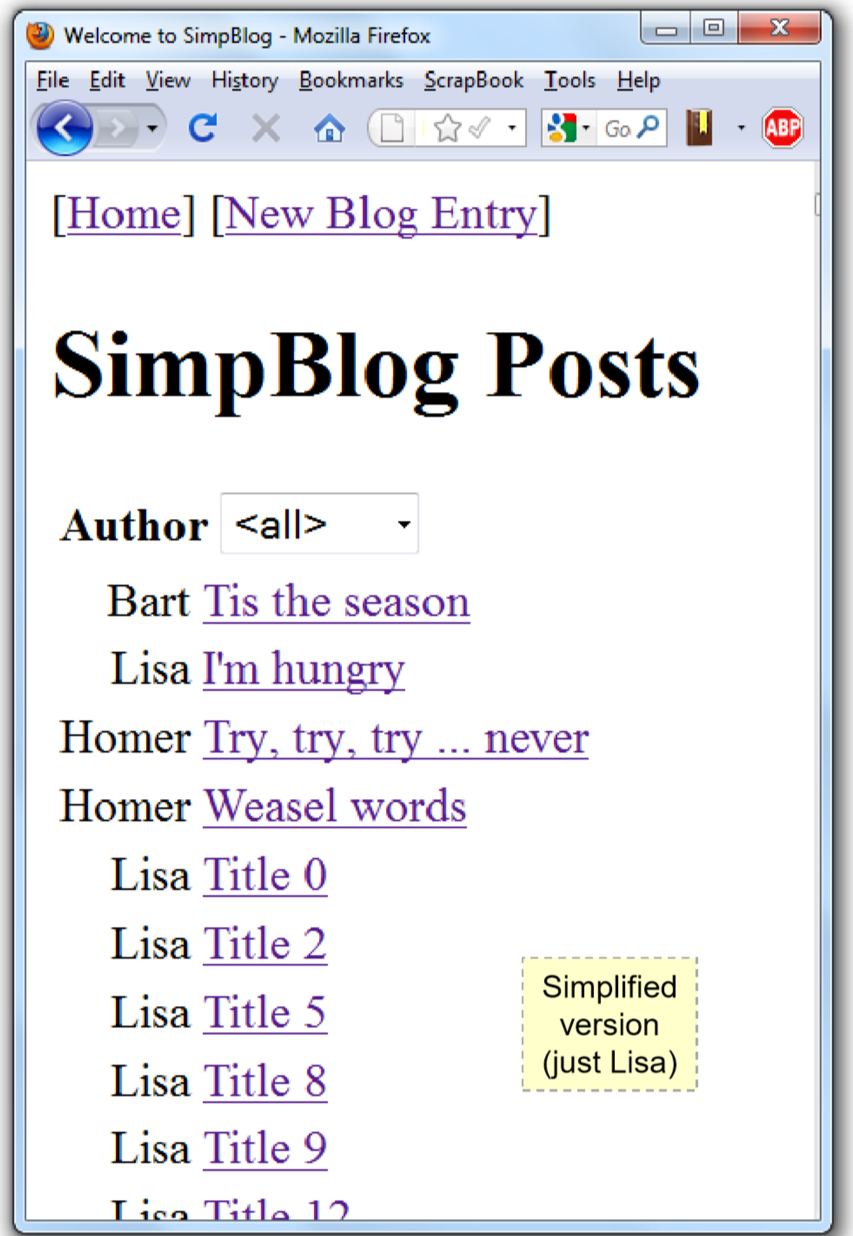
Class Summary	
AbstractListener	An implementation of ModelListener that ignores all events.
AllRoundTester	
GraphListener	This ModelListener builds a graph of the observed parts of the model.
GreedyTester	Test a system by making greedy walks through an EFSM model of the system.
ListenerFactory	This singleton object defines all the pre-defined model listeners (and coverage metrics).
LookaheadTester	A test generator that looks N-levels ahead in the graph.
Model	This class is a wrapper around a user-supplied EFSM model.
ModelTestCase	**Deprecated.** *Use one of the subclasses of Tester instead.*
RandomTester	Test a system by making random walks through an EFSM model of the system.
ResultExtractor	This class runs several random and greedyRandom walks and outputs them to a text file
StopOnFailureListener	An implementation of ModelListener that throws an exception when the first test failure is detected.
Tester	An abstract superclass for all the test generation algorithms.
Transition	A transition represents a triple (StartState,Action,EndState).
TransitionPair	A transition pair is a pair of transitions (incoming,outgoing).
VerboseListener	An implementation of ModelListener that prints event messages to the Model's `getOutput()` stream.

