Course: BSc (Hons) Computing

Module Code: CO2401
Module Title: Software Development

Title of the Brief:
Test-Driven Development (TDD) of a Smart Building Controller Class

Type of assessment:
Software Design and Testing

This assessment is worth **50%** of the overall module mark and should take approximately 120 hours of independent study to complete (including completion of related practical/tutorial exercises, reading about unit testing and TDD, and creating test cases to design, develop, and unit test a single class by iterating through the red-green-refactor cycle).

This Assessment Pack consists of the following:

(1) Information to guide you on **how, where, when and what to submit**.
(2) A detailed **assignment brief** with guidance on what you need to prepare, how class/lab sessions support your ability to complete this assignment successfully, and the **Marking Scheme**.
(3) Additional useful resources. If you need additional support, please make a note of the **services** detailed in this document.
(4) Feedback Guidance & Coursework Cover Sheet.

HOW, WHEN, WHERE, AND WHAT TO SUBMIT

The assessment release date is: 18/12/2023.

The assessment deadline date and time is: Friday 15/03/2024 at 12 noon (Cyprus time).

All assignments must be submitted electronically on **Blackboard** before the deadline.

1) Use the C# project template provided to develop your classes. You must zip your entire completed **Visual Studio Project Repository** (your project folder) into a **.zip file** and submit it on Blackboard.

2) You also need to submit a separate **Project Report** documenting the process of TDD you followed to develop the final classes. Use the report template provided and provide the necessary annotated screenshots and explanations.

3) Finally, in-class **Project Demonstrations** will take place during Week 10 (Wednesday 20th March 2024) and Week 11 (Wednesday 27th March 2024) and will involve hands-on test case design. During the demonstration, you will be given new requirements for the unit under test, and you will need to create the test cases for it, which in turn will drive the development of code that satisfies these new requirements. The demonstrations will take place during your regular lecture & lab sessions. Physical attendance and presentation are compulsory. The demonstration order will be random.

Feedback will be available after the project demonstrations when all submissions are marked.

This is an individual assignment. All parts must be completed individually.

Note: If you have any valid Mitigating Circumstances that mean you cannot meet an assessment submission deadline and you wish to request an extension, you need to apply online. To apply for Mitigating Circumstances, please submit a formal request through your Timetable. Once you access and log in to your Timetable[1], select "Student Letter Requests" → "Mitigating Circumstances" and proceed to complete the form with the requested information along with your evidence at least 3 days before the deadline. Further information on Mitigating Circumstances can be found via the Cyprus Student Hub[2]. Read the information available on this dedicated webpage carefully and thoroughly.

We wish you every success in completing your assessment. Read this guidance carefully, and should you have any questions, please discuss these with your Module Leader (Dr Andriani Piki, apiki@uclan.ac.uk).

ASSIGNMENT BRIEF

Learning Outcomes Assessed

This coursework contributes to the assessment of the following module learning outcomes (LOs), which state that upon completion of this module you should be able to:

LO1 – Design an appropriate OO software solution.
LO3 – Evaluate approaches to ensuring software quality.
LO4 – Choose and apply appropriate software development approaches, methods, and tools for a given problem.

Summary of the Assessment

UCLan has decided to convert the C&T building into a 'Smart Building' controlled by a custom written object-oriented application. During the first iteration of the system, the university would like the system to control the doors, lights, and the fire alarm. Using **C#** and the **NUnit 3** unit testing framework, you are to take a 'Test-Driven Development' (TDD) approach to design, develop, and unit test a single class in this system, the **BuildingController** class. You are to develop your solution to a given specification.

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 (TDD) approach, you are to implement, and unit test, the **BuildingController** class according to the Requirements set out in Appendix B.

[1] https://timetable.cyprus.uclan.ac.uk/
[2] https://msuclanac.sharepoint.com/sites/CyprusStudentHub/SitePages/Mitigating-Circumstances.aspx

What you need to implement

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

- 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.
  - Each unit test should also have a **prefix** matching the Requirement id (e.g., L1R2_).
- 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

Important notes

- It is VERY important you follow the specification provided 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 implement **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.

Marking criteria

Your work will be marked based on 2 criteria:

- The functionality and quality of your BuildingController implementation – I will be running my own set of over 125 test cases on your BuildingController class to test the requirements in Appendix B.
- The quality of your Unit tests – I will be examining the completeness and validity of your unit tests (across all submitted evidence: Project repository, Project report, Demonstration).

