NATIONAL AVIATION UNIVERSITY
FACULTY OF CYBERSECURITY, COMPUTER AND SOFTWARE ENGINEERING
SOFTWARE ENGINEERING DEPARTMENT

Term paper

From subject “Object oriented programming”

Variant 6

Prepared by
Vlad Studennikov
Grout SE-226(E)
Checked by: Dyshleviy O.P.

Kyiv, 2022

Context:

General task:

1. Design a software using a principle of multilayered architecture, where should be next layers:
   - DAL – data access layer (all work with data should be there)
   - BLL - business logic layer (business logic layer should process data from DAL)
   - PL – presentation layer (the layer with which user can contact with an app)

2. Implement some exceptions for BLL and DAL layers that would check data from files and check correctness of the work process.

3. Implement testing using one of the frameworks (MSTest, NUnit, XUnit etc) using triple A principle.

4. Create mnemonic identifiers, formulate your code clearly, add comments if needed. Don’t use public fields in classes. If you need to refer them, use the properties, indexers or individual accessory methods. Classes should be described in separate files.

Individual task:

Functional Software Requirements

1. Client management:

1.1. Ability to add customers

1.2. Ability to remove customers

1.3. Ability to change customer data

1.4. Ability to view data of a specific customer

1.5. Ability to view list of all customers:

1.5.1. Ability to sort the list by name

1.5.2. Ability to sort the list by surname

1.5.3. Ability to sort the list by initial digit of bank account

2. Management of real estate(property) data:

2.1. Possibility to add property

2.2. Ability to delete property

2.3. Possibility to change data of property

2.4. Ability to view data of a particular property

2.5. Ability to view a list of all properties:

2.5.1. Ability to sort the list by type of property (Apartment 1-bedroom, 2-bedroom, 3-

bedroom, private area)

2.5.2. Ability to sort the list by property value

3. Management of Property offers:

3.1. To the list of suggestions client can add n real Estate N < 5

3.2. Possibility according to client's requirements (type of real estate and its value)

Determine if the desired item is in the list of available properties.

3.3. Customer's ability to decline an offer of a specific property.

4. Search:

4.1. Ability to search by keywords among clients

4.2. Ability to search by keyword among properties

4.3. Ability to search for all data (among real estate and clients) by the key word

4.4. Advanced customer search (when given a specific set of data, such as surname and desired type of property)

Description of program layers:

1. **DAL –** data access layer. This layer works directly with files, it can get the data from files, update files or delete some information from them.
   In DAL could be also found classes that are used in a program: class for Customer, class for Property and class for Property:

   1. Class "Customer": has next fields:
   - First name;
   - Last name;
   - Number of bank account;
   - Email;
   - Telephone number;
   - List of requirements for the property;

   All fields could be changed using properties.

   2. Class "Property": has next fields:
   - Type of property: flat or house;
   - Quantity of bedrooms (from 1 to 20);
   - City;
   - District;
   - Price;
   - Information whether customer can buy the property or take it in rent;

   All fields could be changed using properties.

   3. Filters: filters are added to customers when they want to choose a flat or a house.
      There are several filters that customer can add for choosing the property:
   - Filter for the type of property – whether customer wants to get a flat or a house;
   - Filter for city where property is situated;
   - Filter for district where property is situated;
   - Filter for quantity of bedrooms;
   - Filter for the price;
   - Filter that defines whether customer wants to rent a property or to buy it.

All filters implements 1 interface and all filter classes have function “satisfies(). It checks whether the property that was passed as an argument suits or not for the given filter.

In DAL could be found interface “IDataProvider” – it is an interface for all classes that will work with files. It is a generic interface, so it will work with all data types. Main functions in this interface: Read and Write, those functions have to be implemented in all classes of IDataProvider interface. Read function is used for getting information from the file, Write – function for adding new information to a file.
In my program was implemented 1 data provider – binary data provider (class BinaryDataProvider). In this class were implemented 2 functions from IDataProvider interface – Read – return the list of desterilized items, and Write – add object or other information to a binary file.

Another interface that could be found in DAL – IDataContext. It can help to improve working with file, because in data context we can set all data providers that are in DAL layer. Also in IDataContext could be found property Link, and it makes working with program much more easier, because we don’t need to enter links to file all the time when we are working with it.
Class that was built using interface IDataContext – MainDataContext – it has several functions: GetData() – read data from the data provider (from the file), and SetData() – writing the data to the file. Also there is a function clearFile() to delete information from the file. This function would be necessary for deleting of the elements.

2. **BLL** – business logic layer. It process the data that was received from data access layer.
BLL contains several classes: CustomerServices – processing customer’s data, and PropertyServises – processing property data.

Let’s start from processing customers data:

1. In class CustomerServices we can found function addCustomer(), which receives characteristics of the customers as an argument, or receives an object as an argument.
2. There are several ways to find customers: by their name and surname or using a string. If it is needed to find the customer from name and surname, method reads the information about all customers from the list and checks whether some customers have appropriate name and surname. If someone has, this object would be returned. If as an argument was passed a simple

string, function would check whether characteristic of any customer from the list starts with a string. If object's data starts with the string, this object would be added to the created list that would be returned after method will end the work.

3. There is an ability to delete customer – for this was used method deleteCustomer(). If this method found the customer which should be deleted, it will delete him from the list, delete all information from the file where was an information about customers and will rewrite new information to the file without customer that was deleted.
4. Method returnCustomers() returns data about all customers from the file.
5. Method sort() is used for sorting, as an argument this method receives a string.

   If this string starts with "last", all customers would be sorted by their last name. Sorting would be done in another list that would be returned after method ends its work.

   If argument start with "first" – customers would be sorted by their first name. List with sorted customers would be returned after method ends its work.

   If argument start with "bank" – customers would be sorted by their bank account number. Sorted customers would be returned from method as the new list.