Case Study with ModelJUnit

```
done ( __ __ __ __ , pickCategory, __ CA __ __ )
done ( __ CA __ __ , enterContent, __ CA __ CO )
done ( __ CA __ CO , enter title , __ CA TI CO )
done ( __ CA TI CO , chooseAuthor, AU CA TI CO )
done ( AU CA TI CO , submit post , __ __ __ __ )
done ( __ __ __ __ , pickCategory, __ CA __ __ )
done ( __ CA __ __ , chooseAuthor, AU CA __ __ )
done ( AU CA __ __ , enter title , AU CA TI __ )
done ( AU CA TI __ , enterContent, AU CA TI CO )
done ( AU CA TI CO , submit post , __ __ __ __ )
done ( __ __ __ __ , chooseAuthor, AU __ __ __ )
done ( AU __ __ __ , pickCategory, AU CA __ __ )
done ( AU CA __ __ , enter title , AU CA TI __ )
done ( AU CA TI __ , enterContent, AU CA TI CO )
done ( AU CA TI CO , submit post , __ __ __ __ )
done ( __ __ __ __ , enterContent, __ __ __ CO )
done ( __ __ __ CO , pickCategory, __ CA __ CO )
done ( __ CA __ CO , chooseAuthor, AU CA __ CO )
done ( AU CA __ CO , enter title , AU CA TI CO )
done ( AU CA TI CO , submit post , __ __ __ __ )
done ( __ __ __ __ , pickCategory, __ CA __ __ )
done ( __ CA __ __ , enter title , __ CA TI __ )
done ( __ CA TI __ , chooseAuthor, AU CA TI __ )
done ( AU CA TI __ , enterContent, AU CA TI CO )
done ( AU CA TI CO , submit post , __ __ __ __ )
done ( __ __ __ __ , chooseAuthor, AU __ __ __ )
done ( AU __ __ __ , pickCategory, AU CA __ __ )
done ( AU CA __ __ , enter title , AU CA TI __ )
done ( AU CA TI __ , enterContent, AU CA TI CO )
done ( AU CA TI CO , submit post , __ __ __ __ )
...
```

```
...
done ( __ __ __ __ , pickCategory, __ CA __ __ )
done ( __ CA __ __ , enterContent, __ CA __ CO )
done ( __ CA __ CO , chooseAuthor, AU CA __ CO )
done ( AU CA __ CO , enter title , AU CA TI CO )
done ( AU CA TI CO , submit post , __ __ __ __ )
done ( __ __ __ __ , chooseAuthor, AU __ __ __ )
done ( AU __ __ __ , pickCategory, AU CA __ __ )
done ( AU CA __ __ , enterContent, AU CA __ CO )
done ( AU CA __ CO , enter title , AU CA TI CO )
done ( AU CA TI CO , submit post , __ __ __ __ )
done ( __ __ __ __ , chooseAuthor, AU __ __ __ )
done ( AU __ __ __ , enter title , AU __ TI __ )
done ( AU __ TI __ , pickCategory, AU CA TI __ )
done ( AU CA TI __ , enterContent, AU CA TI CO )
done ( AU CA TI CO , submit post , __ __ __ __ )
done Random reset(true)
done ( __ __ __ __ , pickCategory, __ CA __ __ )
done ( __ CA __ __ , enterContent, __ CA __ CO )
done ( __ CA __ CO , enter title , __ CA TI CO )
done ( __ CA TI CO , chooseAuthor, AU CA TI CO )

Metrics Summary:
action coverage: 5/5
state coverage: 12/16
transition coverage: 19/33
transition-pair coverage: 26/56

Graph contains 16 states and 33 transitions.
```

Case Study with ModelJUnit


Welcome to SimpBlog - Mozilla Firefox
File Edit View History Bookmarks ScrapBook Tools Help