Preparation for the Assessment

- For completing this assignment, you should revisit the lecture/lab materials, and pay particular attention to content about testing, test driven development and isolation frameworks.
- You will find all the materials we covered during the lecture and practical sessions on Blackboard.
- Templates will be available for you on Blackboard to use for your C# project and Project Report.

MARKING SCHEME

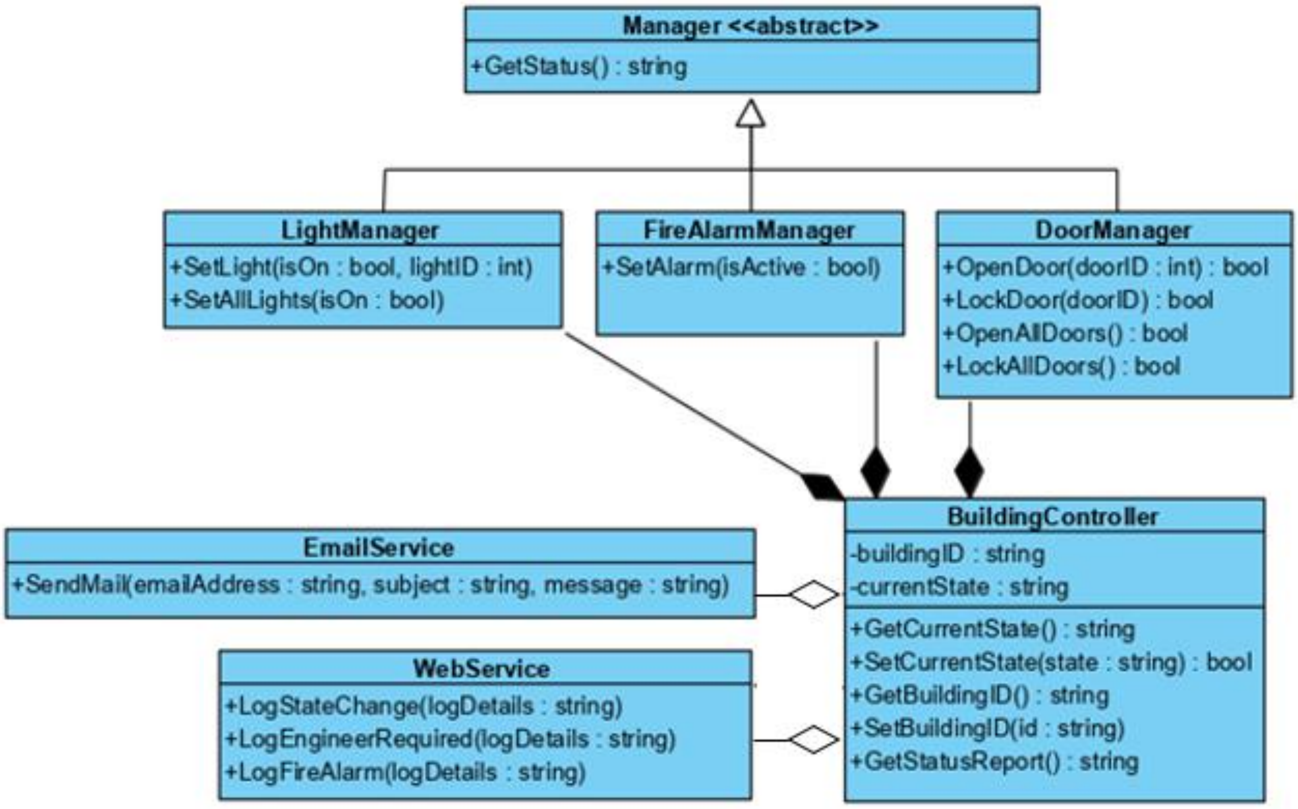
- UCLan assignments are marked using a **banded marking scheme**. See the assessment handbook for more information about banded marking.
- The Marking Scheme identifies the criteria used to mark this coursework. You should carefully examine it to ensure that you are maximising the marks that you can achieve.
- Note 1: If you are unable to implement a lower level requirement, but are able to implement a higher level requirement in it's place, you are recommended to do so, as we may be able to take this into account when marking your work.
- Note 2: 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. Ensure important aspects are documented in your report.