Processing of property data (in class PropertyServices):

1. For adding new property to a file used method addProperty(). As an argument it receives an object or all characteristics of the property that should be added to the file.
2. To delete the property used method deleteProperty(). The way it works is the same as the process of deleting of some customers from the file – method searches for the property that should be added by its characteristics and, if it was found, delete this property from the list, delete all information from the file and write new information to it.
3. To find property used function findProperty(). There are a bunch of overloading for this method:
   - Method can receive property characteristics and return the property that was found;
   - Method can receive property, and, if was found property with same characteristics, it would be returned;
   - Method can receive a string – if so, method compares all fields of properties that are in file, and if at list one property starts with this string, it will be returned.

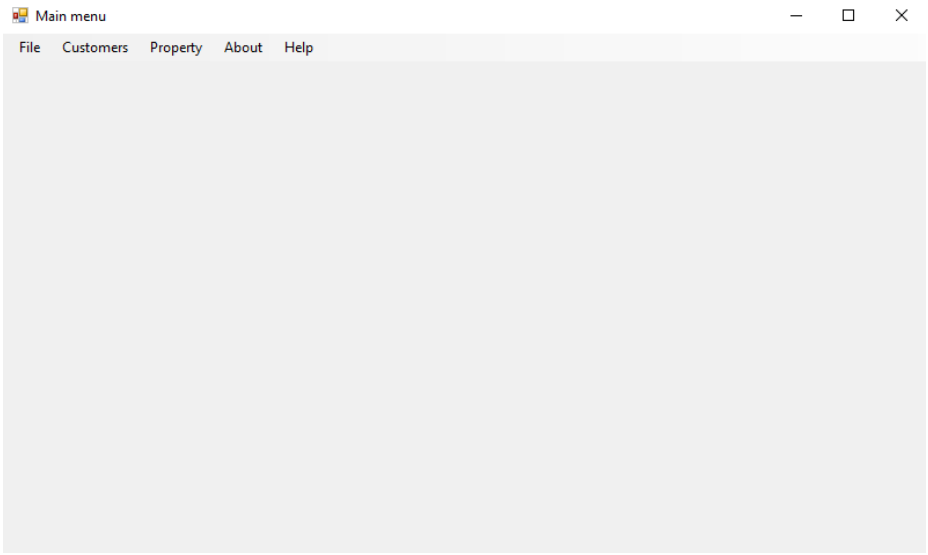
4. The main function of this program – helping to find property for customers from their preferences. For this used method findPropertyFromRequirements(). It compares filters that user chose and returns property that will be suitable for the customer.

3. **PL** – presentation layer – it is needed to interact with user.
Presentation layer has a class "Menu" that defines all functions that could be used in different types of interfaces and types of applications.

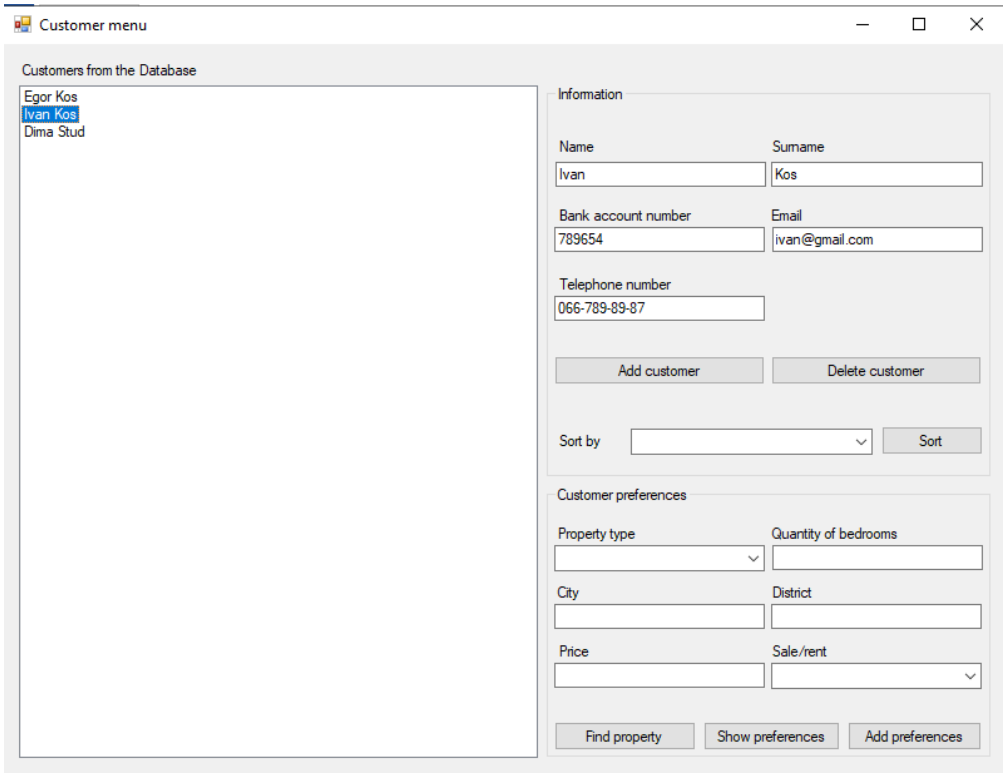
List of methods in class "Menu":
- addCustomer() – adding the customer;
- addProperty() – adding the property;
- deleteCustomer() – deleting of the customer;
- deleteProperty() – deleting of the property;
- sortCustomer() – sorting of the customers;
- sortProperty() – sorting of the property;
- getListOfProperties() – get the list of all properties from the file;
- getListOfCustomers() – get list of all customers rom the file;
- findPropertyFromRequirements() – find property for particular customer;

User interface was implemented using WinForms.

