Mockito and the Hamcrest Matchers

Mocks, Stubs, and Matchers

Contact Info

Ken Kousen

Kousen IT, Inc.

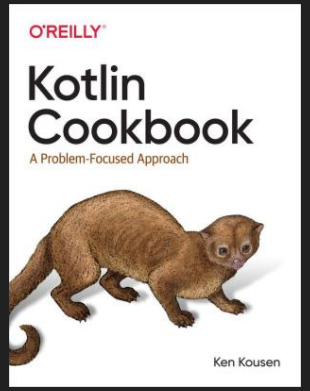
ken.kousen@kousenit.com

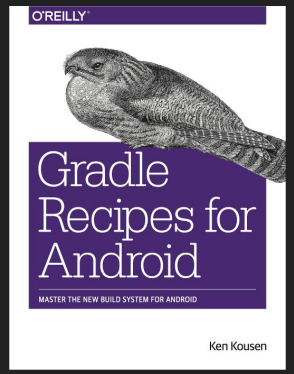
http://www.kousenit.com

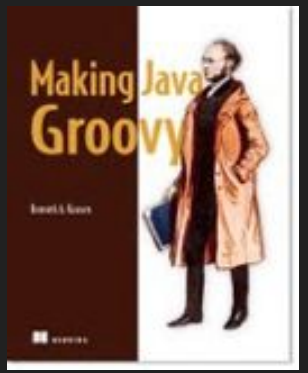
http://kousenit.org (blog)

@kenkousen (twitter)

Tales from the jar side,
https://kenkousen.substack.com (newsletter)







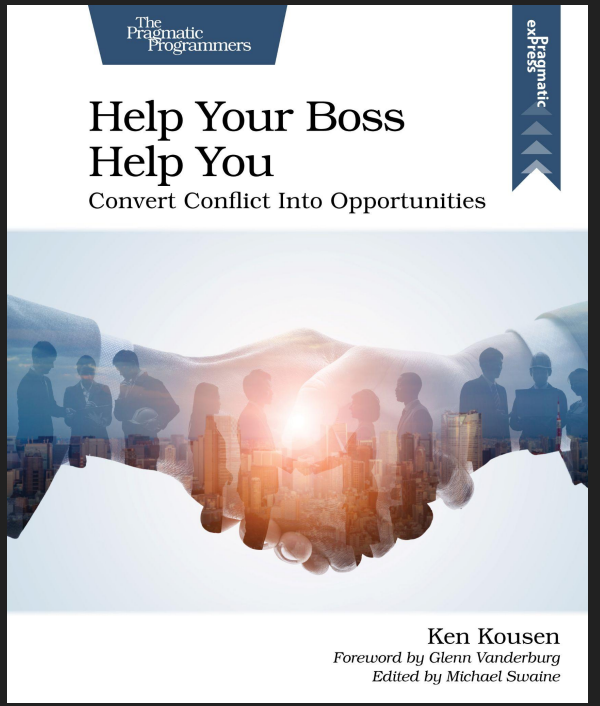
New Book

Help Your Boss Help You

How to manage your manager to get
what you want in your job and your career

At Amazon:

#1 New Release in Business Ethics
#1 New Release in Business Communications



Videos

O'Reilly video courses: See Safari Books Online for details

- Groovy Programming Fundamentals,
  - Practical Groovy Programming, Mastering Groovy Programming
- Learning Android, Practical Android
- Gradle Fundamentals, Gradle for Android
- Spring Framework Essentials
  - Reactive Spring
- Advanced Java Development
  - Understanding Java 8 Generics
- Managing Your Manager
- Grails 3 (several, starting with this)

GitHub Repository

Mockito and the Hamcrest Matchers:

https://github.com/kousen/mockito-hamcrest

Topics

Hamcrest Matchers

A convenient set of methods to make JUnit tests more readable

Mockito

A library for creating mocks and stubs

Hamcrest matchers

- Hamcrest is a framework for software tests
- Syntactic sugar for JUnit tests
  - Behavior is the same
  - Readability is better

Hamcrest

Wikipedia article

https://en.wikipedia.org/wiki/Hamcrest

First generation: simple assert statements

Second generation: family of assert statements,
like assertEquals, assertTrue, assertNotNull

Third generation: assertThat with matchers

```
assertThat(x, is(not(equalTo(y))))
```

