Advanced development technics - Trial Exam

Create an application based on the exercises below written in C#. The solution has to be zipped into a file, named in **firstname\_lastname\_neptuncode** format and uploaded to the location provided by the instructor.

You have 90 minutes to complete the exam exercises.

Create a **single-layer** console application that can extract data from a local database and combine it with a processed XML file.

# DATABASE CREATION (8.5 points)

1. Create a service-based database and name it *FoodRecipes.* (0.5 point)
2. Create a Recipes and *Ingredients* tables using code first approach. (3 points)
   1. *Recipes* table columns:
      1. *Id (int)* – primary key, should be an auto-incremented id (Identity column);
      2. *Name (string)* – Name of the receipt. Max 50 chars, required field;
      3. *Price (int)* – Price of the receipt;
      4. *IsRomantic (bool)* – Indicates whether a recipe is romantic or not.
   2. *Ingredients* oszlopai az alábbiak:
      1. *Id (int)* – primary key, should be an auto-incremented id (Identity column);
      2. *Name (string)* – – Name of the ingredient. Max 50 chars, required field;
      3. *Amount (int)* – Amount of the ingredient;
      4. *ReceiptId (int)* – Receipt id (Foreign key).
3. Set the required parameters for the database connection and also define the foreign key dependencies between the tables. Enable LazyLoading. (4 points)
4. Fill in the tables with test data based on the enclosed *data.txt* file, using the default data input method learned during the semester! (1 point)

# ACCESSING XML FILE DATA (5 points)

1. Create a *Product* class with the following writable, readable public properties: (0 points)

*Name (string) - Name*

*Amount (int) - Amount*

1. Create a *Refrigerator* class with the following writable, readable public properties: (0 points)

*Brand (string) - Brand*

*Capacity (int) - Capacity*

*Products (List<Product>) - Products*

1. Create a static method that can create a *Refrigerator* object from the *frigo.xml* file along with the Product objects in it. (5 points)

# ATTRIBUTES (2.5 points)

1. Create an attribute called DisplayName. (0.5 points)
2. Create the attribute so that it can be applied to properties alone. (0.5 points)
3. The attribute should have a public property in which the display name can be stored. (0.5 points)

This value can also be given as a value in the attribute constructor. (0.5 points)

1. Place the DisplayName attribute on the properties of the Refrigerator and Product classes with meaningful names. (0.5 points)

# HELPER CLASS (4 pont)

1. Create a Helper class called AttributeHelper that can return the display name of a property. (4 points)

Help: *GetPropertyDisplayName<T>(string propertyName)*, where T is the class whose property you want to express, propertyName is the name of the property whose DisplayName attribute you want to display.

# HELPER CLASS UNIT TESTING (6,5 pont)

1. Create a separate DLL in which we can perform unit testing based on what we have learned during the semester. (2 points)
2. Create a class for the AttributeHelper class that tests the operation of the GetPropertyDisplayName() method in three ways:
   1. Create a test for a property that **has** the DisplayName attribute placed on it. (1.5 points)
   2. Create a test for a property that **does not** have the DisplayName attribute placed on it. (1.5 points)
   3. Create a test for a property that **does not exist**. Treat any exceptions that may arise in a way that we learned during the semester (1.5 points)

# PREPARE THE FOLLOWING TASKS BASED ON DATA FROM THE TWO SOURCES (SHOW THEIR RESULTS ON THE CONSOLE) (12.5 POINTS)

1. How many recipes are in the database? (1.5 points)
2. Display the romantic recipes on the console. (1.5 points)
3. Arrange the recipes that contain an ingredient called *Oil* in **descending** order of *price*. (4 points)
4. If we were to make all the recipes, how much of the ingredients would we need?

In the result, display the name of the ingredient and the total amount needed in such a way that it appears in ascending order according to the total amount. (3 points)

1. Display the contents of the *refrigerator* on the console. (1 point)

Use the value of the *DisplayName* attribute on the product properties during display! (2.5 points)

Good luck!