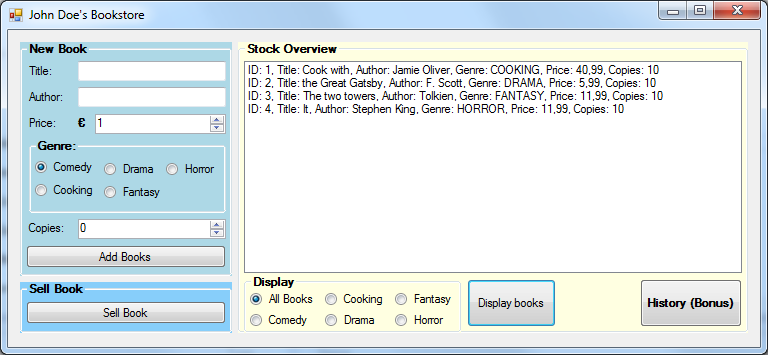
# Exam PCS2 – Book Store January 2016

**Start time: 8:45**

**End time: 11:15**

You are asked to develop an application for managing a book store (See image below).



With this application, the user should be able to add new books by giving the correct information and pressing the “Add Books” button (left side). When the user selects a book in the listbox (right side), he/she can sell one copy by clicking on the “Sell Book” button (left lower corner). If the user wants to display books with a specific genre, he/she has to perform two steps. First select which books to display from the radiobuttons (at the bottom) below “Display”. Second, press the “Display books” button to show the list of books with the same genre (or all books).

In the constructor of Form1 you can find Dummy data commented out, which places the books in the storage as seen above in the image. This way you can start your application with some books. Follow the instructions in the comments to make this work.

Now open the startup-project either in Visual Studio 2013 or Visual Studio 2015 (it is mandatory to use Visual Studio 2013 or Visual Studio 2015 in this exam). You can see that there is one form with a lot of controls on it (see the picture above). During this exam, you will implement two classes: class Book and class Store. Of course, you will also add some code to the Form as well.

Exam points:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Assignment | 1 | 2 | 3 | 4 | Bonus | Total |
| Points | 3 | 16 | 38 | 33 | 10 | 100 |

**Important note about naming -** Whenever you see a name in **bold**, you have to use this word as the name of the class, method, field or property.

**Assignment 1: Genres (3 pts)**

In this assignment you will be asked to implement the enumerator for the **Genres.**

Since there are only five genres we can use an enumerator. Make an enumerator called **Genres** in the file “Genre.cs”. It should contain the following fivevalues: COMEDY, COOKING, DRAMA, FANTASY, HORROR.

**Assignment 2: Class Book (6 + 5 + 5 pts)**

In this assignment you will be asked to implement the class “Book”.

Go to the file “Book.cs” and implement the class “Book”.

1. Add the following properties (and if required, fields) inside the class:   
   (When possible, use auto-implemented properties)
   * Start with the **Id** of the book. This should be an integer. Make sure that this is read-only.
   * We also want to store the **Title** of the book.
   * The **Author** who has written the book is also stored.
   * Next the **Genre** of the book. The type of this property is the enumerator from assignment 1.
   * The **Price** is also needed**.** it should be a number and cannot be less than € 0,01. A possible value could be 6,99.
   * Also make a property which stores the amount of **Copies** this book has in storage. The number of copies cannot be negative and at most 50.
2. Now a constructor should be made to initialize all the values. Use parameters to assign a value to the following properties: “Title”, “Author”, “Genre”, “Price” and “Copies”. The “Id” should be assigned a value inside the constructor without the use of a parameter. Make a field and use this to assign a unique value to “Id”.
3. To complete the “Book” class, make a method named **AsString** without any parameters. This method should return all information about the book as shown in the listbox (right side) in the image above.

**Assignment 3: Class Store (3 + 4 + 6 + 7 + 8 + 4 + 6 pts)**

In this assignment you will be asked to implement the class “Store”.

Go to the file “Store.cs” and implement the class “Store”.

