Lab: Iterators and Comparators

You can check your solutions here: https://judge.softuni.bg/Contests/3183/Additional-Exercises.

1. Library

NOTE: You need the namespace **`IteratorsAndComparators`**.

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.

- **`List<Book> books`**

A Library could 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

```
public class Book
{
    public Book(string title, int year, params string[] authors)
    {
        this.Title = title;
        this.Year = year;
        this.Authors = new List<string>(authors);
    }

    public string Title { get; set; }

    public int Year { get; set; }

    public List<string> Authors { get; set; }
}
```

```
public class Library
{
    public Library(params Book[] books)
    {
        this.Books = new List<Book>(books);
    }

    public List<Book> Books { get; }
}
```

2. Library Iterator

NOTE: You need the namespace **`IteratorsAndComparators`**.

Extend your solution from the prevoius task. The **`Library`** class should implement the **IEnumerable<Book>** interface. Use a **yield return** statement to return each element one at a time. You will need one more member: **`List<Book> books`**.

Now you should be able to iterate through a **`Library`** in the **`Main()`** method.

Examples

Startup.cs	Output

`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 libraryTwo)`
`  {`
`  Console.WriteLine(book.Title);`
`  }`
`}`
	`Animal Farm`
	`The Documents in the Case`
	`The Documents in the Case`

Solution

```
namespace IteratorsAndComparators
{
    public class Library : IEnumerable<Book>
    {
        private readonly List<Book> books;

        public Library(params Book[] books)
        {
            this.Books = new List<Book>(books);
        }

        public List<Book> Books { get; }

        public IEnumerator<Book> GetEnumerator()
        {
            for (int i = 0; i < this.Books.Count; i++)
            {
                yield return this.Books[i];
            }
        }

        IEnumerator IEnumerable.GetEnumerator()
        {
            return GetEnumerator();
        }
    }
}
```

3. Comparable Book

NOTE: You need the namespace **`IteratorsAndComparators`**.

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}"`**

Modify your **`Library`** class, so that it stores the books in the correct order (**sorted**).

- You may use **`SortedSet<Book>`** to hold the books.
- Or you may explicitly sor the array of books: **`this.books.Sort()`**.

Examples

Startup.cs	Output

`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(bookTwo, bookOne, bookThree);`
	`The Documents in the Case - 1930`
	`The Documents in the Case - 2002`
	`Animal Farm - 2003`

```
  foreach (var book in libraryTwo)
  {
    Console.WriteLine(book);
  }
}
```

Solution

```
public class Book : IComparable<Book>
{
    3 references
    public Book(string title, int year, params string[] authors)...
    6 references
    public string Title { get; set; }
    6 references
    public int Year { get; set; }
    1 reference
    public List<string> Authors { get; set; }
    0 references
    public int CompareTo(Book other)
    {
        var result = this.Year.CompareTo(other.Year);
        if (result == 0)
        {
            result = this.Title.CompareTo(other.Title);
        }
        return result;
    }
    0 references
    public override string ToString()
    {
        return $"{this.Title} - {this.Year}";
    }
}
```

4. Book Comparator

NOTE: You need the namespace **`IteratorsAndComparators`**.

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**.

- You may sort the books, e. g. like this: **`this.books.Sort(new BookComparator())`**.

Examples

Startup.cs	Output

`public static void Main()`
`{`
`  Book bookOne = new Book("Animal Farm", 2003, "George Orwell");`
	`Animal Farm - 2003`
	`The Documents in the Case - 2002`

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 library = new Library(bookTwo, bookOne, bookThree); }	The Documents in the Case - 1930

Solution

```
public class BookComparator : IComparer<Book>
{

    public int Compare(Book x, Book y)
    {
        var result = x.Title.CompareTo(y.Title);

        if (result == 0)
        {
            result = y.Year.CompareTo(x.Year);
        }

        return result;
    }
}
```