University of Central Lancashire
UCLan

School of Psychology and Computer Science

UCLan Coursework Assessment Brief

2021-2022

Module Title: Software Development

Module Code: CO2401 | Level 5

Test Driven Development of a Smart Building Controller Class

This assessment is worth 50% of the overall module mark

THE BRIEF/INSTRUCTIONS

Learning Outcomes Assessed by this Assignment:

LO1: Elicit software requirements and design an appropriate software solution
LO2: Evaluate approaches to ensuring software quality
LO3: Choose and apply appropriate software development approaches, methods, and tools for a given problem

UCLan has decided to convert the C&T building into a 'Smart Building' controlled by a custom written application. During the first iteration of the system, the university would like the system to control the doors, lights and fire alarm. Using **C#** and the **NUnit 3** unit testing framework, you are to take a 'test driven development' approach to develop and unit test a single class in this system, the BuildingController class.

The **BuildingController** class is responsible for managing the various smart systems in the building by communicating with three different dependencies (LightManager, DoorManager and FireAlarmManager).

BuildingController is also responsible for logging changes to the buildings state to an online web service by communicating with a WebService object, and sending maintenance emails as appropriate through communications with an EmailService object. Appendix A provides details about the structure and possible states of the **BuildingController** class.

Using a Test-Driven Development approach, you are to implement and unit test the **BuildingController** class according to the Requirements set out in Appendix B.

For this assignment, you should implement the following in a C# project:

- The **BuildingController.cs** class
- A **BuildingControllerTests.cs** class (that contains your unit tests). Each unit test should contain a comment detailing the requirement number that is being tested.
- Interface files for each of your 5 dependencies (60%+). These should be named as follows:
  - **IDoorManager.cs**
  - **ILightManager.cs**
  - **IFireAlarmManager.cs**
  - **IWebService.cs**
  - **IEmailService.cs**

It is VERY important you use these names for your interface files. It is also VERY important you follow the specification and implement the correct method signatures in your interface files as specified in Appendix A and Appendix B. It is VERY important that any messages passed as parameters or returned as results correspond exactly to the requirements. If you do not, my unit tests will not work with your code and I will be unable to award marks for passing these unit tests.

You do not need to develop any other system classes except the **BuildingController** class for this assignment, but for the higher marks (60%+) you will need to create Interface files and use the **NSubstitute** isolation/mocking framework to create stubs and mock objects to simulate the dependencies that the BuildingController class relies on to function properly.

Your work will be marked based on 2 criteria.

1. The functionality and quality of your **BuildingController** implementation – I will be running my own set of over 125 test cases on your Building Controller class to test the requirements in Appendix B
2. The quality of your Unit tests – I will be examining the completeness and validity of your unit tests.

Marking Criteria

40%

- All 'level 1' requirements implemented and tested
- BuildingController class implemented and NUnit 3 framework is used to develop a set of reasonable unit tests
- Contains at least 8 unit tests
- BuildingController class must pass at least four of your own unit tests
- BuildingController class must pass at least four of my unseen unit tests

50%

- Unit tests follow Arrange/Act/Assert pattern
- All 'level 2' requirements implemented and tested
- Unit tests are parameterised where appropriate, making use of multiple [TestCase] labels
- Code is appropriately commented
- Appropriate naming conventions used for tests

60%

- C# Interfaces created for all class dependencies (LightManager, FireAlarmManager, DoorManager, WebService and EmailService)
- NSubstitute mocking framework used to create stubs for use in unit tests
- All 'level 3' requirements developed and tested
- Implementation passes all submitted test cases or the failure has been documented in a comment by the code.
- Each unit test should have a comment mapping the test to a specific numbered requirement
- Implementation passes 70% of my unseen unit tests
- Good quality test cases

70%+

- NSubstitute mocking framework used to create mock objects
- All 'level 4' requirements developed and tested

- High quality, concise comments explaining the operation of the software without simply duplicating the code
- Implementation passes 85% of my unseen test cases
- Excellent quality test cases and structure

Note 1: There is NO REPORT with this assignment. Your test code and comments should be named and structured so that its operation is CLEAR to third party developers/testers who are responsible for taking over and maintaining your tests. If a unit test needs a report to explain its function, your test probably hasn't been structured correctly.

