# Data Structures Exam: VANI Planning

This document defines the exam preparation for ["Data Structures – Advanced (C#)" course @ Software University](https://softuni.bg/trainings/3113/data-structures-advanced-with-csharp-october-2020).

Please submit your solutions (**source code**) of all below described problems in [Judge](https://judge.softuni.bg/Contests/2608/08-Exam-Preparation).

## Description

Your task is to implement a simple system for invoice processing.

You have a class Invoice which has the following properties:

* String SerialNumber – unique number for each invoice
* String CompanyName – the invoice company name
* Double Subtotal – the subtotal of the invoice
* Enum Department – the department to which the invoice belongs
* DateTime IssueDate – the date when the invoice was created
* DateTime DueDate – the deadline for the invoice

You need to support the following operations (and they should be **fast**):

* Create(Invoice)– Add the invoice to the agency archive. You will need to implement the Contains() method as well. If the invoice number already exists throw ArgumentException
* Contains(number) – checks if an invoice with the given number is present in the archive
* Count – returns the number of invoices in the archive
* PayInvoice(dueDate) – find all invoices whose due date is exactly the given one and set their subtotal to 0. If there aren’t any throw ArgumentException.
* ThrowInvoice(SerialNumber) – remove the invoice with the given number. If the invoice doesn’t exist throw ArgumentException
* ThrowPayed() – remove all invoices which were payed
* GetAllInvoiceInPeriod(start, end) – find all invoices which were created in the given period inclusive. Order them by date of creating ascending, then by due date as second parameter ascending. If there aren’t any return empty collection.
* SearchBySerialNumber (String SerialNumber) – return all invoices whose number contains the given one. Order them by their number descending. If there aren’t any throw ArgumentException
* ThrowInvoiceInPeriod(start, end)– remove all invoices which have due date between the given range exclusive. If there aren’t any throw ArgumentException.
* GetAllFromDepartment(department) – finds all invoices in the provided department. Order them by subtotal descending as first parameter, then by creating date ascending. If there aren’t any return empty collection.
* GetAllByCompany(company) – finds all invoices created by the given company. Order them by their number descending. If there aren’t any return empty collection.
* ExtendDeadline(dueDate, days) – find all invoices whose due date is equal to the given one and extend their due date by the days given. If there aren’t any throw ArgumentException.

*Feel free to override Equals() and GetHashCode() if necessary.*

## Input / Output

You are given a **Visual Studio C# project skeleton** (unfinished project) holding the interface IAgency, the classes Agency and Invoice. **Tests** covering the Agency **functionality** and **performance**.

Your task is to **finish this class** to make the tests run correctly.

* You are **not allowed to change the tests**.
* You are **not allowed to change the interface**.
* You can add to the Invoice class, but don't remove anything.
* You can edit the Agency class if it implements the IAgency interface.

## Interface

The interface IAgency in C# looks like the code below:

|  |
| --- |
| public interface IAgency  {  void Create(Invoice invoice);  void ThrowInvoice(string number);  void ThrowPayed();  int Count();  bool Contains(string number);  void PayInvoice(DateTime due);  IEnumerable<Invoice> GetAllInvoiceInPeriod(DateTime start, DateTime end);  IEnumerable<Invoice> SearchBySerialNumber(string serialNumber);  IEnumerable<Invoice> ThrowInvoiceInPeriod(DateTime start, DateTime end);  IEnumerable<Invoice> GetAllFromDepartment(Department department);  IEnumerable<Invoice> GetAllByCompany(string company);  void ExtendDeadline(DateTime dueDate, int days);  } |

## Scoring

Each implemented method brings you a specific amount of points, some of the points are awarded for correct behavior, others for performance. **The performance tests are tested only in the Judge system**. You need to cover all tests in each group to receive points. Bellow is a breakdown of all points by methods:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Correct Behavior | Performance | Total |
| Overall | 50 | 100 | 150 |