Module Title: Object Oriented Programming			
Module Code: CMP5332			
Assessment Type (delete as appropriate) Coursework	**Level** 5	**Weighting** 100%	**Word Count** N/A
Submission Date Details below	**Submission Time** Details below	**Module Leader** Dr. Quanbin Sun	**Time Limit (for in person or oral assessments)** 20 minutes for viva in D2.

Assessment Brief

Assessment Information	
Assessment Task (with genre/type)	D1 [Individual]: Weekly Submission of Lab Exercises (20%) D2 [Group]: Group Project: Program implementation, source code documentation, and viva (80%)
Assessment Title	Object Oriented Programming Assessment Brief
Things to include:	Deliverable 1 will test your understanding of topics learns each week. Deliverables 2 will involve working with a partner to implement a full solution of the provided software system specification, as well as creating the relevant tech document and present your solutions in a Viva. **D1: Weekly Lab Submission – each Friday (from week 2) (20%):** Task will be released at the beginning of each week, and the submission deadline will be the Friday of the following week. **D2: Group Project – 9th Jan 2026 (80%):** One group member should submit a single **zip** file with **source code + relevant documents about the following system implementation:** This programming assignment is to apply the object-oriented programming principles covered in tutorials and lectures to develop an application for an interactive library system. The aim of the exercise is to enhance students' experience of programming by applying object-oriented programming principles to a larger problem through developing and testing a complete command line and GUI based application. Your software system must be developed **according to the detailed requirements specification** provided on Moodle. ***Objectives*** Automation of library management system has been studied over the years by many stakeholders. The main purpose of such system is to

mechanise library housekeeping operations predominantly by computerisation. In this coursework you will create an interactive 'prototype' library management application to focus on basic operations in a library like adding new books, new members, searching books and members and facility to borrow and return books; and updating information about books and members. The application should store all the information about books and members in a persistent storage e.g. files.

Sample Prototype Application
A partial implementation of a prototype library management system will be made available on Moodle. The prototype is implemented as a standard Java project and contains partial implementation for the set of commands that provides the different functionalities for the library system:

listbooks	print all books
listpatrons	print all patrons
addbook	add a new book
addpatron	add a new patron
showbook [book id]	show book details
showpatron [patron id]	show patron details
borrow [patron id] [book id]	borrow a book
renew [patron id] [book id]	renew a book
return [patron id] [book id]	return a book
loadgui	loads the GUI version of the app
help	prints this help message
exit	exits the program

The prototype uses text files as its backend storage to store the information of books, patrons and loans. These files are named as "books.txt", "patrons.txt" and "loans.txt" respectively, and are placed under the "resources/data" project folder. The different properties of each entity are stored as a single line in the text file, where each property is separated by double colon delimiter (i.e. ::). For example, the following book information:

Id: 23
Title: Introduction to Java
Author: Smith Owen
Publisher: Penguin
Publication Year: 2023

Can be stored as the line shown below in the file books.txt:

23::Introduction to Java::Smith Owen::Penguin::2023::

See the **assessment criteria** below for more information about the functionalities required to be completed and the associated marks.

The Viva:

- Will be arranged during Semester 1 Assessment Weeks 2 and 3, i.e., 12 - 23 January 2026.
- You can only attend viva after submitting D2.

	• Failure to attend the viva might result in a NIL mark of D2.

Completion of this assessment will address the following learning outcomes:	
1	Demonstrate knowledge of the fundamental principles of object-oriented programming.
2	Apply object-oriented principles to design and implement programs from high level requirements specifications.
3	Use a unit testing framework in the design, testing and debugging of object-oriented programs.
4	Follow standard software development practices including pair programming and code review.
5	Use and create technical documentation for object-oriented code.

Use of Artificial Intelligence is PERMITTED:

For this assessment if you use generative Artificial Intelligence (AI) in the process of completing this assessment you MUST set out clearly the following:

- *WHY you used generative AI*
- *WHAT it was used for*
- *WHICH AI was used; and*
- *If any generated content has been used directly in this submission, if so where.*

Note that this declaration does NOT contribute towards the word count for the assessment.

