**Computing Department**

BSc Contemporary Software Development

|  |  |
| --- | --- |
| **Module Name:** | **Software Development** |
| **Assignment No:** | **CA2** |
| **Assignment Title:** | **Collaboratively build a software artefact** |
| **Lecturer:** | **Teresa Deeney** |
| **Release Date:** | **8th November 2023** |
| **Submission Date:** | **1st December 2023** |
| **Indicative Content Covered by Assignment** | Implement programs using interfaces, abstract classes and exception handling.  Select and apply appropriate algorithms for manipulating data in data structures.  Analyse a solution in terms of its complexity and performance profile and gather metrics to justify the appropriateness of selected algorithms and data structures.  Design and implement appropriate unit testing strategies for classes.  Collaboratively build a software artefact. |
| **Definition of Plagiarism:** | Plagiarism is the act of taking or copying someone else’s work, including another student’s, and presenting it as if it were one’s own. Plagiarism is said to occur when ideas, texts, theories, data created artistic artifacts or other material are presented without acknowledgement so that the person considering the work is given the impression that what they have before them is the student’s own work when it is not. Plagiarism also occurs when a student’s own work is re-presented without being properly referenced. Plagiarism is a form of cheating and is a disciplinary offence. |
| **Student Declaration:** | I declare that this is my own work and that any material I have referred to has been accurately and consistently referenced. I have read and understand the definition of plagiarism given above. If it is shown that material has been plagiarised, or I have otherwise attempted to obtain an unfair advantage for myself or others, I understand that I may face sanctions. A mark of zero may be awarded and the reason for that mark will be recorded on my file. |
| **Student Name:** |  |
| **Student Signature:** |  |
| **Date Submitted:** |  |
| **Mark:** |  |
| **Assessor:** |  |

**Software Development CA2**

Create a Java-based library catalogue system that manages information about resources, authors and users.

The objective of this assignment is to design and implement a file-based data management system using Java. Students will work in groups to create a system that stores, retrieves and manipulates data objects, making use of Java interfaces, abstract classes, regular classes, stacks, queues and/or linked lists (one or more), external files (CSV) for storage and Maven dependencies for enhanced functionality.

Requirements

1. Define the following classes:
   1. **Book/Audio Book**: Represent with attributes like title, author, ISBN, availability status
   2. **Theses/Dissertation**: Represent with attributes like title, author, topic, abstract, date published, availability status
   3. **CD/DVD**: Represent with attributes like title, producer, director, playtime, availability status
   4. **Author**: Represent attributes like name, list of authored books
   5. **Library User**: Represent with attributes like name, ID, list of borrowed assets

***Note***: The above list of classes and attributes is the minimum requirement and is not definitive. You can add any others you deem necessary for your design.

1. Implement appropriate constructors and getters/setters
2. Use inheritance and interfaces to design an efficient class hierarchy, eg, you could create an abstract class to represent library item attributes and behaviours shared by both books and media or an interface that defines methods for adding, editing, deleting and retrieving content from a data file. These would then by used by classes which would implement the required methods accordingly.

***Ensure that you include several lambda functions in your desigm.***

1. Use stacks, queues and/or linked lists to store and manipulate the created objects
2. File Handling
   1. Create external CSV files, eg, books.csv, authors.csv, users.csv, etc, to store information about books, authors and users, respectively (Apache Commons CSV)
   2. Implement operations to read and write data to these files, enabling the system to load and save catalogue data
3. Error Handling
   1. Implement error handling for file operations, user input validation and any other potential exceptions, creating user defined error handling classes as required
4. Sorting & Searching
   1. Implement a sort and search facility for books, authors and users
   2. Consider the various sorting and searching algorithms available and justify the appropriateness of your selected algorithms and data structures in terms of complexity and performance profile when compared to others
5. Implement methods to perform the following library operations:
   1. Add a book to the library
   2. Add an author to the library
   3. Add a library user to the system
   4. Borrow a book (update book availability and library user's borrowed books)
   5. Return a book (update book availability and library user's borrowed books)
   6. List available books
   7. List books borrowed by a user
   8. List books authored by an author
   9. Any other options you desire
6. Implement the following Maven Dependencies:
   1. External library for reading and writing to external files
   2. JUnit to thoroughly test the program
7. User Interface
   1. Create a simple command-line user interface to interact with the library system
   2. Allow users to execute library operations through text-based menus
8. Submission
   1. Each group should submit their Maven project with a well-organised codebase
   2. The external data files should be included in the submission
   3. A report that explains the design decisions, classes, interfaces, source code, external files and Maven dependencies used in the project
9. Grading
   1. Correctness and functionality of the library system
   2. Proper use of Java interfaces, abstract classes, classes and lambdas
   3. Effective use of external file handling for data storage
   4. Successful use of Maven for dependency management/unit testing
   5. Code quality, organisation, and comments
   6. Documentation including system design and research/justification on algorithms used
   7. Appropriate testing