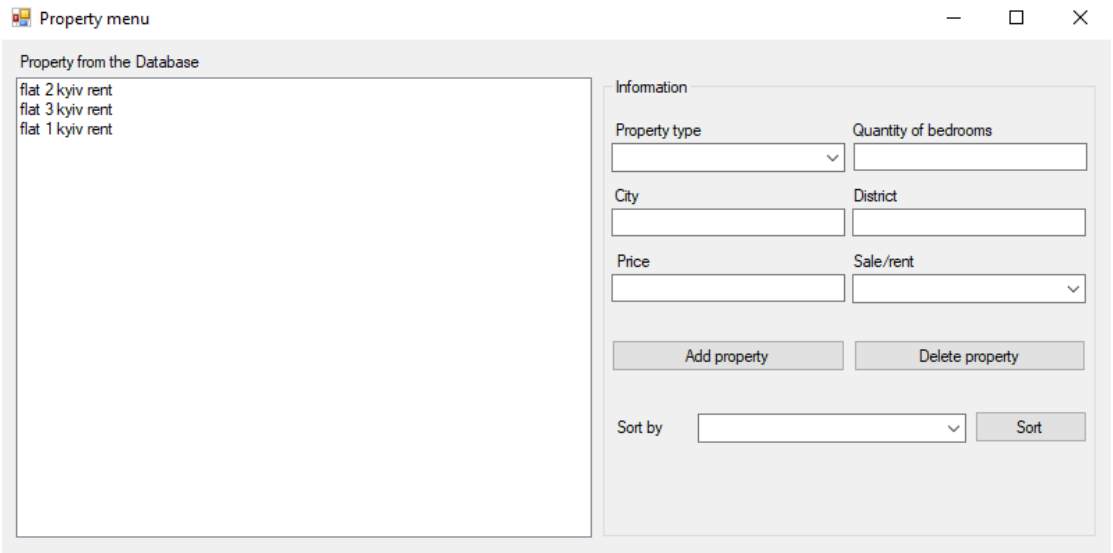
Form1 – main form in the program. Here we can find a menu, where we can open new windows:



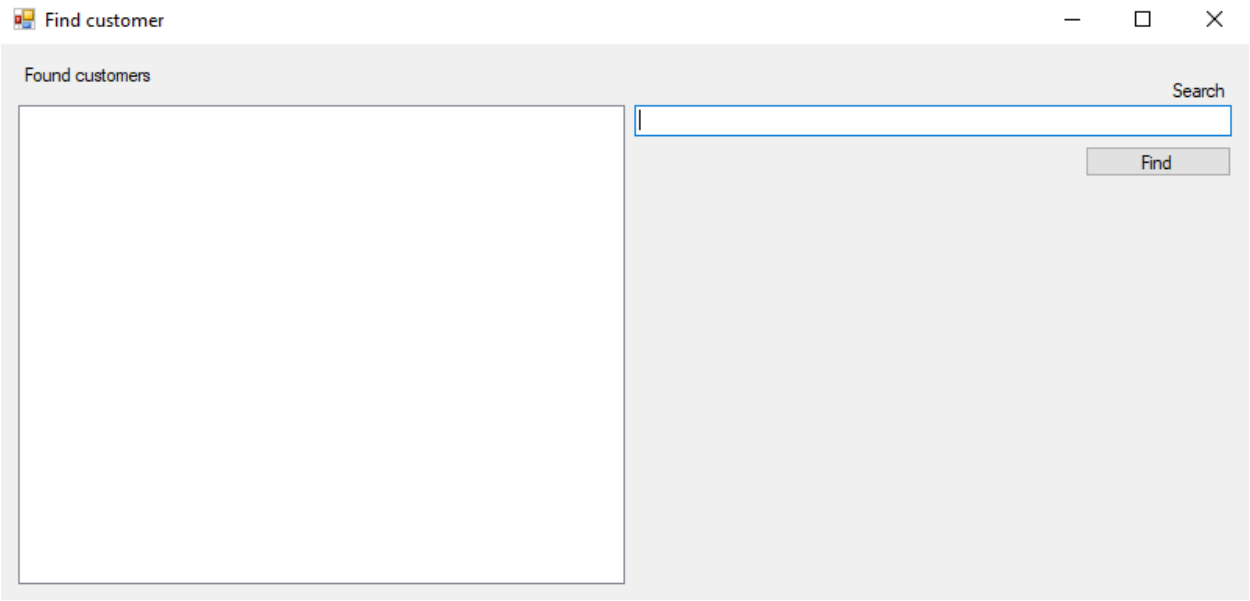
Form2 – customers menu. From this menu we can add new customers, change their data, delete customers, sort them and add some requirements to the property.



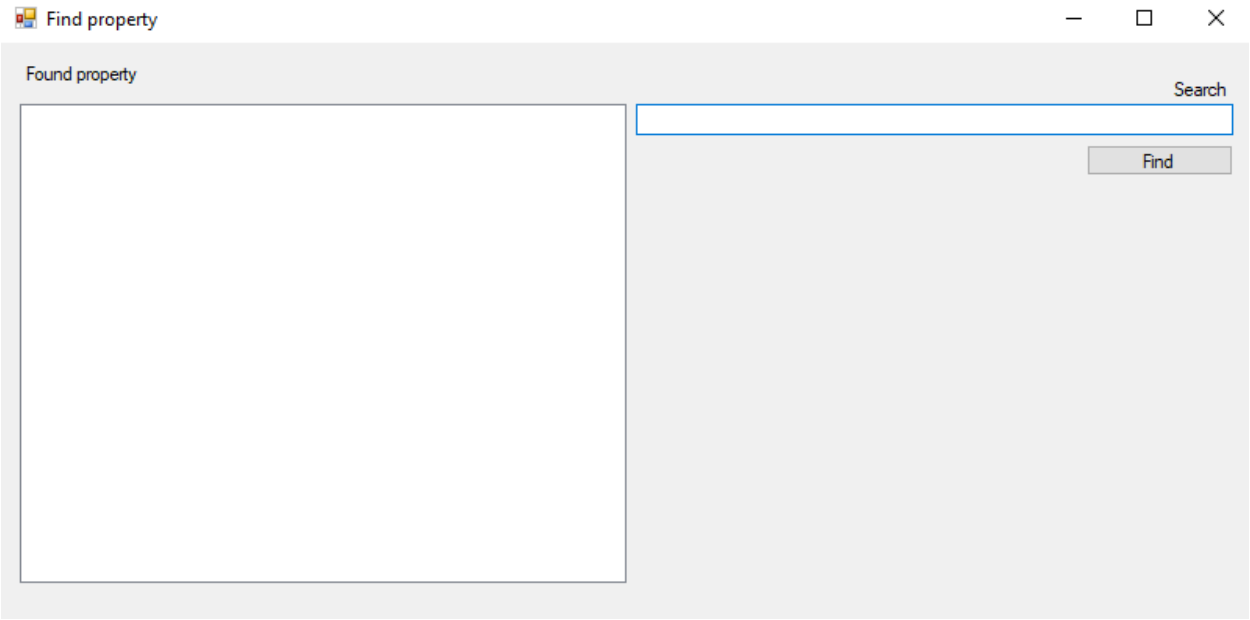
Form3 – property menu. Here we can add new property, delete property, sort property and change the data about it.