PREPARATION FOR THE ASSESSMENT

- For completing this assignment, you need to revisit all lectures we covered so far, and pay particular attention to week 8 onwards, where we start learning about testing. You will find all the materials we covered during the lecture and practical sessions on Blackboard. I also have added links to additional resources such as LinkedIn Learning links.

RELEASE DATES AND HAND IN DEADLINE

Assessment Release date: [14/12/2021] Assessment Deadline Date and time: [18/03/2022 23:59]

Please note that this is the final time you can submit – not the time to submit!
Your feedback/feed forward and mark for this assessment will be provided on 08/04/2022

SUBMISSION DETAILS

You must zip your visual studio project repository (your project folder) into a .zip file and submit it through the appropriate link on Blackboard.

Important: you must zip your file using the .zip format. Other archive formats may not open on the university machines and may result in you receiving a mark of 0%

Late Work: Except where an extension of the hand-in deadline date has been approved, work that is handed in within 5 working days late will receive a maximum mark of 40%. Work handed in later than this will receive 0%.

Extenuating Circumstances: Sometimes students have problems that affect their ability to perform in an assessment to their full potential or to complete an assignment by the set deadline. If you believe that this is happening to you, then it is your responsibility to let your School know as soon as possible as you may be able to apply for an extension/extenuating circumstances. See the following for more information: https://www.uclan.ac.uk/students/support/extensions.php

Cheating: The consequences of cheating in assessments are serious - you will fail the module. Cheating is using or attempting to use unfair means to enhance performance. This includes plagiarism (presenting someone else's work, or work found online as if it was your own), collusion (working with others on an individual assignment) and allowing other students to access your work. Make sure that you do not give someone the opportunity to steal your work (e.g. by asking them to print it out for you). If you have any doubt about what cheating is or how to reference material properly, please ask a tutor.

