Lehrstuhl für Computergestützte Modellierung und Simulation
TUM School of Engineering and Design
Technical University of Munich

Professional Software Engineering

Lecture 8: Exercises

Exercise 1 LINQ

In this exercise, you are given an xml file named "*pokedex.xml*", which contains a list of Pokémons based on the Pokédex entry[1].

Your task is to create a C# program that imports the data from *pokedex.xml*, selects pokémons according to their type (e.g. electric, grass etc.), and sorts them by species name. Use LINQ to query the data, then store your sorted list of pokémons in a container (e.g. List<Pokemon>).

Once you are done, output the list to the console.

Hint: The first part of this task i.e. importing data from xml is very similar to Lecture-5 exercise. You would need to create a class with the name "Pokemon" and use properties to store the pokémon's attributes. Only 'Species','Dex' and 'Type' attributes are required to be read from the xml file for this exercise. Also, you'd need another class named "Pokedex" to store the Pokémon list.

Note: You'll need to add *using System.Linq* to header.

Exercise 2 ORM

Note: You need to finish Exercise 1 before proceeding with this exercise and also need to install the following NuGet packages:

- *Microsoft.EntityFrameworkCore.Design*
- *Microsoft.EntityFrameworkCore.Sqlite*
- *EntityFramework Tool, use: dotnet tool install –global dotnet-ef*

Create an SQLite database using the library "Entity Framework". The database should contain the list of pokémons you imported in Exercise 1.

Before exporting data to database, you'd need to create migration and database itself. For that you'd have to use the following commands on terminal:

- *dotnet ef migrations add InitialCreate*
- *dotnet ef database update*

You need to add an 'Id' property to *Pokemon* class to write its objects to database otherwise the program gives an error because of a non-unique id for each database entry.

Note: You can download *"DB Browser for SQLite"* to view databases.

Hint: Before starting, list down the object relations you want in your database. There is no right answer to this problem but a good rule of thumb is to think about which filters a user will most likely use when querying your data.

[1] If you are not familiar with Pokémon, you can think of them as arranged in a random order.