

<b>CSYM025: Visual Object Software</b>			
Date of Issue	<b>Friday, 2<sup>nd</sup> December 2022</b>	<b>Last Date for Submission:</b>	<b>Sunday, 22<sup>nd</sup> January 2023 23:59:59</b>
		Module Tutor:	Dr Suraj Ajit

### **Guidelines – Please read carefully:**

- I. The University of Northampton’s Policy on Academic Integrity and Misconduct will be strictly implemented.
- II. This is not a group project and by submitting this assignment you are asserting that this submission is entirely your own individual work. **Sharing your work with another student or submitting code that was written by someone else may be deemed academic misconduct.**
- III. If you have used any external code that you did not write, you must clearly reference it within your report. See <https://libguides.northampton.ac.uk/c.php?g=683675&p=4918336>
- IV. You must submit all items of the assessment according to the submission procedure stated in this document. **Failure to follow the submission procedure may result in a penalty or capped grade.**
- V. This assignment/module is used to assess your object-oriented design and programming skills. Hence, **any use of databases and query languages (e.g., SQL) is strictly prohibited and would result in a fail grade.**

### **Brief:**

Design, implement and test a rental property management system using object-oriented principles and data structures in Java FX. Design should include class diagrams. Testing should include both white box (JUnit tests) and black box (test logs). Consider the datasets supplied together with following requirements:

### **Basic System Requirements:**

The system must allow the user to:

1. Display details of properties available for rent
2. Register details of new customer
3. Rent property to customers (update availability status of property) for a particular period
4. Generate an invoice with relevant information (including deposit of six months’ rent and agent fee of 20 percent of 1 month rent)
5. Permanent storage and retrieval of data using object serialisation

### **Enhancements (in order of importance – high to low):**

1. Generate invoice at the end of tenancy (including any deduction for damage)
2. Input/update property information and points of interest
3. Import/export both datasets in csv format (object mapping)
4. Display distance of property from points of interest (e.g., distance from train station, nearest supermarket)
5. Search/Sort property by price, listing date, no. of bedrooms, history and so on.

A good quality system should be easily adaptable/extensible. You may include any other useful features relevant to this application. You need to clarify and extract detailed requirements from the tutor during class hours and are encouraged to use an agile methodology for software development.

## Deliverables:

All requirements (A, B and C below) **MUST** be delivered to achieve a passing grade for this assignment.

### A) Technical Report

The report should consist of the following sections (in the same order):

1. A list of all the features implemented in a tabular format with remarks. For example:

Functionalities	Implemented (Partial/Full)	Remarks
Display details of properties available for rent	Full	Error validation
Register details of new customer	Full	No error validation
Rent property to customers (update availability status of property) for a particular period	Partial	Error validation

2. UML Class Diagrams showing relationships between the main classes in the model
3. Brief explanation of the main sections/fragments of the code. Provide information that would be useful for another developer (not an end user!) who may want to extend/maintain your system. You may want to refer to the class diagrams to explain code.
4. Screenshots of the system showing key features
5. Evidence of Testing:
  - a. Blackbox Testing: Test **logs** providing information of all the tests carried out (including any failed tests for functionality not implemented)
  - b. Whitebox Testing: **Code Listing** of the JUnit test cases.
  - c. List of any bugs and/or weaknesses in your system (if you do not think there are any, then say so). Bugs that are declared in this list will lose you fewer marks than ones that you do not declare.
6. References (use Harvard referencing):  
If you have borrowed some code from elsewhere (e.g., from a book or some resource on the web you **must** indicate clearly what they are and include references).

### B) Source Code

The source code must be well documented with necessary comments. Consistent and clear indentation of the code is also important. Source code needs to be submitted in two forms:

- (i) As a single ZIP archive (.zip file consisting of all “.java” files, unit tests, data files, executable jar).
- (ii) A commented full listing in a separate Word document named “Full Source Code Listing”.

### C) Video Demonstration

In addition to the report, you **must** submit a video demo (URL) of your assignment. The demo should be no longer than 10 minutes. Your face and voice need to be clear for the marker to see/hear. This should include concise code explanation and a walkthrough of the functionalities. **The module tutor reserves the right to invite you for a viva-voce. Poor demo, viva or report could negatively influence all the marking criteria and may result in a fail grade.**

## Submission Procedure:

- E-Submission of documents through Turnitin on NILE as TWO separate WORD documents.  
[Document 1 = **Report** & Document 2 = **FullSourceCodeListing**]  
To do this, go to the NILE site for this module and use the link labelled ‘Submit your work’.
- E-Submission of a single ZIP archive that contains all the source code files (.java), unit tests, data files and ReadMe file. The archive must be named with your student ID, e.g., *12345678.zip* where

12345678 is your student ID. To do this, go to the NILE site for this module and use the link labelled 'Submit your work'. Clicking on the link (*SourceCodeSubmission*), will take you into the submission form, where you can upload your ZIP archive using the 'Attach File' button (*Browse for Local File*). Finally, click the *Submit* button.

- When submitting your video demonstration, use of Kaltura (<https://video.northampton.ac.uk/>) is recommended. You must ensure that the video link is accessible to the marker (do not set it to private access).
- Failure to follow the above submission guidelines may result in a capped or fail grade.

## Marking Criteria:

The grade for this assignment will form 100% of the overall assignment grade for the module. Marks are split according to the following scheme. In general, the following criteria will act as a guide to what you should expect:

	A	B	C	F	G
Design (20%)	Excellent design of program and user interface. Adherence to object-oriented principles. Class diagrams are very well designed and presented.	Good quality design of program and user interface. Adequate adoption of object-oriented principles. Class diagrams are well designed and presented.	Satisfactory design of program and user interface. Class diagrams are satisfactory.	Faulty design of class diagrams. Very little discussion of the overall design.	No submission or no submission of merit
Functionality (35%)	All criteria for (B) and many significant additional features.	All criteria for (C) and some significant additional features.	Most basic system requirements are met.	Most basic system requirements are not met.	No submission or no submission of merit
Testing (15%)	Evidence of both white box and black box testing with extensive code coverage.	Evidence of both white box and black box testing with good code coverage.	Evidence of either white box or black box testing with satisfactory code coverage.	No evidence of any white box or black box testing.	No submission or no submission of merit
Code quality and Efficiency (10%)	Code is very well structured to enable white box testing, reusability and debugging. Excellent work on error handling.	Code is well structured to enable white box testing, reusability and debugging. Good work on error handling.	Some thought has been given on how the code is structured. Some work on error handling.	Hardly any thought on how the code is structured.	No submission or no submission of merit
Codio exercises/engagement (20%)	Timely submission of all exercises. Excellent implementation and documentation.	Timely submission of most exercises. Very good implementation.	Satisfactory submission and implementation of exercises.	Poor engagement and untimely submission for exercises	No submission