# Homework: Functional Programming

This document defines the homework assignments from the ["OOP" Course @ Software University](https://softuni.bg/trainings/coursesinstances/details/8). Please submit as homework a single zip / rar / 7z archive holding the solutions (source code) of all below described problems. The solutions should be written in C#.

## StringBuilder Extensions

Implement the following **extension methods** for the class **StringBuilder**:

* **Substring(int startIndex, int length)** – returns a new **String** object, containing the elements in the given range. Throw an exception when the range is invalid.
* **RemoveText(string text)** – removes all occurrences of the specified text (case-insensitive) from the **StringBuilder**. The method should not create a new **StringBuilder**, but should modify the existing one and return it as a result.
* **AppendAll<T>(IEnumerable<T> items)** – appends the string representations of all items from the specified collection. Use **ToString()** to convert from **T** to **string**.

Write a program to demonstrate that your new extension methods work correctly.

## Custom LINQ Extension Methods

Create your own LINQ extension methods:

* **public static IEnumerable<T> WhereNot<T>(this IEnumerable<T> collection, Func<T, bool> predicate) { … }** – works just like **Where(predicate)** but filters the non-matching items from the collection.
* **public static IEnumerable<T> Repeat<T>(this IEnumerable<T> collection, int count) { … }** – repeats the collection count **times**.
* **public static IEnumerable<string> WhereEndsWith(this IEnumerable<string> collection, IEnumerable<string> suffixes) { … }** – filters all items from the collection that ends with some of the specified **suffixes**.

## Class Student

Create a class **Student** with properties **FirstName**, **LastName**, **Age**, **FacultyNumber**, **Phone**, **Email**, **Marks** (**IList<int>**), **GroupNumber**. Create a **List<Student>** with sample students. These students will be used for the next few tasks.

## Students by Group

Print all students from group number 2. Use a LINQ query. Order the students by **FirstName**.

## Students by First and Last Name

Print all students whose first name is before their last name alphabetically. Use a LINQ query.

## Students by Age

Write a LINQ query that finds the first name and last name of all students with age between 18 and 24. The query should return only the **first name**, **last name** and **age**.

## Sort Students

Using the extension methods **OrderBy()** and **ThenBy()** with lambda expressions sort the students by first name and last name in descending order. Rewrite the same with LINQ query syntax.

## Filter Students by Email Domain

Print all students that have email **@abv.bg**. Use LINQ.

## Filter Students by Phone

Print all students with phones in Sofia (starting with **02** / **+3592** / **+359 2**). Use LINQ.

## Excellent Students

Print all students that have **at least one mark Excellent (6)**. Using LINQ first select them into a new anonymous class that holds **{ FullName + Marks}**.

## Weak Students

Write a similar program to the previous one to extract the **students with exactly two marks "2"**. Use extension methods.

## Students Enrolled in 2014

Extract and print the **Marks** of the students that **enrolled in 2014** (the students from 2014 have 14 as their 5-th and 6-th digit in the **FacultyNumber**).

## \* Students by Groups

Add a **GroupName** property to Student. Write a program that extracts all students **grouped by** **GroupName** and then prints them on the console. Print all group names along with the students in each group. Use the "**group by into**" LINQ operator.

## \* Students Joined To Specialties

Create a class **StudentSpecialty** that holds **specialty name** and **faculty number**. Create a list of **student specialties** that specifies for eachs student his specialty. Print all student names alphabetically along with their faculty number and specialty name. Use the "**join**" LINQ operator. Example:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Student Specialties** | | **join** | **Students** | | **🡪** | **Result (Joined Students with Specialties)** | | |
| **SpecialtyName** | **FacNum** | **FacNum** | **Name** | **Name** | **FacNum** | **Specialty** |
| Web Developer | 203314 | 203814 | Milena Kirova | Asya Manova | 203314 | Web Developer |
| Web Developer | 203114 | 203114 | Stefan Popov | Asya Manova | 203314 | QA Engineer |
| PHP Developer | 207713 | 203314 | Asya Manova | Diana Petrova | 203914 | PHP Developer |
| PHP Developer | 203914 | 203914 | Diana Petrova | Diana Petrova | 203914 | Web Developer |
| QA Engineer | 203314 | 203814 | Ivan Ivanov | Ivan Ivanov | 203814 | PHP Developer |
| Web Developer | 203914 |  |  | Stefan Popov | 203114 | Web Developer |

## \*\* LINQ to Excel

Write a C# program to create an Excel file like the one below using an external library such as [LinqToExcel](https://github.com/paulyoder/LinqToExcel).

You are given as **input** course data about **1000 students** in a **.txt** file (tab-separated values). Each line in the input holds **ID**, **first name**, **last name**, **email**, **gender**, **student type**, **exam result**, **homework sent**, **homework** **evaluated**, **teamwork score**, **attendances count**, **bonus**.

* Create a class **Student** that holds all aforementioned data fields from the file. Add a field **Result** and a method **CalculateResult()** that calculates the total course **result** of a student using the formula ***(exam result + homework sent + homework evaluated + teamwork + attendances + bonus) / 5****.*
* Create a **Student** object for each student from the **.txt** file and store it in some collection. **Filter** only the **online students** and sort them by their **course result**. Print the resulting student collection in an Excel table. Styling the table is not required.