Criteria	Range
No or poor submission. No or poor report. No show or poor demonstration/presentation.	Fail (0-35)
40% • All 'level 1' requirements implemented and tested (Appendix A) • 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 (4) of your own unit tests • BuildingController class must pass at least four (4) of my unseen unit tests • *Basic report quality and demonstration*	3rd (42-48)
50% • Unit tests follow Arrange/Act/Assert pattern • All 'level 2' requirements implemented and tested (Appendix A) • Unit tests are parameterised where appropriate, making use of multiple [TestCase] labels • Code is appropriately commented • Appropriate naming conventions used for tests • *Reasonable report quality and demonstration*	2.2 (52-58)
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 (Appendix A) • 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 75% of my unseen unit tests • Good quality test cases • *Good report quality and demonstration*	2.1 (62-68)
70%+ • NSubstitute mocking framework used to create mock objects • All 'level 4' requirements developed and tested (Appendix A) • High quality, concise comments explaining the operation of the software without simply duplicating the code • Implementation passes 90% of my unseen test cases • Excellent quality test cases and structure • *Excellent report quality and demonstration*	1st (74-100)

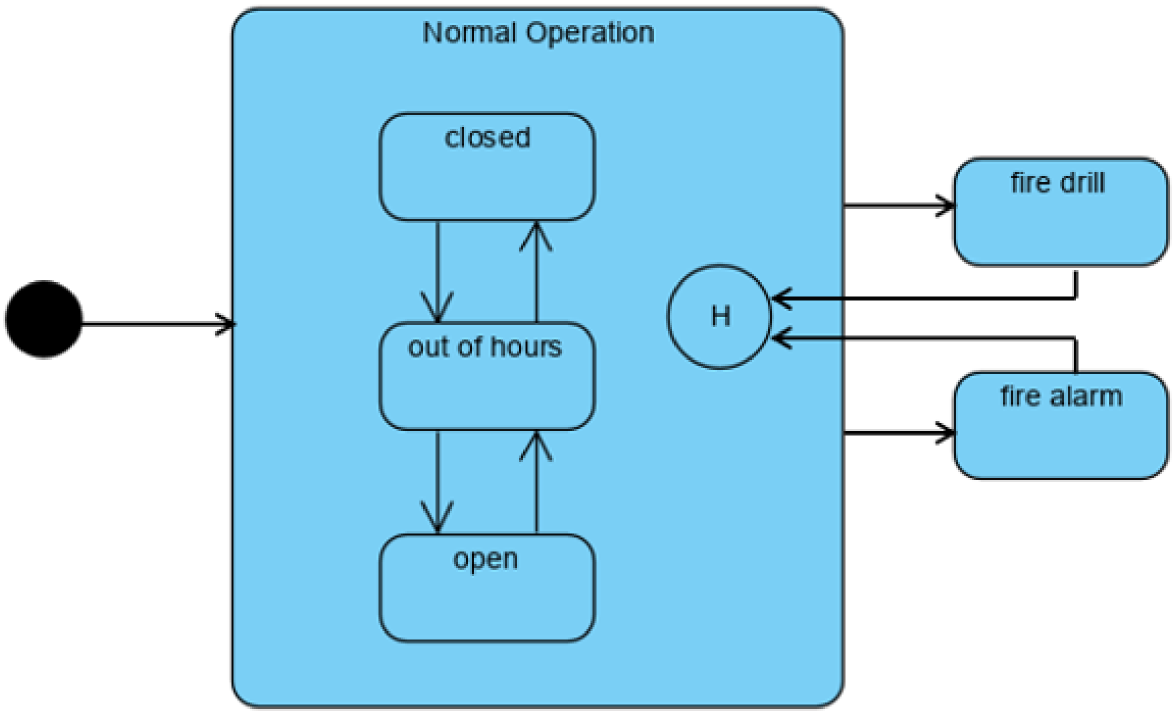
University of Central Lancashire UCLan

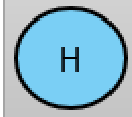
APPENDIX A

Class Diagram Illustrating (Partial) System Structure


Manager <<abstract>>
+GetStatus() : string
LightManager
+SetLight(isOn : bool, lightID : int)
+SetAllLights(isOn : bool)
FireAlarmManager
+SetAlarm(isActive : bool)
DoorManager
+OpenDoor(doorID : int) : bool
+LockDoor(doorID) : bool
+OpenAllDoors() : bool
+LockAllDoors() : bool
BuildingController
-buildingID : string
-currentState : string
+GetCurrentState() : string
+SetCurrentState(state : string) : bool
+GetBuildingID() : string
+SetBuildingID(id : string)
+GetStatusReport() : string
EmailService
+SendMail(emailAddress : string, subject : string, message : string)
WebService
+LogStateChange(logDetails : string)
+LogEngineerRequired(logDetails : string)
+LogFireAlarm(logDetails : string)

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 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 write a unit test for this requirement. 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,Fir
eAlarm,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.

