# Lab: Iterators and Comparators

Problems for exercises and homework for the [["C# OOP Advanced" course @ Software University](https://softuni.bg/trainings/1637/c-sharp-oop-advanced-july-2017)](https://softuni.bg/courses/java-oop-advanced). You can check your solutions here: <https://judge.softuni.bg/Contests/707/Iterators-and-Comparators-Lab>

## Library

Create a class **Book** which should have three public properties:

* **string Title**
* **int Year**
* **List<string> Authors**

Authors can be **anonymous, one or many**. A Book should have only **one** **constructor**.

Create a class **Library** which should store a collection of books and implement the **IEnumerable<Book>** interface.

* **List<Book> books**

A Library should be intilized without books or with any number of books and should have only **one** **constructor**.

### Examples

|  |
| --- |
| Startup.cs |
| public static void Main()  {  Book bookOne = new Book("Animal Farm", 2003, "George Orwell");  Book bookTwo = new Book("The Documents in the Case", 2002, "Dorothy Sayers", "Robert Eustace");  Book bookThree = new Book("The Documents in the Case", 1930);  Library libraryOne = new Library();  Library libraryTwo = new Library(bookOne, bookTwo, bookThree);  } |

### Solution





## Library Iterator

Extend your solution from the prevoius task. Inside the Library class create a **nested class** **LibraryIterator** which should implement the **IEnumerator<Book>** interface. Try to implement the bodies of the inherited methods by yourself. You will need two more members:

* **List<Book> books**
* **int currentIndex**

Now you should be able to iterate through a Library in the Main method.

### Examples

|  |
| --- |
| Startup.cs |
| public static void Main()  {  Book bookOne = new Book("Animal Farm", 2003, "George Orwell");  Book bookTwo = new Book("The Documents in the Case", 2002, "Dorothy Sayers", "Robert Eustace");  Book bookThree = new Book("The Documents in the Case", 1930);  Library libraryOne = new Library();  Library libraryTwo = new Library(bookOne, bookTwo, bookThree);  foreach (var book in libraryFull)  {  Console.WriteLine(book.Title);  }  } |

|  |
| --- |
| **Output** |
| Animal Farm  The Documents in the Case  The Documents in the Case |

### Solution



## Comparable Book

Extend your solution from the prevoius task. Implement the **IComparable<Book>** interface in the existing class **Book**. The comparison between two books should happen in the following order:

* First sort them in **ascending chronological** order (by year)
* If two books are published in the **same year**, sort them **alphabetically**

Override the **ToString()** method in your Book class so it returns a string in the format:

* {**title**} - {**year**}

Change your Library class so that it stores the books in the correct order.

You don’t need to change anything in your **Main** method from the previous task except for the way to print a book on the Console.

### Examples

|  |
| --- |
| **Output** |
| The Documents in the Case - 1930  The Documents in the Case - 2002  Animal Farm - 2003 |

### Solution



## Book Comparator

Extend your solution from the prevoius task. Create a class **BookComparator** which should implement the **IComparer<Book>** interface and thus include the following method:

* **int Compare(Book, Book)**

**BookComparator** must **compare** two books by:

1. Book title - **alphabetical order**
2. Year of publishing a book - **from the newest to the oldest**

Modify your Library class once again to implement the **new sorting**.

### Examples

|  |
| --- |
| Startup.cs |
| public static void Main()  {  Book bookOne = new Book("Animal Farm", 2003, "George Orwell");  Book bookTwo = new Book("The Documents in the Case", 2002, "Dorothy Sayers", "Robert Eustace");  Book bookThree = new Book("The Documents in the Case", 1930);    } |

|  |
| --- |
| **Output** |
| Animal Farm - 2003  The Documents in the Case - 2002  The Documents in the Case - 1930 |

### Solution

