Testing with Mocks

Mocks

- Mock object is a pseudo object, replacing the dependency we are not interested in testing and helping to isolate the object in which we are interested
- Mock code does not have to be written, mock objects are generated using third-party libraries
- Mock object will implement the dependent interface on the fly
- Developer can configure the behavior: set which methods will be called and what will they return. Then expectations can be checked in order to decide the test result.

Mocking Frameworks

- EasyMock provides an easy way to replace collaborators of the unit under test.
- jMock is a small library, and the team that created it brags about making it quick and easy to define mock objects. The API is very nice and is suitable for complex stateful logic.
- Mockito is a mocking framework that provides a really clean and simple API for writing tests.It provides the possibility of partial mocking; real methods are invoked but still can be verified and stubbed.
- PowerMock is a framework that extends other mock libraries such as EasyMock with more powerful capabilities. It was created in order to provide a way of testing code considered untestable.

Mockito

- org.mockito.Mockito.mock - Creates mock object of given class or interface
- @Mock - Creates mock object of given class or interface
- @InjectMocks - instantiate testing object instances and inject fields annotated with @Mock
- MockitoAnnotations.initMocks - Initializes objects annotated with Mockito annotations for given testClass
- @RunWith(MockitoJUnitRunner.class) - alternative for MockitoAnnotations.initMocks