1. Add the following field and property to this class:
   * The store should have a list of “Book” objects named **storage** it should not be accessible outside this class.
   * We also want a property for the **Name** of the Store.
2. Make a constructor for the **Store** class. It should have one parameter to assign a value to the “Name”. Also make sure that the “storage” is initialized.
3. Now make a method named **GetBook**. This method should search and return a book that matches the given “title” and “author” from “storage”. If the book cannot be found, return null.
4. Make a method that adds books to “storage” named **AddBook**. This method should have the following parameters: “title”, “author”, “genre”, “price” and “copies”. Be aware of the rule that the combination of “title” and “author” acts as a primary key in the store. In other words, the method works as follows: if the “storage” does not contain a book with the same “title” and “author”, a new book for those parameter-values is created and added to the storage and the method returns the value true. If the storage already contains a book with the given “title” and “author”, then nothing is added to the storage and the returned value is false.
5. Make a method called **SellBook**.It should use the “title” and “author” and sell one copy of this book. You may assume that this book exists. One of the following messages can be returned:  
   - If there is at least one copy in “storage”: **Successfully sold a book!**  
   - If there are no copies in “storage”: **This book is sold out!**
6. Now make a method called **GetListOfBooks.** This method has no parameters and should return all books in “storage”.
7. Last, make another method called **GetListOfBooks** with a parameter for the “genre”**.** This method should return a list of books of that specified genre, e.g.:
   * If that genre is about COMEDY, the method returns a list of all COMEDY-Books.
   * If that genre is about HORROR, the method returns a list of all HORROR-Books.

**Assignment 4: Form (3 + 6 + 7 + 8 + 9 pts)**

Consider the file “Form1.cs” and implement the code required by this assignment.

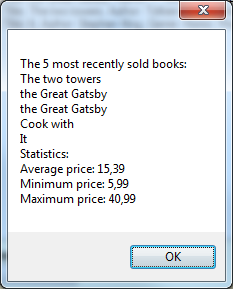
1. Start off with creating a field for the ”Store”, named **myStore.**
2. Give code to make “myStore” at startup:
   * Initialize “myStore” with your own name as the owner.
   * Make sure the text at the top of the form is equal to the name of myStore (like in the above picture it is "John Doe's bookstore").
3. Implement the button-click of “buttonAddBook”. When the user clicks, the application checks if “myStore” already has a book with the “Title” and “Author” as specified in the textboxes. If the book already exists (same “Title” and “Author”), a message is displayed saying: “This book already exists!”. If the book does not exist, the book gets added to “myStore”.
4. Now implement the button-click of “buttonDisplayBooks”. Make sure that the “listBoxBooks” displays all books with the selected genre from the radioButtons at the bottom below the text “Display”.
5. Last implement the button-click of “buttonSellBook”. By clicking, the user should be able to sell the book which is selected in the “listBoxBooks”. Use a messagebox to tell if the sale was successful or not. (see assignment 3e for the possible messages).

**THE FOLLOWING ASSIGNMENT IS ONLY FOR THOSE WHO DID NOT RECEIVE THE BONUSPOINTS FROM THE PRACTICAL, IF YOU DID RECEIVE THEM, YOU WILL NOT GET POINTS FOR THIS ASSIGNMENT.**

**Bonus Assignment: (4 + 6)pts**

In this assignment you will be asked to add and implement a button with name **buttonHistory**.

It will display (up to) 10 most recent purchases and some additional statistics seen in the image below:



Example of output.

1. Start off by going to the “Store” class and add the following fields:
   1. Make a field which can store up to ten “Book”s and initialize it in the constructor.
   2. To make sure you know which book you added last, you should store the current index.
2. Make sure when you sell a book, the purchase gets added (at the correct index) to the history. The order of when the books were sold does **not** matter.
   1. If you are at the last index, the next book can be added at the beginning.
   * Make a method named **GetHistory** in the “Store” class,which returns a text. This method should give the following information,
     1. The amount of books most recently sold (never more then 10).
     2. The titles of the books most recently sold.
     3. Some statistics: the average price, the minimum price and the maximum price of these books.

If the store didn’t sell books yet, a message should tell this to the user.

* + Let a messagebox show the history when the “buttonHistory” is clicked in the form.