7SENG011W Object Oriented Programming – Coursework (2024/25)	
Module Leader	FRANCESCO TUSA (F.TUSA@WESTMINSTER.AC.UK)
Unit	Coursework
Weighting	50%
Description	Design and Development of a basic *Hotel Booking Management System*
Learning Outcomes Covered in this Assignment	This assignment contributes towards the following Learning Outcomes (LOs): *(LO1)* Demonstrate an understanding of a significant object-oriented programming language and development environment. *(LO2)* Critically evaluate the extensive classes provided by the language and demonstrate their use in the design and construction of an application. *(LO3)* Demonstrate a thorough understanding of a contemporary framework (e.g., Java SE), together with a critical understanding of the issues and problems associated with the use of third-party packages and frameworks. *(LO4)* Critically evaluate and communicate their work by both written and oral means.
Handed Out Date	Friday, 22nd of November 2024
Due Date	**Tuesday, 7th of January 2025—submissions by 1 pm via Blackboard**
Expected deliverables	• The system Class Diagram and Design Report (based on the provided template); • A ZIP file with the Java source files (.java only) (and the Netbeans project files) of your software project; • A link to a recorded video demonstration of the implemented system (e.g., link to YouTube or Google Drive).
Submission Method	Electronic submission on Blackboard via the provided link. The submitted files should have the following naming format: *StudentNumber_firstName_lastName_filename.xyz*
Type of Feedback and Due Date	Electronic feedback and marks will be provided via the module's Blackboard board within *15 working days* from the submission deadline. **All marks will remain provisional until formally agreed by an Assessment Board.**

Assessment regulations

Please refer to the *University Assessment Regulations* on the following website:

http://www.westminster.ac.uk/study/current-students/resources/academic-regulations

for a detailed description of *how you are assessed*, *penalties* and *late submissions*, what constitutes ***plagiarism***, etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, ten marks will be deducted from the final mark as a penalty for late submission, except for work which obtains a mark in the range of 50-59%, in which case the mark will be capped at 50%. Suppose you submit your coursework more than 24 hours or more than one working day after the specified deadline. In that case, you will be given a zero mark for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

Mitigating Circumstances

It is recognised that, on occasion, illness or a personal crisis can mean that you fail to submit a work on time. In such cases, you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the appropriate Assessment Board, which decides whether the zero mark will stand.

Plagiarism

By submitting the work through Blackboard, you acknowledge that this is solely your work. Any code not created by you MUST be commented as such. Any code discovered not to have been created by you will mean that the work will be submitted to academic standards for a potential assessment offence, which may result in a zero mark in the component or whole module.

Coursework Description

Problem Statement

This coursework assesses the knowledge and skills you acquired about Object Oriented Programming (OOP) throughout the module. The specific problem to be solved is designing and implementing a basic *Hotel Booking Management* software system.

Two types of users will interact with the system via a text menu that provides them with access to specific functionalities:

- An *admin* can add and remove rooms from the booking systems, list registered rooms and generate reports with the booking information.
- A *customer* can check the list of rooms available at a given timeframe using various criteria and make (or delete) a booking.

For simplicity, the implementation of registration and login functionalities for the above types of users and a database are not requested for this module coursework.

System Design and Documentation (20 marks)

As part of the coursework, you are asked to analyse the problem, identify a set of required system functionalities, and provide the related design and implementation. **Submitting the following two items is mandatory: your final mark will be capped at 40% if any are missing**.

UML Class Diagram (mandatory—10 marks)

A UML Class Diagram that describes the OOP system design. The marking of the diagram will be done based on the following criteria:

- All the required entities are correctly represented with their relevant attributes and methods;
- Object relationships (i.e., associations, aggregations, compositions) and their multiplicities, as well as the class relationships (i.e., generalisations), are modelled correctly;
- The design is consistent with the software implementation that was submitted.

Project Design Report (mandatory—10 marks)

A report (based on the provided template, with an expected length of no more than 1000 words) that highlights your reflection on the design process, i.e., the role of each class, interface, abstract class, of their members and relationships.

You will also have to reflect on the OOP principles you applied, i.e., Abstraction, Encapsulation, Inheritance and Polymorphism and how they have been applied in your design and implementation.

System Functionalities (implementation: 80 marks)

The implementation of the system functionalities must be based on the topics taught during the module—no other solutions will be accepted without proper justification and

explanation. For example, features related to sorting must be implemented using the *Comparable* or *Comparator* interfaces.

