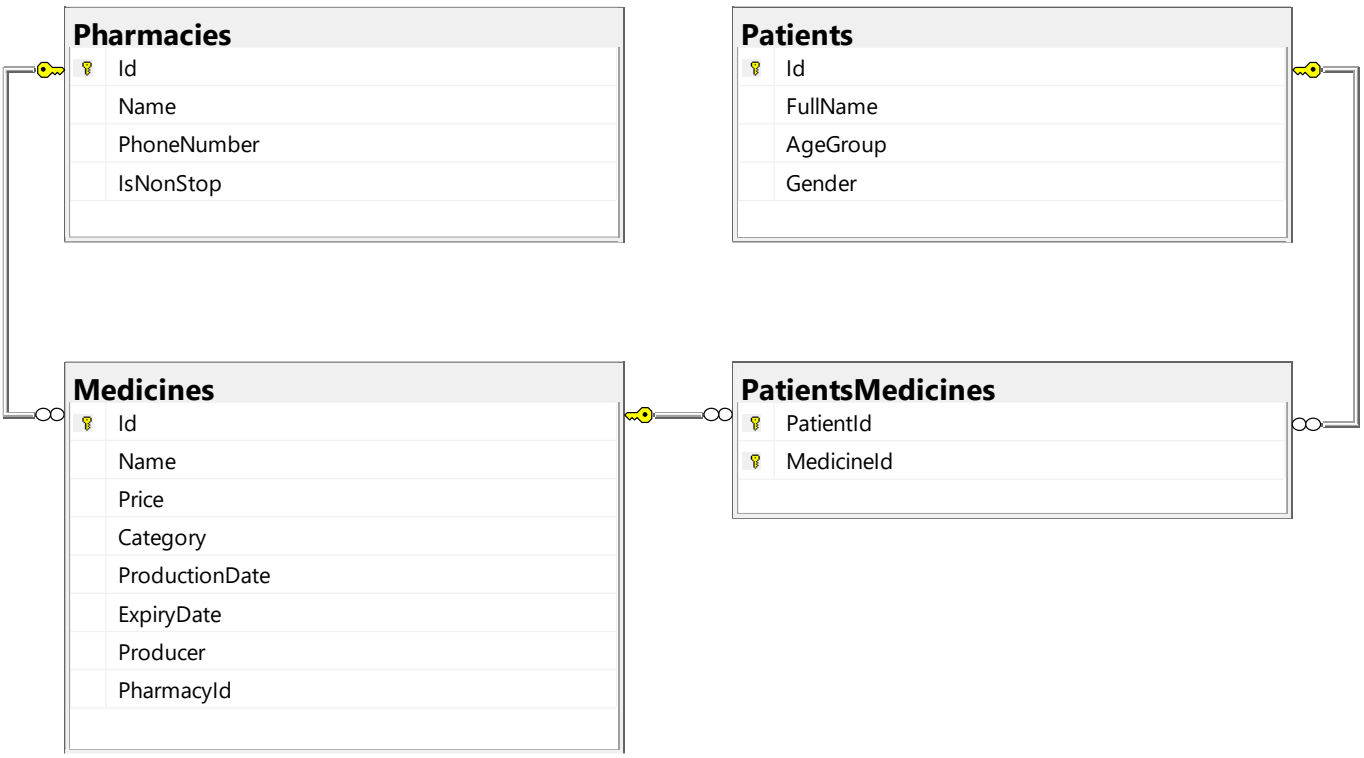
Databases Advanced Exam - 02 December 2023

Exam problems for the Databases Advanced - Entity Framework course @ SoftUni.
Submit your solutions in the **SoftUni Judge** system (delete all **bin/obj** and **packages** folders) here.

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.

