Day 02.2. Время отдыхать

## Общие требования

* Убедитесь, что на вашем компьютере установлен [SDK для разработки на .NET 5](https://dotnet.microsoft.com/download) и вы используете именно его.
* Помните, ваш код будут читать! Обратите особое внимание на оформление вашего кода и именование переменных. Обязательно придерживайтесь общепринятых стандартов [C# Coding Conventions](https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/inside-a-program/coding-conventions).
* Самостоятельно выберите удобную для себя IDE.
* Программа должна иметь возможность запуска через командную строку dotnet.
* В каждом из заданий указаны примеры ввода и вывода. Решение должно использовать их как верный формат.
* В начале каждого задания приведен список рекомендованных языковых конструкций.
* Если затрудняетесь в решении задачи, обратитесь с вопросами к другим участникам бассейна, интернету, Google, посмотрите на StackOverflow.
* С основными возможностями языка C# можно ознакомиться в [официальной спецификации](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/language-specification/introduction).
* Избегайте **хардкода** и “**магических чисел**”.
* Вы демонстрируете все решение, верный результат работы программы – лишь один из способов проверки ее корректной работы. Поэтому когда необходимо получить определенный вывод в результате работы ваших программ, запрещено показывать пред рассчитанный результат.
* Обращайте особое внимание на термины, выделенные **bold** шрифтом: их изучение пригодится вам как в выполнении текущего задания, так и в вашей дальнейшей карьере .NET разработчика.
* Have fun :)

## Требования к заданиям дня

* Все сборки должны быть в одном решении.
* Каждому из заданий должно соответствовать отдельное консольное приложение, созданное на основе стандартного шаблона .NET SDK.
* Используйте **top-level-statements** и **var**.
* Название проекта (и его отдельного каталога) должно выглядеть как d{*xx*}\_ex{*yy*}, где *xx* - цифры текущего дня, *yy* - цифры текущего задания.
* Название решения (и его отдельного каталога) - d{*xx*}, где *xx* - цифры текущего дня.
* Для форматирования выходных данных используйте **культуру** en-GB: N2 для вывода денежных сумм, d для дат.
* Файлы с данными в заданиях считать верными и не нуждающимися в валидации. Файлы можно найти в соответствующих номеру задания папках example files.

## Что нового

* Принцип подстановки Лисков
* Обобщенные типы, ограничения типов
* LINQ, методы-расширения
* Сложность алгоритмов
* Конфигурация приложений
* Деревья выражений

## Структура проекта

d04/

Program.cs

appsettings.json

Model/

ISearchable.cs

Book.cs

Movie.cs

## Интро

Жизнь на работе и учебе начинают налаживаться, но отдыхать тоже важно. И как было бы здорово, думаете вы, если бы определиться с этим было проще. Открыл готовый список и выбрал.

Здесь вам на помощь приходит архив New York Times - издание постоянно публикует списки бестселлеров и обзоры на лучшие фильмы, почему бы не воспользоваться? Остается всего ничего - скомпилировать это в единый каталог увеселений.

Итак, вам необходимо разработать консольное приложение, которое при запуске будет собирать информацию из двух файлов **JSON** в две разных коллекции с элементами.

# Задание 00. Книги

Итак, у вас есть выгрузка книг-бестселлеров этой недели, взятых из разных категорий. Считаем, что файл периодически обновляется и содержит актуальную информацию. Из приведенной в списке информации нас интересуют: название книги, ее автор, описание, место в рейтинге и его (рейтинга) название, а также ссылка на страницу в магазине.

Список книг должен подгружаться из файла book\_reviews.json, их нужно **десериализовать** из **JSON** в [соответствующие сущности](https://docs.microsoft.com/en-us/dotnet/standard/serialization/system-text-json-customize-properties).

Реализуйте для получившегося класса *BookReview* отдельный формат вывода с помощью **перегрузки метода** ToString(): пусть он выводит:

{book.Title} by {book.Author} [{book.Rank} on NYT’s {book.ListName}]

{book.SummaryShort}

{book.Url}

Выведите список книг в консоль.

# Задание 01. Фильмы

Для кино-настроения рядом есть выгрузка свежих ревью на кино. Считаем, что и этот файл обновляется и содержит актуальную информацию. Из приведенных в этом списке данных нам нужны: название фильма, его рейтинг, является ли фильм выбором критиков, саммари его ревью и ссылка на страницу публикации.

Список фильмов должен подгружаться из файла movies.json, их нужно **десериализовать** из **JSON** в [соответствующие сущности](https://docs.microsoft.com/en-us/dotnet/standard/serialization/system-text-json-customize-properties).

Реализуйте для получившегося класса *MovieReview* отдельный формат вывода с помощью **перегрузки метода** ToString(): пусть он выводит:

{movie.Title} {movie.IsCriticsPick ? “[NYT critic’s pick]” : “”}

{movie.SummaryShort}

{movie.Url}

Выведите список фильмов в консоль.

# Задание 02. Поиск

Было бы здорово, думаете вы, реализовать поиск в одном методе - искать-то придется по заголовку и там, и там, а дублировать код - [дурной тон](https://x-team.com/blog/principles-clean-code/). И это значит, что здесь пригодится **принцип подстановки Лисков**.

Реализуем поиск по заголовку в статическом **методе-расширении** *Search.* Чтобы методу *Search* было все равно, по книгам идет поиск или по фильмам, сделайте тип T **обобщенным**. Однако важно, чтобы этот тип имел заголовок, по которому идет поиск: давайте добавим [ограничение](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/where-generic-type-constraint):

static T[] Search<T>(this IEnumerable<T> list, string search)

where T : ISearchable

Реализуйте в теле метода поиск по введенной строке запроса, метод должен возвращать список всех найденных элементов: книг и отзывов на фильмы.

