**BookLibrary**

**Avaliable from**: Sunday, 3 July 2022, 10:00  
**Due date**: Sunday, 3 July 2022, 16:00  
**Requested files**: Program.cs, Book.cs, BookLibrary.cs ([Download](https://it-kariera.mon.bg/e-learning/mod/vpl/views/downloadrequiredfiles.php?id=7623))  
**Type of work**: Individual work

**Библиотека**

**Общ преглед**

Във вашата фирма постъпва проект за създаване на приложение, обслужващо библиотека.

Вашият софтуер трябва да описва **библиотека** **(BookLibrary)** и **книга (Book).**

Tрябва да реализирате функционалност, която да позволява някои основни команди с книгите в библиотеката. Поредицата от команди приключва с "**END**". За ваше удобство ще получите готов **Program.cs** файл, и ще трябва да реализирате само необходимите класове **BookLibrary.cs** и **Book.cs**

**Подзадача 1: 30 точки**

**Book**

Всяка книга има заглавие и рейтинг:

* **title** – текст
* **rating** – рейтинг (дробно число)

Реализирайте **конструктор**, който да създава обект от класа Book при зададени **заглавие и рейтинг**.

Методът **ToString()** трябва да връща текст в следния формат:

* **"Book {title} is with {rating} rating."**, където рейтингът трябва да бъде форматиран до **първия знак** след десетичната запетая.

|  |
| --- |
| **Book.cs** |
| private string title;  private double rating;  public Book(string title, double rating)  {  //TODO: Добавете вашия код тук …  }  public string Title  {  //TODO: Добавете вашия код тук …  }  public double Rating  {  //TODO: Добавете вашия код тук …  }  public override string ToString()  {  //TODO: Добавете вашия код тук …  } |

**BookLibrary**

Всякa библиотека има име и списък с книги:

**name** – текст

**books** – списък от книги

|  |
| --- |
| **BookLibrary.cs** |
| private string name;  private List<Book> books;  public BookLibrary (string name)  {  //TODO: Добавете вашия код тук …  }  public string Name  {  //TODO: Добавете вашия код тук …  }  public List<Book> Books  {  //TODO: Добавете вашия код тук …  }  public void AddBook(string title, double rating)  {  //TODO: Добавете вашия код тук …  }  public double AverageRating()  {  //TODO: Добавете вашия код тук …  }  public List<string> GetBooksByRating(double rating)  {  //TODO: Добавете вашия код тук …  }  public List<Book> SortByTitle()  {  //TODO: Добавете вашия код тук …  }  public List<Book> SortByRating()  {  //TODO: Добавете вашия код тук …  }  public string[] ProvideInformationAboutAllBooks()  {  //TODO: Добавете вашия код тук …  }  public bool CheckBookIsInBookLibrary (string title)  {  //TODO: Добавете вашия код тук …  } |

**Командa за добавяне на книга**

Вашето приложение трябва да обслужва следната команда за добавяне на книга:

* **AddBook** <**заглавие рейтинг**> - тази команда има за цел да добави книга с нейното заглавие и рейтинг.

**Команда за извеждане на информация**

Вашето приложение във всеки един момент може да получи заявка да отпечата информация за всички книги. Командата за това е следната:

* **PrintBookLibraryInfo** - отпечатва информация за всички книги в структурата във формат:  
  **Book {title} is with {rating} rating.**
* За успешна реализация трябва да реализирате **ToString()** метода за класа **Book.**

**Подзадача 2: 30 точки**

* **SortByTitle** - Трябва да сортира в библиотеката книгите от списъка по име във възходящ (нарастващ). След което трябва да бъде върнат сортираният списък.

*Забележка: След изпълнение на тази операция, списъкът с книгите трябва да остане в състоянието си на сортиран вид, според условието на командата.*

* **SortByRating** - Трябва да сортира в библиотеката книгите в списъка по рейтинг в низходящ (намаляващ) ред. След което трябва да бъде върнат сортираният списък.

*Забележка: След изпълнение на тази операция, списъкът с книгите трябва да остане в състоянието си на сортиран вид, според условието на командата.*

* **CheckBook <title**> - Трябва да бъде намерена книга, на която заглавието да съвпада с подаденото. При успешно намиране трябва да върнете булева стойност **true,** а в обратен случай **false.**

**Подзадача 3: 20 точки**

* **GetBestBooksByRating <рейтинг**> - Трябва да бъдат намерени всички книги от списъка, на които рейтингът им е по-висок от дадения в командата. Да се върне списък със загалвията на намерените книги.

**Подзадача 4: 20 точки**

* **AverageRating** - Трябва да бъде намерен средноаритметичният рейтинг на книгите в списъка.

