being marked. You must retain a copy of all electronic work you have submitted and re-submit if requested.

Learning Outcomes to be Assessed:

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.**

Assessment Details:

Title: Interactive Library System (ILS)

Type: Coursework submission

Style: Implementation/ Presentation

Rationale:

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 implementation is based on the topics covered in the lectures and lab sessions for this module.

There are mainly two reasons behind the selection of library system as the topic for this coursework. Firstly, the students are familiar with the dynamics of a library system, therefore they do not need to spend time in order to understand the functionalities of the system that they are going to develop. Rather they can directly be involved in the design and development of the command line system by applying the programming knowledge and skills they learnt. Secondly, implementing such programs make it easier to interact with the developed system and demonstrate a direct relationship between the user interaction and the system classes, methods and objects.

Introduction

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** for component B below for more information about the functionalities required to be completed and the associated marks.

Additional information:

Your group are required to give a 15 minutes presentation covering the following:

- 10 Minutes - Describe the protocol design and implementation for your library system application
- 5 Minutes - Demonstrate your implementation

For advice on writing style, referencing and academic skills, please make use of the Centre for Academic Success: https://icity.bcu.ac.uk/celt/centre-for-academic-success

Additional information:

For advice on writing style, referencing and academic skills, please make use of the Centre for Academic Success: Centre for Academic Success - student support | Birmingham City University (bcu.ac.uk)

Workload:

This is the main assessment for the module (weighing 80% of the overall mark), and is expected to take approximately 40 hours of work in total. This is roughly equivalent to a 4000-word report.

Transferable skills:

This assignment develops numerous transferable skills that are highly prized by employers. These include:

- Programming and OOP modelling skills
- Software application troubleshooting, testing and debugging
- Oral and written communication skills
- Time management and planning

This assessment should help students to develop various transferable skills which are necessary to obtain a decent job in the future.

Problem Solving
Being a programmer involves the ability to identify real-world issue and coming up with a technological solution to address it. This requires having strong analytical skills to understand the issue and evaluate different solutions in order to find the one that best fits the problem.

Ability to express your thoughts effectively
Using flowcharts provide an effective way of communicating the logic of a system to all people involved in the project regardless of their background and expertise. They can serve as an excellent bridge between software developers and end users.

Project evaluation skills
It is the process that critically examines a program. It involves collecting and analysing information about program activities, characteristics, and outcomes in relation to customer specifications. Its purpose is to make judgements about a programme that you have developed, to improve its effectiveness and inform programming decisions.

Time management and planning
Design documentation serve as the basis for the estimating, scheduling, and validating efforts.

Marking Criteria

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 for Component A (Lab Exercises):

Table of Assessment Criteria and Associated Grading Criteria

Assessment Criteria →	**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

Assessment Criteria for Component B (Program Implementation):

Table of Assessment Criteria and Associated Grading Criteria

Assessment Criteria →	**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

Submission Details:

Format: All work should be submitted on Moodle as a single zip file. The zip file should contain your PowerPoint presentation and all of your code with associated project files. Include any additional components that are required for your application to work. The zip file should be structured so that the location of its contents is clear and unambiguous. Please submit only one version of your work.

You must submit a zip file containing:
(Group Submission)
1. The Java source code for your product implementation.
2. A PowerPoint Presentation describing the design and implementation of your ILS. Your presentation should clearly indicate individual contribution for each member of the team.

Also, the zip file should be named using the following format StudetName_studentID.zip
For example: ARTawil_123456789.zip

Regulations:

- The minimum pass mark for a module is 40%.
- Re-sit marks are capped at 40%

Full academic regulations are available for download using the link provided above in the IMPORTANT STATEMENTS section.

Late Penalties
If you submit an assessment late at the first attempt, then you will be subject to one of the following penalties:

- if the submission is made **between 1 and 24 hours** after the published deadline the original mark awarded will be reduced by **5%**. For example, a mark of 60% will be reduced by 3% so that the mark that the student will receive is 57%.
- if the submission is made between **24 hours** and **one week (5 working days)** after the published deadline the original mark awarded will be reduced by 10%. For example, a mark of 60% will be reduced by 6% so that the mark the student will receive is 54%.

- **if the submission is made after 5 days following the deadline, your work will be deemed as a fail and returned to you unmarked.**

The reduction in the mark will not be applied in the following two cases:

- the mark is below the pass mark for the assessment. In this case the mark achieved by the student will stand.
- where a deduction will reduce the mark from a pass to a fail. In this case the mark awarded will be the threshold (i.e.,40%).

Please note:

- **If you submit a re-assessment late, then it will be deemed as a fail and returned to you unmarked.**

Feedback:

Marks and Feedback on your work will normally be provided within 20 working days of its submission deadline via Moodle.

Where to get help:

Students can get additional support from the library for searching for information and finding academic sources. See their iCity page for more information: http://libanswers.bcu.ac.uk/

The Centre for Academic Success offers 1:1 advice and feedback on academic writing, referencing, study skills and maths/statistics/computing. See their iCity page for more information: https://icity.bcu.ac.uk/celt/centre-for-academic-success

Additional assignment advice can be found here: https://libguides.bcu.ac.uk/MA

Fit to Submit:

Are you ready to submit your assignment – review this assignment brief and consider whether you have met the criteria. Use any checklists provided to ensure that you have done everything needed.

Check Link on Moodle