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:*** *Uploaded .PDF and .ZIP files to Moodle* *Demonstrations may be made during weeks 14 and 15.*
9. Arrangements for submission	***By the end of week 14 you should have:*** *Uploaded .PDF and .ZIP files to Moodle* *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 Moodle 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 may make an appointment with the teaching team (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: Holiday reservation System.

The Napier Holiday Village requires a reservation system. The system should be able to accept a variety of holiday bookings for customers. Each basic booking comprises the following attributes:

Attribute	Type
Arrival Date	Date
Departure Date	Date
Booking Reference Number	Auto increment starting at 1

In addition each booking is associated with a customer who has the following attributes:

Attribute	Type
Name	String
Address	String
Customer Reference Number	Auto increment starting at 1

A customer may have many bookings, each booking must be associated with a customer.

Each booking will have a number of guests,up to a maximum of 4 (a customer is not necessarily a guest):

Attribute	Type
Name	String
Passport Number	String (max 10 chars)
Age	Integer 0-101

The cost of a basic holiday is calculated as follows:

- Basic cost per night per person £50 (£30 if under 18)
- For each person the basic cost is multiplied by the number of nights booked
- The total cost the cost for each person, plus extras (see below)

Holidays may have a number of extra options added on

- Evening meals: A holiday may have evening meals added on to it, the cost will be an extra £15 per person per night. When meals are added to a booking the dietary requirements of the booking should be noted. This will take the form of a text string (e.g. “2 vegetarian and 1 nut allergy”) associated with the booking.

- Breakfast: A holiday may have breakfast added on to it, the cost will be an extra £5 per person per day. When meals are added to a booking the dietary requirements of the booking should be noted. This will take the form of a text string (e.g. “2 vegetarian and 1 nut allergy”) associated with the booking.
- Car hire: Car hire costs £50 per day extra. When a car hire is associated with a booking the following should be noted, hire start date, hire end date and name of driver.

You will create a system that will allow bookings to be added, amended and deleted. The system should be able to cope with any number of customers, each of whom may have multiple bookings. It should be possible to amend a booking (including adding or removing extras) once it has been entered. The system should also be able to produce invoices showing the cost of a booking, the invoice should show the costs per night and the costs of any extras.

Tasks

1. Create a class diagram showing your design (use the diagramming tool 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.
2. Implement the classes identified in your diagram using C#
3. Add a GUI (using WPF) to allow the following
    a. Add a new customer, booking or guest
    b. Add extras to a booking
    c. Display the invoice (costs) for a specific booking
    d. Delete a customer (assuming there are no bookings for them)
    e. Delete a booking
    f. Amend an existing customer, booking or guest (including adding/removing extras)

4. Create a Unit test for your Booking class (or equivalent), this should test the correctness of the methods and properties associated with these classes
5. 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

OO Design and use of design patterns

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)
- Where you have made use of a design pattern, add a brief note to the diagram explaining which pattern(s) you've used and how they were used.

Appropriate use of design patterns should be made, ideas could include:

- Use of factories to create objects which need an ID
- Use of a facade to hide the complexity of the system from the GUI
- Use of decorators to add extra features to a booking

You can incorporate design patterns in any other ways you feel appropriate, marks are available for the use of up to 2 design patterns.

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 & Simon read access to your repository, which must be maintained until the module results have been published.
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 describe:
    i. Author name
    ii. Description of class purpose
    iii. Date last modified
    iv. Any design patterns that the class is part of

6. Methods should all have a comment to describe their purpose
7. Properties should all have a comment to describe their purpose
8. Use descriptive variable and method names (.e.g. no use of 'x' or 'y'!)
9. 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.

Submission

You must make your submission via Moodle please upload the following:

- A .zip archive containing your complete Visual Studio Project
- A single .PDF containing
    - Your class diagram
    - A note explaining any design patterns used
    - Printed listings for all of the non-GUI classes in your system.

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
Create a new customer.	
Create a new booking with guests	
Amend an existing booking	
Add an extra option onto a booking	
Delete a booking	
Delete a customer	
Produce an invoice for a booking	

2. **User Interface**

Item	**Mark 0,2,4** 0 – difficult to use 2 – usable 4 – exemplary
Clarity and functionality of UI	

3. **Testing**

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

4. **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 _____ /26

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	
Class diagram: GUI classes separated from business classes	
Appropriate data types used for properties	
Properties declared as private and accessors used	
Code is adequately commented	
Code matches class diagram	

Design patterns	**Mark (0-6)**
Appropriate use and description of up to **2** design patterns.	0 No attempt 1-2 Marks for an attempt 3 Marks for a well-documented pattern

Total____/20

Appendix 1 : Sample Data

Menu Items

Description	Price	Vegetarian
Tomato Pizza	£4.50	Y
Peperoni Pizza	£5.00	N
Spaghetti Bolognaise	£6.00	N
Tomato Pasta	£3.50	Y

Servers

Name	ID
Ben	1
Louise	2
Neil	3

Drivers

Name	ID	Car Reg
Jim	4	VSC 86
David	5	BFS 1L
Sian	6	CSG 773S