Medicines



1. Project Skeleton Overview

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

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

2. Model Definition (50 pts)

The application needs to store the following data:

Pharmacy

- **Id** – integer, **Primary Key**
- **Name** – **text** with length **[2, 50]** (**required**)
- **PhoneNumber** – text with **length 14.** (**required**)
    - All phone numbers must have the **following structure**: three digits enclosed in parentheses, followed by a space, three more digits, a hyphen, and four final digits:
        - Example -> `(123) 456-7890`
- **IsNonStop** – **bool** (**required**)
- **Medicines** - collection of type **Medicine**

Medicine

- **Id** – integer, **Primary Key**
- **Name** – **text** with length **[3, 150]** (**required**)
- **Price** – **decimal** in range **[0.01…1000.00]** (**required**)
- **Category** – **Category enum (Analgesic = 0, Antibiotic, Antiseptic, Sedative, Vaccine)** (**required**)
- **ProductionDate** – **DateTime** (**required**)
- **ExpiryDate** – **DateTime** (**required**)
- **Producer** – **text** with length **[3, 100]** (**required**)
- **PharmacyId** – **integer**, **foreign key** (**required**)
- **Pharmacy** – **Pharmacy**
- **PatientsMedicines** - collection of type **PatientMedicine**

Patient

- **Id** – integer, **Primary Key**
- **FullName** – **text** with length **[5, 100]** (**required**)
- **AgeGroup** – **AgeGroup enum (Child = 0, Adult, Senior)** (**required**)
- **Gender** – **Gender enum (Male = 0, Female)** (**required**)
- **PatientsMedicines** - collection of type **PatientMedicine**

PatientMedicine

- **PatientId** – **integer**, **Primary Key**, **foreign key** (**required**)
- **Patient** – **Patient**
- **MedicineId** – **integer**, **Primary Key**, **foreign key** (**required**)
- **Medicine** – **Medicine**

3. 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 **Data Transfer Objects** 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 Pharmacies

Using the file "**`pharmacies.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 **pharmacy** entity (such as **invalid name, invalid phone number, invalid boolean value (***valid boolean values are only true/false***)**, **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 **medicine** entity such as:
  - invalid **price** or **missing producer**;
  - **production date is on the same day or after the expiry date** or **category** is invalid, **do not import only the medicine entity** and **append an error message to the method output**.
    - The **DateTime data** in the document will be in the following fomat: `"yyyy-MM-dd"`
    - Make sure you use **`CultureInfo.InvariantCulture`**
- If the medicines collection of the **current pharmacy contains another medicine** with the **same name** and **same producer**, the **record should NOT be added and an error message should be appended** to the **method output**.
  - However, **if the producer is different**, or the **medicine is available in another pharmacy** with the same name and producer, **the record will be added**.

Success message
`Successfully imported pharmacy - {pharmacyName} with {medicinesCount} medicines.`

Example

`pharmacies.xml`

```
<?xml version='1.0' encoding='UTF-8'?>
<Pharmacies>
  <Pharmacy non-stop="true">
    <Name>Vitality</Name>
    <PhoneNumber>(123) 456-7890</PhoneNumber>
    <Medicines>
      <Medicine category="1">
        <Name>Ibuprofen</Name>
        <Price>8.50</Price>
        <ProductionDate>2022-02-10</ProductionDate>
        <ExpiryDate>2025-02-10</ExpiryDate>
        <Producer>ReliefMed Labs</Producer>
      </Medicine>
      <Medicine category="4">
        <Name>Lorazepam</Name>
        <Price>25.30</Price>
        <ProductionDate>2022-03-20</ProductionDate>
        <ExpiryDate>2023-03-20</ExpiryDate>
        <Producer>Central Pharma</Producer>
```

```
        </Medicine>
                ...
      </Pharmacy>
</Pharmacies>
```

Output

```
Invalid Data!
Invalid Data!
Invalid Data!
Successfully imported pharmacy - Vitality with 11 medicines.
Invalid Data!
Invalid Data!
Invalid Data!
Invalid Data!
Invalid Data!
Successfully imported pharmacy - GreenLeaf with 5 medicines.
...
```

Upon **correct import logic**, you should have imported **10 pharmacies** and **29 medicines**.

JSON Import

Import Patients

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

Constraints

- If **any validation error occurs** for the **patient** entity (such as invalid **name, age group, gender value**), **do not** import any part of the entity and **append an error message** to the **method output**.
- **If a medicine id is already added to the medicines collection of the patient, do not** add the duplicated id and **append an error message** to the **method output.**

Success message
Successfully imported patient - {**patientName**} with {**patientsMedicinesCount**} medicines.

