[TestLab1]
Testing

Introduction: In computer programming, unit testing is a software testing method by which individual units of source code—sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures—are tested to determine whether they are fit for use. Unit tests are typically automated tests written and run by software developers to ensure that a section of an application (known as the "unit") meets its design and behaves as intended. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. By writing tests first for the smallest testable units, then the compound behaviors between those, one can build up comprehensive tests for complex applications.

Objectives:

- Understand the importance of testing.
- Implement unit tests.

Classwork:

1. Add maven dependency to [JUnit4] if it is not already done.
2. Create JUnit 4 tests for most important domain logic methods of your feature. Note: Swing and JUnit extensions often works best with JUnit 4.
3. Write JUnit tests for best case scenario
4. Write JUnit tests for identified boundary cases
   (a) A unit test should test a single code-path through a single method. When the execution of a method passes outside of that method, you have a dependency and should apply mocks/stubs to avoid the dependency, see [mockito.org].
5. Use JAVA Assertions to test invariants, see [JAVA Asserts].
   (a) Assertions should be used to check something that should never happen
   (b) Note, an assertion should stop the program from running, but an exception should let the program continue running.

Report Work:

- At class level document unit test of important business functionality. Write part of Section 6 from [Report Template].