Mockito
Cheat Sheet

SOURCE ARTISTS

```
public class SutTest{
```

Initialize Mockito engine

```
@BeforeEach
public void init(){
   MockitoAnnotations.initMocks(this);
}
```

Declare mocks and inject into SUT

```
@Mock
private Service serviceStub;

@Mock
private Controller controllerMock;

@InjectMocks
private Sut sut;
```

Set-up and verify mocks

```
@Test
public void shouldTestThis(){
      // Arrange
      when(serviceStub.getUser())
            .thenReturn(testUser);

      // Act
      sut.execute("test");

      // Assert
      verify(controllerMock)
            .callRemote(testUser);
}
```

Set-up behaviour

```
/* set-up method on a mock */
when(userService.getUser(firstName
  , lastName).thenReturn(expectedUser);

/* set-up an exception to be thrown */
when(userService.getUser(firstName
  , lastName).thenReturn(expectedUser);

/* set-up method on a spy */
doReturn(invoiceService.getUser(firstName
  , lastName).thenReturn(expectedInvoice);

/* set-up a dummy method  on spy */
doNothing().when(invoiceService)
  .saveInvoice(expectedInvoice);

/* set-up method with wildcards */
when(userService.getUserByTitleAndAge(
  anyString(), anyInt()).thenReturn(user);

/* set-up method with a mix of
  wildcards and real values */
when(userService.getUserByTitleAndAge(
  anyString(), eq(25)).thenReturn(user);

/* set-up dynamic behaviour */
when(userService.getUserById(id))
  .thenAnswer((invocation) ->{
    int id = invocation.getArgument(0);
    if(id.equals(13){ return luckyUser; }
    else { return commonUser; }
});
```

Verify behaviour

```
/* verify nothing happened */
verify(userService, never())
     .saveUser(any(User.class));

/* verify method called  n times*/
verify(userService, times(5))
     .refresh(any(User.class));

/* verify methods invoked once in order*/
InOrder inOrder = inOrder(stub, mock);

inOrder.verify(stub).getUser(id);
inOrder.verify(mock).saveUser(user);

/* capture and verify arguments */
@ArgumentCaptor
private ArgumentCaptor<User> userCaptor;

verify(userService).saveUser(
   userCaptor.capture());

assertThat(userCaptor.getValue(),
   equalTo(luckyUser));
```

Remember to:

```
import static org.mockito.Mockito.*;
import static org.mockito.ArgumentMatchers.*;
```