Ch2 Code Unit Testing

Write Code to Test Code(4)

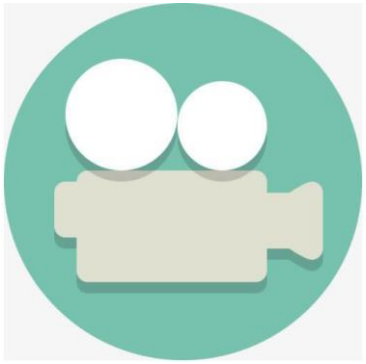
Instructor: Haiying SUN

E-mail: hysun@sei.ecnu.edu.cn

Office: ECNU Science Build B1104

Available Time: Wednesday 8:00 -12:00 a.m.

Practice



- 实验任务
  - 阅读MeetHereMaven项目中UpcomingReservationNotifier和SmtpMessageSender的代码，理解预约提醒功能run()的实现逻辑
  - 设计测试用例验证UpcomingReservationNotifier实现的预约提醒功能

10-Mockito Foundations

```
@Test
void notifyReservation(){
    //arrange
    UpcomingReservationNotifier notifier =
            new UpcomingReservationNotifier(new MeetCalendar(),new SmtpMessageSender());
    //action
    notifier.run();
    //assert???
}
```

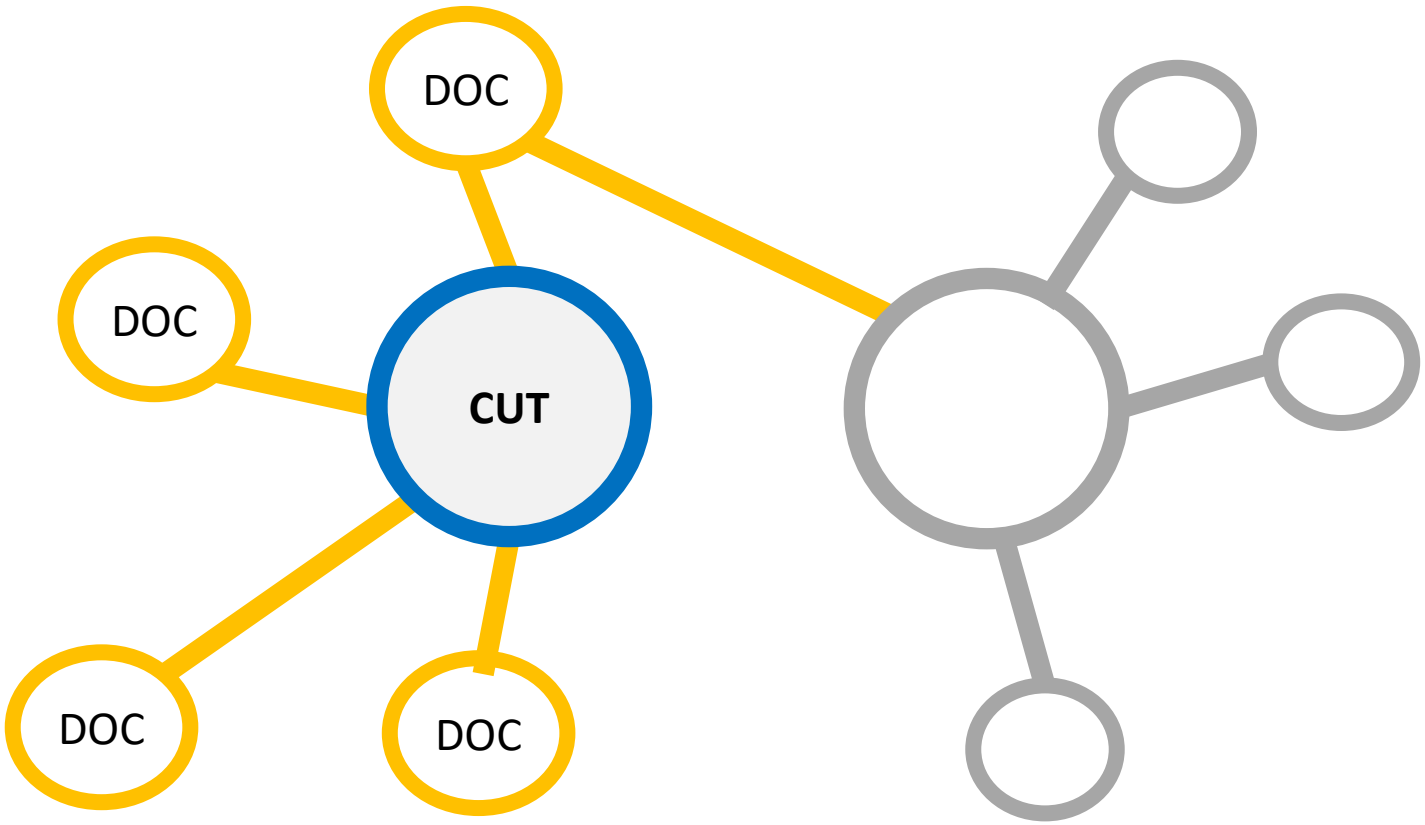
Practice

- 实验准备
  1. 课程网站下载**MockitoDemo.zip**： 视频资料→ JUnit + mockito视频
  2. 解压**MockitoDemo.zip**到当前目录（**目录中不要有中文**）
  3. Idea中打开**MockitoDemo.zip注意JDK版本**）
- 实验任务
  - 阅读PhoneBookDAO.java，PhoneBookH2DAO.java, PhoneEntry.java
  - 编写测试用例测试PhoneBookH2DAO的create（）