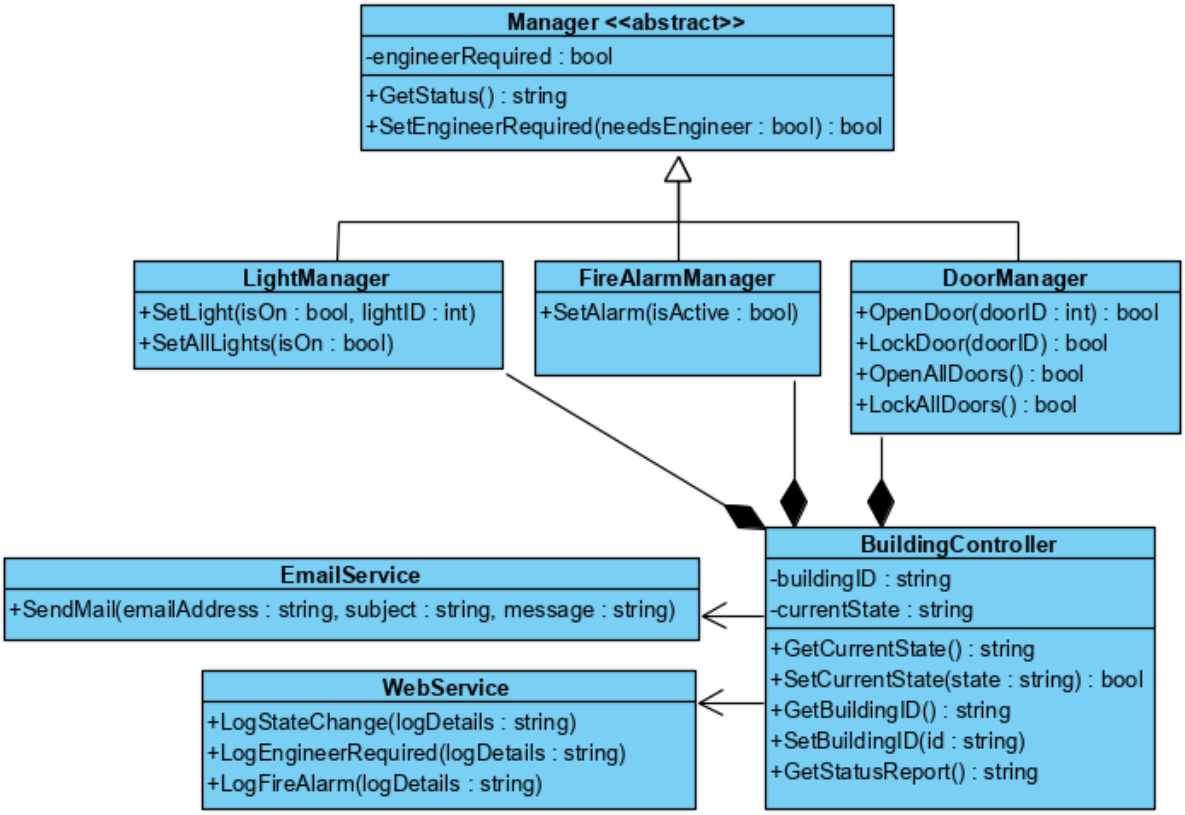
HELP AND SUPPORT

- For academic support related to this assessment, contact Dr. Mahsa Honary mhonary@uclan.ac.uk.
- For support with using library resources, please contact our subject librarian Bob Frost RSFrost@uclan.ac.uk. You will find links to lots of useful resources in the My Library tab on Blackboard.
- If you have not yet made the university aware of any disability, specific learning difficulty, long-term health or mental health condition, please complete a Disclosure Form. The Inclusive Support team will then contact to discuss reasonable adjustments and support relating to any disability. For more information, visit the Inclusive Support site.

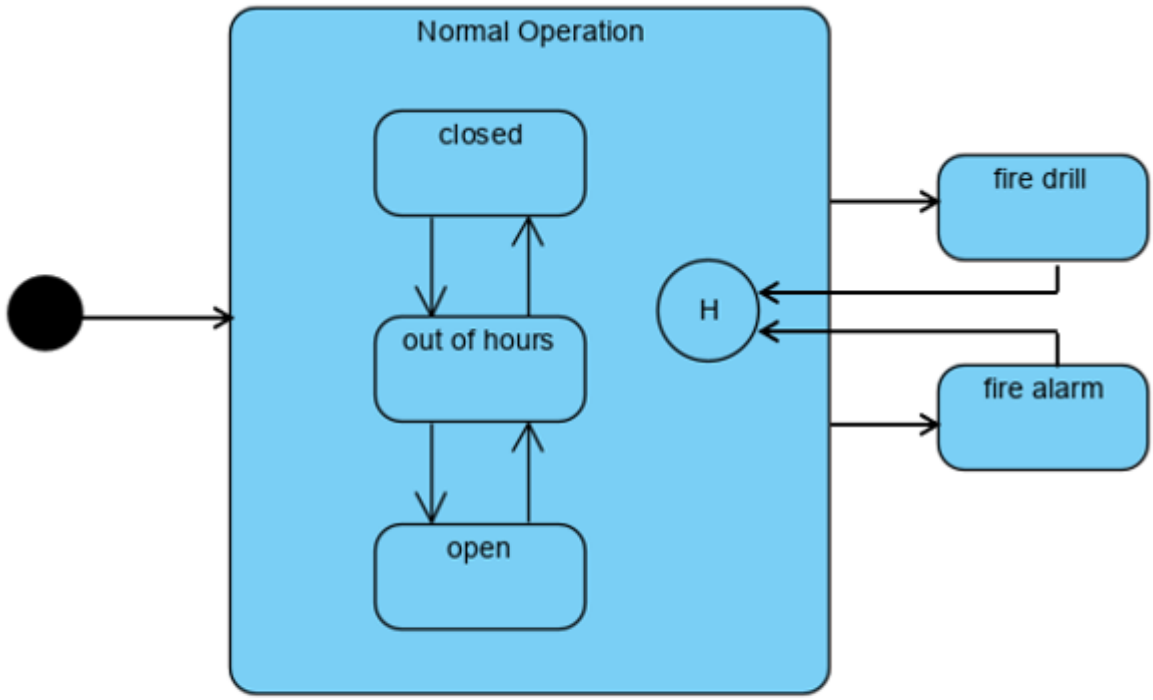
- To access mental health and wellbeing support, please complete our online referral form. Alternatively, you can email wellbeing@uclan.ac.uk, call 01772 893020 or visit our UCLan Wellbeing Service pages for more information.
- If you have any other query or require further support you can contact The Student Information and Support Centre. Speak with us for advice on accessing all the University services as well as the Library services. Whatever your query, our expert staff will be able to help and support you. For more information , how to contact us and our opening hours visit Student Information and Support Centre.
- If you have any valid mitigating circumstances that mean you cannot meet an assessment submission deadline and you wish to request an extension, you will need to apply online prior to the deadline.