Documentation

- Home page: http://hamcrest.org
- Java implementation: http://hamcrest.org/JavaHamcrest/
- JavaDocs: http://hamcrest.org/JavaHamcrest/javadoc/
- Tutorial: https://code.google.com/archive/p/hamcrest/wikis/Tutorial.wiki
- Mailing list: https://groups.google.com/forum/?fromgroups#!forum/hamcrest-java
- Source code: https://github.com/hamcrest/JavaHamcrest

Set Up

Gradle dependency for JUnit 4:

```
dependencies {
    testImplementation 'junit:junit:4.13.1'
    testImplementation 'org.hamcrest:hamcrest:2.2'
}
```

JUnit 5 instead:

```
    testImplementation 'org.junit.jupiter:junit-jupiter:5.8.1'
```

Set Up

Maven

```
<dependency>
    <groupId>org.hamcrest</groupId>
    <artifactId>hamcrest</artifactId>
    <version>2.2</version>
    <scope>test</scope>
</dependency>
```

Set Up

Prior implementations divided into multiple jars:

Only hamcrest-core is required

hamcrest-library adds additional matchers

hamcrest-integration provides integration with
JUnit, TestNG, jMock, and EasyMock

hamcrest-all contains all the above

See
https://code.google.com/archive/p/hamcrest/wikis/HamcrestDistributables.wiki

assertThat

Start with MatcherAssert.assertThat(...)

```
static void assertThat(String reason, boolean assertion)

static <T> void assertThat(String reason,
     T actual, Matcher<? super T> matcher)

static <T> void assertThat(T actual, Matcher<? super T> matcher)
```

Simple Matchers

Lots of static methods in Matchers class

equalTo(obj) → invoke equals(Object) method

is(T val) → shorthand for is(equalTo(val))

"is" → "syntactic sugar" designed to make test more readable

Simple Matchers

allOf → true if all matchers match

anyOf → true if any matchers match

not → flips boolean

isA, instanceOf → type test

nullValue, notNullValue

sameInstance

Simple Matchers

greaterThan, greaterThanOrEqualTo

lessThan, lessThanOrEqualTo

closeTo → used for floating point and BigDecimals

is, equalTo

equalToIgnoreCase, equalToCompressingWhiteSpace

white space trimmed, then internal collapsed to single space

containsString, endsWith, startsWith

Bean Matchers

hasProperty("property")

samePropertyValuesAs(obj)

Custom Matchers

Create your own matcher by extending TypeSafeMatcher

Already implements null checks, checks the type, and casts

delegates to matchesSafely

See tutorial for details

Mockito

Mocks, Stubs, and Spies

Consider an order processor

```
public Confirmation processOrder() {

    // calculate total cost
    // look up customer info
    // process any discounts
    // add required taxes and fees
    // add in shipping costs
    // process credit card

}
```

Order processor

Most of that is local, but what about the credit card processor?

```
public Confirmation processOrder() {

    // …
    cardService.chargeCard(String number);
}
```

Don't want to call the real credit card service while testing

Mocks vs Stubs

What we need for the credit card processor is a

Mock object

… or is that a stub?

Mocks vs Stubs

A stub

stands in for the real object

provides specific responses to method calls

mockCardService.chargeCard("12345") → true

This is called setting expectations

Mocks vs Stubs

A mock

- stands in for the real object
- verifies that methods were called:
  - the right number of times
  - in the right order

You verify a mock

Mocks vs Stubs

```
@Test
public void testProcessOrder() {

    // verify:
    // customer lookup called first
    // total price calculator called next
    // shipping service called after that
    // credit card service called last

    // each called exactly once
}
```

Mocks vs Stubs

The created object (mock or stub) is the same

The difference is how they are used:

stubs provide known outputs for method calls

mocks verify the protocol
the interaction between our test class and the mock

Spies

A spy is a partial mock

method calls pass through to underlying real object

can replace some calls with specific outputs

Mockito makes a copy of the real instance

Mockito discourages their use, but allows it when necessary

Mockito

A mocking (and stubbing) tool

Enables true unit tests in Java

Programmatic stubbing via

- Mockito.when(mock.action()).thenReturn(true)
- BDDMockito.given(mock.action()).willReturn(true)