Difficulties

DOC Dependence On Component

CUT Code Under Test



Difficulties

测试断言写什么？

测试断言写什么？

测试断言写什么？

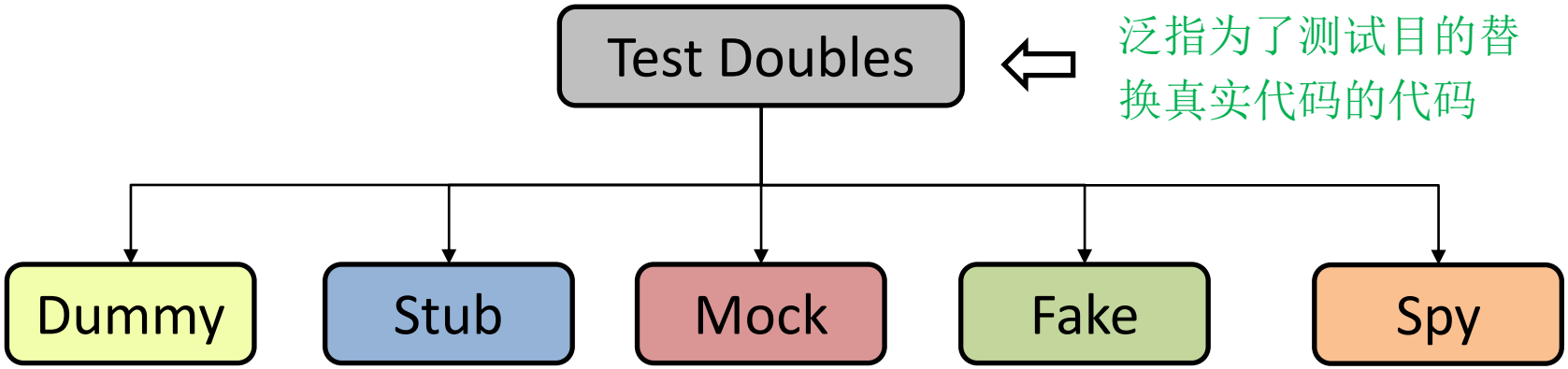
Overview

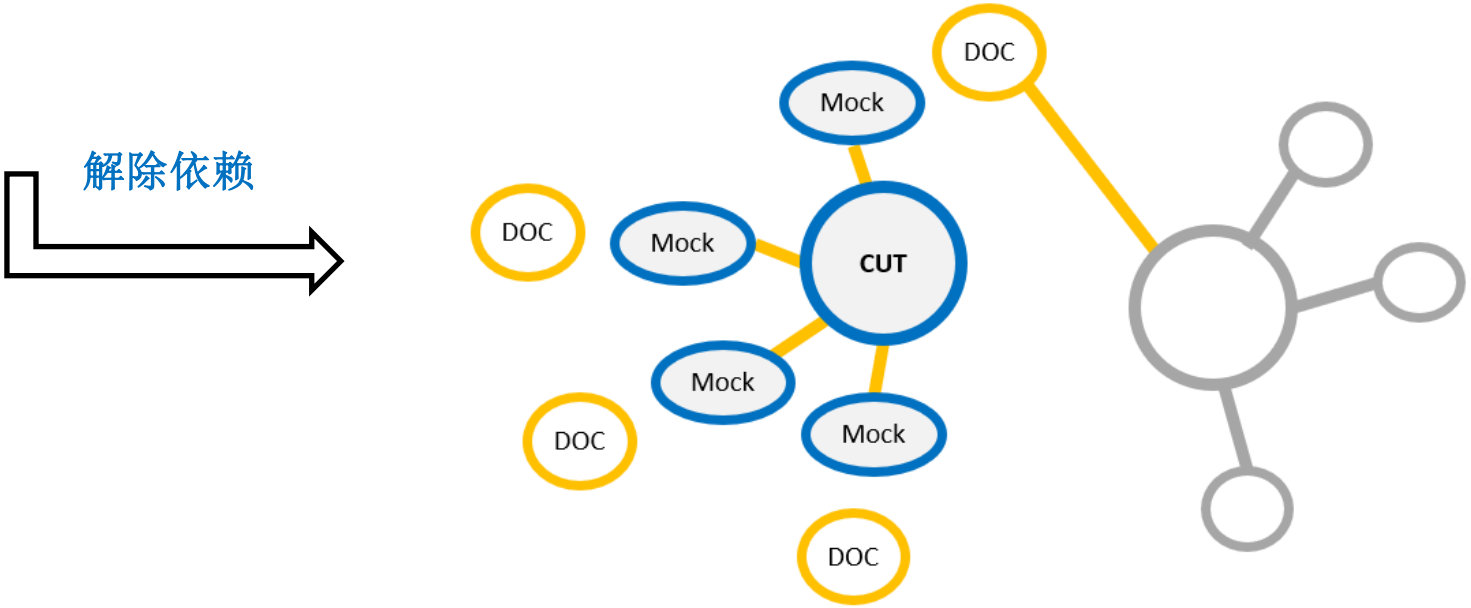
- Test Doubles
- Behavior based test
- Test Framework
  - Mockito

Overview

- Test Doubles
- Behavior based test
- Test Framework
  - Mockito

Test Doubles





Test Doubles

- Dummy
  - 在测试方法中不使用其任何方法的测试替身
  - 一般出现在方法的参数处
  - 通常为了防止NullPointerException的出现，保证测试方法可以顺利执行

```
class DummyDemoTest {
    @Test
    void s1_is_null_return_null() {
        DummyDemo dd = new DummyDemo();
        PhoneEntry entry1 = null;
        PhoneEntry entry2 = new PhoneEntry();
        assertNull(dd.replaceEntryName(entry1,entry2));
    }
}
```

A dummy

MockitoDemo项目DummyDemoTest示例

Test Double

- Fake
  - 一种简化真实代码的测试替身，通常采用继承被Fake对象（真实代码），成为其子类的方法实现
  - 不能作为产品代码，单纯为了测试快，不在测试中出现耗时行为
  - Example: In-memory database

```
class PhoneBookH2DaoFake extends PhoneBookH2Dao {
    protected void loadDriver() {
    }

    protected Connection getConnection() throws SQLException {
        return connection;
    }
}
```

MockitoDemo项目PhoneBookDatabaseDaoTest示例

Test Doubles

- Stub
  - Provide canned answer：为其调用者提供测试过程中需要使用的信息
  - 通常应用响应待测系统的请求，然后返回特定的值

```
// You can mock concrete classes and interfaces
TrainSeats seats = mock(TrainSeats.class);

// stubbing appears before the actual execution
when(seats.book(Seat.near(WINDOW).in(FIRST_CLASS))).thenReturn(BOOKED);
```