Disclaimer: The information provided in this assessment brief is correct at time of publication. In the unlikely event that any changes are deemed necessary, they will be communicated clearly via e-mail and a new version of this assessment brief will be circulated.	Version: 1

APPENDIX A
Class Diagram Illustrating (Partial) System Structure



State Transition Diagram for BuildingController



Note: 'H' is notation for the 'History' state. Transitions into this state represent a returning to the state the system was previously in before transitioning out of the 'normal operation' superstate. E.g. If the system transitions from open -> fire alarm, then when exiting the 'fire alarm' state it must go from fire alarm -> open

APPENDIX B

Requirements: (During your 'Test Driven Development' it is suggested you begin with the level 1 requirements and work upwards)

Level 1 Requirements:

L1R1: *BuildingController contains a constructor method with a single string parameter that assigns the buildingID object variable. The constructor will have the following signature:*

```
BuildingController(string id)
```

L1R2: *GetBuildingID returns the value of the buildingID variable*

L1R3: *When the buildingID object variable is set in the constructor, uppercase letters will be converted to lower case.*

L1R4: *SetBuildingID sets the value of buildingID, converting uppercase letters to lowercase*

L1R5: *Constructor sets the initial value of currentState to "out of hours"*

L1R6: *GetCurrentState() function returns the value of the currentState object variable*

L1R7: *SetCurrentState() function checks that the string supplied is a valid state ("closed", "out of hours", "open", "fire drill" or "fire alarm"). If the string supplied is valid the function will set the currentState variable and return true. If the string supplied is invalid the function will return false and the state will be unchanged.*

Level 2 Requirements:

L2R1: *SetCurrentState() function will only allow changes of state according to the state transition diagram illustrated in Appendix A*

L2R2: *SetCurrentState() will return true by default and remain in the same state if an attempt is made to set the BuildingController's state to the present value of currentState*

L2R3: The BuildingController class has an additional constructor that takes two string parameters that set the default values for buildingID and currentState (the constructor should accept parameters in upper case, lower case, or a mixture of the two, but should store the value of currentState in lower case). The BuildingController class can only be initialised to one of the three normal operation states *("closed", "out of hours" or "open"). If it is not set to one of these states, an ArgumentException is thrown with the following message:*

```
"Argument Exception: BuildingController can only be initialised to the following states 'open', 'closed',
'out of hours'"
```

The constructor will have the following signature:

```
BuildingController(string id, string startState)
```

Level 3 Requirements:

L3R1: To make the BuildingController class 'unit test friendly', it allows dependency injection through an additional constructor method. The method takes 6 parameters providing the class with interfaces for all 5 dependencies outlined on the class diagram. The constructor will have the following signature:

```
BuildingController(string id, ILightManager iLightManager, IFireAlarmManager iFireAlarmManager,
IDoorManager iDoorManager, IWebService iWebService, IEmailService iEmailService)
```

L3R2: The GetStatus() methods of all 3 manager classes return a string containing comma-separated values. The first value is the type of device (either Lights, Doors, or FireAlarm) followed by a sequence of either OK, or FAULT values indicating the state of each managed device. E.g., LightManager may return:

"Lights,OK,OK,OK,OK,OK,OK,OK,OK,OK,OK," if the manager is managing 10 lights and all 10 lights are working normally. (note that the final value also includes a trailing comma). It may return the following:

"Lights,OK,OK,FAULT,OK,OK,OK,OK,FAULT,OK,OK," when two of the 10 lights are faulty.

NOTE: *You do not need to implement any of the manager classes, but you will need to simulate return types like this when you create your stubs*

L3R3: GetStatusReport() method calls the GetStatus() methods of all 3 manager classes (LightManager, DoorManager and FireAlarmManager) and appends each string returned together into a single string (in the following order: lightStatus, doorStatus, fireAlarmStatus) before returning the result. E.g.,

"**Lights,OK,OK,FAULT,OK,OK,OK,OK,FAULT,OK,OK,Doors,OK,OK,OK,OK,OK,OK,OK,OK,OK,OK,FireAlarm,OK,OK,FAULT,OK,OK,OK,OK,FAULT,OK,OK,**"

L3R4: When the BuildingControllers SetCurrentState() method moves to the "open" state, all doors should be set to open by calling the OpenAllDoors() method of the DoorManager object. If DoorManager.OpenAllDoors() returns false (indicating there was a failure to unlock all the doors) 'false' should be returned from the SetCurrentState() function and the building should remain in its current state.

L3R5: When moving to the "open" state, if DoorManager.OpenAllDoors() returns 'true' when attempting to move to the open state (indicating unlocking all the doors was a success) 'true' should be returned from SetCurrentState() and the building should move to the "open" state.

Level 4 Requirements:

L4R1: When the BuildingController's SetCurrentState() method moves to the "closed" state, all doors should be set to closed by calling the DoorManager.LockAllDoors() method and all lights must be turned off by calling the LightManager.SetAllLights(false)

L4R2: When the BuildingController.SetCurrentState() method moves to the "fire alarm" state, the alarm should be triggered by calling FireAlarmManager.SetAlarm(true), all doors should be unlocked by calling DoorManager.OpenAllDoors(), all lights should be turned on using LightManager.SetAllLights(true) and an online log should be made by calling WebService.LogFireAlarm("fire alarm").

L4R3: The GetStatusReport() method will parse each of the three status reports (for Lights, FireAlarm and Doors) and if a fault is detected, the WebService object will be used to log that an engineer is required to fix the fault by calling the WebService.LogEngineerRequired() method, passing a string parameter. The string parameter should contain the type of device that has shown a fault (e.g. Lights, FireAlarm or Doors). If multiple device types have shown a fault then these should be separated by a comma. E.g. If both Lights and Doors status reports indicate a fault the following string should be logged to the web server "Lights,Doors,".

L4R4: In addition to requirement L4R2, If WebService.LogFireAlarm( ) throws an Exception when called, an email should be sent using the EmailService's SendMail() method. To smartbuilding@uclan.ac.uk, with the subject "failed to log alarm" and the message parameter should contain the exception message returned from the failed call to the LogFireAlarm() function.

NOTE: You shouldn't ACTUALLY be sending an email to smartbuilding@uclan.ac.uk (the address doesn't exist). You should be checking your BuildingController class makes the appropriate call to a mock WebService object with the appropriate parameters.