Example

patients.json

```
[
  {
  "FullName": "Ivan Petrov",
  "AgeGroup": "1",
  "Gender": "0",
  "Medicines": [
    15,
    23
  ]
},
{
  "FullName": "Maria Ivanova",
  "AgeGroup": "2",
  "Gender": "1",
  "Medicines": [
    16,
```

```
    26
  ]
},
{
  "FullName": "Georgi Dimitrov",
  "AgeGroup": "0",
  "Gender": "0",
  "Medicines": [
    1,
    2,
    5
  ]
},…
]
```

Output

```
Successfully imported patient - Ivan Petrov with 2 medicines.
Successfully imported patient - Maria Ivanova with 2 medicines.
Successfully imported patient - Georgi Dimitrov with 3 medicines.
Successfully imported patient - Stafaniya Angelova with 4 medicines.
Successfully imported patient - Dimitar Stoyanov with 3 medicines.
Successfully imported patient - Lyubomir Vasilev with 3 medicines.
Successfully imported patient - Elena Dimitrova with 2 medicines.
Successfully imported patient - Petar Georgiev with 4 medicines.
Invalid Data!
...
```

Upon **correct import logic**, you should have imported **64 patients** with **139 patientsmedicines**.

4. Data Export (25 pts)

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

JSON Export

Export Medicines From Desired Category existing in Non Stop Pharmacies

Select all the **medicines, from a specific category (for this task the category is hardcoded in the StartUp class and passed to the method),** that can be found in **pharmacies working 24/7 (non-stop). Select** them with their **name, price, pharmacy**. For the **pharmacy**, export its **name** and **phone number.** Order the **medicines** by **price (ascending)** and then by **name (alphabetically)**.

In the exported document, the **price** should be formatted to the **second decimal** place and exported **to string** format.

Example

`ExportMedicinesFromDesiredCategoryInNonStopPharmacies(context, medicineCategory)`

```
[
{
  "Name": "Clindamycin",
  "Price": "15.30",
  "Pharmacy": {
    "Name": "Revive",
    "PhoneNumber": "(654) 987-0123"
  }
},
```

```
{
  "Name": "Erythromycin",
  "Price": "16.85",
  "Pharmacy": {
    "Name": "Serenity",
    "PhoneNumber": "(890) 123-4567"
  }
},
{
  "Name": "Ciprofloxacin",
  "Price": "19.20",
  "Pharmacy": {
    "Name": "Vitality",
    "PhoneNumber": "(123) 456-7890"
  }
},
…
]
```

XML Export

Export Patients with Their Medicines

Export all **patients** that have bought at least one **medicine**, **produced after** the given date. For each **Patient**, export their **full name, age group** and **gender**. For each **medicine**, export its **name, price, category, producer** and **expiry date.** Order the **medicines** by **expiry date** (**descending**), then by **price** (**ascending**). Order the **patients** by **medicines count** (**descending**), then by **name** (**alphabetically**).

- The **price** should be exported **to string** format and formatted to the **second decimal** place.
- The **DateTime data** in the document will be in the following fomat: `"yyyy-MM-dd"`
- Make sure you use **`CultureInfo.InvariantCulture`**

Example

`ExportPatientsWithTheirMedicines(context, date)`

```
<?xml version="1.0" encoding="utf-16"?>
<Patients>
  <Patient Gender="male">
    <Name>Stanimir Pavlov</Name>
    <AgeGroup>Adult</AgeGroup>
    <Medicines>
      <Medicine Category="antibiotic">
        <Name>Aleve (Naproxen)</Name>
        <Price>10.50</Price>
        <Producer>HealthCare Pharma</Producer>
        <BestBefore>2025-09-01</BestBefore>
      </Medicine>
      <Medicine Category="antiseptic">
        <Name>Ciprofloxacin</Name>
        <Price>19.20</Price>
        <Producer>ReliefMed Labs</Producer>
        <BestBefore>2025-07-22</BestBefore>
      </Medicine>
…
    </Medicines>
  </Patient>
```

```
…
</Patients>
```