School of Computing, Edinburgh Napier University

1. Module number	*SET08119*
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 13 you should have uploaded .PDF and .ZIP files to Moodle* *Demonstrations may be made during week 13 or before.*
9. Arrangements for submission	*By the end of week 13 you should have uploaded .PDF and .ZIP files to Moodle* *Demonstrations may be made during week 13 or before.*
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. 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: Railway Planning System.

A system is required to manage the planning of a railway network and allow passengers and parcels to be booked onto train services.

Trains run between two points and stop at a number of intermediate stations. Trains run between Edinburgh (Waverley) and London (Kings Cross). Trains may stop at Peterborough, Darlington, York or Newcastle. Trains may be classed as stopping, express or sleeper. An express may not have any intermediate stops, a stopper may have intermediate stops and a sleeper may have intermediate stops, but must depart after 21:00 hours. A train may offer first class and a sleeper may offer a cabin.

Each train should hold the following information:

Item	Notes
Train-id	A 4 character code (e.g. 1A45), no two trains may have the same code
Departure	Select from “Edinburgh (Waverley)” or “London (Kings Cross)”
Destination	Select from “Edinburgh (Waverley)” or “London (Kings Cross)”
Type	Select from “Express”, ”Stopping” or “Sleeper”
Intermediate	Select from “Peterborough”, “Darlington”, “York” or “Newcastle” – can have more than 1
Departure time	hh:mm – 24 hour clock
Departure day	dd/mm/yy
SleeperBerth	True/false – true if a sleeper
FirstClass	True if a train offers first class

Once a train (or trains) has been added passengers may book tickets to travel on that train. Each booking records the following information

Item	Notes
Name	String (not blank)
Train	Train-id
Departure station	Select from “Edinburgh (Waverley)”, “London (Kings Cross)”, “Peterborough”, “Darlington”, “York” or “Newcastle”
Arrival Station	Select from “Edinburgh (Waverley)”, “London (Kings Cross)”, “Peterborough”, “Darlington”, “York” or “Newcastle”
FirstClass	True/False – can only be true if the train offers first class
Cabin	True/False – can only be true if the train is a sleeper
Coach	Coach in which seat is reserved (A-H)
Seat	Seat which is reserved (1-60)

Fares are calculated as follows:

Edinburgh – London: £50
Between any 2 intermediate stations or any intermediate station and Edinburgh or London : £25
First class £10 surcharge
Sleeper £10 surcharge
Sleeper cabin £20 surcharge

Basic functionality:

1. The user should be able to create new trains, setting destinations and types etc.as required. Data should be validated, please note the following:
    a. No two trains can have the same id
    b. Sleepers depart after 21:00

2. The user should be able to add passengers to a train and store their details, please note the following:
    a. Once a coach/seat is booked it may not be booked by another passenger
    b. The user should be shown the price of the ticket, prior to confirming the booking.

Optional extra functionality

1. *Implement a search function to find all of the trains running between two specified stations – show dates and departure time of the train.*
2. *Have the data layer store the trains and bookings in a file or database in order to make them persistent.*

Tasks

1. Create a class diagram showing your design (automatically generated diagrams **are not acceptable**). You do not need to include GUI classes (presentation layer) at this stage. Your diagram should be divided into Presentation, Business and Data layers. A suggested approach is that the business object layer should only handle 1 train at a time with the associated 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 Train
    b. Add a new passenger booking
    c. Show a list of all trains running on a specific day

   d. Show a list of all passengers booked onto a specific train (name, coach, seat)

4. Create a Unit test for your Train 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 hints on Implementation

1. 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.

2. 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.

3. 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

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

Before demonstrating add the following trains:

Item			
Train-id	1S45	1E05	1E99
Departure	London (Kings Cross)	Edinburgh (Waverley)	Edinburgh (Waverley)
Destination	Edinburgh (Waverley)	London (Kings Cross)	London (Kings Cross)
Type	Express	Stopping	Sleeper
Intermediate		Peterborough, Darlington, York	
Departure time	10:00	12:00	21:30
Departure day	01/11/18	01/11/18	01/11/18
SleeperBerth	false	false	true
FirstClass	true	true	false

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 Trains (including validation)	Basic	6	All works 6 marks. Partially works 2-5 marks. Major errors 1 mark.
Add Booking	Basic	3	All works 3 marks. Partially works 2marks. Major errors 1 mark.
Calculate Train Fare	Basic	3	All works 3 marks. Partially works 2marks. Major errors 1 mark.
Handle multiple Trains and bookings	Basic	2	Works 2 marks.
Persistance	Advanced	4	All works 4 marks. Read or write only 2 marks.
Advanced Search	Advanced	4	Works 4 marks, 1-3 marks for partially working

Total ___/ 22

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.

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

Total ___/30