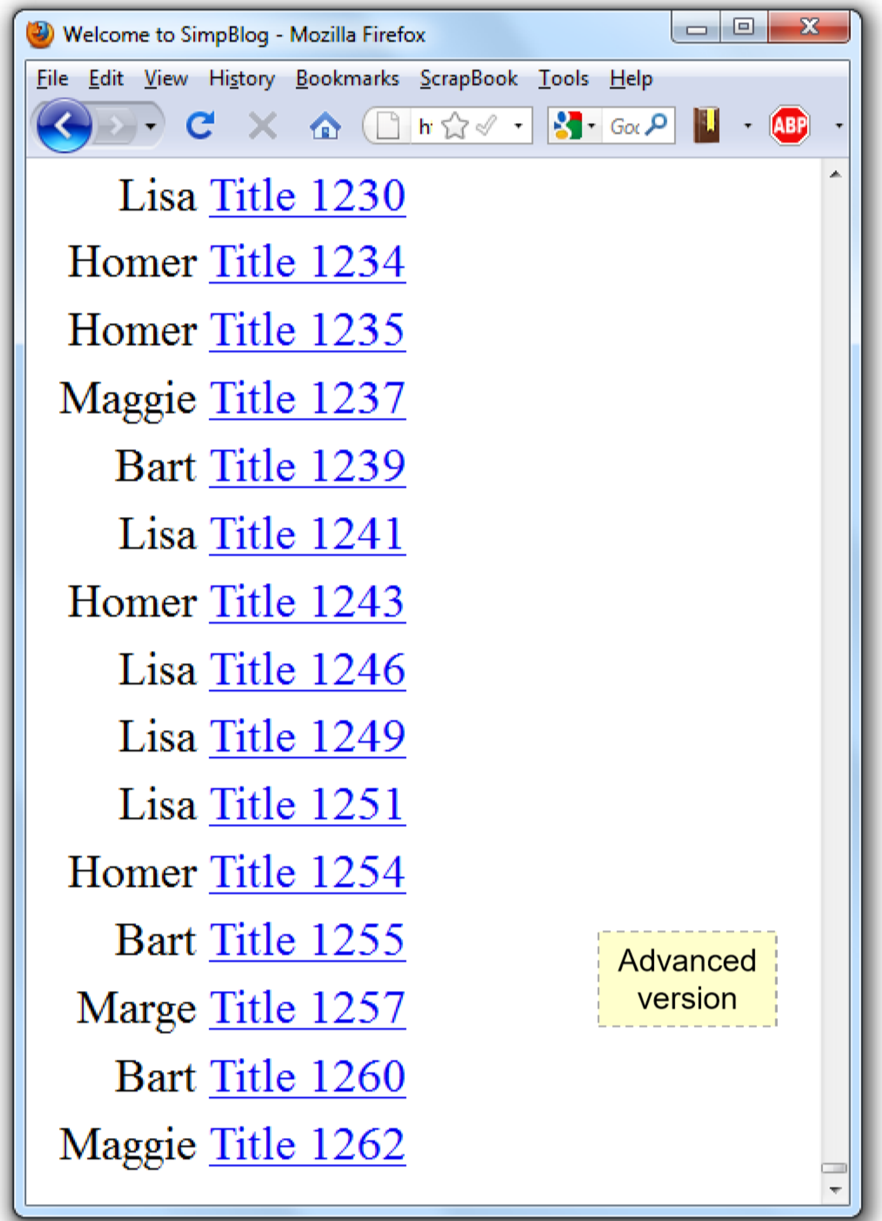
[Home] [New Blog Entry]

SimpBlog Posts

Author <all>

Bart Tis the season
Lisa I'm hungry
Homer Try, try, try ... never
Homer Weasel words
Lisa Title 0
Lisa Title 2
Lisa Title 5
Lisa Title 8
Lisa Title 9

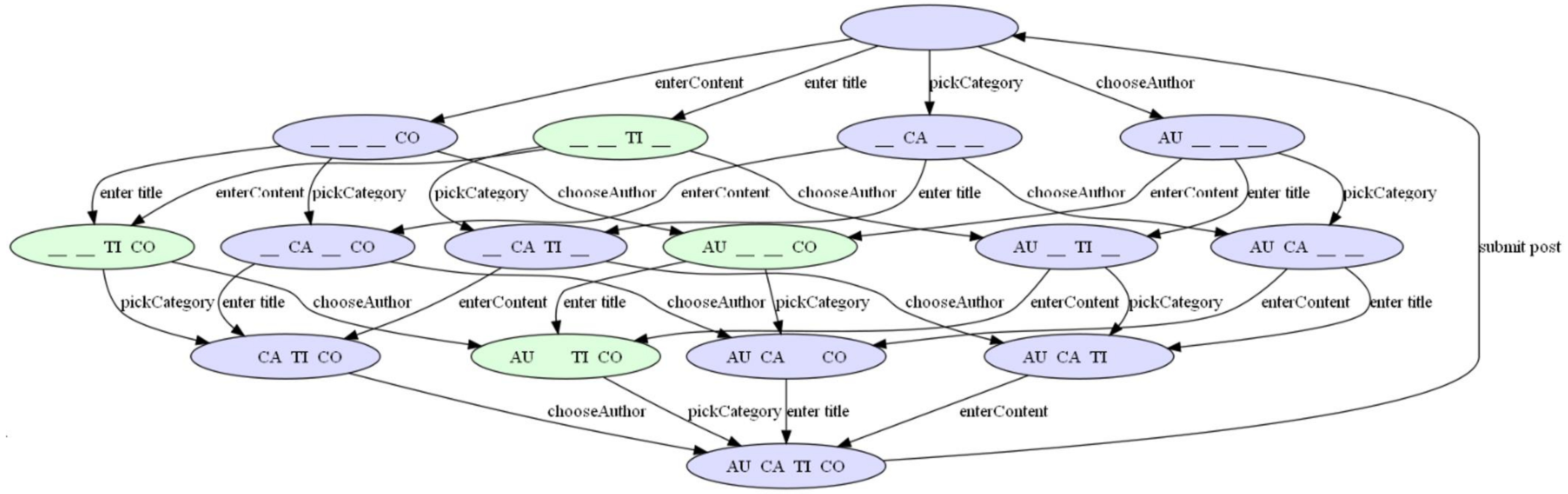
Simplified version (just Lisa)

