# Databases Advanced Exam - 11 December 2023

Exam problems for the [Databases Advanced - Entity Framework course @ SoftUni](https://softuni.bg/trainings/4234/entity-framework-core-october-2023).  
Submit your solutions in the **SoftUni Judge** system (delete all **bin**/**obj** and **packages** folders) [here](https://judge.softuni.org/Contests/4468/CSharp-DB-Advanced-Regular-Exam-02-December-2023).

Before submitting your solutions in the **SoftUni Judge** system, delete all **bin**/**obj** and **packages** folders. If the **zip** file is still too large, you can delete the **ImportResults**, **ExportsResults** and **Datasets** folders too.

Your task is to create a **database application**, using **Entity Framework Core,** using the **Code First** approach. Design the **domain models** and **methods** for manipulating the data, as described below.

# Cadastre



## Project Skeleton Overview

You are given a **project skeleton**, which includes the following folders:

1. Data – contains the CadastreContext class, Models folder, which contains the **entity classes** and the **Configuration** class with **connection string**
2. DataProcessor – contains the Serializer and Deserializerclasses, which are used for **importing** and **exporting** data
3. Datasets – contains the .json and .xml files for the import part
4. ImportResults – contains the **import** results you make in the Deserializer class
5. ExportResults – contains the **export** results you make in the Serializer class

## Model Definition (50 pts)

The application needs to store the following data:

### District

* Id– integer, **Primary Key**
* Name– **text** with length **[2, 80]** (**required**)
* PostalCode – text with **length** **8.** All postal codes must have the **following structure**:starting with two capital letters, followed by e dash **'-'**, followed by five digits. *Example: SF-10000* (**required**)
* Region– Region enum (SouthEast = 0, SouthWest, NorthEast, NorthWest) (**required**)
* Properties - collection of type Property

### Property

* Id– integer, **Primary Key**
* PropertyIdentifier – **text** with length **[16, 20] (required)**
* Area – **int** not negative (**required**)
* Details - **text** with length **[5, 500]** (**not** **required**)
* Address– **text** with length **[5, 200]** (**required**)
* DateOfAcquisition – DateTime (**required**)
* DistrictId– integer, foreign key **(required)**
* District– District
* PropertiesCitizens - collection of type PropertyCitizen

### Citizen

* Id– integer, **Primary Key**
* FirstName– **text** with length **[2, 30]** (**required**)
* LastName – **text** with length **[2, 30]** (**required**)
* BirthDate – DateTime (**required**)
* MaritalStatus - MaritalStatus enum (Unmarried = 0, Married, Divorced, Widowed) (**required**)
* PropertiesCitizens - collection of type PropertyCitizen

### PropertyCitizen

* PropertyId– integer, Primary Key, foreign key (required)
* Property– Property
* CitizenId– integer, Primary Key, foreign key (required)
* Citizen – Citizen

## Data Import (25pts)

For the functionality of the application, you need to create several methods that manipulate the database. The **project skeleton** already provides you with these methods, inside the Deserializer class. Usage of DataTransferObjects and **AutoMapper** is **optional**.

Use the provided **JSON** and **XML** files to populate the database with data. Import all the information from those files into the database.

You are **not allowed** to modify the provided **JSON** and **XML** files.

**If a record does not meet the requirements from the first section, print an error message:**

|  |
| --- |
| **Error message** |
| Invalid Data! |

### XML Import

#### Import Districts

Using the file "**Districts.xml"**, import the data from the file into the database. Print information about each imported object in the format described below.

##### Constraints

* If there are **any validation errors** for the **district** entity (such as **invalid name, invalid postal code**), **do not** import any part of the entity and **append an error message** to the **method output**.
* If there is **already added a district with the same name** in the database, **do not duplicate** the record. **Do not** import any part of the entity and **append an error message** to the **method output**.
* If there are **any validation errors** for the **property** entity (such as invalid **property identifier**, **details , address**), **do not import only the property entity** and **append an error message to the method output**.
  + The **DateTime** **data** in the document will be in the following fomat: "dd/MM/yyyy"
  + Make sure you use CultureInfo.InvariantCulture
* If the **database** or the **current district** contains **another property with the same PropertyIdentifier**, the **property should not be duplicated**. **Do not import only the property entity** and **append an error message to the method output**.
* If the **database** or the **current district** contains **another property** with the **same Address, do not import the property entity** and **append an error message to the method output**.

|  |
| --- |
| **Success message** |
| Successfully imported district - {**districtName**} with {**propertiesCount**} properties. |

##### Example

|  |
| --- |
| **Districts.xml** |
| <?xml version="1.0" encoding="utf-8" ?>  <Districts>  <District Region="SouthWest">  <Name>Sofia</Name>  <PostalCode>SF-10000</PostalCode>  <Properties>  <Property>  <PropertyIdentifier>SF-10000.001.001.001</PropertyIdentifier>  <Area>71</Area>  <Details>One-bedroom apartment</Details>  <Address>Apartment 5, 23 Silverado Street, Sofia</Address>  <DateOfAcquisition>15/03/2022</DateOfAcquisition>  </Property>  <Property>  <PropertyIdentifier>SF-10000.003.002.001</PropertyIdentifier>  <Area>120</Area>  <Details>Spacious two-bedroom apartment near central park</Details>  <Address>Apartment 8, 47 Green Street, Sofia</Address>  <DateOfAcquisition>01/02/2022</DateOfAcquisition>  </Property>  <Property>  <PropertyIdentifier>SF-10000.004.002.002</PropertyIdentifier>  <Area>150</Area>  <Details>Luxury penthouse with panoramic city views</Details>  <Address>Penthouse 2, 55 High Tower Road, Sofia</Address>  <DateOfAcquisition>10/02/2023</DateOfAcquisition>  </Property>  …  <Properties>  …  </District>  …  </Districts> |
| **Output** |
| Successfully imported district - Sofia with 5 properties.  Successfully imported district - Plovdiv with 4 properties.  Successfully imported district - Varna with 5 properties.  Successfully imported district - Burgas with 5 properties.  Successfully imported district - Blagoevgrad with 4 properties.  Successfully imported district - Veliko Tarnovo with 4 properties.  ... |