You will also have to confirm in your declaration that the work remains yours and you have intellectual ownership of it. You may be invited for an informal conversation to discuss the approach to your assessment. A failure to disclose the use of AI, or the use of a misleading description of its use will be considered academic misconduct. As a result, keeping good records of your interactions is strongly advised.

Student AI Guidelines

Academic Misconduct

Academic misconduct is conduct that has or may have the effect of providing you with an unfair advantage by relying on dishonest means to gain advantage and which therefore compromises your academic integrity.

The Academic Misconduct Procedure sets out the process we will follow, and the penalties we may apply, in cases where we believe you may have compromised your academic integrity by committing academic misconduct.

Late Submission

Where you are required to submit assessment by a certain deadline (for example essays, case studies or physical artifacts) but you fail to meet the deadline, your mark will be reduced in accordance with the Late Submissions Policy. This Policy does not apply where the assessment is 'in-person' such as exams and in-class tests.

For support on submission, please review the Submitting Your Assignment e-learning resource.

---------- *FOR STAFF ONLY*----------

Staff Use Only: Retained Work Record	
Details of Retained Work:	D1 Lab submissions + D2 Code/Word/PDF
Location of Retained Work:	Moodle: https://moodle.bcu.ac.uk/course/view.php?id=98576
Type of Work Retained *(tick all that apply)*:	Code + Written report + Viva recording where applicable

D1 [Individual]: Lab Exercises-- Table of Marking Criteria (20%)

Assessment Criteria à	1. Quality of code implementation (targeting LO 1, 2 and 4)
Weighting:	20%
Grading Criteria 0 – 29%	Significantly incomplete program. Does not compile and execute.
30 – 39%	Program compiles but gives errors during execution.
40 – 49%	Program compiles and demonstrates only few basic features of the given specification.
50 – 59%	Program demonstrates most of the basic features of the given specification.
60 – 69%	Program demonstrates all of the basic features of the specification and some other advanced features of the design.
70 – 79%	Program demonstrates all of the basic features of the specification and some other advanced features of the design.
80 – 100%	Program demonstrates all of the basic features of the specification and all other advanced features of the design.
Checklist	Source Code

D2 [Group]: Group Project -- Table of Marking Criteria (80%)

The implementation should achieve the following (in those orders):

To add new Books to the system. The application should store at least the following information for each book: ID, Title, Author, Publisher and Publication Year. In addition, the Book should have a reference to a Loan object (see below) to provide information about its Availability (i.e. if the book is issued to someone) and Due Date (i.e. when the book must be returned). – *this functionality is already implemented*

The listing of all Books stored in the system is also implemented.

Achieving a mark of 40% to maximum of 49%

The application must implement all the following:

- Add new Patrons (members) to the system. System should store at least the following information for each member: ID, Name, Phone Number and List of Books Borrowed.
- List all Patrons stored in the system.
- Issue Books to Patrons. When a book is issued to a Patron a Loan object must be created holding a reference to the Book being issued, to the Patron borrowing the Book and the Due Date of the Loan. This object should be added to the Book and the book should be added to the Patron's list of borrowed Books.
- Return issued books. When a member returns a book the status of the returned book should be updated to reflect its availability. Also book must be removed from the Patron's list of borrowed Books.
- Save the status of the system to the backend storage (i.e. text file storage) when the system is closed. The library data should be stored in three different files (books.txt, patrons.txt and loans.txt). A sample format to save the different properties for each object is given in the Sample Prototype Application section above. When the system starts it should load the status of the library from the text files to the memory.

Achieving a mark of 50% to maximum of 59%
The application must implement all the above and the following:

- Add a publisher property to the Book object and make the appropriate changes to the program to ensure that this information can be captured when a new Book is created. Also ensure that this information will be stored to and correctly loaded from the file storage.
- Add an email property to the Patron object and make the appropriate changes to the program to ensure that this information can be captured when a new Patron is created. Also ensure that this information will be stored to and correctly loaded from the file storage.
- Implement Unit Tests to validate and demonstrate that the above changes made to the Book and Patron classes work as expected.