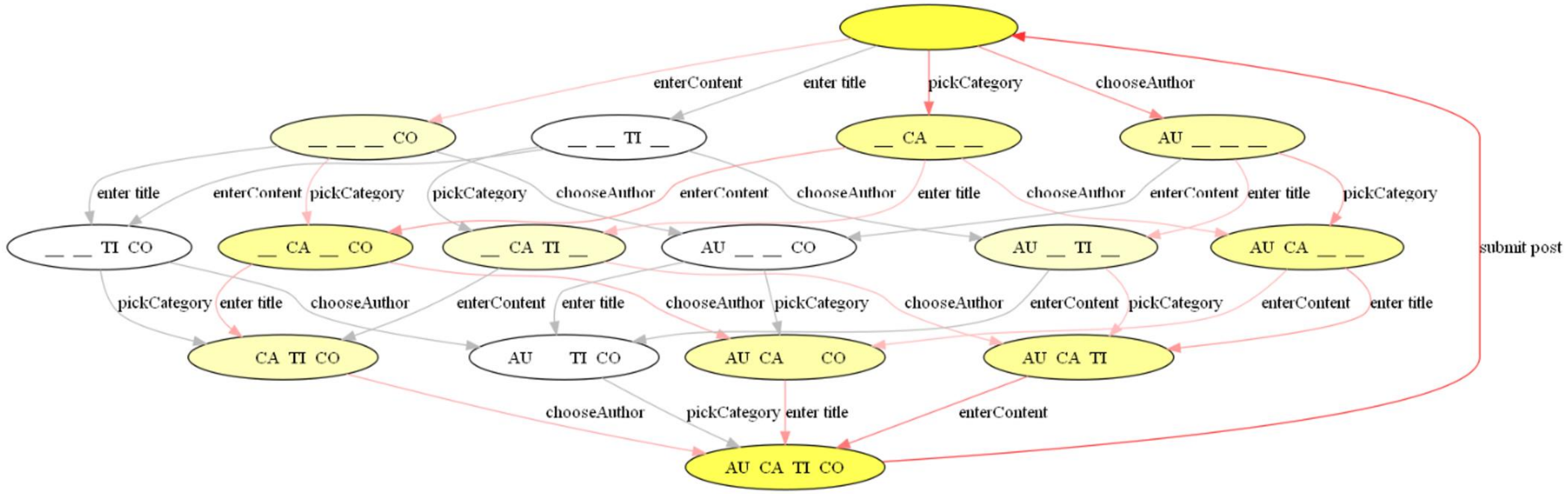

Welcome to SimpBlog - Mozilla Firefox
File Edit View History Bookmarks ScrapBook Tools Help

Lisa Title 1230
Homer Title 1234
Homer Title 1235
Maggie Title 1237
Bart Title 1239
Lisa Title 1241
Homer Title 1243
Lisa Title 1246
Lisa Title 1249
Lisa Title 1251
Homer Title 1254
Bart Title 1255
Marge Title 1257
Bart Title 1260
Maggie Title 1262

