#### NMock 2.0 (RC1) Cheat Sheet

|  |  |
| --- | --- |
| **Setting Up** | |
| Create a new Mockery and use it to create mocks for all the interfaces you're interested in.  **Mockery mocks = new Mockery();**  **InterfaceToBeMocked aMock = (InterfaceToBeMocked)**  **mocks.NewMock(typeof(InterfaceToBeMocked));**  If you're using .NET 2.0, there's a generic NewMock() method so you don't need the cast:  **Mockery mocks = new Mockery();**  **InterfaceToBeMocked aMock = mocks.NewMock<InterfaceToBeMocked>();** | |
| **Defining Basic Expectations** | |
| Expect.Once.On(aMock)  .Method( ... )  .With( ... )  .Will(Return.Value( ... )); | Expect.Once.On(aMock)  .Method( ... )  .With( ... )  .Will(Throw.Exception( ... )); |
| **Defining Expectations on Properties** | |
| Expect.Once.On(aMock)  .GetProperty( ... )  .Will(Return.Value( ... ); | Expect.Once.On(aMock)  .SetProperty( ... )  .To( ... ); |
| **Stubs** | **Constraining Order** |
| Expect can be replaced with Stub which essentially means “zero or more”. Behavior of the stub will be defined and invoked if called, but the stub will not cause the test to fail.  **Stub**.On(aMock)  .Method( ... )  .With( ... )  .Will(Return.Value( ... ); | Mocks by default can be in any order. To constrain the order of a set of expectations, wrap the expectations with a using block.  using (mocks.Ordered)  {  Expect.Once.On( ...  Expect.Once.On( ...  } |
| **Possible Method Call Expectations** | |
| Expect**.Once** Expect**.Never** Expect**.AtLeastOnce**  Expect**.AtLeast(<# times>)** Expect**.AtMost(<# times>)** Expect**.Exactly(<# times>)**  Expect**.Between(<# times>, <# times>)** | |
| **Arguments** | |
| **.With(...)** can be replaced with **.WithAnyArguments()** or **.WithNoArguments()** | |
| **Verification** | |
| Mockery mocks = new Mockery();  ...  mocks.VerifyAllExpectationsHaveBeenMet(); | |