Subs and Mocks clarification

Meszaros uses the term **Test Double** as the generic term for any kind of pretend object used in place of a real object for testing purposes. The name comes from the notion of a Stunt Double in movies. (One of his aims was to avoid using any name that was already widely used.) Meszaros then defined five particular kinds of double:

- **Dummy** objects are passed around but never actually used. Usually they are just used to fill parameter lists.
- **Fake** objects actually have working implementations, but usually take some shortcut which makes them not suitable for production (an in memory database is a good example).
- **Stubs** provide canned answers to calls made during the test, usually not responding at all to anything outside what's programmed in for the test.
- **Spies** are stubs that also record some information based on how they were called. One form of this might be an email service that records how many messages it was sent.
- **Mocks** are what we are talking about here: objects pre-programmed with expectations which form a specification of the calls they are expected to receive.

Java tests with Jmock library

```
public class OrderInteractionTester extends MockObjectTestCase {
  private static String TALISKER = "Talisker";

  public void testFillingRemovesInventoryIfInStock() {
    //setup - data
    Order order = new Order(TALISKER, 50);
    Mock warehouseMock = new Mock(Warehouse.class);

    //setup - expectations
    warehouseMock.expects(once()).method("hasInventory")
      .with(eq(TALISKER),eq(50))
      .will(returnValue(true));
    warehouseMock.expects(once()).method("remove")
      .with(eq(TALISKER), eq(50))
      .after("hasInventory");

    //exercise
    order.fill((Warehouse) warehouseMock.proxy());

    //verify
    warehouseMock.verify();
    assertTrue(order.isFilled());
  }

  public void testFillingDoesNotRemoveIfNotEnoughInStock() {
    Order order = new Order(TALISKER, 51);
    Mock warehouse = mock(Warehouse.class);

    warehouse.expects(once()).method("hasInventory")
      .withAnyArguments()
      .will(returnValue(false));

    order.fill((Warehouse) warehouse.proxy());

    assertFalse(order.isFilled());
  }
```

Java tests with EasyMock library

```
public class OrderEasyTester extends TestCase {
  private static String TALISKER = "Talisker";

  private MockControl warehouseControl;
  private Warehouse warehouseMock;

  public void setUp() {
    warehouseControl = MockControl.createControl(Warehouse.class);
    warehouseMock = (Warehouse) warehouseControl.getMock();
  }

  public void testFillingRemovesInventoryIfInStock() {
    //setup - data
    Order order = new Order(TALISKER, 50);

    //setup - expectations
    warehouseMock.hasInventory(TALISKER, 50);
    warehouseControl.setReturnValue(true);
    warehouseMock.remove(TALISKER, 50);
    warehouseControl.replay();

    //exercise
    order.fill(warehouseMock);

    //verify
    warehouseControl.verify();
    assertTrue(order.isFilled());
  }

  public void testFillingDoesNotRemoveIfNotEnoughInStock() {
    Order order = new Order(TALISKER, 51);

    warehouseMock.hasInventory(TALISKER, 51);
    warehouseControl.setReturnValue(false);
    warehouseControl.replay();

    order.fill((Warehouse) warehouseMock);

    assertFalse(order.isFilled());
    warehouseControl.verify();
  }
}
```

Java tests with Stubs

```
public interface MailService {
  public void send (Message msg);
}
public class MailServiceStub implements MailService {
  private List<Message> messages = new ArrayList<Message>();
  public void send (Message msg) {
    messages.add(msg);
  }
  public int numberSent() {
    return messages.size();
  }
}
```

We can then use state verification on the stub like this.

class OrderStateTester...

```
  public void testOrderSendsMailIfUnfilled() {
    Order order = new Order(TALISKER, 51);
    MailServiceStub mailer = new MailServiceStub();
    order.setMailer(mailer);
    order.fill(warehouse);
    assertEquals(1, mailer.numberSent());
  }
```

Using mocks this test would look quite different.

class OrderInteractionTester...

```
  public void testOrderSendsMailIfUnfilled() {
    Order order = new Order(TALISKER, 51);
    Mock warehouse = mock(Warehouse.class);
    Mock mailer = mock(MailService.class);
    order.setMailer((MailService) mailer.proxy());

    mailer.expects(once()).method("send");
    warehouse.expects(once()).method("hasInventory")
      .withAnyArguments()
      .will(returnValue(false));

    order.fill((Warehouse) warehouse.proxy());
  }
}
```

https://martinfowler.com/articles/mocksArentStubs.html - you can find comprehensive article about all tests doubles here