Software Development: Object Oriented Programming

Assessment: ALL OUTCOMES

Introduction

Please read the following documentation carefully.
This assessment can be completed IN and OUT of class.
There are several separate stages of this project, please ensure that you manage your time effectively to meet the overall deadline.
Your assessor will check the authenticity of any unsupervised work.
Please read all of the evidence requirements for each stage and clarify any points with your assessor.
The assessment is divided into the following sections:

- **Section 1 – Assessment Brief:**
- This section contains the UML design documents and Use Case diagrams.
- **Section 2 – Implementation**
- Contains instructions and a checklist of assessment criteria for you to use while you implement the project.
- **Section 3 – Testing**
- Contains instructions and a checklist of criteria for you to use while you test the project.
- **Remediation and Reassessment**
    - One remediation attempt will be permitted on areas that do not meet the criteria.
    - If after remediation the work still does not meet criteria, a 2nd attempt on a new brief will be required

Section 1: Assessment Brief

Please read the following design brief carefully:
As a software developer, you have been approached to develop a computer program to manage local authority library stock. This system is developed separately from the library loans system, which is a separate system outside of the scope of this brief.

Each library offers a selection of books, journals and audio/video material on offer to the public. This system is managed by library staff, who input this information centrally.
All **borrowable items** have a unique id (integer), a type (text), title (text), a location (text) and a cost (float) and whether or not the item has been issued (boolean)
In addition:
• **Books** have an author (text), a publisher (text) , and a number of pages (integer)
• **Journals** have a publisher (text), issue number (integer), subject area (text), and a number of pages (integer)
• **Audio/Video items** have a duration (int minutes), and format (text - CD/DVD)

The program must display a list of all items and their type in the main program window. Users should have the ability to add, delete and search for items of the appropriate type and display them on screen.
The program must have an option to add up the total cost of all items in the inventory and display this on screen.

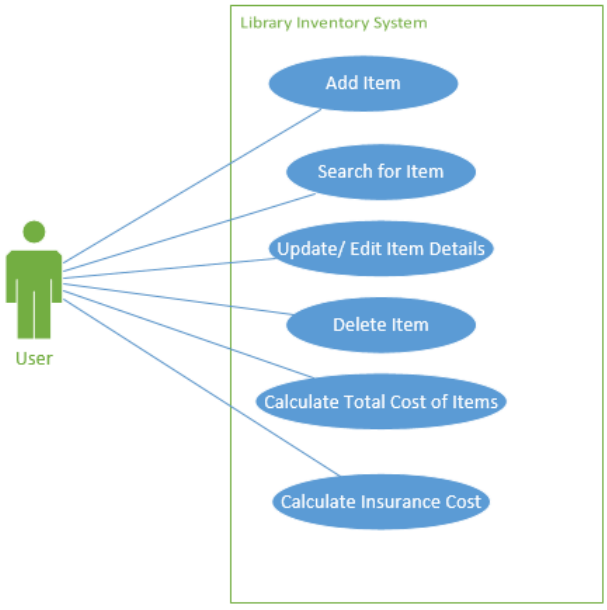
Your program should have appropriate input validation and provide error messages.

In addition it should provide an insurance cost. The insurance cost is 50% of the total cost of all of the items. The cost of insurance should be displayed on the same screen / popup as the total cost. No matter how much the cost works out to be, the final cost of insurance will be capped at £400.

