7.9: Assessment 2 - summative, system development using Python

Start Assignment

Assessment - *Summative, System Development using Python*

The following learning outcomes will be assessed:

1. Understanding object oriented paradigm and object oriented programming concepts.
2. Understand and implement the software development lifecycle, and development methodologies pertinent within current industry practice
3. Solve complex problems using an object oriented approach to software development.
4. Design and evaluate a complex object oriented system through appropriate software engineering and design models, using UML diagrams, notations, & techniques.

Important Information

You are required to submit your work within the bounds of the University Infringement of Assessment Regulations (see your Programme Guide). Plagiarism, paraphrasing and downloading large amounts of information from external sources, will not be tolerated and will be dealt with severely. Although you should make full use of any source material, which would normally be an occasional sentence and/or paragraph (referenced) followed by your own critical analysis/evaluation. You will receive no marks for work that is not your own. Your work may be subject to checks for originality which can include use of an electronic plagiarism detection service.

For this assessment you are asked to submit an individual piece of work, therefore the work must be entirely your own. The safety of your assessments is your responsibility. You must not permit another student access to your work.

Referencing for this assessment should be done using the Harvard referencing system. (see your Programme Guide).

Please ensure that you retain a duplicate of your assignment. We are required to send samples of student work to the external examiners for moderation purposes. It will also safeguard in the unlikely event of your work going astray.

Submission Location - Digital copy via Canvas

Document Format

1. A fully documented (i.e. containing suitable comments) Python program which should implement the system requirements. You must submit your entire system in a zip file titled "studentID_system_source_code.zip".
2. Microsoft word document titled "studentID_class_diagram.docx" This file should contain System design using UML methodology.
3. Screencast. Your screencast should be in MPEG or MOV file format titled "studentID_screencast"
4. Replace "studentID" with your university student ID.
5. Submit the file in the allocated assignment submission windows on Canvas.

Assignment weight - 65%

Other requirements

You must adhere to the above assessment requirements.

Your source code will be checked for plagiarism using Pycode open source software

System Specification

The assignment for this module is to implement a library record system in Python. You are asked to build a system by using object oriented programming concepts. It should be possible to create objects from your implemented Python classes. Each class should contain information about different parts of the system. The objects should be created from the classes and interact with each other to achieve the correct functionality of the system. There are several Python classes to be written for this assignment. The system should include the following Python classes as minimum: Books, BookList, Users, UserList and Loans. See the Tasks section below for specific details about each class.

Programming Tasks (30 points)

Design and implement a software system for a fictitious library system. Your system should follow object oriented programming approach. It should contain the following components each represented in a Python class:

Books: Define a Python class with methods to do the following:

1. Define a constructor to create new book records. Each record should have include the following attributes:
    1. Randomly generated book ID, title, author, year, publisher, number of available copies and publication date.

2. Define different methods to set each of the following book attributes, one method per attribute:
    1. title, author, year, publisher, number of available copies and publication date.

getter & setter:

3. Define different methods to return each the following book attribute, one method per attribute:
    1. title, author, year, publisher, number of copies, available number of copies and publication attribute.
4. The class should include error checking (e.g., exception handling).
5. The class should be documented by comments.

BookList: Define a Python class with methods to do the following:

1. Define a constructor to create new object from this class.
2. Define a method to store a collection (e.g., dictionary). The collection should store book instances that are created from the Book object.
3. Define a method to search through the collection and find a book by one of the following data: title, author, publisher OR publication date.
4. Define a method to remove a book from the collection. The book should be specified by its title.
5. Define a method to return the total number of books stored in the collection.
6. The class should include error checking (e.g., exception handling).
7. The class should be documented by comments.

Users: Define a Python class with functions to do the following:

1. Define a constructor to create a user with the following attributes:
    - username, firstname, surname, house number, street name, postcode, email address, and date of birth.
2. Define different method to return the following attributes: username, firstname, surname, house number, street name, postcode, email address, and date of birth. You should have one method per attribute.
3. Define different methods to edit the following attribute: firstname, surname, email address, and date of birth. You should have one method per attribute.
4. The class should include appropriate error checking.

5. The class should have be well documented by comments.

UserList: Define a Python class with functions to do the following:

1.
    1. Define a constructor to create new object from this class.
    2. A method to store a collection (e.g., dictionary) of user instances that are created with the class Users.
    3. A method to remove a user from the collection by giving the user's first name. This operation must inform program users if there are two or more users with same first name.
    4. A method to count the number of users in the system. This should be based on the number of user object in the collection.
    5. A method to return a user's detail by the username.
    6. The class should include appropriate error checking (e.g., exception handling).
    7. The class should be well documented by comments.

Loans: Define a Python class with methods to do the following:

1.
    1. Define a constructor to create new object from this class.
    2. A method for a user to borrow a book. This method should have appropriate features to assign a book to a user. The information could be stored in an appropriate data structure for further processing.
    3. A method for a user to return a book. This method should un-assign a book previously assigned to a user.
    4. A method to count and return the total number of books a user is currently borrowing.
    5. A method to print out all the overdue books along with the users' username and first name. The username and first name of the user should be retrieved through the appropriate methods in the User class.
    6. The class should include appropriate error checking (e.g., exception handling).
    7. The class should be well documented by comments.

Extras (5 points)

Extend your program to include the following features:

1. Books: Modify a book’s title, author, year, and publisher and number of copies from an easy to use command line user interface.
2. Users: Modify a user’s first name, surname, house number, street name, postcode from an easy to use command line user interface.

Class diagram (15 points)

Create a UML class diagram for the software modelling. The UML diagram should contain the full system design and should reflect your system implementation.

1. The UML diagram should include all the properties of the classes, the correct methods and the correct association between the classes. The diagram should reflect the system and class implementations.

Demonstration (15 points)

Prepare up to 10 minutes of recorded presentation of all the features and functionalities of your implemented software. Your presentation should show at least the following aspects of your system:

- Clear view of the user interface.
- Clear view of the source code generating the user interface.
- Clear view of the source code of different classes you have implemented.
- Show all the system functionalities using different types of input and the way your system handles errors and potential problems.
- Appropriately pitched and paced audio.

Guidelines

This assignment should be submitted no later than Monday of week 8 at 23.59 UK time

Grading

This activity will be graded and you will receive feedback on it. The following rubric is provided to help you understand how you will be graded.