Mockito中的打桩代码

Test Doubles

- spy
  - Spy are stubs that also record some information based on how they were called
    1. 使用真实代码的测试替身，**返回其真实值**
    2. 可以打桩
    3. 可以记录使用轨迹，便于在后续的测试活动中验证是不是安排的事情按照期望发生

```java
package edu.ecnu.sei.mockito.trading;

public class DemoClass {

    public String foo() {

        return "I like mock";

    }
}
```

```java
package edu.ecnu.sei.mockito.trading;

import static org.mockito.Mockito.spy;

import org.junit.Test;

public class FirstSpy {

    DemoClass demo = spy(DemoClass.class);

    @Test
    public void what_is_a_mock() {

        System.out.println(demo.foo());
    }
}
```

Output: I like mock

```java
package edu.ecnu.sei.mockito.trading;

public class DemoClass {

    public String foo() {

        return "I like mock";

    }
}
```

```java
package edu.ecnu.sei.mockito.trading;

import static org.mockito.Mockito.mock;

import org.junit.Test;

public class FirstMock {

    DemoClass demo = mock(DemoClass.class);

    @Test
    public void what_is_a_mock() {

        System.out.println(demo.foo());
    }
}
```

Output: null

Test Double

- Mock
  - 按照期望实现的用于测试方法中的行为代码

  1. 正常路径：返回正常值
  2. 异常路径：返回期望的错误/异常

- Mockito
  - mock被测对象外部依赖的Java开源测试框架

Typical Scenario for Mock Object

- objects supplies nondeterministic results
  1. maximum/minimum value
  2. random result
  3. current time
- objects difficult to create or reproduce
  1. Network error
- objects not yet exist or may change behavior.
  1. want database query return same result

Overview

- Test Doubles
- Behavior based test
- Test Framework
  - Mockito

Two Approaches of Test Verdict Construction

- State based testing (Test by Result)
  - determine whether the CUT worked correctly by examining the state of the SUT and its collaborators after the method was exercised
- Behavioral based testing (Test by Process)
  - determine whether the CUT worked correctly by examining its action process

Example

- 实验任务

阅读MeetHereMaven项目

UpcomingReservationNotifierTest 的

1、test_notify_Reservation_with_state()

2、test_notify_Reservation_with_behavior()

Overview

- Test Doubles
- Behavior based test
- Test Framework
  - Mockito

Mockito Foundation

- Mocking Class/Interface
- Stubbing Method
- Verify Invocation
- Argument Match & Capture

Basic Usage Example

```
// You can mock concrete classes and interfaces
TrainSeats seats = mock(TrainSeats.class);

// stubbing appears before the actual execution
when(seats.book(Seat.near(WINDOW).in(FIRST_CLASS))).thenReturn(BOOKED);

// the following prints "BOOKED"
System.out.println(seats.book(Seat.near(WINDOW).in(FIRST_CLASS)));

// the following prints "null" because
// .book(Seat.near(AISLE).in(FIRST_CLASS))) was not stubbed
System.out.println(seats.book(Seat.near(AISLE).in(FIRST_CLASS)));

// the following verification passes because
// .book(Seat.near(WINDOW).in(FIRST_CLASS)) has been invoked
verify(seats).book(Seat.near(WINDOW).in(FIRST_CLASS));

// the following verification fails because
// .book(Seat.in(SECOND_CLASS)) has not been invoked
verify(seats).book(Seat.in(SECOND_CLASS));
```

Mockito Foundation

- Mocking Class/Interface
- Stubbing Method
- Verify Invocation
- Argument Match & Capture

Create Mock Object

- Create mock object : **mock()**

```
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>3.12.4</version>
    <scope>test</scope>
</dependency>
```

```
SmtpMessageSender sender =
                    mock(SmtpMessageSender.class);
```

Create Mock Object

- Create mock object : **@Mock**

```
<!-- https://mvnrepository.com/artifact/org.mockito/mo
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>3.12.4</version>
    <scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.mockito/mo
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-junit-jupiter</artifactId>
    <version>3.12.4</version>
    <scope>test</scope>
</dependency>
```