Documentation

- Home page: http://mockito.org/
- JavaDocs: http://javadoc.io/page/org.mockito/mockito-core/latest/org/mockito/Mockito.html
- Release Notes: https://github.com/mockito/mockito/blob/release/2.x/doc/release-notes/official.md
- FAQ: https://github.com/mockito/mockito/wiki/FAQ
- Mailing list: http://groups.google.com/group/mockito

Mockito

Limitations (some by design):

- Cannot mock static methods (until 3.4.0)
- Cannot mock constructors
- Cannot mock equals(), hashCode()
- Cannot mock private methods

Capabilities:

- Can mock both classes and interfaces

Mockito Versions

Mockito 2 → significant changes from 1

Mockito 3 → requires Java 8+

Mockito 4 → removes deprecated classes and methods

Everything from 2.* on up works with 4

Mockito

Gradle dependency:

```
repositories {
    jcenter()
}

dependencies {
    testImplementation 'org.mockito:mockito-core:4.0.0'
}
```

Using Mockito

Create mocks with:

- static mock() method
- @Mock annotation

Mockito

Programmatic verification via

- Mockito.verify(mock).action()
- BDDMockito.then(mock).should().action()

Annotations available for mocking

- @Mock
- @Spy
- @Captor
- @InjectMocks

Mockito

Provides its own JUnit 4 runner:

```
@RunWith(MockitoJUnitRunner.class)
```

For JUnit 5, add dependency:

```
testImplementation "org.mockito:mockito-junit-jupiter:4.0.0"
```

Then use Mockito extension:

```
@ExtendWith(MockitoExtension.class)
```

Using Annotations

Add @Mock (or @Spy) to attributes

Three ways to use Mockito with JUnit 4:

Call MockitoAnnotations.openMocks(this)

Use JUnit 4 Rule:

```
@Rule
public MockitoRule mockitoRule = MockitoJUnit.rule();
```

Use Mockito JUnit 4 Runner

Configuring Mocks

Mocked objects return default values if not specified

- null for object references
- zero for numbers
- false for booleans
- empty collections for collections
- etc.

Configuring Mocks

Setting expectations

After mock(MyClass.class),

when(...).thenReturn(...)

Can chain thenReturn(...) calls

Returns in order, then the final one repeatedly

when(...).thenReturn(...).thenReturn(...)

Configuring Mocks

Configure mock based on specific argument

when(42).thenReturn(true)

Can use ArgumentMatchers

anyInt(), anyBoolean(), anyString(), …

any(), any(Class<T>)

Verifying Invocations

verify(mock).method() checks that method() is called on mock

verify(mock, times(1)).method()

- times(int)
- never()
- atLeastOnce()
- atLeast(int)
- atMost(int)

Configuring Mocks

Can not mock methods that return void the same way

Can not mock methods that throw exceptions the same way

Use the "doReturn" methods instead

```
doReturn(...).when(...).action()
```

```
doThrow(new RuntimeException()).when(...).action()
```

Ordering

Can verify methods invoked in the proper order

Use InOrder class, which takes one or more mocks as arguments

Spies

Wraps a real object

Every call is delegated to the object unless specified otherwise

Use the spy() method or @Spy

Verifying Behavior

- Mockito keeps track of all method calls and parameters on real object
- Use verify() on mock
- Distinguishes between true mocks and true stubs
  - Stubs just return specified values
  - Mocks verify the protocol
    - Methods are called the right number of times, in proper order

Verifying Behavior

Use ArgumentMatchers to check argument types

- static methods like eq(), anyInt(), any()

Check multiplicity

- never(), atLeastOnce(), atLeast(num)
- times(...), atMost(...)

ArgumentCaptor

ArgumentCaptor allows access to arguments of method calls

verify(mockedList).addAll(captor.capture())

Then captor.getValue() returns actual value

BDD

Behavior Driven Development

Use BDDMockito class

given(mock.method()).willReturn(value)

then(mock).should(times(1)).method()

Mocking final types

Incubating capability to mock:

- final types
- enums
- final methods

Use Mockito extension mechanism

/mockito-extensions/org.mockito.plugins.MockMaker

containing "mock-maker-inline"

Mocking final types

Alternatively, use "mockito-inline" artifact in build

Still restrictions; see docs for details

Summary

Hamcrest

- Simplified assertions
- Useful for closeTo, containsInAnyOrder, custom matchers

Mockito

- Stubs provide known values
- Mocks verify interactions