Distributed Systems

v2021

Lab 2:
Essential Programming with C# 2

Contents

Exercise 1: Create application to convert .txt file data to XML

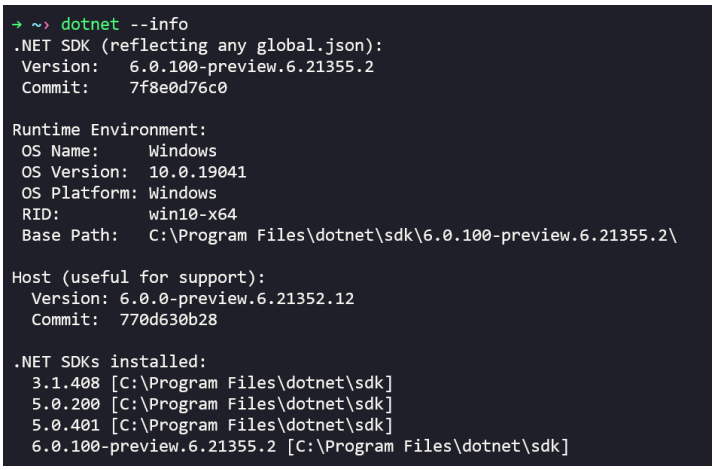
In this exercise you will create C# .NET 5 console application which will convert .txt file data into C# object and then these objects should be saved to XML file.

Creating Solution and Parser Project

Creating .NET C# **solution** and **project** can be done using .NET SDK.

1. Open Powershell and verify you have installed correct version of SDK

```
dotnet --info
```



2. Create new folder and move to its location in console

```
mkdir TxtToXmlParser
cd TxtToXmlParser
```

3. Create solution file

```
dotnet new sln --name TxtToXmlParser
```

4. Create new C# .NET console project

```
dotnet new console --name TxtToXmlParser.Parser
```

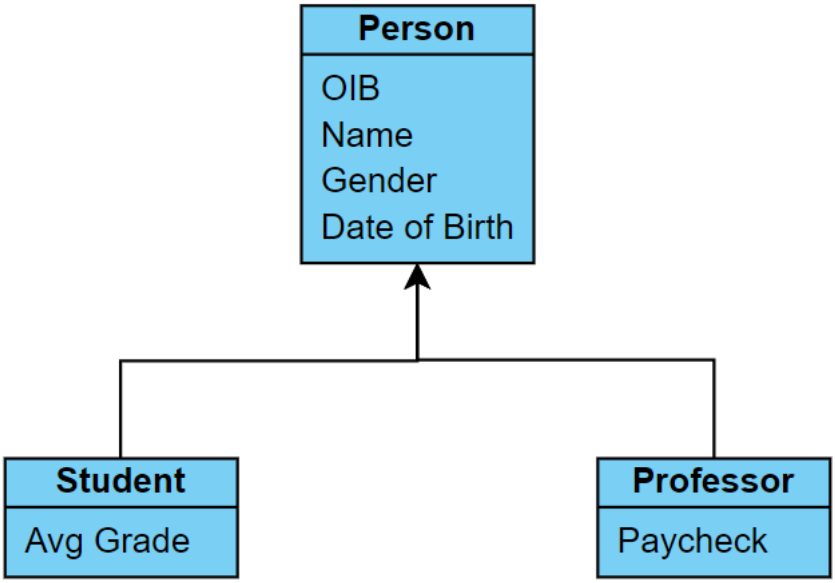
5. Add newly created project to solution

```
dotnet sln add
.\TxtToXmlParser.Parser\TxtToXmlParser.Parser.csproj
```

Exercise 2: Creating Model Project and Classes

In this exercise you will create a C# classes and define its *properties*. From that class you will instantiate *objects* and test their functionality.

The classes you will create is shown in UML below:



Creating Solution and Parser Project

Creating .NET C# **solution** and **project** can be done using .NET SDK.

1. Create new C# .NET class library project

   ```
   dotnet new classlib --name TxtToXmlParser.Model
   ```

2. Add newly created project to solution

Add a Classes

1. From UML diagrams add classes in separate files to *TxtToXmlParser.Model* project

Exercise 3: Create .txt Parser Service

In this exercise you will create service to parse .txt file into C# objects.

TXT Parser Service

1. Add new class named *TxtParserService.cs*
2. *TxtParserService.cs* should have public method:

```
public IEnumerable<Person> ParseTxtFile(string fileName){…
```

3. Implement method to:
    a. Load all text from file
    b. Rows are separated by new line character
    c. Data is separated by ; character
    d. First row represents column names
    e. Validate data
    f. Returns list of valid objects as output

Exercise 4: Create Application Entry point

In this exercise you will create application entry point in *Program.cs*

Parse command line arguments

First argument is full path to txt file location
Second argument is full path of file to save.

1. Parse arguments

2. Validate arguments:

    a. Both arguments should be valid file paths
    b. .txt file should exist
    c. Folder of file to save should exist

Parse data and save data

1. Invoke parser service and receive parsed objects
2. Serialize parsed objects to .xml file on given file location

Exercise 5: Run and test

Verify and test application functionality by parsing few .txt files

Example .txt file contents

```
Type;OIB;Name;Gender;DateOfBirth;AvgGrade;Paycheck
Student;001212;Iva Ivić;Female;1998/02/02;4.3;
Professor;001213;Ivan Zoraja;Male;1961/01/01;;100.00
```

Run application

```
dotnet run --project
.\TxtToXmlParser.Parser\TxtToXmlParser.Parser.csproj
C:\path\to\file.txt C:\path\to\new\file.xml
```

Exercise 6 (Homework): Add JSON serializer

Add additional argument in which user will be able to choose serializer (XML or JSON)

To reduce usage of command line arguments usage of library https://github.com/commandlineparser/commandline is recommended.

Source control

1. Create github account if you don't have one.
2. Create new repository named dis-02-txt-parser
3. Push your code (preferably with meaningful commits) to branch development
4. Create pull request (to main or master branch) in github and add user ipazanin as reviewer to your pull request
5. After ipazanin has approved your pull request merge it