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, magazines and newspapers and audio/video material on offer to the public. This system is managed by library staff who input this information centrally.

All items have a unique id (integer), a 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 number of pages (integer) and a publisher (text)
- Journals have a publisher (text), a subject (text) and a number of pages (integer)
- Audio/Video items have a type (text – audio or video), format (text - CD/DVD) and duration (int minutes)

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.

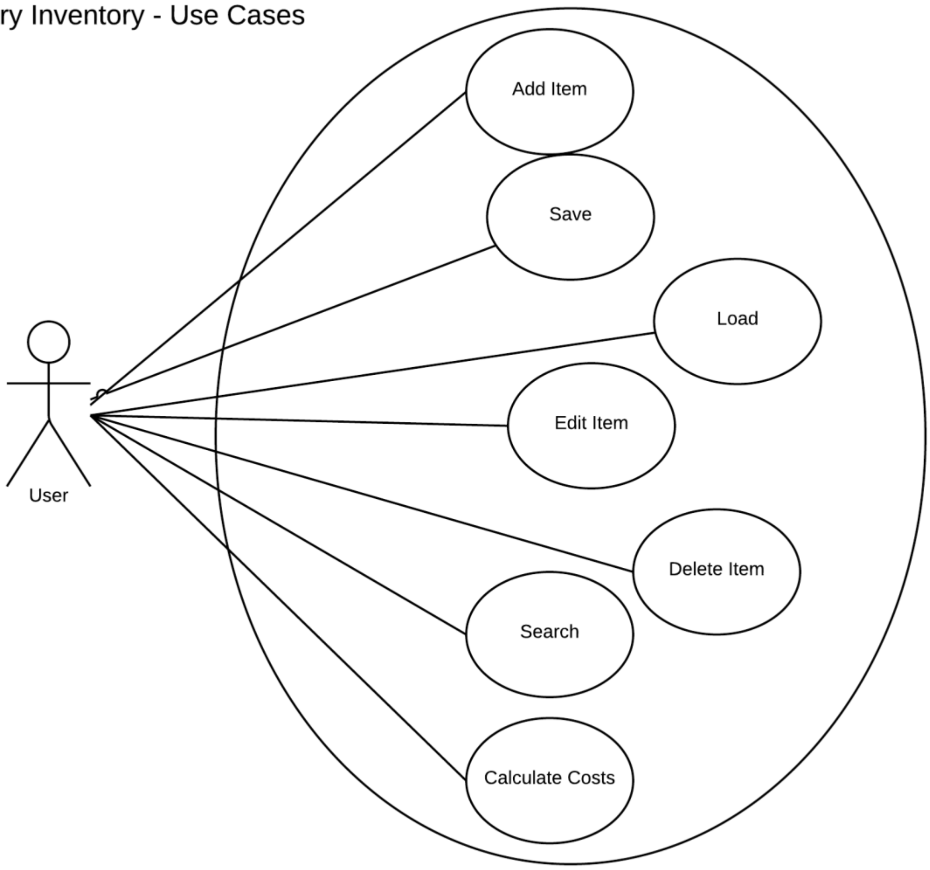
Your program should have appropriate input validation and provide error messages.

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.

