# Упражнения: Комбиниране на структури от данни

Тествайте решението в Judge: <https://judge.softuni.org/Contests/4174/13-Combining-Data-Structures-More-Exercises>.

## Re-Play

**Re-Play** e интерактивна музикална платформа, която осигурява функция за регистриране, добавяне в опашка за слушане и слушане на песни.

Даден ви е скелет с клас **RePlayer**, който трябва да имплементира интерфейса **IRePlayer**.

Класът **RePlayer** има **песни**. Всички **песни** има уникално id.

Класът **Track** (песен) има следните свойства:

* **Id** – низ
* **Title** – низ
* **Artist** – низ (име на артист)
* **Plays** – число (броя на пусканията на песента)
* **DurationInSeconds** - число

Имплементирайте следните функции:

* **void AddTrack(Track track, string albumName)** – **добавя** **песен** в **RePlayer**. Ако няма албум с такова име, трябва да го създадете. Албумите имат уникални имена. Имената на **песните** в албума също са уникални.
* **bool Contains(Track track)** –връща дали **съществува** дадена **песен** в **RePlayer**.
* **int Count –** връща броя на **песните**.
* **Track GetTrack(string title, string albumName) –** връща **песен** с дадено име от албум. Ако няма такъв албум или песен, хвърлете **ArgumentException()**
* **IEnumerable<Track> GetAlbum(string albumName)** –връща всички **песни** от албум с дадено име. Подредете песните по брой пускания в низходящ ред. Ако няма такъв албум, хвърлете **ArgumentException()**
* **void AddToQueue(string title, string albumName) –** добавя **песен** с дадено име от албум в **опашка за слушане**. Ако няма такъв албум или песен, хвърлете **ArgumentException()**
* **Track** **Play()** – връща **първата песен**, която е влязла в опашката за слушане. След това **увеличете** броя на пусканията с **1** като премахнете песента от опашката за слушане. Ако няма песен в опашката, хвърлете **ArgumentException()**
* **void RemoveTrack(string title, string albumName)** – премахва **песен** с дадено име от албум. Ако я има в опашката за слушане, трябва да я премахните от там**.** Ако няма такава песен или албум, хвърлете **ArgumentException()**
* **IEnumerable<Track> GetTracksInDurationRangeOrderedByDurationThenByPlaysDescending(int lowerBound, int upperBound)** – връща всички песни с определена продължителност, която е по-голяма или равна на **lowerBound** и по-малка или равна на **upperBound**. Подредете резултата по продължителност във възходящ ред. След това по брой пускания в низходящ ред. Ако няма песни, върнете празна колекция
* **IEnumerable<Track> GetTracksOrderedByAlbumNameThenByPlaysDescendingThenByDurationDescending()** –

връща всички **песни** подредени по името на албума във възходящ ред. След това ги подредете по брой пускания и по продължителност в низходящ ред. Ако няма такива **песни** върнете, празна колкеция.

* **Dictionary<string, List<Track>> GetDiscography(string artistName)** – връща всички **песни**, които са **ГРУПИРАНИ** по име на албум, на даден артист. Ако няма такъв **артист** или няма такива **песни**, хвълете **ArgumentException()**.

## Фактури

Вашата задача е да имплементирате система за обработка на фактури.

Имате клас **Invoice (фактура)**, който има свойствата:

* String SerialNumber – уникално число за всяка
* String CompanyName – името на фирмата за фактурата
* Double Subtotal – обща сума на фактурата
* Enum Department – отделение към което принадлежи фактурата
* DateTime IssueDate – дата на създаване на фактурата
* DateTime DueDate – краен срок на фактурата

Трябва да направите следните операции:

* Create(Invoice)– Добавя фактура в агенцията за архивиране. Ако вече съществува има фактора с даден SerialNumber, хвърлете ArgumentException.
* Contains(number) – Проверете дали има фактура с даден SerialNumber в архива
* Count – Връща броя на фактурите в архива
* PayInvoice(dueDate) – Намира всички фактури, чийто краен срок е дадена дата. След това задайте техните общи суми на 0. Ако няма такива, хвърлете ArgumentException.
* ThrowInvoice(SerialNumber) – Премахнете фактура с даден SerialNumber. Ако няма, такава хвърлете ArgumentException
* ThrowPayed() – Премахва всички фактури, които са платени
* GetAllInvoiceInPeriod(start, end) – Намерете всички фактури, които са създадени в даден от start до включително end. Подредете ги по **дата на създаване**, след това по **дата на краен срок**. Ако няма фактури, върнете празна колекция.
* SearchBySerialNumber (String SerialNumber) – връща всички фактури, които имат даден SerialNumber. Сортирайте ги по техния номер. Ако няма такива, ArgumentException
* ThrowInvoiceInPeriod(start, end)– премахва всички фактури, чийто краен срок е в даден период. Ако няма такива, хвърлете ArgumentException.
* GetAllFromDepartment(department) – намира всички фактури, които са в даден отдел. Подредете ги по обща сума в низходящ ред като първи параметър, след това по дата на създаване във възходящ ред. Ако няма такива фактури, върнете празна колекция.
* GetAllByCompany(company) – намира всички фактури, които са създадени от дадена компания. Подредете ги по тяхното число в низходящ ред. Ако няма такива, върнете празна колекция.
* ExtendDeadline(dueDate, days) – намира всички фактури, чийто краен срок е дадена дата. Удължете ги с даденото време. Ако няма такива, хвърлете ArgumentException