**Вход**

* Програмата ще получава множество редове с информация. Всеки ред представлява команда. Самият вход се обработва изцяло от примерния Program.cs.
* **Всички команди приключват с въвеждането на END**

**Изход**

За някои от командите не е нужно да извеждате нищо. За други е необходимо форматиране на изход – напр. Book.ToString(), Book.Title()

**Ограничения**

* Всички реални числа ще бъдат в диапазона **–10000** до **+10000**
* Имената няма да съдържат интервал

**Примери**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| AddBook, TheMatrix, 8.6  AddBook, Parasite, 8.5  AddBook, GreenBook, 8.2  PrintBookLibraryInfo  END | Added book: TheMatrix -> 8.6.  Added book: Parasite -> 8.5.  Added book: GreenBook -> 8.2.  Book TheMatrix is with 8.6 rating.  Book Parasite is with 8.5 rating.  Book GreenBook is with 8.2 rating. |
| AddBook, Passengers, 7.0  AddBook, Star, 7.7  AddBook, Troy, 7.2  PrintBookLibraryInfo  SortByTitle  PrintBookLibraryInfo  SortByRating  PrintBookLibraryInfo  CheckBook, Troy  CheckBook, IT  AverageRating  GetBestBooksByRating, 7.1  END | Added book: Passengers -> 7.  Added book: Star -> 7.7.  Added book: Troy -> 7.2.  Book Passengers is with 7.0 rating.  Book Star is with 7.7 rating.  Book Troy is with 7.2 rating.  First book is: Passengers  Book Passengers is with 7.0 rating.  Book Star is with 7.7 rating.  Book Troy is with 7.2 rating.  The worst book is with rating: 7  Book Star is with 7.7 rating.  Book Troy is with 7.2 rating.  Book Passengers is with 7.0 rating.  Book Troy is available.  Book IT is not available.  Average rating of books: 7.3  Most rated books are: Star, Troy |

**Точки**

Разбивката по подзадачи е следната:

1. **30** точки
2. **30** точки
3. **20** точки
4. **20** точки

Общ брой точки: **100**

**Requested files**

**Program.cs**

using System;

using System.Collections.Generic;

namespace Program

{

public class Program

{

static BookLibrary bookLibrary = new BookLibrary("BookLibrary");

static void Main(string[] args)

{

string line;

while ("END" != (line = Console.ReadLine()))

{

string[] cmdArgs = line.Split(", ");

switch (cmdArgs[0])

{

case "AddBook":

AddBook(cmdArgs[1], double.Parse(cmdArgs[2]));

break;

case "AverageRating":

AverageRating();

break;

case "GetBestBooksByRating":

GetBooksByRating(double.Parse(cmdArgs[1]));

break;

case "SortByTitle":

SortByTitle();

break;

case "SortByRating":

SortByRating();

break;

case "CheckBook":

CheckBookIsInBookLibrary(cmdArgs[1]);

break;

case "PrintBookLibraryInfo":

ProvideInformationAboutAllBooks();

break;

}

}

}

private static void ProvideInformationAboutAllBooks()

{

string[] info = bookLibrary.ProvideInformationAboutAllBooks();

foreach (string item in info)

{

Console.WriteLine(item);

}

}

private static void CheckBookIsInBookLibrary(string title)

{

if (bookLibrary.CheckBookIsInBookLibrary(title))

{

Console.WriteLine($"Book {title} is available.");

}

else

{

Console.WriteLine($"Book {title} is not available.");

}

}

private static void SortByRating()

{

bookLibrary.SortByRating();

Console.WriteLine("The worst book is with rating: " + bookLibrary.Books[bookLibrary.Books.Count - 1].Rating);

}

private static void SortByTitle()

{

bookLibrary.SortByTitle();

Console.WriteLine("First book is: " + bookLibrary.Books[0].Title);

}

private static void GetBooksByRating(double rating)

{

List<string> bestBooks = bookLibrary.GetBooksByRating(rating);

Console.WriteLine("Most rated books are: " + string.Join(", ", bestBooks));

}

private static void AverageRating()

{

double averageRating = bookLibrary.AverageRating();

Console.WriteLine($"Average rating of books: {averageRating:f1}");

}

private static void AddBook(string title, double rating)

{

bookLibrary.AddBook(title, rating);

Console.WriteLine($"Added book: {title} -> {rating}.");

}

}

}

#### Book.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Program

{

public class Book

{

private string title;

private double rating;

public Book(string title, double rating)

{

Title = title;

Rating = rating;

}

public string Title

{

get

{

return title;

}

set

{

title = value;

}

}

public double Rating

{

get

{

return rating;

}

set

{

rating = value;

}

}

public override string ToString()

{

return $"Book {title} is with {rating} rating.";

}

}

}

#### BookLibrary.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Program

{

public class BookLibrary

{

private string name;

private List<Book> books;

public BookLibrary(string name)

{

Name = name;

Books = new List<Book>();

}

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public List<Book> Books

{

get

{

return books;

}

set

{

books = value;

}

}

public void AddBook(string title, double rating)

{

Book book = new Book(title, rating);

books.Add(book);

}

public double AverageRating()

{

return books.Average(x => x.Rating);

}

public List<string> GetBooksByRating(double rating)

{

return books.Where(x => x.Rating > rating).Select(x => x.Title).ToList();

}

public List<Book> SortByTitle()

{

books = books.OrderBy(x => x.Title).ToList();

return books;

}

public List<Book> SortByRating()

{

books = books.OrderByDescending(x => x.Rating).ToList();

return books;

}

public string[] ProvideInformationAboutAllBooks()

{

return books.Select(x => x.ToString()).ToArray();

}

public bool CheckBookIsInBookLibrary(string title)

{

foreach (Book book in books)

if (book.Title == title)

return true;

return false;

}

}

}