Migrando de jMock a Mockito

Evoluciona o muere

Wences Arana

lugusac, debian-gt, guatejug

14 de noviembre de 2015

JAVA DAY 2015 Guatemala

Programa

- Frameworks
  - jMock
  - Mockito

- Diferencias entre jMock y Mockito
- Creando mocks
- Verificación de llamadas y valores de retorno
- Argument matchers
- Number of invocations
- Llamadas consecutivas
- Verificación de orden de llamadas
- Manejo de excepciones

Programa

- Frameworks
  - jMock
  - Mockito
- Diferencias entre jMock y Mockito
- Creando mocks
- Verificación de llamadas y valores de retorno
- Argument matchers
- Number of invocations
- Llamadas consecutivas
- Verificación de orden de llamadas
- Manejo de excepciones

jMock

Features:[1]

[1] http://www.jmock.org/index.html

jMock

Features:[1]

[1] http://www.jmock.org/index.html

jMock

Features:[1]

- Provee una extensa librería de mocking objects

[1] http://www.jmock.org/index.html

jMock

Features:[1]

- Provee una extensa librería de mocking objects
- Fácil y rápido

[1] http://www.jmock.org/index.html

jMock

Features:[1]

- Provee una extensa librería de mocking objects
- Fácil y rápido
- Librería extensa

[1] http://www.jmock.org/index.html

jMock

Features:[1]

- Provee una extensa librería de mocking objects
- Fácil y rápido
- Librería extensa
- Completamente declarativa

[1] http://www.jmock.org/index.html

jMock

Features:[1]

- Provee una extensa librería de mocking objects
- Fácil y rápido
- Librería extensa
- Completamente declarativa
- Fácil de extender

[1] http://www.jmock.org/index.html

jMock

Features:[1]

- Provee una extensa librería de mocking objects
- Fácil y rápido
- Librería extensa
- Completamente declarativa
- Fácil de extender
- expect-run-verify

[1] http://www.jmock.org/index.html

Programa

- Frameworks
  - jMock
  - Mockito

- Diferencias entre jMock y Mockito
- Creando mocks
- Verificación de llamadas y valores de retorno
- Argument matchers
- Number of invocations
- Llamadas consecutivas
- Verificación de orden de llamadas
- Manejo de excepciones

Mockito

Features:[2]

[2] https://code.google.com/p/mockito/wiki/FeaturesAndMotivations

Mockito

Features:[2]

- Desarrollo continuo (push)

[2] https://code.google.com/p/mockito/wiki/FeaturesAndMotivations

Mockito

Features:[2]

- Desarrollo continuo (push)
- JDK8 compatible (Defenders)

[2] https://code.google.com/p/mockito/wiki/FeaturesAndMotivations

Mockito

Features:[2]

- Desarrollo continuo (push)
- JDK8 compatible (Defenders)
- API limpia y simple

[2] https://code.google.com/p/mockito/wiki/FeaturesAndMotivations

Mockito

Features:[2]

- Desarrollo continuo (push)
- JDK8 compatible (Defenders)
- API limpia y simple
- Fácil lectura

[2] https://code.google.com/p/mockito/wiki/FeaturesAndMotivations

Mockito

Features:[2]

- Desarrollo continuo (push)
- JDK8 compatible (Defenders)
- API limpia y simple
- Fácil lectura
- Bastante utilizada

[2] https://code.google.com/p/mockito/wiki/FeaturesAndMotivations

Mockito

Features(continuación...)

Mockito

Features(continuación...)

- Mockea clasess además de interfaces

Mockito

Features(continuación...)

- Mockea clasess además de interfaces
- ANNOTATIONS o/

Mockito

Features(continuación...)

- Mockea clasess además de interfaces
- ANNOTATIONS o/
- Stack trace limpio

Funcionamiento

Funcionamiento

- No es una librería expect-run-verify

Funcionamiento

- No es una librería expect-run-verify
- NO EXISTEN LAS EXPECTATIONS

Funcionamiento

- No es una librería expect-run-verify
- NO EXISTEN LAS EXPECTATIONS
- Preguntar sobre interacciones despues de ejecutar

Funcionamiento

- No es una librería expect-run-verify
- NO EXISTEN LAS EXPECTATIONS
- Preguntar sobre interacciones despues de ejecutar
- Verifica solo lo que necesites (I can test what I want)

Funcionamiento

- No es una librería expect-run-verify
- NO EXISTEN LAS EXPECTATIONS
- Preguntar sobre interacciones despues de ejecutar
- Verifica solo lo que necesites (I can test what I want)
- Stubbing va antes de la ejecución, verificaciones de interacción después