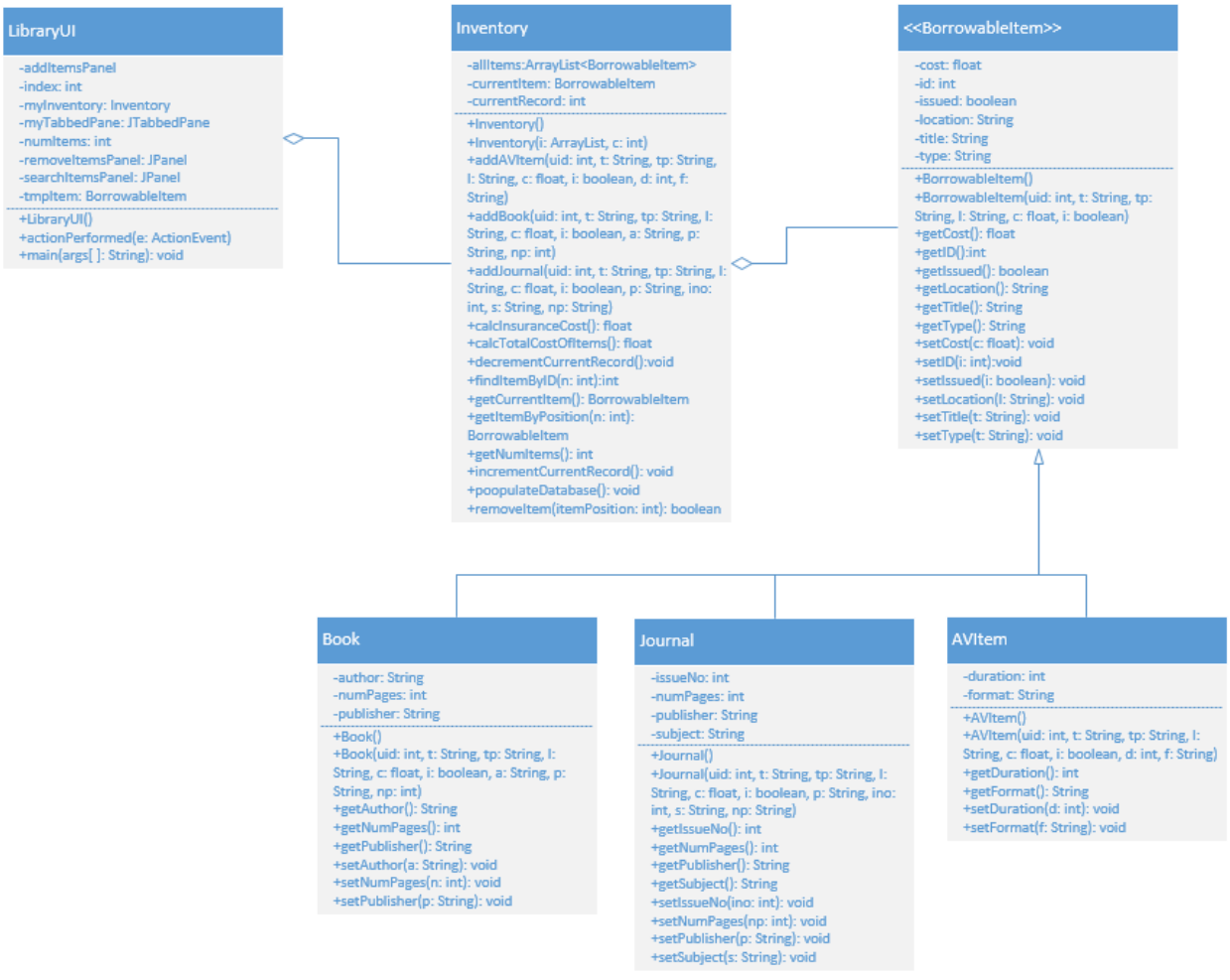
The user should be able to interact with the program using an appropriate selection of buttons, drop down menus, text boxes and so on.

Design Diagrams

Use Case Diagram



Class Diagram



Section 2: Implementation

Learning Outcomes 1 and 2

You are required to create an object oriented program for the brief provided in Section 1.

The design brief should be the **basis** of your solution. You may add additional features if you wish.

Your solution must demonstrate each of the criteria in the assessment checklist.

Section 3: Testing

Learning Outcome 3

You are now required to develop a test plan and test the completed program. You should produce appropriate test logs to identify any areas where the program fails and detail any fixes and retests required.

Your testing must demonstrate each of the criteria in the assessment checklist.

Assessment Checklist (All Outcomes)

Candidate name		**Candidate ID**	
Class		**Date**	
Assessment Requirement	**Satisfactory (S)/ Unsatisfactory (U)**	**Comments**	
Outcome 1			
Abstraction, Encapsulation and information hiding used where appropriate.			
Inheritance used if appropriate to the solution.			
Polymorphism used only if appropriate to the solution.			
Class-wide variables are private to prevent content coupling.			
Class-wide variables are kept to a minimum to ensure a minimum of common coupling.			
Data coupling is used (using parameter passing) in preference to content or common coupling.			
Program does not contain a lot of unnecessary data coupling.			
Classes are highly cohesive.			
Outcome 2			
Implement a working solution which meets the requirements of the given brief.			
Declare and initialise variables.			
Correctly use arithmetic and/ or logical operators.			
Implement a range of control structures.			
Access and manipulate a data structure.			
Create a minimum of four classes, which contain attributes, methods, and a constructor method.			
Create a minimum of three objects from the classes, with			

appropriate initial attribute values set through the Constructor methods.		
Assessment Requirement	**Satisfactory (S)/ Unsatisfactory (U)**	**Comments**
Implement at least one overloaded method (this may be the constructor).		
Link the classes appropriately through association, aggregation, or inheritance relationships.		
Pass parameters correctly both within and between objects.		
Define the access type (public, private, or protected) for methods, attributes and classes, and the access modifiers chosen are appropriate.		
Make use of pre-defined Classes and/ or Methods from the standard object library.		
Appropriately handle errors with exceptions or pre-validation.		
Implement code commented appropriately throughout.		
Outcome 3		
Implement a test plan using a defined strategy.		
Maintain test documentation.		
Evaluate the results of test runs.		
Amend code as necessary.		

The comment column can be used to highlight any re-assessment that may be needed.

Overall comments

Assessor's signature **Date**