Upon **correct import logic**, you should have imported **27 districts** and **117 properties**.

### JSON Import

#### Import Citizens

Using the file **"**Citizens.json**"**, import the data from that file into the database. Print information about each imported object in the format described below.

##### Constraints

* If there **any validation errors occur** for the **citizen** entity (such as invalid **first or last name, marital status value(**check if string is "Unmarried" ||"Married" || "Divorced" || "Widowed"**)**), **do not** import any part of the entity and **append an error message** to the **method output**.
  + The **DateTime** **data** in the document will be in the following fomat: "dd-MM-yyyy"
  + Make sure you use CultureInfo.InvariantCulture

|  |
| --- |
| **Success message** |
| Successfully imported citizen - {**citizenFirstName**} {**citizenLastName**} with {**propertiesCount**} properties. |

##### Example

|  |
| --- |
| **Properties.json** |
| [  {  "FirstName": "Ivan",  "LastName": "Georgiev",  "BirthDate": "12-05-1980",  "MaritalStatus": "Married",  "Properties": [ 17, 29 ]  },  {  "FirstName": "Stefan",  "LastName": "Dimitrov",  "BirthDate": "22-08-1972",  "MaritalStatus": "Divorced",  "Properties": [ 33, 47 ]  },  {  "FirstName": "Elena",  "LastName": "Petrova",  "BirthDate": "03-03-1985",  "MaritalStatus": "Unmarried",  "Properties": [ 12, 54, 60 ]  },  …  ] |
| **Output** |
| Succefully imported citizen - Ivan Georgiev with 2 properties.  Succefully imported citizen - Stefan Dimitrov with 2 properties.  Succefully imported citizen - Elena Petrova with 3 properties.  Succefully imported citizen- Nikolai Vasilev with 2 properties.  Succefully imported citizen - Dimitrina Ilieva with 2 properties.  Invalid Data!  **...** |

Upon **correct import logic**, you should have imported **76** **citizens** with **156 propertiesCitizens**.

## Data Export (25 pts)

**Use the provided methods in the** Serializer class**.** Usage of **Data Transfer Objects and AutoMapper** is **optional**.

### JSON Export

#### Export Properties with Their Owners

Select all the **properties** from that have **date of acquisition equal or later than 01/01/2000. Select** them with their **property identifier, area, address, date of acquisituon and owners(all citizens related to the property)**. For the **citizens**, export their **last name** and **marital status.** Order the **properties** by **date of aquisition (descending)** and then by **property identifier (alphabetically, ascending)**. Foreach property **order** the **citizens** **by last name**(**alphabetically, acsending**).

##### Example

|  |
| --- |
| Serializer.ExportPropertiesWithOwners(CadastreContext dbContext) |
| [  {  "PropertyIdentifier": "SF-10000.004.002.002",  "Area": 150,  "Address": "Penthouse 2, 55 High Tower Road, Sofia",  "DateOfAcquisition": "10/02/2023",  "Owners": [  {  "LastName": "Petrov",  "MaritalStatus": "Married"  },  {  "LastName": "Todorov",  "MaritalStatus": "Married"  }  ]  },  {  "PropertyIdentifier": "SF-10000.006.003.002",  "Area": 100,  "Address": "Apartment 21, 33 Family Street, Sofia",  "DateOfAcquisition": "15/07/2022",  "Owners": [  {  "LastName": "Iliev",  "MaritalStatus": "Married"  }  ]  },…  ] |

### XML Export

#### Export All Properties Larger Than 100 sq.m. with District

Export all **properties** that have area **equal or larger** than **100** square meters. Select them with their **property identifier, area and date of acquisition**. For each **property**, export its related district with its **postal code**. Order the **properties** by **area** (**descending**), then by **date of acquisition** (**ascending**).

##### Example

|  |
| --- |
| **Serializer.ExportFilteredPropertiesWithDistrict(CadastreContext dbContext)** |
| <?xml version="1.0" encoding="utf-16"?>  <Properties>  <Property postal-code="VA-90000">  <PropertyIdentifier>VA-90000.003.005.005</PropertyIdentifier>  <Area>2300</Area>  <DateOfAcquisition>28/08/2008</DateOfAcquisition>  </Property>  <Property postal-code="ST-60000">  <PropertyIdentifier>ST-60000.004.002.002</PropertyIdentifier>  <Area>1150</Area>  <DateOfAcquisition>14/06/2002</DateOfAcquisition>  </Property>  <Property postal-code="PL-40000">  <PropertyIdentifier>PL-40000.002.004.004</PropertyIdentifier>  <Area>1050</Area>  <DateOfAcquisition>03/03/2010</DateOfAcquisition>  </Property>  …  </Properties> |