Create Mock Object

- Create mock object : **@Mock**

```
@ExtendWith(MockitoExtension.class)
class UpcomingReservationNotifierTest {
    @Mock
    SmtpMessageSender sender;
    @Test
    void checkEmailContent() {
        //....
    }
```

Mockito Foundation

- Mocking Class/Interface
- Stubbing Method
- Verify Invocation
- Argument Match & Capture

Stubbing Method

- Defines the behavior of a mock method when the method is invoked
  1. value to be returned
  2. the exception to be thrown.
- Method with return value
- Method without return value

Stubbing Method

- Method with return value
  1. when(mockObject.method()). thenReturn(testNeededValue)
  2. when(mockObject.method()). thenThrow(Throwable)
  3. when(mockObject.method()).thenAnswer(Answer answer)
  4. when(mockObject.method()).*thenCallRealMethod()*

When Example

```
//You can mock concrete classes, not just interfaces
LinkedList mockedList = mock(LinkedList.class);

//stubbing
when(mockedList.get(0)).thenReturn("first");
when(mockedList.get(1)).thenThrow(new RuntimeException());

//following prints "first"
System.out.println(mockedList.get(0));

//following throws runtime exception
System.out.println(mockedList.get(1));

//following prints "null" because get(999) was not stubbed
System.out.println(mockedList.get(999));
```

thenAnswer

- 根据调用该方法的参数动态地返回与之匹配的值（on the fly value）

- Answer接口：

```
public interface Answer<T>{
  T answer(InvocationOnMock invocation) throws Throwable;
}
```

- InvocationOnMock
  - Object[] args = invocation.getArguments();
  - Object mock = invocation.getMock();

Stubbing Method

- Method without return value
  1. doReturn(Object). when(mockObject.method())
  2. doThrow(Throwable).when(mockObject.method())
  2. doNothing().when(mockObject.method())
  3. doAnswer().when(mockObject.method())
  4. doCallRealMethod().when(mockObject.method())

Mockito Foundation

- Mocking Class/Interface
- Stubbing Method
- Verify Invocation
- Argument Match & Capture

Mockito Foundation

- Verify Invocation
  - 在面向对象编程里，很多情况下方法实现的主体是一系列调用其它对象相应方法的代码，此时该如何实现测试断言呢？
- verify()用于验证方法的调用是否符合预期
  1. 是否调用了正确的方法，包括方法名和参数
  2. 调用次数是否正确
  3. 方法不应被调用
  4. 调用顺序是否符合要求

Verify Demo

```java
public class UpcomingReservationNotifier {
    private MeetCalendar calendar;
    SmtpMessageSender notifier;

    public UpcomingReservationNotifier(MeetCalendar calendar,SmtpMessageSender notifier) {
        this.calendar = calendar;
        this.notifier = notifier;
    }

    public void run(){
        for(UserReservation reservation:calendar.getReservations()){
            String email = "user@foo.com";
            //System.out.println(buildmail(reservation));
            notifier.sendNotification( subject: "预约提醒",buildmail(reservation),email);
        }
    }
```

Verify Demo

```
@Test
void checkEmailContent() {
    MeetCalendar meet = new MeetCalendar();
    meet.addReservation( userName: "sun", siteKey: "gymb1", dateTime: "2019-09-20 18:00");

    SmtpMessageSender sender = mock(SmtpMessageSender.class);
    when(sender.sendNotification( subject: "预约提醒",
            body: "2019-09-20 18:00您预约了体育馆1号篮球场地", address: "user@foo.com"))
            .thenReturn("提醒邮件发送成功");

    UpcomingReservationNotifier notifier = new UpcomingReservationNotifier(meet,sender);
    notifier.run();
    verify(sender).sendNotification(anyString(),anyString(),anyString());
    verify(sender,times( wantedNumberOfInvocations: 1))
            .sendNotification(anyString(),anyString(),anyString());
}
```

Verify Invocation Times

- Verify in Depth
  - times(int wantedNumberOfInvocations)
  - never()
  - atLeatOnce()
  - atLeast(int minNumberOfInvocations)
  - atMost(int maxNumberOfInvocations)：
  - only(): 期望mock对象只调用指定方法，调用该mock对象的除期望方法以外的任何其它方法，测试失败