Advanced version

Case Study with ModelJUnit





Wrapup: Key Testing Practices…

Use testing DSL’s

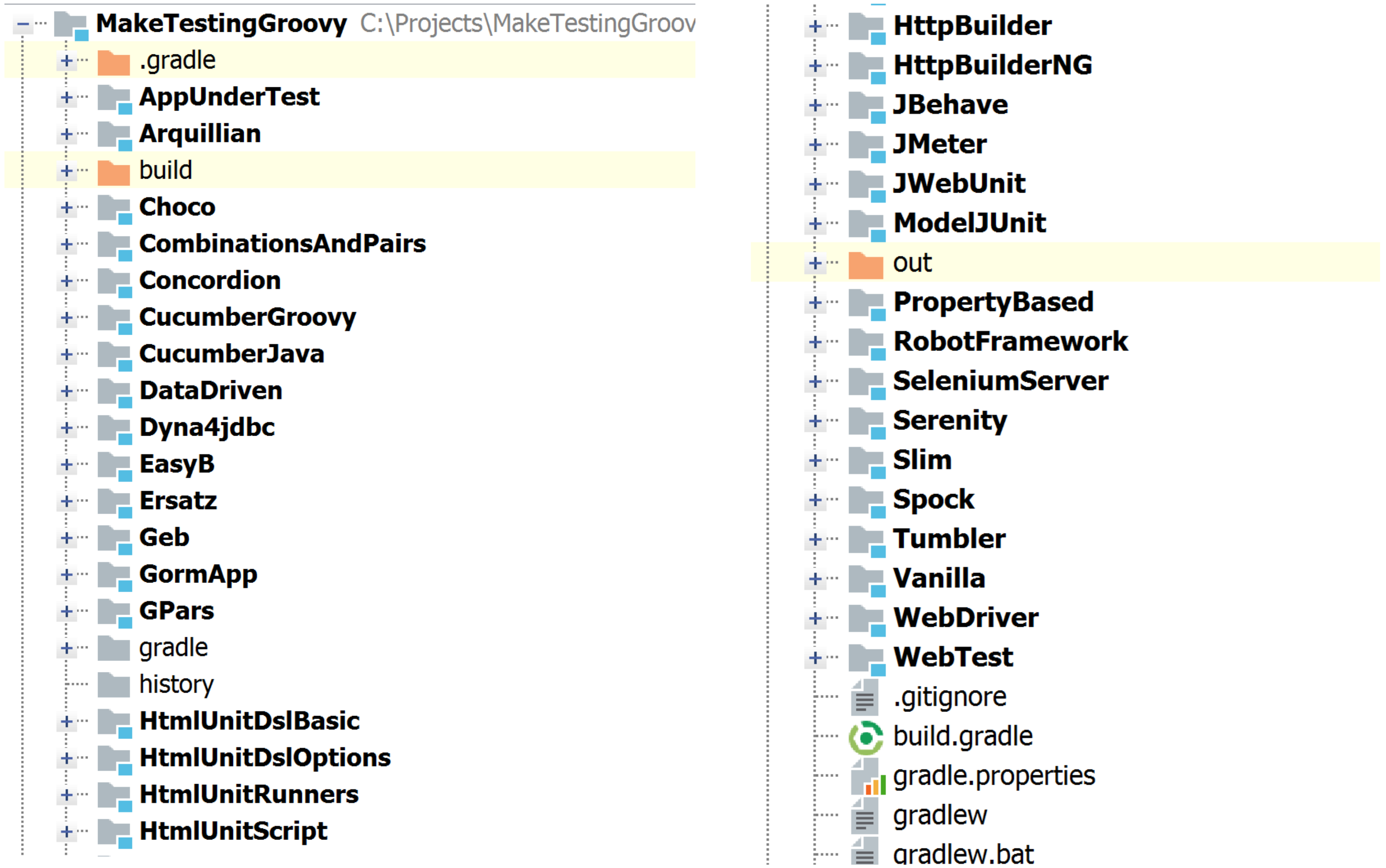
Look to move up the testing stack

- It used to be all about the driver
- Now the driver is hidden in the framework or tool stack

Apply good testing practices

- Pareto analysis, bug clusters, mutation testing, test early, all pairs/equivalence partitions/orthogonal array testing, risk-based test selection, coding for testability, use CI, boundary value analysis, defensive programming

https://github.com/paulk-asert/MakeTestingGroovy



MakeTestingGroovy C:\Projects\MakeTestingGroov
- .gradle
- AppUnderTest
- Arquillian
- build
- Choco
- CombinationsAndPairs
- Concordion
- CucumberGroovy
- CucumberJava
- DataDriven
- Dyna4jdbc
- EasyB
- Ersatz
- Geb
- GormApp
- GPars
- gradle
- history
- HtmlUnitDslBasic
- HtmlUnitDslOptions
- HtmlUnitRunners
- HtmlUnitScript
- HttpBuilder
- HttpBuilderNG
- JBehave
- JMeter
- JWebUnit
- ModelJUnit
- out
- PropertyBased
- RobotFramework
- SeleniumServer
- Serenity
- Slim
- Spock
- Tumbler
- Vanilla
- WebDriver
- WebTest
- .gitignore
- build.gradle
- gradle.properties
- gradlew
- gradlew.bat

More Information: Groovy in Action