Form5 – form for searching for customers.



Form7 – form for searching for properties.



Also there are some projects for testing: BLL.Tests (testing of business logic) and DAL.Tests (testing of data access layer). For testing was used framework MSTest.

Description of project components:

1. C#:

The C# programming language was designed by Anders Hejlsberg from Microsoft in 2000 and was later approved as an international in 2003. Microsoft introduced C# along with .NET Framework and Visual Studio, both of which were closed-source.

The musical notation whereby a sharp symbol indicates that the written note should be made a semitone higher in pitch inspired the name «C sharp». This is similar to the language name of C++, where "++" indicates that a variable should be incremented by 1 after being evaluated. The sharp symbol also resembles a ligature of four "+" symbols (in a two-by-two grid), further implying that the language is an increment of C++

The core syntax of the C# language is similar to that of other C-style languages such as C, C++ and Java.

2. WinForms:

Windows Forms (WinForms) is a free and open-source graphical (GUI) class library included as a part of Microsoft .NET.

The code for the application can be written in a .NET programming language such as C# or Visual Basic.

All visual elements in the Windows Forms class library derive from the Control class. This provides the minimal functionality of a user interface element such as location, size, color, font, text, as well as common events like click and drag/drop. The Control class also has docking support to let a control rearrange its position under its parent.

WinForms is old and successful technology. It is simple to use WinForms, but now Microsoft develops other frameworks for user interface, such as WPF and UPF and others.

3. MSTest:

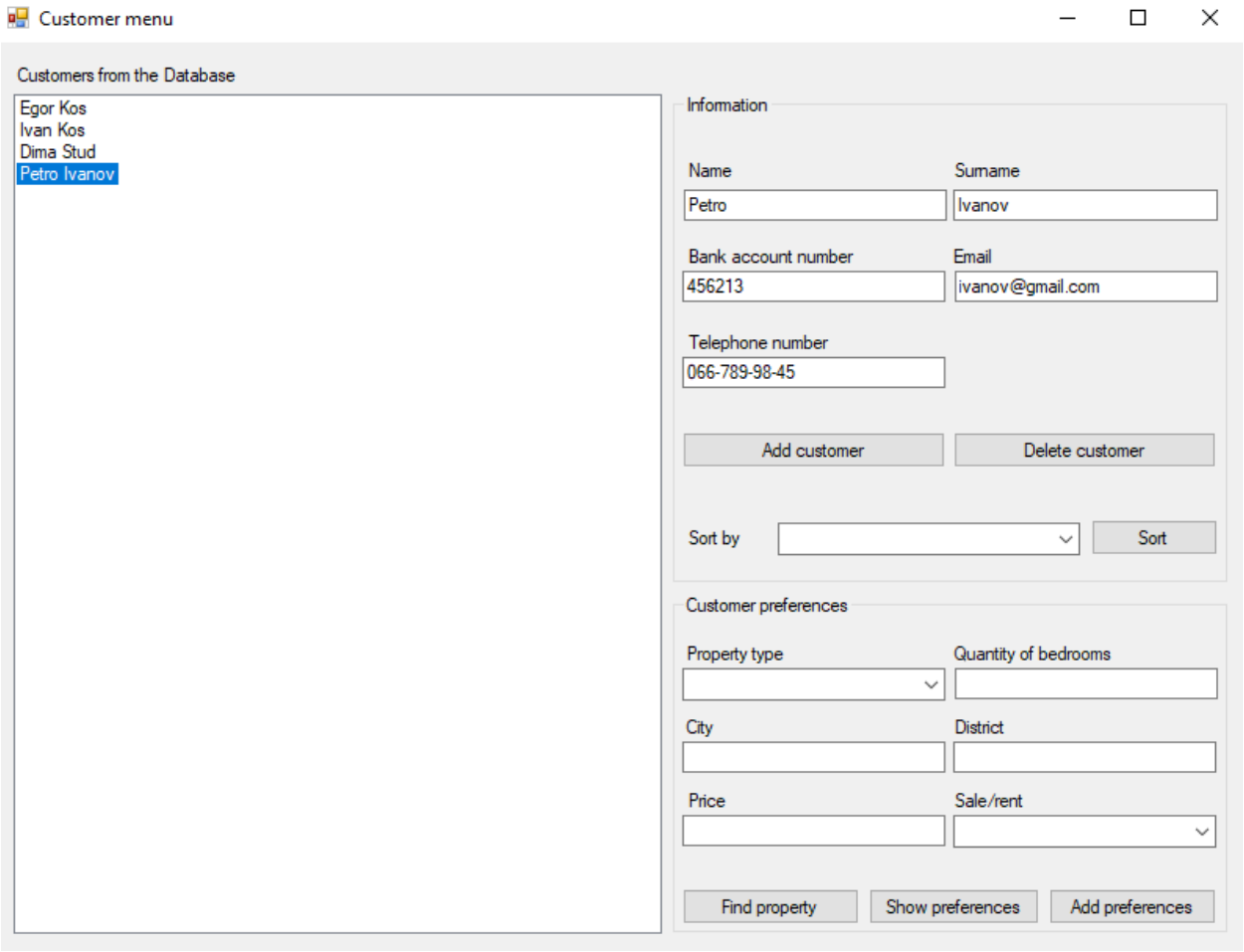
MSTest is a framework for testing developed by Microsoft. MSTest is used as a part of Visual Studio IDE and usually also called Visual Studio unit testing framework.