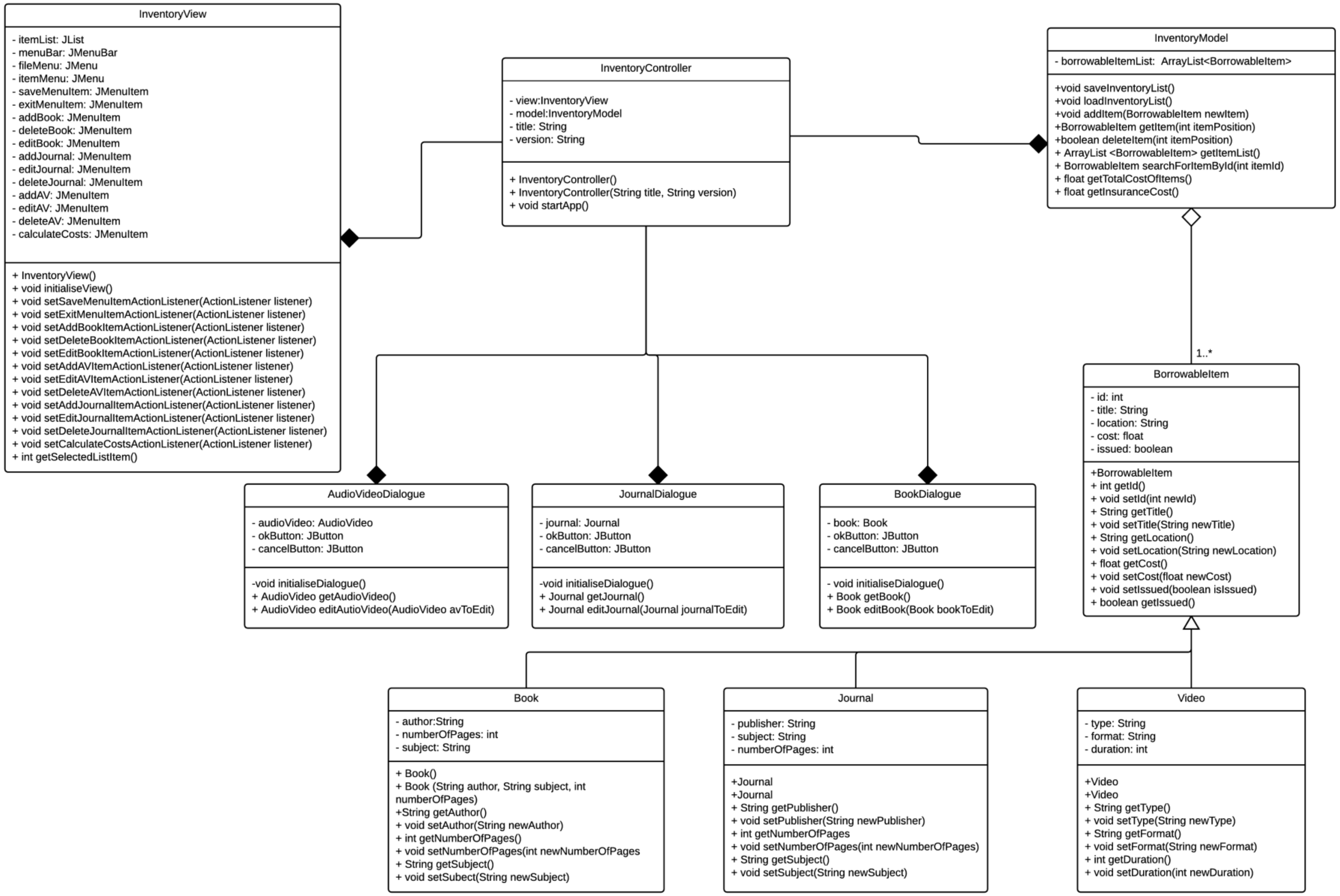
The user should be able to interact with the program using an appropriate selection of buttons, drop down menus, text boxes and so on.

The program should be able to save the list of items when requested and should always try and load the same list from disk on start up.

