School of Computing, Edinburgh Napier University

1. Module number	*SET08108*
2. Module title	*Object Oriented Software Development*
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 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 your .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 Chalet Management System.

The Napier Holiday Village requires a reservation system, which can create holiday chalet bookings for customers. Each booking should be allocated to a chalet; there are 10 chalets that may be allocated.

Each basic booking comprises the following attributes:

Attribute	**Type**
Arrival Date	Date
Departure Date	Date
Booking Reference Number	Auto increment starting at 1
ChaletID	In range 1 to 10

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 6 (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 chalet is calculated as follows:

- Basic cost per night for a chalet is £60 plus £25 for each guest staying
- Any extras are charged per person, per night
- The total cost of the booking is the sum of the nightly costs/

Holidays may have a number of extra options added on

- Evening meals: A chalet may have evening meals I provided, the cost will be an extra £10 per guest per night Meals can only be added for all guests for the duration of the booking.

- Breakfast: A holiday may have breakfast added on to it, the cost will be an extra £5 per guest per day. Meals can only be added for all guests for the duration of 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.

Optional extra functionality

1. *When creating a booking check to see if the selected chalet is available, if the new booking clashes with an existing booking then display an error message.*
2. *Have the data layer store the bookings in a file or database in order to make them persistent.*

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.

   Your diagram should be divided into Presentation, Business and Data layers. A suggested approach is that the business layer should only handle one booking and the data layer should store multiple bookings.

2. Implement business layer and data layers identified in your diagram using C#

3. Add the presentation layer (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. Create a report (max 1 A4 side of 12pt text and 1A4 side of diagrams) that answers the following questions:
    a. What advantages are of the 3 layered approach to building applications?
    b. With an example, explain why using design patterns can make the design of an OO system easier to understand.

General Points on Implementation

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 link the layers together
- 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.

1. 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

2. Methods should all have a comment to describe their purpose
3. Properties should all have a comment to describe their purpose
4. Use descriptive variable and method names (.e.g. no use of 'x' or 'y'!)
5. 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.
    - Your answers to both questions.

Demonstration Sheet

Name___________________________ **Matric Number**__________________

Demonstrator ____________________ **Date/Time** ______________________

Item	Level	Mark	Notes
Create customer, booking and guests (including validation)	Basic	6	All works 6 marks. Partially works 2-5 marks. Major errors 1 mark.
Add extras	Basic	3	All works 3 marks. Partially works 2marks. Major errors 1 mark.
Calculate Bill	Basic	3	All works 3 marks. Partially works 2marks. Major errors 1 mark.
Handle multiple bookings	Basic	2	Works 2 marks.
Persistance	Advanced	4	All works 4 marks. Read or write only 2 marks.
Conflict checking	Advanced	4	Works 4 marks, 1-3 marks for partially working

Notes:

Design/Code Review Sheet

Name___________________________ **Matric Number**_________________

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.

Basic OO design		6	identification of classes, use of inheritance, use of relations
Adherence to architecture		6	2 marks per level (1 for design and 1 for imll with separate projects)
Use of design patterns		4	2 marks per pattern used. Max 2 patterns
Code standards		2	Comments, accessors, sensible names etc
Report - item A		4	
Report - item B		4	
Unit test		4	Test Created. Methods tested