BSE3211 - Unit Testing Exercise

You have been hired to test our new calendar app! Congratulations!(?) This program allows users to book meetings, adding those meetings to calendars maintained for rooms and employees. It will actively prevent multiple bookings, and will manage the busy and open status for employees and rooms.

The system enables the following high-level functions:

- Booking a meeting
- Booking vacation time
- Checking availability for a room
- Checking availability for a person
- Printing the agenda for a room
- Printing the agenda for a person

Normally, actions are conducted through a command line user interface provided by the main method in the PlannerInterface class. As a tester, you - of course - have full access to the source code to employ in testing the system. The code is available in the folder: https://muele.mak.ac.ug/mod/folder/view.php?id=142683 as meetingplanner.zip

(as a VS Code project, but feel free to import into the IDE of your choice)

You are to do the following:

1. Formulate an informal test plan.

a. Given the above features and the code documentation, plan out a series of unit tests to ensure that these features can be performed without error.

i. Make sure you think about both the normal execution and illegal inputs and actions that could be performed. Think of as many things that could go wrong as you can! For instance, you will probably be able to add a normal meeting, but can you add a meeting for February 35th? Try it out.

2. Write tests in the jUnit framework.

1. If a test is supposed to cause an exception to be thrown. Make sure you check for that exception.
2. Make sure that your expected output is detailed enough to ensure that - if something is supposed to fail - that it fails for the correct reasons. Use appropriate assertions.

3. Submission

- The test plan
    - Unit Testing
- Unit tests – VS project code

Deadline: April 10, 2025