School of Computing, Edinburgh Napier University

1. Module number	*SET08108*
2. Module title	*Software Development 2*
3. Module leader	*Neil Urquhart*
4. Tutor with responsibility for this Assessment Student's first point of contact	*Neil Urquhart* E: n.urquhart@napier.ac.uk T: 0131 455 2655
5. Assessment	*Practical*
6. Weighting	*80%*
7. Size and/or time limits for assessment	*OO design and development as specified. You should not spend more than 50 hours on this. A short (~5 minute) demonstration will be required prior to hand-in.*
8. Deadline of submission Your attention is drawn to the penalties for late submission	***By the end of week 14 you should have:*** • *Handed in a copies of your class diagrams and code to the School Office* • *Demonstrated your work in class* • *Created a working repository on BitBucket* • *Uploaded a .ZIP to Moodle containing your project* MAKE SURE THAT YOU CHECK THE CLOSING TIME OF THE SCHOOL OFFICE!
9. Arrangements for submission	***By the end of week 14 you should have:*** • *Handed in a copies of your class diagrams and code to the School Office* • *Demonstrated your work in class*

	• *Uploaded a .ZIP to Moodle containing your project* Demonstrations may be made during weeks 14 and 15.
10. Assessment Regulations All assessments are subject to the University Regulations**.**	*No exemptions*
11. The requirements for the assessment	*See attached*
12. Special instructions	The teaching team will make arrangements for demonstrations and publicise this to students during the lectures, via WebCT and via Email. Students must remain in contact with the teaching team.
13. Return of work and feedback	Instant, personalised oral feedback will be available at the demonstration. A session will be offered in **weeks 14 or 15 to provide** cohort feedback. Individuals who wish to discuss their specific submission (priority will be given to those who have reassessments).
14. Assessment criteria	See attached. Please note that checks for plagiarism will be made on electronic submissions. Students may be required to attend a further demonstration if there exists doubts as to the authorship of work.

Coursework 2

Edinburgh Napier University requires a student records system. The system must be capable of storing student and lecturer's details.

Students have the following attributes:

Attribute	**Type**	**Validation**
Name	String	Not blank
Address	String	Not blank
Email	String	Not blank contains '@'
Matriculation Number	Int	In range 1000 to 9000 Must be unique

Staff have the following attributes:

Attribute	**Type**	**Validation**
Name	String	Not blank
Address	String	Not blank
Email	String	Not blank contains '@'
Payroll Number	Int	In range 9000 to 9999 Must be unique
Department	String	Not blank
Role	String	Must be: "Lecturer" "Senior Lecturer" "Professor"

The University run a number of modules, each module has the following attributes:

Modules have the following attributes:

Attribute	**Type**	**Validation**
Name	String	Not blank
Module Code	String	Not blank Unique
Module Leader	Staff	Not blank

Each module has a member of staff allocated as the module leader, a member of staff may be module leader for many modules. Students are enrolled on modules, each module may have many students enrolled on it and each student may be enrolled upon many modules. When a student is enrolled upon a module they will have a mark and a status within that module. Initially their mark will be -1 and their

status will be studying. When a mark is entered for that student on that module the status will change to “Failed” if the mark is <40 or “Passed” if the mark is 40.

Create a class diagram showing your design (use the diagrammer from the Architecture menu in Visual Studio, automatically generated diagrams are not acceptable). You do not need to include GUI classes (forms) at this stage.

1. Implement the classes identified in your diagram using C#
2. Add a GUI (using WPF) to allow the following
    a. List all members of staff
    b. List all modules
    c. List all students
    d. Enrol a student on a module
    e. Add/update a mark for a student on a module- the students’ status on the module should be automatically updated depending on the mark
    f. Edit a student’s details
    g. Remove a student from a module
    h. List all the students on a module, along with their marks and status
    i. Reassign module leader
3. Create Unit tests for your Student and Module classes. They should test the correctness of the methods and properties associated with these classes, including operations such as enrolment of students on modules.

4. Update your class diagram to show the classes added to implement the GUI

Optional Tasks

You may wish to pick one of the following optional tasks (which may be carried out in addition to the basic or advanced tasks). You will need to research how these tasks will be carried out.