Achieving a mark of 60% to maximum of 69%
The application must implement all the above and the following:

- Extend the prototype implementation for the GUI application provided to allow for the following basic functionalities:
  - Display a popup window that will show the Patron details if a Book is on loan.
  - List all Patrons and their details including the number of books they have on loan.
  - Display a popup window that will show the Books details if a Patron has Books on loan.
  - Display a popup window when the "Add" submenu of the "Patrons" menu item is selected. The popup should display a form that allows the addition of a new Patron to the system.

- Extend the functionality of the library system to allow for storing data to the file storage after the execution of commands that change the state of the system (e.g. “addbook”, “renewbook”). If the system fails to store the data on the file storage due to an error (e.g. file is already in used or corrupted), the program must inform the user and rollback any changes made to the system prior to the error. You can change the file permission to “read-only” in order to test this functionality.

Achieving a mark of 70% to maximum of 79%
The application must implement all the above and the following:

- Remove (hide) existing books from the system. When a book is removed, it should not appear in the books view. Instead of completely deleting a book, use a Boolean property in the Book class to indicate whether the book is deleted. Change the affected functions appropriately to return only the books that are not deleted.
- Remove (hide) existing Patrons from the system. When a patron is removed, it should not appear in the patrons view. Instead of completely deleting a patron, use a Boolean property in the Patron class to indicate that the patron is deleted. Change the affected functions appropriately to return only the books that are not deleted.
- Impose a limit on the maximum number of books that a patron can borrow and use that during issuing books for a patron.
- Extend the implementation for the GUI application to add the Delete functionality for both Books and Patrons using the GUI.
- Add Javadoc documentation for the newly created methods.

Achieving a mark of 80% and over
The application must implement all the above and the following:

- Keep record of all loan history for Patrons. To implement this functionality you need to introduce a Boolean property in the Loan class to indicate whether the loan is terminated and a property that will hold the date that the book was returned. You have the flexibility to decide on how to complete the implementation of this functionality.
- Complete the implementation for the GUI application provided to allow for the following functionalities:
  - Implement the Borrow functionality to borrow a book using the GUI.
  - Implement the Renew functionality to allow for renewing a book loan using the GUI.
  - Implement the Return functionality to return a book using the GUI.

Assessment Criteria à	1. Code Quality (targeting LO 1, 2 and 5)	2. Application Implementation (targeting all LOs)	3. Project Presentation (targeting LO 1 and 4)

Weighting:	20%	45%	15%
Grading Criteria 0 – 29%	Code disorganised, indentation astray, comments absent or scattered.	Significantly incomplete. Does not compile	Incomplete presentation.
30 – 39%	Little evidence of thought applied to which code should be implemented in which module or class.	Implementation is missing features to achieve a pass as described for the application.	Presentation provides only few screen shots of the implemented application. Usage scenarios are not demonstrated.
40 – 49%	Application functionality implemented but at the cost of good code structure. Some comments.	Source code implements specific features for the application as defined in the 40% to 49% criteria	Presentation describes detailed usage scenarios for all the features related to this mark band.
50 – 59%	Code appropriately commented and structured based on appropriate class and object design.	Source code implements specific features for the application as defined in the 50% to 59% criteria	Presentation describes detailed usage scenarios for all the features related to this mark band.
60 – 69%	Good class and object design with good judgement evident in code layout and design	Source code implements specific features for the application as defined in the 60% to 69% criteria	Presentation describes detailed usage scenarios for all the features related to this mark band.
70 – 79%	Good class and object design, well-structured code, appropriate method and variable names with enough comments to understand the code.	Source code implements specific features for the application as defined in the 70% to 79% criteria	Presentation describes detailed usage scenarios for all the features related to this mark band. Good presentation skills are demonstrated.
80 – 100%	Excellent class and object design and a high level of attention to detail achieving high quality in code design and coding documentation.	Source code implements extra features as described in the specific criteria for the application.	Presentation describes detailed usage scenarios for all the features related to this mark band. Excellent presentation skills are demonstrated.
Checklist	Source Code	Source Code	Project presentation