Verify Invocation

- verifying zero and no more interactions
  - verifyZeroInteractions(Object mocks)
    - 验证在mock对象上没有发生任何调用
  - verifyNoMoreInteractions(Object mocks)
    - 验证在mock对象上没有过多的调用，即期望mock对象上在测试方法中的所有调用都被验证
    - 请注意，当设置调用次数times时的使用

Verify Invocation

```
73      @Test
74      public void no_more_interactions_are_invoked_on_mock_objects() {
75
76          Stock noStock= null;
77          portfolio.getAvgPrice(noStock);
78
79          portfolio.sell(noStock, 0);
80
81          verify(portfolio).getAvgPrice(eq(noStock));
82          verifyNoMoreInteractions(portfolio);
83      }
```

测试失败！ 没有验证对portfolio的sell方法的调用

Verify Invocation

```
73    @Test
74    public void no_more_interactions_are_invoked_on_mock_objects() {
75
76        Stock noStock= null;
77        portfolio.getAvgPrice(noStock);
78        portfolio.sell(noStock, 0);
79        verify(portfolio).getAvgPrice(eq(noStock));
80        verify(portfolio).sell(noStock, 0);
81        verifyNoMoreInteractions(portfolio);
82    }
```

测试通过！ 增加了portfolio的sell方法调用的验证

Verify Invocation

```
73      @Test
74      public void no_more_interactions_are_invoked_on_mock_objects() {
75
76          Stock noStock= null;
77          portfolio.getAvgPrice(noStock);
78          portfolio.sell(noStock, 0);
79          verify(portfolio).getAvgPrice(eq(noStock));
80
81          verifyNoMoreInteractions(portfolio);
82
83          verify(portfolio).sell(noStock, 0);
84      }
```

测试失败！ 顺序不正确，81行代码应该在83行之后

Verify Invocation

```
73      @Test
74      public void no_more_interactions_are_invoked_on_mock_objects() {
75
76          Stock noStock= null;
77          portfolio.getAvgPrice(noStock);
78          portfolio.getAvgPrice(noStock);
79
80          verify(portfolio).getAvgPrice(eq(noStock));
81
82          verifyNoMoreInteractions(portfolio);
83      }
```

测试失败！不是因为少写了一个verify（portfolio).getAvgPrice...，而是因为测试80行代码说得是期望getAvgPrice被调用1次，因为verify方法中不写times，表示times的缺省值是1

Verify Invocation

```
73  @Test
74  public void no_more_interactions_are_invoked_on_mock_objects() {
75
76      Stock noStock= null;
77      portfolio.getAvgPrice(noStock);
78      portfolio.getAvgPrice(noStock);
79
80      verify(portfolio,times(2)).getAvgPrice(eq(noStock));
81      verifyNoMoreInteractions(portfolio);
82  }
```

测试通过！验证次数改为了2次

Verify Invocation Order

- 交互顺序是否正确也是判断被测对象行为是否正确的标志
- InOrder类
  - 验证指定mock对象的方法的交互顺序是否满足期望

Verify Invocation Order

```
@Test
public void ArgumentOrder() {

    when(mockedList.add(anyInt())).thenReturn(true);
    when(mockedList.get(anyInt())).thenReturn("the argument of add is a integer");

    mockedList.get(10);
    mockedList.add(10);

    InOrder order = inOrder(mockedList);

    order.verify(mockedList).add(anyInt());
    order.verify(mockedList).get(anyInt());
}
```

测试失败！Order里指出add应该在get之前调用，但是mockedList.get(10)在mockedlist.add(10)之前调用

Verify Invocation Order

```
@Test
public void ArgumentOrder() {

    when(mockedList.add(anyInt())).thenReturn(true);
    when(mockedList.get(anyInt())).thenReturn("the argument of add is a integer");

    mockedList.get(10);
    mockedList.add(10);
    mockedList.get(10);

    InOrder order = inOrder(mockedList);

    order.verify(mockedList).add(anyInt());
    order.verify(mockedList).get(anyInt());
}
```

