**Unit Test Plan**

We will indicate which classes need to be tested and which do not. Some methods like getters and setters are too trivial and with a big program like this it is better to leave them untested. We will not test code that deals with UI either because unit tests are back-end tests.

In the application package, the FTMSApplication class calls upon the PersistenceFTMS subsystem to load persistence data, and starts the UI, so we will not have unit tests for it.

In the controller package, we will test 3 of the 4 methods of the OrderController class: FromStringToMenuItem, canItemBeMade and increasePopularity. placeOrder uses some of these methods so it does not belong in the unit tests.

We will test all methods of the ScheduleMaker class.

In the model package, we will test almost all the methods of the FTMSManager class: numberOfEquipment, hasEquipment, indexOfEquipment, hasMenu, numberOfStaff, hasStaff, indexOfStaff, numberOfSupplies, hasSupplies, indexOfSupplies, addEquipment, removeEquipment, addEquipmentAt, addOrMoveEquipmentAt, addStaff, removeStaff, addStaffAt, addOrMoveStaffAt, addSupply, removeSupply, addSupplyAt, addOrMoveSupplyAt, delete.

We will test 2 methods of the equipment class: delete and toString because the rest of the methods are getters and setters.

We will test all of the methods of the Menu class except for getters and setters and trivial methods (like minimumNumberOfMenuItems which only returns an int of value 0): numberOfMenuItems, hasMenuItems, indexOfMenuItem, addMenuItem, removeMenuItem, addMenuItemAt, addOrMoveMenuItemAt, delete.

We will do the same thing with MenuItem class: indexOfIngredient, addIngredient, removeIngredient, addIngredientAt, addOrMoveIngredientAt, delete, toString.

We will test the same methods for the Order class as we tested for the MenuItem class except that instead of dealing with ingredients we deal with menu items.

We will test 3 methods of the Schedule class: setSunday, delete and toString.

Aside from getters and setters, the Staff and Supply classes only have 2 methods: delete and toString which we will unit test.

For the persistence package, there is no particular method to test, but while unit testing other methods, we should add assertions and evaluate the persistence file to test if that works well.

Finally, the View package display the content retrieved via the controllers so there is no unit test we can create for it.

We will use different tools to unit test our different applications. JUnit for the desktop app and PHPUnit for the web app. Since our android app uses the jar from the java desktop app, we don’t need to test the same methods again. These tests are run on simple methods that should rarely be modified, so we can afford to run tests every time we do modify some. Our goal is to have a 75% test coverage of all our code.