Faculty of Science, Engineering and Technology

Object Oriented Programming

Pass Task 8.1: Semester Test

Overview

> **Note**: This task is a time-bound test, replacing the usual Semester Test. You have 5 days to complete it (within Week 8 of semester), and only **one** opportunity to resubmit the task after receiving feedback. The test is not a hurdle requirement, but counts as a pass task to be included in your final portfolio.

In this unit, you have been using object-oriented programming to implement all of your programs. For this task, you will need to show your understanding of the principles associated with this programming paradigm.

Purpose:	Demonstrate your understanding of object-oriented programming and the core concepts of object-oriented design.
Task:	You must complete two questions. The first is a coding question, to be submitted as C# source code files, a UML diagram, and a screenshot showing your program's output. The second question asks for a written response, to be submitted as a PDF.
Time:	This task should be completed during week 8.

Submission Details

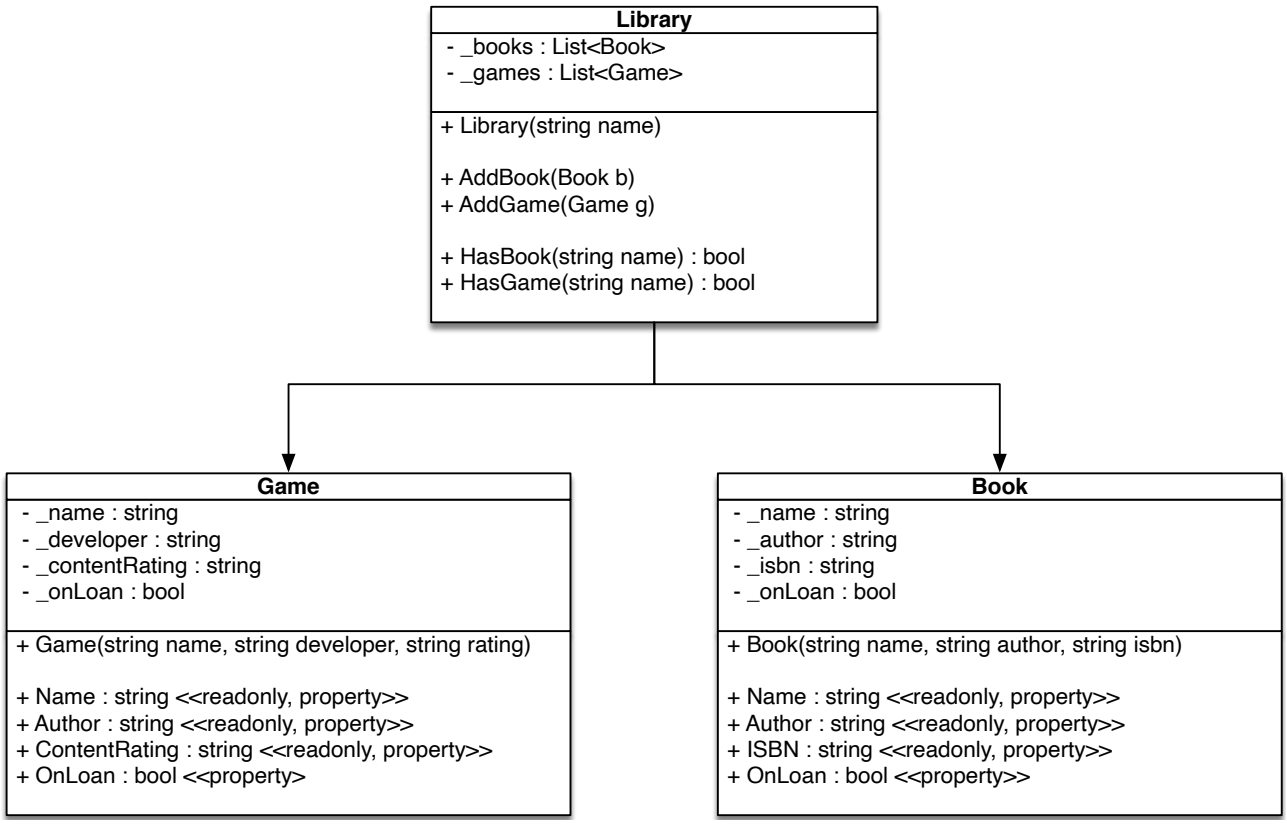
You must submit the following files to Doubtfire:

- For Question 1:
  - C# code files of the classes created
  - An image showing your modified design as a UML class diagram
  - A screenshot of the program output
- For Question 2:
  - A PDF document with your answer

Make sure that you submit code that is readable and is appropriately documented.

Question 1

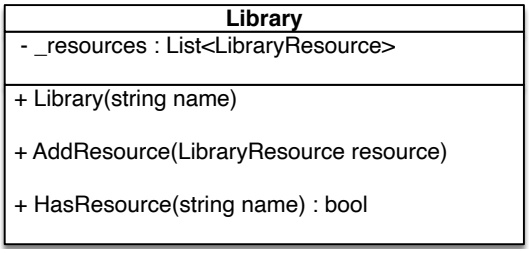
Consider the following program design:



When a Game or Book object is created, it is **not on loan**. This can be changed later through the **OnLoan** property. When the **HasBook** or **HasGame** methods are used, they return **true** if the Library has a Book or Game object with the **name** specified in the parameter, **and** that Book or Game object is **not on loan**. Otherwise, these methods should return **false**.

Tasks

Your task is to redesign this program to better follow the principles of object-oriented programming, focusing on the concept of **polymorphism**. To do this, we want to restructure the program to use an **abstract LibraryResource class**. To support this restructuring, the Library class should be modified to match the following UML:



Using this design, the **AddResource** method can be used to add Book and Game objects to the Library. The **HasResource** method should work the same way as **HasBook** and **HasGame** in the original design, but be able to search for any Book or Game based on the value passed to the **name** parameter.

Your task is to design the LibraryResource abstract class, and redesign the Game and Book classes to inherit from it.

> **Tip**: It is bad object-oriented design to duplicate information in subclasses. Look for common fields between **Game** and **Book** and move them to the abstract parent class.

You are required to:

a) Provide a new UML class diagram for your updated design (hand drawn is fine).

b) Write the code for all classes, including the new abstract class, and all methods/fields/constructors required.

c) Write a simple **Main** method to demonstrate how your new design works:

1. Create a Library object.
2. Create at least two Book and two Game objects, and add them to the Library using **AddResource**. Set the **OnLoan** value of one Book and one Game object to **true**.
3. Call **HasResource**, passing in the name of one of the Book or Game objects that are **not on loan**, and print the result (should print **true**).
4. Repeat step 3, but pass in the name of one of the Book or Game objects that **are on loan**, and print the result (should print **false**).

Submit a photo or image of your UML, all source code files, and a screenshot showing your program working to Doubtfire.

Question 2

In your own words, explain the object-oriented principles of **polymorphism** and **abstraction**. In your answer, reference the solution you have created for Question 1, or another piece of work you have completed for this unit. Your answer should be roughly half a page long, plus any diagrams you want to include.

> **Tip**: In object-oriented programming we have talked about the concepts of **abstraction** and **abstract classes**. Remember that they are different, and we are asking you to explain the first one!

> **Note**: Write your answer **in your own words**. You can use as many reference materials as you like, but you must not directly copy text from anywhere.

Submit your answer as a PDF document to Doubtfire.