测试结果？？

Stubbing consecutive calls

- 支持builder pattern构造打桩链
  - when/thenXXXX
  - doXXXX

```
152     @Test
153     public void consecutive_calls() throws Exception{
154
155         Stock stock = new Stock(null,null,null);
156         when(portfolio.getAvgPrice(stock)).thenReturn(BigDecimal.TEN,BigDecimal.ONE);
157         Assert.assertEquals(BigDecimal.TEN, portfolio.getAvgPrice(stock));
158         Assert.assertEquals(BigDecimal.ONE, portfolio.getAvgPrice(stock));
159         Assert.assertEquals(BigDecimal.ONE, portfolio.getAvgPrice(stock));
160         Assert.assertEquals(BigDecimal.ONE, portfolio.getAvgPrice(stock));
161     }
```

打桩语句：第一次调用getAvgPrice时，返回BigDecimal.TEN，第二次调用及以后调用getAvgPrice时，返回BigDecimal.ONE

Stubbing consecutive calls

```
164     @Test
165     public void consecutive_calls_other_formal() throws Exception{
166
167         Stock stock = new Stock(null,null,null);
168         when(portfolio.getAvgPrice(stock)).thenReturn(BigDecimal.TEN).thenReturn(BigDecimal.ONE);
169         Assert.assertEquals(BigDecimal.TEN, portfolio.getAvgPrice(stock));
170         Assert.assertEquals(BigDecimal.ONE, portfolio.getAvgPrice(stock));
171         Assert.assertEquals(BigDecimal.ONE, portfolio.getAvgPrice(stock));
172         Assert.assertEquals(BigDecimal.ONE, portfolio.getAvgPrice(stock));
173     }
```

另一种连续打桩方式

Stubbing consecutive calls

- 区别打桩链和多次独立调用打桩语句when/thenReturn

```
175     @Test
176     public void may_be_confused_with_consecutive_calls() throws Exception{
177
178         Stock stock = new Stock(null,null,null);
179         when(portfolio.getAvgPrice(stock)).thenReturn(BigDecimal.TEN);
180         when(portfolio.getAvgPrice(stock)).thenReturn(BigDecimal.ONE);
181         Assert.assertEquals(BigDecimal.TEN, portfolio.getAvgPrice(stock));
182         Assert.assertEquals(BigDecimal.ONE, portfolio.getAvgPrice(stock));
183         Assert.assertEquals(BigDecimal.ONE, portfolio.getAvgPrice(stock));
184         Assert.assertEquals(BigDecimal.ONE, portfolio.getAvgPrice(stock));
185     }
```

测试失败：打桩语句使用的方法，方法的参数完全一致的话，后续的打桩语句会覆盖前面的打桩语句

Mockito Foundation

- Mocking Class/Interface
- Stubbing Method
- Verify Invocation
- Argument Match & Capture

Using Argument Matcher

- Argument Matcher
  - 为打桩的方法执行参数匹配，确定打桩方法应该返回的测试数据
  - when(mockObject.method(**ArgumentMatcher**)).theReturn(testValue)
  - org.hamcrest.BaseMatcher的子类
  - 内建匹配器：anyInt(), anyDouble(), anyString(), anyList()和 anyCollection(), isA(T), any(T), eq(T), eq(primitive type)
- **注意**: 如果被打桩方法的一个参数使用了参数匹配器的话，那么该方法的所有参数都需要使用参数匹配器，否则测试失败
- 在verfiy中同样可以使用Argument匹配器