Поиск не должен зависеть от регистра. Результаты в каждой группе должны быть отсортированы по заголовку.

Элементы *BookReview* и *MovieReview* должны реализовать один интерфейс *ISearchable*: и у книг, и у фильмов есть заголовки, по которым будет вестись поиск.

Результат поиска должен быть сгруппирован по типу медиа (кино/книга). Не используйте для поиска циклы, посмотрите в сторону **LINQ**! Обратите внимание на **методы расширения LINQ - Any, Count, GroupBy, OrderBy, Where**, они делают код куда более читаемым и простым для восприятия. Используйте [лямбда-выражения](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-expressions).

В случае отсутствия соответствующих поиску элементов приложение должно выдавать сообщение “There are no “{search}” in media today.”

В случае пустой строки поиска приложение должно выдавать все элементы.

## Входные параметры

> Input search text:

| search | string | Строка поиска |
| --- | --- | --- |

## Пример запуска приложения из папки проекта и вывода

$ dotnet run

> Input search text:

ar

Items found: 6

Book search result [3]:

- THE MIDNIGHT LIBRARY by Matt Haig [6 on NYT's Hardcover Fiction]

Nora Seed finds a library beyond the edge of the universe that contains books with multiple possibilities of the lives one could have lived.

https://www.amazon.com/dp/0525559477?tag=NYTBSREV-20

- THE INVISIBLE LIFE OF ADDIE LARUE by V.E. Schwab [9 on NYT's Hardcover Fiction]

A Faustian bargain comes with a curse that affects the adventure Addie LaRue has across centuries.

https://www.amazon.com/dp/0765387565?tag=NYTBSREV-20

- KLARA AND THE SUN by Kazuo Ishiguro [13 on NYT's Hardcover Fiction]

An "Artificial Friend" named Klara is purchased to serve as a companion to an ailing 14-year-old girl.

https://www.amazon.com/dp/059331817X?tag=NYTBSREV-20

Movie search result [3]:

- IN OUR MOTHERS' GARDENS

The Netflix documentary sets out to show how maternal lineages have shaped generations of Black women.

https://www.nytimes.com/2021/05/06/movies/in-our-mothers-gardens-review.html

- MARIGHELLA [NYT critic's pick]

Wagner Moura's provocative feature debut chronicles the armed struggle led by Carlos Marighella against Brazil's military dictatorship in the 1960s.

https://www.nytimes.com/2021/04/29/movies/marighella-review.html

- THINGS HEARD & SEEN

Amanda Seyfried and James Norton move into a haunted house in this busy, creaky Netflix thriller.

https://www.nytimes.com/2021/04/29/movies/things-heard-and-seen-review.html

# Задание 03. Лучший вариант

Если при запуске приложения указан аргумент best реализуйте вместо поиска вывод лучшего варианта книги и фильма.

Лучшими здесь считаем первую книгу с максимальным местом в рейтинге и первый обзор на фильм, который является выбором критика.

Не используйте циклы, посмотрите в сторону **LINQ**! Обратите внимание на **методы расширения LINQ**, они делают код куда более читаемым и простым для восприятия. Используйте [лямбда-выражения](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-expressions).

Обратите внимание, что методы выбора одного элемента - например, First(), - имеют пары First()/FirstOrDefault(). Задумайтесь, в чем разница и применимость каждого из них.

## Пример запуска приложения из папки проекта и вывода

$ dotnet run best

Best in books:

- SOOLEY by John Grisham [1 on NYT's Hardcover Fiction]

Samuel Sooleymon receives a basketball scholarship to North Carolina Central and determines to bring his family over from a civil war-ravaged South Sudan.

https://www.amazon.com/dp/0385547684?tag=NYTBSREV-20

Best in movie reviews:

- MARIGHELLA [NYT critic's pick]

Wagner Moura's provocative feature debut chronicles the armed struggle led by Carlos Marighella against Brazil's military dictatorship in the 1960s.

https://www.nytimes.com/2021/04/29/movies/marighella-review.html

## Примечание. Эффективность на примере метода Last()

У каждой коллекции в C# доступен метод расширения Last(). Однако, работает он за **O(1)** только для коллекций, реализующих интерфейс IList (**список с доступом к элементам по индексу**). Для остальных коллекций он работает за **O(N)**, перебирая её элементы до конца. Будьте осторожны.

Задумывайтесь о том, как работают переборы. Особенно если используете их внутри циклов.

# Задание 04. Конфигурация

Днем ранее мы уже писали собственную конфигурацию, но на деле в .NET Core для этого существуют уже готовые средства. Давайте использовать их: файл конфигурации ***appsettings.json*** и **nuget-пакеты** для управления конфигурацией из JSON файла.

Установите [Microsoft.Extensions.Configuration.Json](https://www.nuget.org/packages/Microsoft.Extensions.Configuration.Json/6.0.0-preview.4.21253.7) и обратите внимание на пакеты-зависимости, которые nuget-менеджер установит для него. Пример настройки и использования конфигураций приложения можно найти в [статье](https://docs.microsoft.com/en-us/archive/msdn-magazine/2016/february/essential-net-configuration-in-net-core). Туда же можно будет вынести дополнительные настройки.

Настройте конфигурацию приложения так, чтобы путь к файлам брался из appsettings.json.

# Дополнительно

Для большей глубины понимания того, как устроен язык запросов LINQ, будет очень полезно разобраться в том, что такое [деревья выражений](https://docs.microsoft.com/en-us/dotnet/csharp/expression-trees) и с чем их едят.