Types of Rooms Modelling (10 marks)

The system should support different types of rooms, i.e., *StandardRoom*, *DeluxeRoom,* and *SuiteRoom*, whose modelling should follow OOP design practices and be derived from a common parent class. Regardless of their type, all the rooms will have information about the *room number*, *floor*, *occupancy* (single, double or triple) and *price* (per night).

Properly apply the *encapsulation*, *inheritance* and *polymorphism* principles to ensure the design allows the modelling of the following additional information:

- *StandardRoom* has a given number of *Window*s (at least 1);
- *DeluxeRoom* room has a *Balcony* (of a given size in m^2) and a *View* (either sea view, landmark view, or mountain view);
- *SuiteRoom* has a *LivingArea* (of a given size in m^2), a *NumberOfBathrooms* and may or may not have a *Kitchenette*.

Room Booking Management (10 marks)

The system should keep track of the dates when each room has been booked. It should be assumed that a booking includes *check-in* and *check-out* dates. The check-in is always expected at 3 pm on the day the booking starts. Similarly, the check-out is always expected at 11 am on the last day of the booking. Therefore, for simplicity, only consider the date as a combination of day, month, and year.

A *Room* can have multiple non-overlapping bookings. For instance, *Room 1* can have 2 bookings: *Booking 1* with check-in on 10/11/2024, check-out on 15/11/2024 and *Booking 2* with check-in on 15/11/2024, check-out on 18/11/2024. Define an interface *Overlappable* with a method *bool overlaps(Booking other)* for checking whether two *Booking* objects overlap.

When a new *Booking* is made, the information about the requesting *customer* must be added, namely their *name*, *surname*, *date of birth*, and contact details. Information about the *ID number* should also be included for later use at check-in time.

Hotel System Admin and Customer Functionalities (60 marks)

A class *HotelSystem* containing the program's main logic has to be developed. This class implements the *HotelManager* and *HotelCustomer* interfaces, with the expected functionalities reported below. You should define a proper set of methods and data structures required to support the functionalities indicated by those interfaces and also provide a thoughtful description of your design and implementation decisions in the report.

HotelManager (25 marks)

Defines the operations that the *admin* can perform.

1. Add a new room to the system. **(3 marks)**
2. Delete an existing room from the system using the room number. **(2 marks)**
3. List all the registered rooms and the associated information, sorted according to the following options:

   a. By room number in ascending order. **(6 marks)**
   b. By room floor number with the top floor rooms displayed first. **(6 marks)**
4. Generate a text file with information about all the room bookings in a given timeframe. The output data should include the room information and the customers who booked it. **(8 marks)**

HotelCustomer (25 marks)

Defines the operations that a *customer* can perform.

5. List all available rooms on a specified timeframe, sorted by price (lower price first). **(7 marks)**
6. List all available rooms on a specified timeframe that match a type (e.g., SuiteRoom) and occupancy (e.g., double). **(7 marks)**
7. Book a room in a specified timeframe using the associated room number. For simplicity, information about the customer booking the room, i.e., full name and contact details, can be provided as input while booking (there is no need to implement member registration and login features). A booking ID is generated as output if a booking is successful, with information about the booking cost. **(8 marks)**
8. Delete an already booked room by providing the booking ID. **(3 marks)**

Text Menu (10 marks)

Once started, the program should show on the screen a *text menu* for:

9. The *customer*, with the functionalities defined by the *HotelCustomer* interface, plus an additional option to access the *admin* menu **(5 marks)**

10. The *admin*, with the functionalities of the *HotelManager* interface and the option to return to the *customer* menu. **(5 marks)**

Coding Standards and Robustness

The evaluation of the system functionalities mentioned above **will not only consider the code's accuracy**, i.e., whether the program generates the *correct output* for each set of *input data*.

The evaluation process will also consider the application of **well-known OOP principles**, **code robustness** (including *error handling* and *input validation*), and adherence to recognised **coding standards**, such as *readability*, *code organisation* and *reuse*, proper use of *comments*, and the choice of *meaningful identifiers*.

Video with System Explanation and Demonstration (mandatory)

A pre-recorded **video with voiceover**—**not exceeding 20 minutes**—must be submitted and available for download or streaming. **The video content exceeding the maximum limit will not be considered for the marking**.

