DEPENDENCIES EXERCISES

TEST SUITES

1. Create both a simple JUnit4 and JUnit5 test file not named with test as usually done
2. Write a number of simple basic tests in each of the test files
3. Create both a JUnit4 and JUnit5 test suite and connect them to each of the test files
4. Set up 2 different test suites in a project, each responsible for running a number of tests, one suite based on all test files in a specific package and one suite based on some specific test files in any package
5. Change order of tests with @Order and @TestMethodOrder annotations
6. Use @Tag, @IncludeTags and @ExcludeTags annotations to include and exclude some of the tests in the test suite
7. Use @IncludeClassNamePatterns and @ExcludeClassNamePatterns annotations to change the naming of test files in the test suites

CALENDAR MOCKING

1. Set up a mock for a calendar object and the methods for hour, year, time and some other method
2. Assert the calendar object before and after setting up the behavior
3. Verify the method calls of the calendar object in some different ways

ORDER STUBBING AND MOCKING

1. Download OrderBehaviorTester.java and OrderBehaviorTester.java from testdoubles folder on git
2. Import test files OrderStateTester.java and OrderBehaviorTester.java in a test project
3. Figure out implementation of Order class, interfaces, methods, stubs and logic based on the two test files

The test files are testing if an order is requested with more than 50 products
The test files should not be changed, but implementations, interfaces, stubs and logic required by the tests should be created, so that the tests pass

OWN MOCKING EXAMPLE

1. Create two classes with some methods and inject one of the classes into the other class
2. Set up behavior tests that demonstrates as many of the Mockito features as possible for the two classes, such as inorder, argument captor, exception throwing and interaction verification