# Упражнения: Комбиниране на структури от данни

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

## Discord

Дискорд е просто хранилище за съобщения, което позволява съобщенията в чат каналите да се съхраняват.

Даден ви е скелет с клас **Discord**, който трябва да имплементира **интерфейса** **IDiscord**.

Дискорд работи с обекта **Message**. Всички обекти от **Message** са идентифицирани с **уникално id**.

**Message** съдържа следните свойства:

* **Id** – низ
* **Content** – низ
* **Timestamp** – число
* **Channel** – низ
* **Reactions** – списък от низове

Имплементирайте следните операции, за да направите дискорд пълно функционален:

* **void SendMessage(Message message, string channel)** – **добавя** **съобщения** в даден канал.
* **bool Contains(Message message)** –връща дали съобщението се **съдържа** в **дискорд**.
* **int Count –** връща **броя** на **съобщенията**.
* **Message GetMessage(string messageId)** – **връща** **съобщение** с дадено **id.**   
  Ако няма такова **съобщение**, **хвърлете ArgumentException()**
* **void DeleteMessage(string messageId)** – **премахва** **съобщение** с дадено **id.**   
  Ако няма такова **съобщение**, **хвърлете ArgumentException()**
* **void ReactToMessage(string messageId, string reaction)** – **добавя** **реакция** на **съобщение** с дадено **id**  
  Ако няма такова **съобщение**, **хвърлете ArgumentException()**
* **IEnumerable<Message> GetChannelMessages(string channel)** – **връща всички** съобщеня, които са в **даден чат-канал**.

Ако няма **съобщения** в даденият **канал**, **хвърлете** **IllegalArgumentException()**

* **IEnumerable<Message> GetMessagesByReactions(List<string> reactions)** –връща всички **съобщения**, които имат дадени реакции. Подредете съобщенията по брой реакции в **низходящ ред** и по **timestamp** във **възходящ** **ред**.
* **IEnumerable<Message> GetMessagesInTimeRange(int lowerBound, int upperBound)** –

Връща всички съобщения с **timestamp**, които са по–големи или равни на **lowerBound** и са по-малки или равни на **upperBound**. Подредете резултата по брой съобщения на канала, където се намира съобщението. Ако няма съобщения в обхвата, върнете **празна колекция**.

* **IEnumerable<Message> GetTop3MostReactedMessages()** – връща топ 3 съобщения с най-много реакции в **низходящ ред**. Ако няма съобщения, върнете **празна колекция**.
* **IEnumerable<Message> GetAllMessagesOrderedByCountOfReactionsThenByTimestampThenByLengthOfContent()** – връща всички **съобщения** подредени по **брой реакции** в **низходящ ред**, след това по **timestamp** във **възходящ ред** и по дължина на съобщението във **възходящ ред**. Ако няма съобщения, върнете **празна колекция**.

## Фактури

Вашата задача е да имплементирате система за обработка на фактури.

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

* String SerialNumber – уникален низ за всяка фактура
* 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