- Store the data held by the system in a file so that it may be saved and reloaded between sessions
- Use a database to store the data held by the system

General Points

Class Diagrams

Class diagrams should show:

- Private properties
- Public properties (with get/set as appropriate)
- Public and private methods
- Relations between classes (e.g. 1:1, 1:M or inheritance)

Coding standards & Use of Bitbucket.

1. Your Bitbucket username must be your matriculation number
2. The repository used for this assessment must be called “assessment2”
3. You must grant Neil & Ben read access to your repository, which must be maintained until the module results have been published.

IF YOU DO NOT FOLLOW THE ABOVE STEPS YOU MAY NOT RECEIVE ANY MARKS FOR THE SOURCE CONTROL SECTION

4. Code should be committed and pushed on a regular basis with comments added to the commit.

5. All classes should have comments at the top to note
    i. Author name
    ii. Description of class purpose
    iii. Date last modified
6. Methods should all have a comment to describe their purpose
7. Properties should all have a comment to describe their purpose
8. All code should be indented as appropriate. EG

...

```
for (int counter = 1; counter <= 1000; counter++)
{
        if (counter == 10)
                break;
        Console.WriteLine(counter);
}
```

...

9. Use descriptive variable and method names (.e.g. no use of ‘x’ or ‘y’!)
10. Code is written in a succinct fashion (i.e. no unnecessary code.)

Demonstration

When demonstrating you must have a copy of the demonstration sheet printed out, this will be filled in during the demonstration. You must also have printed copies of the class diagrams.

For the purposes of demonstration, enter the names and id codes of the 6 modules that comprise this year of your programme. Use the module leaders' names as well (supply your own test data for items such as payroll number and address). Add yourself as a student with enrolments on all 6 modules.

Submission

Please submit a printed copy of your class diagrams and all code that you have written to the School Office no later than the deadline shown on the cover sheet. You must include the code for your test harnesses and screenshots of the successful tests. Your design and your code will be subject to code/design review.

Please make sure that your BitBucket repository contains a valid Visual studio 2013 solution, that may be downloaded and compiled. Please make sure that all of your code/projects/documentation is included. Finally please upload .ZIP archive of your repository into Moodle, these files may be accessed by the external examiners or module moderators at a future date.

Demonstration Sheet

Name_____ **Matric Number**_____

Demonstrator _____ **Date/Time** _____

1. **Development tasks**

Item	**Mark 0,1,2** 0 – no attempt 1 – partial attempt 2 – fully working
Add a new member of staff	
Add a new student	
Enrol a student on a module	
Add a mark for a student on a module	
Update a mark for a student on a module	
Reassign a module to another member of staff	
Remove a student from a module	
Produce module list, showing students (including marks and status)	

2. **Testing**

Item	**Mark 0,1,2** 0 – no attempt 1 – partial attempt 2 – fully working
Execute automated tests for Student	
Execute automated tests for Module	

3. **Optional tasks**

Item	**Mark 0,2,4** 0 – no attempt 2 – partial attempt 4 – fully working
Use of database OR Serialisation (Delete as appropriate)	

Notes:

TOTAL _____

Design/Code Review Sheet

Name_____ **Matric Number**_____

All sections are marked out of 2 (0= no attempt,1=reasonable attempt,2= fully working).

Please note that in situations where substantial sections of the code are missing or incorrect as evidenced by the demonstration then marks in this section may be reduced or capped at the discretion of the module leader.

Item	**Mark (0,1,2)**
Class diagram: Identification of classes	
Class diagram: Use of inheritance and relations	
Appropriate data types used for properties	
Properties declared as private and accessors used	
Tests on Student cover all appropriate methods and properties	
Tests on Module cover all appropriate methods and properties	
Code is adequately commented	
Code matches class diagram	
Class diagram separates GUI classes from business classes	

Source Control	**Mark (0,2,6)**
BitBucket repository contains source, with evidence of regular commits	0 – Not used 2 – Code in repository 6 – Evidence of regular commits

Total____/24

Total marks

From demo _____ /24

From review _____/24

Total _____ /48