ADDITIONAL SUPPORT AVAILABLE

All links are available through the online Student Hub

1. Academic support for this assessment will be provided by contacting the Module Leader (Dr Andriani Piki, apiki@uclan.ac.uk).

2. Our **Library resources** link can be found in the library area of the Student Hub or via your subject librarian at CyprusLibrary@uclan.ac.uk

3. Support with your academic skills development (academic writing, critical thinking and referencing) is available through WISER on the Student Hub.

4. For help with Turnitin, see Blackboard and Turnitin Support on the Student Hub.

5. If you have a disability, specific learning difficulty, long-term health or mental health condition, and not yet advised us, or would like to review your support, **Student Support Officers** (cyprusstudentsupport@uclancyprus.ac.cy) can assist with reasonable adjustments and support. To find out more, you can visit the **Student Support Service** page of the Student Hub. You can also call +357 24694026 or +357 24694108 or +357 24694073.

6. For mental health and wellbeing support, please complete our online referral form or email the Psychological Wellbeing and Counselling Centre at UCLan Cyprus at wellbeing@uclancyprus.ac.cy.

7. For any other support queries, please contact **Student Support** via CyprusStudentSupport@uclan.ac.uk.

8. For consideration of Academic Integrity, please refer to detailed guidelines in our policy document. All assessed work should be genuinely your own work, and all resources fully cited.

9. For advice on the use of Artificial Intelligence, please refer to Categories of AI tools guidance. For this assignment, you are not permitted to use any category of AI tools to generate the test cases, the code, report, or any other part of the assessment.

FEEDBACK GUIDANCE

Reflecting on Feedback - How to improve

From the feedback you receive, you should understand:

- The grade you achieved.
- The best features of your work.
- Areas you may not have fully understood.
- Areas you are doing well but could develop your understanding.
- What you can do to improve in the future – feedforward.

Next Steps:

- List the steps have you taken to respond to previous feedback.
- Summarise your achievements.
- Evaluate where you need to improve here (keep handy for future work):

University of Central Lancashire UCLan

Coursework Cover Sheet

Students should complete the input fields contained in this form and attach it in front of your formal assessment submission. All fields within this form are required. Please ensure that check boxes and radio buttons are appropriately selected. The last three questions are just for you to personally consider.

Department and assessment information:

School Name: Enter the school name here.

Assessment title: Enter the title of the assessment here

Course Title: Enter the Course Title here

Module Title: Enter the Module Title here

Module Code: Enter the Module code here

Year of Study: Enter the your year of study eg. 2023

Academic Misconduct / Plagiarism Declaration

By attaching this front cover sheet to my assessment I confirm and declare that **I am the sole author of this work**, except where otherwise acknowledged by appropriate referencing and citation, and that I have taken all reasonable skill and care to ensure that no other person has been able, or allowed, to copy this work in either paper or electronic form, and that prior to submission I have read, understood and followed the University regulations as outlined in the Academic Integrity Policy and Procedure for Academic Misconduct

Have you checked the following? This will help your assessment achievement.

I have applied the learning outcomes for this module ☐

I have checked for Academic Integrity via Turn-it-in ☐

I have followed the guidance in the Assessment Brief and have not used AI to boost my grade unfairly. ☐

I have used references in accordance with instructions in the Assessment Brief ☐

I have proofread my work for spelling, grammar and punctuation. ☐

I have checked that the word count/size of this submissions accords with the guidance provided in the Assessment Brief.☐

Well-being

We wish to support any student who is experiencing mitigating circumstances which prevents students from performing to the best of their ability when completing or submitting assignments. If you are experiencing such circumstances, then you may apply for Mitigating Circumstances. Wherever possible this must be done prior to handing in your assignment.

Do you need to apply for mitigating circumstances for this assignment Please select Yes / No

Please refer to the Mitigating Circumstances Policy

Questions you may wish to consider:

1. Have I allowed sufficient time to prepare this assessment?
2. Have I reflected on previous feedback and made improvements in accordance with advice?
3. What grade am I expecting?