Programa

Frameworks
- jMock
- Mockito

Diferencias entre jMock y Mockito
- Creando mocks
- Verificación de llamadas y valores de retorno
- Argument matchers
- Number of invocations
- Llamadas consecutivas
- Verificación de orden de llamadas
- Manejo de excepciones

Programa

Frameworks
- jMock
- Mockito

Diferencias entre jMock y Mockito
- Creando mocks
- Verificación de llamadas y valores de retorno
- Argument matchers
- Number of invocations
- Llamadas consecutivas
- Verificación de orden de llamadas
- Manejo de excepciones

Mockito vs jMock

- jMock usa un contexto para manejar los mocks
- mockito lo hace con metodos importados estáticamente
- jMock esta diseñado para mockear interfaces, no clases
- mockito puede mockear clases
- mockito soporta anotaciones

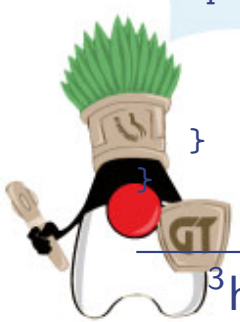
Mockito vs jMock

[3] jMock

```
import org.jmock.Mockery;
import org.junit.Test;

public class PosTerminalTest {
    @Test
    public void shouldPrintSuccessfulWithdrawal() {
        Mockery context = new Mockery();

        BankConnection bankConnection = context.mock(BankConnection.class);
        ReceiptPrinter receiptPrinter = context.mock(ReceiptPrinter.class);
        // the test case continues...
    }
}
```

Mockito

```
{
import static org.mockito.Mockito.mock;
import org.junit.Test;

public class PosTerminalTest {
    @Test
    public void shouldPrintSuccessfulWithdrawal() {
        BankConnection bankConnection = mock(BankConnection.class);
        ReceiptPrinter receiptPrinter = mock(ReceiptPrinter.class);
        // the test case continues...
    }
}
```



[3] http://zsoltfabok.com/blog/2010/08/jmock-versus-mockito/

Mockito(continuación ...)

```
import org.mockito.Mock;
// Other import statements ...

@RunWith(MockitoJUnitRunner.class)
public class PosTerminalTest {
    @Mock
    private BankConnection bankConnection;
    @Mock
    private ReceiptPrinter receiptPrinter;
    @Mock
    private Account customerAccount;
    @Mock
    private Account shopAccount;

    @Test
    public void shouldPrintSuccessfulWithdrawal() {
    }
```

Programa

- Frameworks
  - jMock
  - Mockito
- Diferencias entre jMock y Mockito
  - Creando mocks
  - Verificación de llamadas y valores de retorno
    - Argument matchers
    - Number of invocations
    - Llamadas consecutivas
    - Verificación de orden de llamadas
    - Manejo de excepciones

- mockito usa verify
- mockito no revisa si se hace una llamada no esperada
- jMock tiene allowing e ignoring

jMock

```
@Test
    public void shouldPrintSuccessfulWithdrawal() {
        PosTerminal posTerminal = new PosTerminal(bankConnection, receiptPrinter);

        context.checking(new Expectations() { {
            oneOf(bankConnection).getAccountByCardNumber("1000-0000-0001-0003");
            will(returnValue(customerAccount));

            // Other oneOf() will() statements ...

            oneOf(receiptPrinter).print("Successful withdrawal");
        } });

        posTerminal.buyWithCard(100, "1000-0000-0001-0003");
    }
```

mockito

```
@Test
    public void shouldPrintSuccessfulWithdrawal() {
        PosTerminal posTerminal = new PosTerminal(bankConnection, receiptPrinter);

        when(bankConnection.getAccountByCardNumber("1000-0000-0001-0003"))
            .thenReturn(customerAccount);

        // Other when(...).thenReturn(...) calls ...

        posTerminal.buyWithCard(100, "1000-0000-0001-0003");

        verify(receiptPrinter).print("Successful withdrawal");
    }
```

Programa

- Frameworks
  - jMock
  - Mockito
- Diferencias entre jMock y Mockito
  - Creando mocks
  - Verificación de llamadas y valores de retorno
  - Argument matchers
  - Number of invocations
  - Llamadas consecutivas
  - Verificación de orden de llamadas
  - Manejo de excepciones