In tests was used “Triple A” principle (Arrange, Act, Assert):

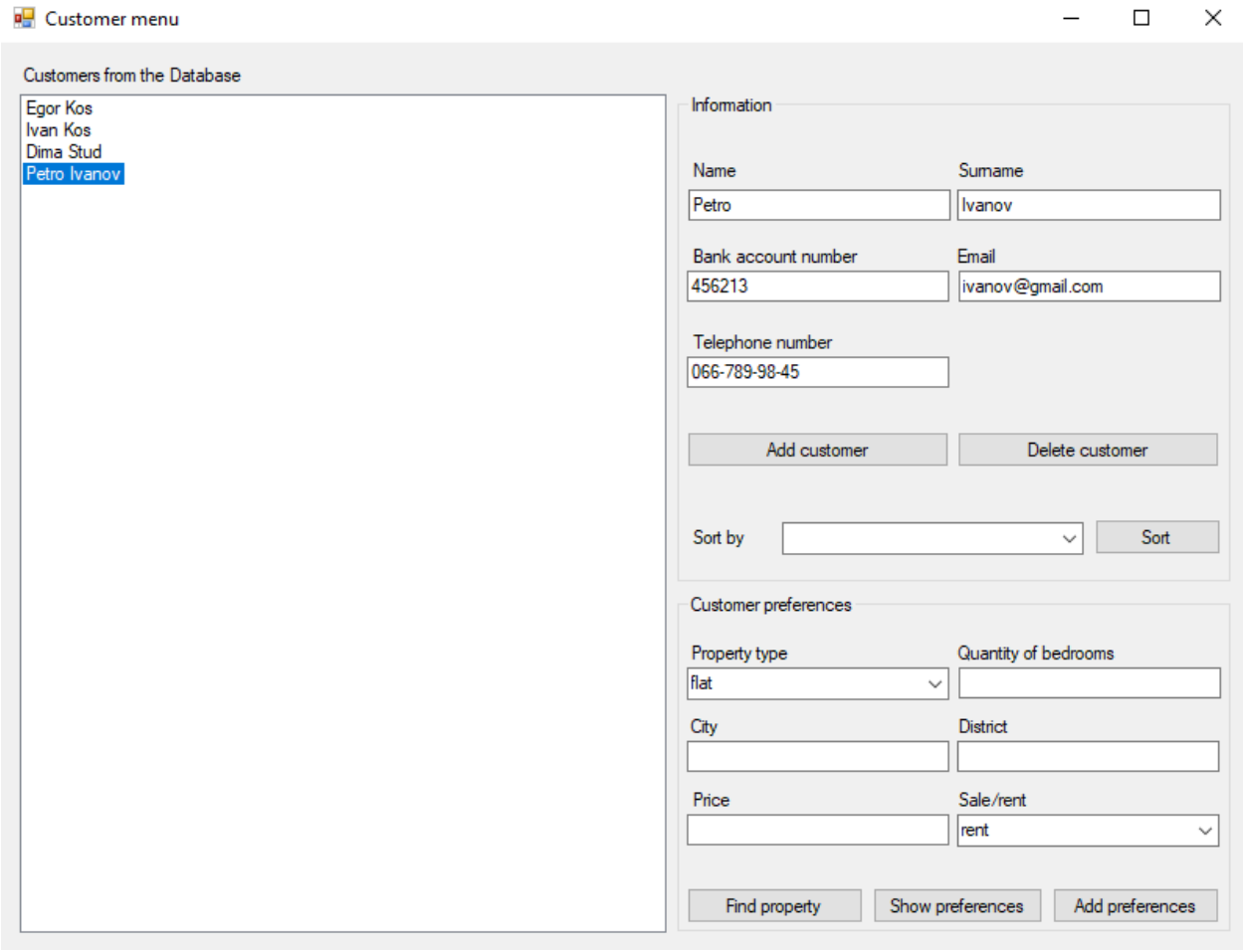
- Arrange – arranging variables or literals that would be used in testing;
- Act – Calling of the testing methods;
- Assert – checking the result of method work and the correct result (in MSTest for this used class Assert).

Result of program work:

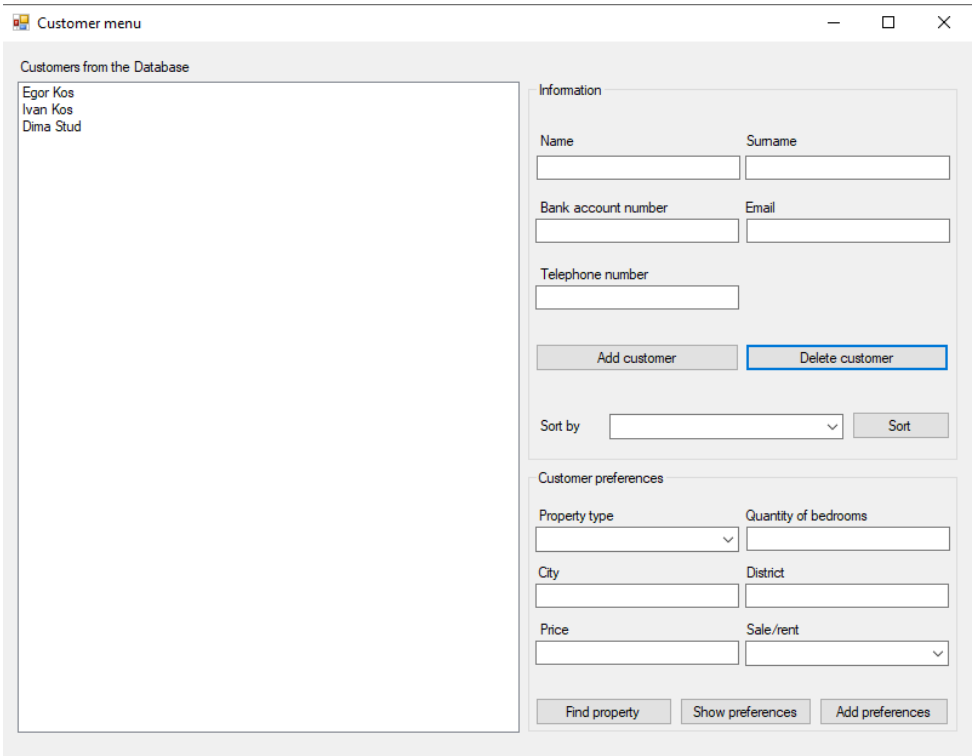
1. Adding a customer:



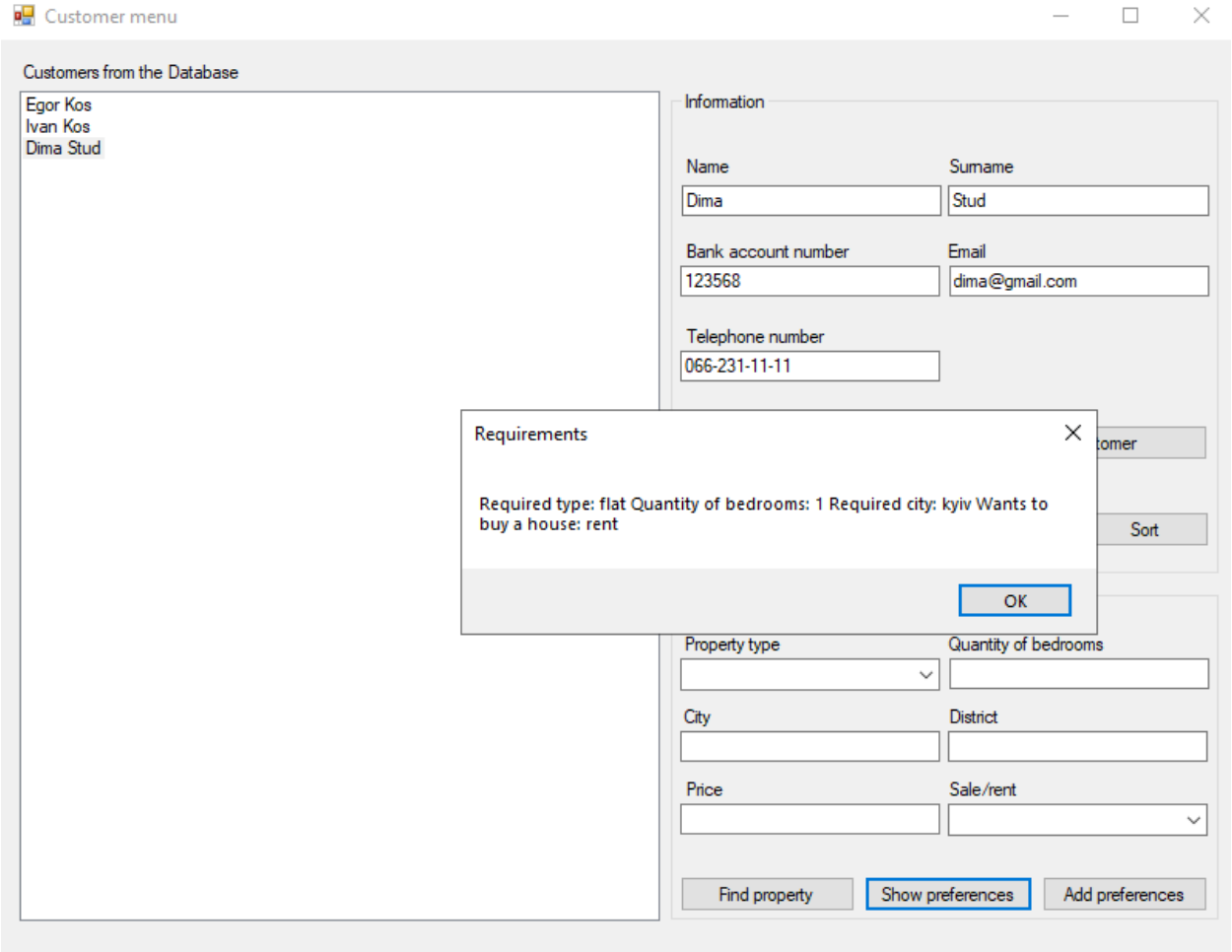
2. Setting preferences to a customer:



