Creación de Mock
+ **mock**(ClaseSimulada.class);

Patrones de Clase de Equivalencia
+ anyBoolean(), anyByte(), anyChar(), anyDouble(), anyFloat(), anyInt(), anyLong(), anyShort(), anyString().
+ anyCollection(), anyList(), anyMap(), anyObject().
+ booleanThat(), byteThat(), charThat(), doubleThat(), floatThat(), intThat(), longThat(), shortThat().
+ argThat(), matches().
+ contains(), endsWith(), eq(), refEq(), same(), startsWith().
+ isA(), isNotNull(), isNull(), notNull().

Descripción del comportamiento
Sigue el modelo: when().thenXXX()
+ **when**(clasemockeada.SuMetodo(Parámetros con patrones o valores)).**then**Return(ValoresRespuesta)
* Clausula WHEN
1.- Admite como parámetro el método invocado con "Patrones de Clase de Equivalencia"
2.- Admite como parámetro el método invocado con valores concretos
3.- Para combinar ambos es necesario usar el modelo de patrones con el patrón eq(valor concreto)
* Clausula THEN.
1.- Admite varias posibilidades
A.- **thenReturn**(Respuesta concreta)
B.- **thenReturn**(Lista de respuestas). Equivalente a 2.A
C.- **thenThrow**(Excepción a lanzar)
D.- **then**(Answer). Respuesta programada por el usuario.
E.- **thenCallRealMetothod**(). llama a un método real combinado con mock/spy.
2.- No tiene porque ser única se pueden encadenar n then.then.then.
A.- Cada llamada al método moqueado del que se describe el comportamiento irá dando por orden las respuestas indicadas, una vez se alcanza la última respuesta ésta queda repitiéndose indefinidamente.
B.- Permite encadenar combinados los distintos then. Es posible thenReturn().thenThrow().thenReturn().

Descripción Alternativa de comportamientos
Sique el modelo doXXX.when. Su necesidad está relacionada con comportamientos infrecuentes y el uso de spy
+ **doReturn**(objetoRetornado)
+ **doThrow**(excepcion)
+ **doAnswer**(answer)
+ **doNothing**()

Modos de verificación
Todos están declarados como métodos estáticos.
atLeast(int minNumeroDeInvocaciones)
atLeastOnce()
atMost(int maxNumeroDeInvocaciones)
never()
times(int numeroRequeridoDeInvocaciones)

Verificación

Sigue el modelo: Verify(ClaseSimulada, [Modo de verificación]).MetodoInvocado(Parámetros con patrones o valores)

+ Si no se incluye Patron de verificación. Se verifica simplemente si es invocada
+ Es posible comprobar que una serie de invocaciones se realizan con una determinada secuencia para ello se añade inOrder antes de verify
  * Se pone. Comprueba que ambos métodos han sido invocados
    verify(Clase).metodoInvocado1()
    verify(Clase).metodoInvocado2()
  * Se pone. Comprueba que los métodos han sido invocados exactamente en ese orden
    inOrder.verify(Clase).metodoInvocado1()
    inOrder.verify(Clase).metodoInvocado2()

Otras verificaciones

+ **verifyNoMoreInteractions**(ClaseSimulada). Verifica que no se han realizado mas interacciones que las comprobadas en la prueba
+ **verifyZeroIneractions**(ClaseSimulada). Verifica que no ha habido ninguna interacción con la ClaseSimulada

Más información:

http://docs.mockito.googlecode.com/hg/latest/org/mockito/Mockito.html