**DIT635 - Mutation Testing Exercise**

**First, if you have not finished the activity from Lecture 12 (Mutation Testing), do so!**

In a previous exercise, you wrote unit tests for a Meeting Planner system based on both the functionality and code structure. We will not return to the Meeting Planner one last time to assess the sensitivity of your test cases to seeded faults in the code.

1. Generate at least four mutants for classes of your choice in the MeetingPlanner code.
   1. You must create at least one invalid, one valid-but-not-useful (non-equivalent), one useful, and one equivalent mutant.
   2. Each mutant must be created by applying a different mutation operator, and you must use at least one mutation operator from each of the three categories in the attached handout.
   3. You do not have to use the same classes or methods for all mutant categories. Try mutating different parts of the code.
2. Assess your test suite that you created in previous exercises, with respect to the set of mutants that you derived - Are you able to kill all of the non-equivalent mutants with your test suite? If not, write additional tests that can kill those non-equivalent mutants.
3. Identify a minimal subset of tests from your test suite that is sufficient to kill all of the non-equivalent mutants.

If you finish early, try adding mutations to the CoffeeMaker classes from Homework Assignment 1. Do your unit tests detect those mutations?