To receive the marks corresponding to a functionality you implemented, a demonstration (test) of that functionality and an explanation of the related code implementation must be included in the video.

The video should show a **window with the code** and a **window with the console terminal**. When a system functionality is demonstrated (tested), you must explain the related code. Your understanding of OOP design and implementation principles and your ability to explain the code will be assessed and used to mark each implemented functionality.

Therefore, if you fail to provide a video recording of your system according to the above guidelines, the mark shall be 1%-20% (based on the marking of the submitted design artefacts only).

Viva

Your final mark will be determined based on your ability to showcase your understanding of the code implementation in the video.

Additionally, the evaluator is authorised to request a viva (oral examination) and/or initiate a misconduct case if there are any doubts about the originality of your work. **Please carefully review your submission and ensure it is your original—and individual—work.**

Marking Rubric

7SENG011W Object Oriented Programming Coursework 2024/25						
Student Name:		**Student Number:**		**Marker:**		
Coursework Item		**Assessment Criteria and Mark Range**			**Mark**	**Feedback**
		System Design (Mandatory Submission—20 marks)				
UML Class Diagram (10)	Not Done (0) **Overall Mark capped to 40**	Model lacks significant detail or it is not aligned with the coursework specification *(1-3)*	Core classes correctly identified with relevant attributes and methods; some issues with the modelling of the relationships *(4-7)*	All classes correctly identified with relevant attributes and methods; correct modelling of relationships and multiplicities. The diagram is consistent with the implementation *(8-10)*		
Design Report (10)	Not Done (0) **Overall Mark capped to 40**	The report lacks significant detail or it is not aligned with the provided Class Diagram *(1-3)*	The report discusses the core entities of the system but there is no clear justification of why they have been included in the design *(4-7)*	The report is well written and properly describes the system design, highlighting excellent knowledge and application of OOP principles *(8-10)*		
		System Implementation (80 marks)				
Modelling of Rooms (10)	Not Done or not presented in the video (0)	The implementation does not follow proper OOP principles or it is not aligned with the class diagram; or not properly presented in the video (1-3)	A working implementation with abstraction and encapsulation; some minor issues, e.g., some missing attributes, methods or usage of not efficient data structures (4-7)	A working implementation that follows all the relevant OOP principles, including inheritance and polymorphism; makes proper usage of data structures ad object relationships (8-10)		
Booking a Room: check-in and check-out dates, overlapping and customer information (10)	Not Done or not described in the report (0)	The implementation does not follow proper OOP principles or it is not aligned with the class diagram; or not properly described in the report (1-3)	A working implementation with abstraction and encapsulation; some minor issues, e.g., some missing attributes, methods or usage of not efficient data structures (4-7)	A working implementation that follows all the relevant OOP principles, makes proper usage of data structures and object relationships (8-10)		
	Not Done or not presented in the video	**Not** working or **partially** working due to **major issues**; or **not properly presented** in the video	Working with **minor issues** or **not fully** adhering to relevant OOP principles	**Fully** working, well implemented following relevant OOP principles, with proper error/exception handling		
IRentalManager (25)						
1. Add Room (3)	0	1	2	3		
2. Delete Room (2)	0	1		2		
3a. List rooms sorted by room number (ascending order) (6)	0	(1-2)	(3-4)	(5-6)		
3b. List rooms sorted by floor number (top floor rooms first) (6)	0	(1-2)	(3-4)	(5-6)		
4. Generate a booking report on a text file (8)	0	(1-3)	(4-6)	(7-8)		
IRentalCustomer (25)						
5. List all available rooms on a specified timeframe, sorted by price (lower price first) (7)	0	(1-2)	(3-5)	(6-7)		
6. List all available rooms on a specified timeframe that match a type and occupancy (7)	0	(1-2)	(3-5)	(6-7)		
7. Book a room in a specified timeframe using the associated room number (8)	0	(1-2)	(3-5)	(6-8)		
8. Delete an already booked room by providing the booking ID (3)	0	1	2	3		
Text Menu (10)						
9. User Menu (5)	0	(1-2)	(3-4)	5		
10. Admin Menu (5)	0	(1-2)	(3-4)	5		
		Video (Mandatory Submission—Contributes to the marking of each of the above items)				
Video with System Explanation and Demonstration	Not Submitted **Overall Mark capped to 20**	Submitted				
				Final Mark	0	
				General Feedback		