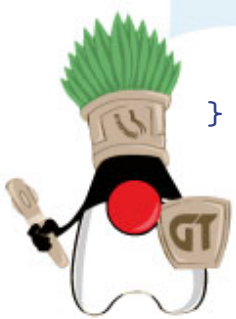
- Uso de any
- Ambos tienen problemas combinando

jMock

```
context.checking(new Expectations() { {
            // Other oneOf() will() statements ...

            oneOf(customerAccount).withdraw(with(any(Integer.class)),
                    with(any(String.class)));
            will(returnValue(true));

            oneOf(shopAccount).enter(with(100), with(any(String.class)));
            will(returnValue(true));

           // Other oneOf() will() statements ...
        } });
```

mockito

```
import static org.mockito.Matchers.anyInt;
import static org.mockito.Matchers.anyString;
import static org.mockito.Matchers.eq;
// Other parts of the code ...
    @Test
    public void shouldPrintSuccessfulWithdrawal() {
       // Other when(...).thenReturn(...) calls ...

        when(customerAccount.withdraw(anyInt(), anyString())).thenReturn(true);
        when(shopAccount.enter(eq(100), anyString())).thenReturn(true);

        // Other when(...).thenReturn(...)  and verify(...) calls ...
    }
```



JAVA DAY 2015 Guatemala

Programa

jMock

```
context.checking(new Expectations() { {
          // Other oneOf() will() statements ...
           exactly(1).of(receiptPrinter).print(with(any(String.class)));
       } });
or
context.checking(new Expectations() { {
          // Other oneOf() will() statements ...
           atLeast(1).of(receiptPrinter).print(with(any(String.class)));
       } });
or
context.checking(new Expectations() { {
          // Other oneOf() will() statements ...
           never(receiptPrinter).print(with(any(String.class)));
       } });
```

mockito

```
// Other when(...).thenReturn(...) calls ...
        verify(receiptPrinter, times(1)).print(anyString());
        // Other verify() calls ...
or
// Other when(...).thenReturn(...) calls ...
        verify(receiptPrinter, atLeast(1)).print(anyString());
        // Other verify() calls ...
or
// Other when(...).thenReturn(...) calls ...
        verify(receiptPrinter, never()).print(anyString());
        // Other verify() calls ...
```

Programa

Frameworks

jMock

Mockito

Diferencias entre jMock y Mockito

Creando mocks

Verificación de llamadas y valores de retorno

Argument matchers

Number of invocations

Llamadas consecutivas

Verificación de orden de llamadas

Manejo de excepciones

jMock

```
context.checking(new Expectations() { {
            // Other oneOf() will() statements ...

            oneOf(shopAccount).enter(with(100), with(any(String.class)));
            will(onConsecutiveCalls(returnValue(true), returnValue(false),
                 returnValue(false)));
        } });
```

mockito

```
// Other when(...).thenReturn(...) calls ...
        when(shopAccount.enter(eq(100), anyString())).thenReturn(true, false, false);
```

Programa

Mockito vs Jmock

- jMock usa sequence
- mockito usa inOrder

jMock

```
@Test
    public void shouldPrintSuccessfulWithdrawal() {
        PosTerminal posTerminal = new PosTerminal(bankConnection, receiptPrinter);

        final Sequence sequence = context.sequence("account order");
        context.checking(new Expectations() { {
            // Other oneOf() will() statements ...

            oneOf(customerAccount).withdraw(with(any(Integer.class)),
                    with(any(String.class)));
            will(returnValue(true));
            inSequence(sequence);

            atLeast(1).of(shopAccount).enter(with(100), with(any(String.class)));
            will(returnValue(true));
            inSequence(sequence);
        } });

        posTerminal.buyWithCard(100, "1000-0000-0001-0003");
    }
```

mockito

```
@Test
    public void shouldPrintSuccessfulWithdrawal() {
        PosTerminal posTerminal = new PosTerminal(bankConnection, receiptPrinter);

        posTerminal.buyWithCard(100, "1000-0000-0001-0003");

        InOrder inOrder = Mockito.inOrder(customerAccount, shopAccount);
        inOrder.verify(customerAccount).withdraw(anyInt(), anyString());
        inOrder.verify(shopAccount).enter(anyInt(), anyString());

        verify(receiptPrinter).print(anyString());
    }
```

Programa

- Frameworks
  - jMock
  - Mockito
- Diferencias entre jMock y Mockito
  - Creando mocks
  - Verificación de llamadas y valores de retorno
  - Argument matchers
  - Number of invocations
  - Llamadas consecutivas
  - Verificación de orden de llamadas
  - Manejo de excepciones

jMock

```
context.checking(new Expectations() { {
            oneOf(bankConnection).getAccountByCardNumber("1000-0000-0001-0003");
            will(throwException(new RuntimeException()));
        } });
```

mockito

```
when(bankConnection.getAccountByCardNumber("1000-0000-0001-0003"))
          .thenThrow(new RuntimeException());
```

DEMO

¡Gracias!

JAVA DAY 2015 Guatemala