Library Inventory - Use Cases



Library Inventory Class Diagram


InventoryView
- itemList: JList
- menuBar: JMenuBar
- fileMenu: JMenu
- itemMenu: JMenu
- saveMenuItem: JMenuItem
- exitMenuItem: JMenuItem
- addBook: JMenuItem
- deleteBook: JMenuItem
- editBook: JMenuItem
- addJournal: JMenuItem
- editJournal: JMenuItem
- deleteJournal: JMenuItem
- addAV: JMenuItem
- editAV: JMenuItem
- deleteAV: JMenuItem
- calculateCosts: JMenuItem
+ InventoryView()
+ void initialiseView()
+ void setSaveMenuItemActionListener(ActionListener listener)
+ void setExitMenuItemActionListener(ActionListener listener)
+ void setAddBookItemActionListener(ActionListener listener)
+ void setDeleteBookItemActionListener(ActionListener listener)
+ void setEditBookItemActionListener(ActionListener listener)
+ void setAddAVItemActionListener(ActionListener listener)
+ void setEditAVItemActionListener(ActionListener listener)
+ void setDeleteAVItemActionListener(ActionListener listener)
+ void setAddJournalItemActionListener(ActionListener listener)
+ void setEditJournalItemActionListener(ActionListener listener)
+ void setDeleteJournalItemActionListener(ActionListener listener)
+ void setCalculateCostsActionListener(ActionListener listener)
+ int getSelectedListItem()
InventoryController
- view:InventoryView
- model:InventoryModel
- title: String
- version: String
+ InventoryController()
+ InventoryController(String title, String version)
+ void startApp()
InventoryModel
- borrowableItemList: ArrayList<BorrowableItem>
+void saveInventoryList()
+void loadInventoryList()
+void addItem(BorrowableItem newItem)
+BorrowableItem getItem(int itemPosition)
+boolean deleteItem(int itemPosition)
+ ArrayList <BorrowableItem> getItemList()
+ BorrowableItem searchForItemById(int itemId)
+ float getTotalCostOfItems()
+ float getInsuranceCost()
1..*
BorrowableItem
- id: int
- title: String
- location: String
- cost: float
- issued: boolean
+BorrowableItem
+ int getId()
+ void setId(int newId)
+ String getTitle()
+ void setTitle(String newTitle)
+ String getLocation()
+ void setLocation(String newLocation)
+ float getCost()
+ void setCost(float newCost)
+ void setIssued(boolean isIssued)
+ boolean getIssued()
AudioVideoDialogue
- audioVideo: AudioVideo
- okButton: JButton
- cancelButton: JButton
-void initialiseDialogue()
+ AudioVideo getAudioVideo()
+ AudioVideo editAutioVideo(AudioVideo avToEdit)
JournalDialogue
- journal: Journal
- okButton: JButton
- cancelButton: JButton
-void initialiseDialogue()
+ Journal getJournal()
+ Journal editJournal(Journal journalToEdit)
BookDialogue
- book: Book
- okButton: JButton
- cancelButton: JButton
- void initialiseDialogue()
+ Book getBook()
+ Book editBook(Book bookToEdit)
Book
- author:String
- numberOfPages: int
- subject: String
+ Book()
+ Book (String author, String subject, int numberOfPages)
+String getAuthor()
+ void setAuthor(String newAuthor)
+ int getNumberOfPages()
+ void setNumberOfPages(int newNumberOfPages
+ String getSubject()
+ void setSubect(String newSubject)
Journal
- publisher: String
- subject: String
- numberOfPages: int
+Journal
+Journal
+ String getPublisher()
+ void setPublisher(String newPublisher)
+ int getNumberOfPages
+ void setNumberOfPages(int newNumberOfPages)
+ String getSubject()
+ void setSubject(String newSubject)
Video
- type: String
- format: String
- duration: int
+Video
+Video
+ String getType()
+ void setType(String newType)
+ String getFormat()
+ void setFormat(String newFormat)
+ int getDuration()
+ void setDuration(int newDuration)

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 following criteria:

Design Requirements Checklist (Learning Outcome 1):

Criteria	Completed (Student)	Completed (Lecturer)	Evidence
Abstraction, encapsulation and information hiding used where appropriate			
Inheritance used			
Polymorphism used			
All class wide variables private to prevent content coupling			
Class wide variables kept to a minimum to reduce common coupling			
Parameter passing is used			
Program does not contain a lot of unnecessary data coupling			
Classes are highly cohesive			

Implementation Requirements (Learning Outcome 2):

Criteria	Completed (Student)	Completed (Lecturer)	Evidence
A working solution that meets the requirements of the brief			

Variables declared and initialised			
Arithmetic and logical operators used correctly			
Range of control structures implemented correctly			
At least two data structures are implemented correctly			
The program contains a minimum of four classes			
Classes contain attributes (member variables)			
Classes contain methods			
Classes contain constructors			
At least 3 objects are created with values set through the constructor.			
Program contains at least ONE overloaded method (this may be a constructor)			
Classes are linked appropriately through association, aggregation or inheritance.			
Parameters are passed correctly both within and between objects.			
Appropriate access types are defined for methods, attributes and classes			
Use of pre-defined classes and/or methods from the standard object library (API)			

Program handles errors through exceptions or pre-validation			
The program code is commented appropriately throughout			

Section 3 – Testing

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.

Testing Requirements (Learning Outcome 3)

Criteria	Completed (Student)	Completed (Lecturer)	Evidence
Test Plan			
Testing Carried Out			
Evaluation of Testing			
Amending code as necessary			

Final Check List

Learning Outcome	Complete	Signed
1		
2		
3		

1st Attempt

2nd Attempt