Using Argument Matcher

```
66      @Test
67      public void when_ten_percent_gain_then_the_stock_is_sold() {
68
69          Stock  seiStock = new Stock("SEI","ECNU",new BigDecimal("50.00"));
70          Stock  stStock = new Stock("ST","ECNU",new BigDecimal("44.00"));
71
72          //stubbing the getQuote method
73          when(marketWatcher.getQuote(eq("SEI"))).thenReturn(seiStock);
74          when(marketWatcher.getQuote(eq("ST"))).thenReturn(stStock);
75
76          //stubbing the getAvgPrice method so that the percent gained is morn than 10%
77          when(portfolio.getAvgPrice(eq(seiStock))).thenReturn(new BigDecimal("40.00"));
78          when(portfolio.getAvgPrice(eq(stStock))).thenReturn(new BigDecimal("40.00"));
79
80          StockBroker broker = new StockBroker(marketWatcher);
81
82          broker.perform(portfolio, seiStock);
83          verify(portfolio).sell(seiStock,10);
84
85          broker.perform(portfolio, stStock);
86          verify(portfolio).sell(stStock,10);
87      }
```

Using Argument Matcher

```
344     @Test
345     public void argument_matcher_demo() {
346
347         Stock  seiStock = new Stock("SEI","ECNU",new BigDecimal("50.00"));
348
349         //stubbing the getQuote method
350         when(marketWatcher.getQuote(eq("SEI"))).thenReturn(seiStock);
351
352
353         //stubbing the getAvgPrice method so that the percent gained is morn than 10%
354         when(portfolio.getAvgPrice(eq(seiStock))).thenReturn(new BigDecimal("40.00"));
355
356         StockBroker broker = new StockBroker(marketWatcher);
357
358         broker.perform(portfolio, seiStock);
359         verify(portfolio).sell(eq(seiStock),10);
360
361     }
```

verify(portfolio).**sell(eq(seiStock),eq(10))**

Using Argument Matcher

- 自定义参数匹配器
  1. 通过实现ArgumentMatcher接口的matches()
  2. 在打桩方法时, 通过argThat使用自定义的参数匹配器

Using Argument Matcher

```
4   import org.mockito.ArgumentMatcher;
5
6   public class BlueChipStockMatcher implements ArgumentMatcher<String>{
7
8       @Override
9       public boolean matches(String symbol) {
10          return "SEI".equals(symbol)||"ST".equals(symbol);
11      }
12
13  }
```

```
3   public class OtherStockMatcher extends BlueChipStockMatcher{
4
5       @Override
6       public boolean matches(String symbol) {
7           return !super.matches(symbol);
8       }
9   }
```

Using Argument Matcher

```
330  @Test
331   public void customized_argument_matcher_demo() {
332
333       Stock seiStock = new Stock("SEI","ECNU",new BigDecimal("10.00"));
334       Stock otherStock = new Stock("XY", "XY Corp", new BigDecimal("5.00"));
335
336       when(marketWatcher.getQuote(argThat(new BlueChipStockMatcher()))).thenReturn(seiStock);
337
338       when(portfolio.getAvgPrice(isA(Stock.class))).thenReturn(new BigDecimal("10.00"));
339
340       StockBroker broker = new StockBroker(marketWatcher);
341       broker.perform(portfolio, seiStock);
342   }
```

ArgumentCaptor

- 获取when/thenReturn的参数信息或者verify的参数信息

```
365     @Test
366     public void argument_captor_demo() throws Exception {
367
368         Stock seiStock = new Stock("SEI", "ECNU", new BigDecimal("11.20"));
369
370         when(marketWatcher.getQuote(anyString())).thenReturn(seiStock);
371         when(portfolio.getAvgPrice(isA(Stock.class))).thenReturn(new BigDecimal("10.00"));
372
373         StockBroker broker = new StockBroker(marketWatcher);
374
375         broker.perform(portfolio, seiStock);
376
377         ArgumentCaptor<String> stockIdCaptor = ArgumentCaptor.forClass(String.class);
378
379         verify(marketWatcher).getQuote(stockIdCaptor.capture());
380         assertEquals("SEI", stockIdCaptor.getValue());
381
382          ArgumentCaptor<Stock>  stockCaptor = ArgumentCaptor.forClass(Stock.class);
383          ArgumentCaptor<Integer> stockSellCountCaptor = ArgumentCaptor.forClass(Integer.class);
384
385          verify(portfolio).sell(stockCaptor.capture(), stockSellCountCaptor.capture());
386          assertEquals("SEI", stockCaptor.getValue().getSymbol());
387          assertEquals(10, stockSellCountCaptor.getValue().intValue());
388     }
```

The End