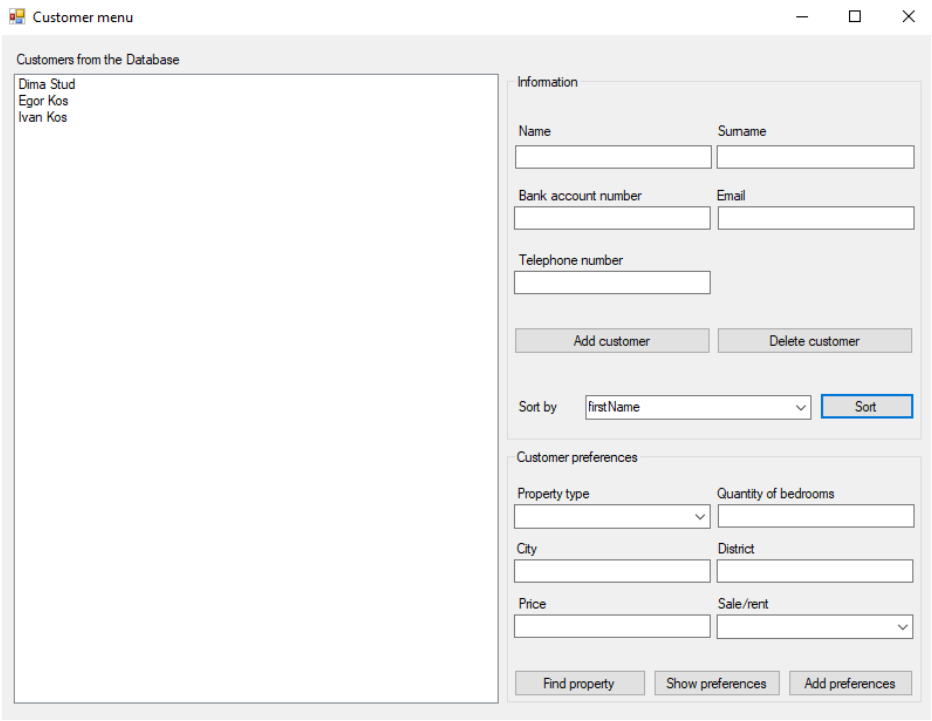
3. Deleting of the customer (pressing button “Delete customer”):



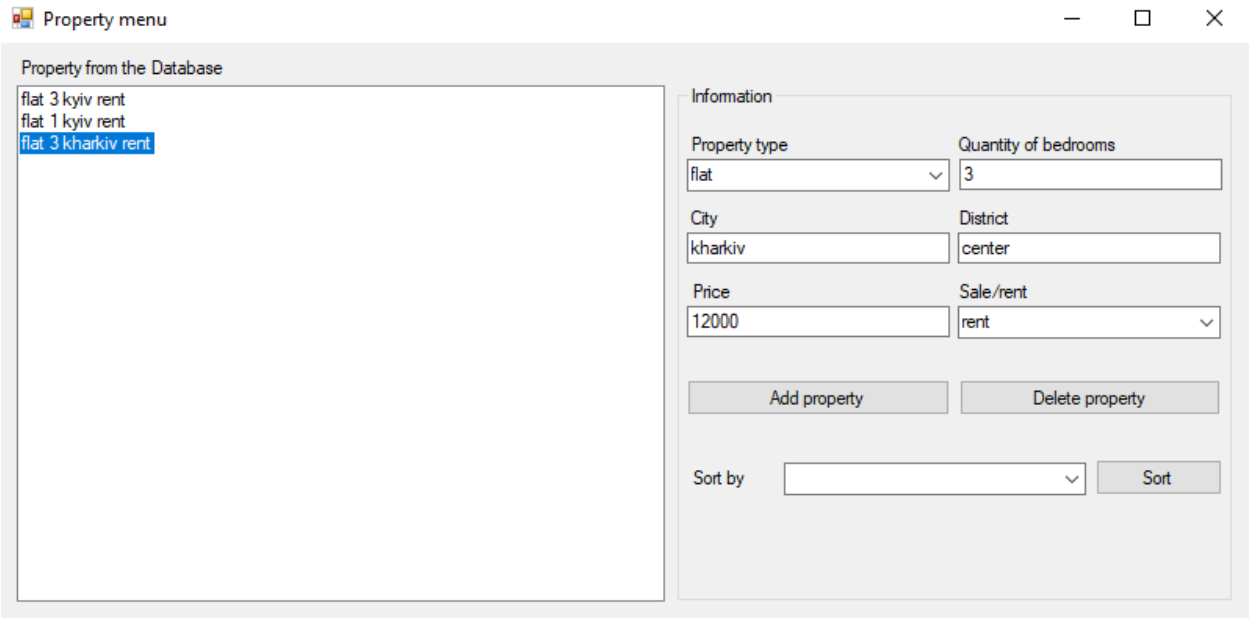
4. Considering customer preferences:



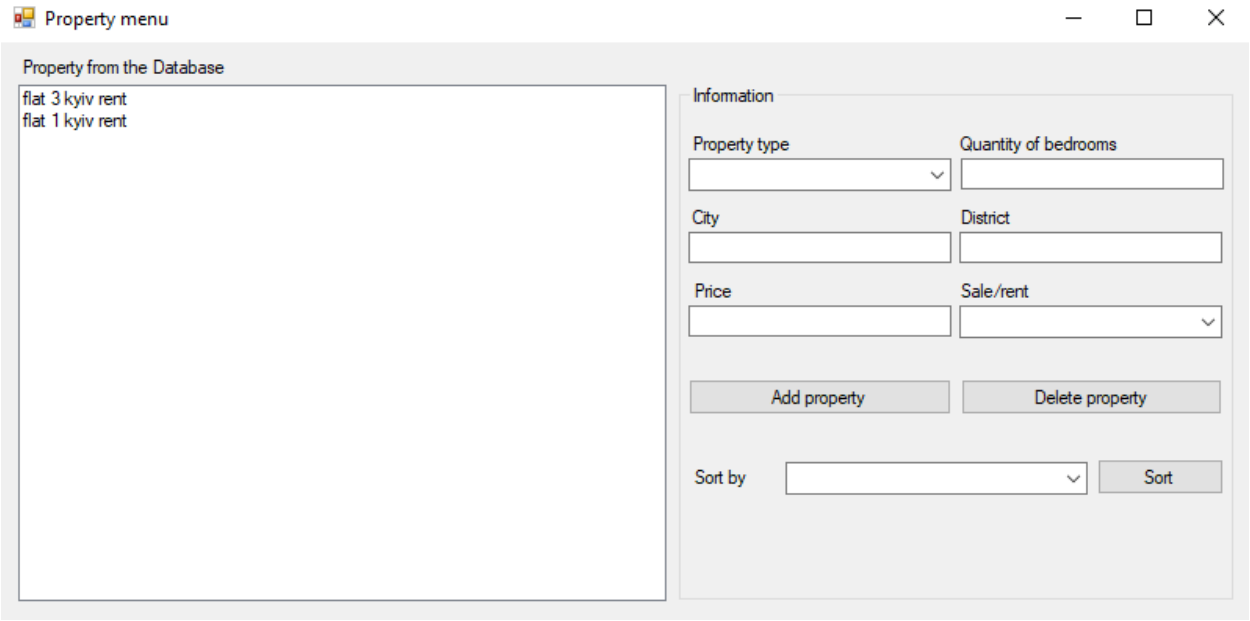
5. Sorting customers:



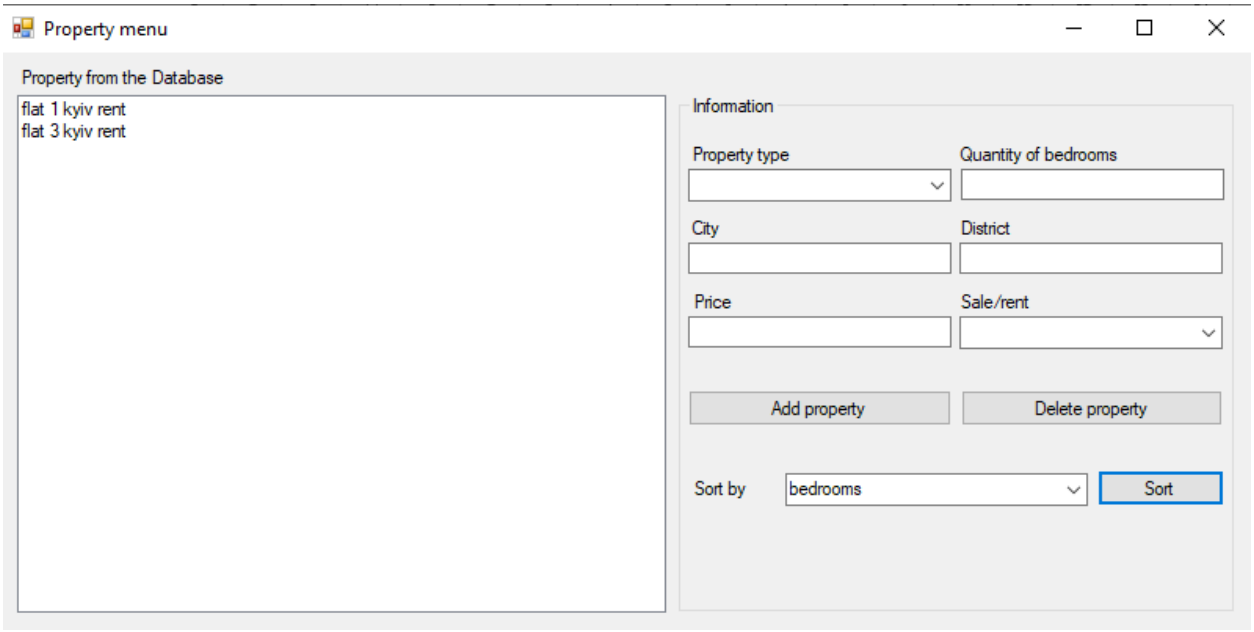
6. Adding property:



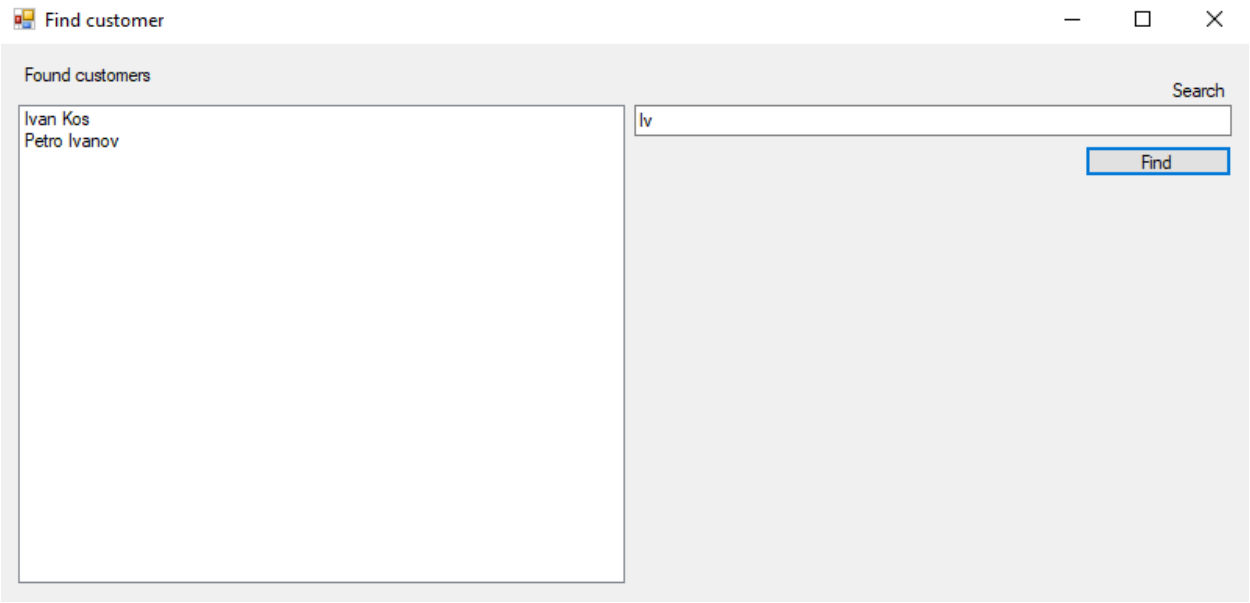
7. Deleting property:



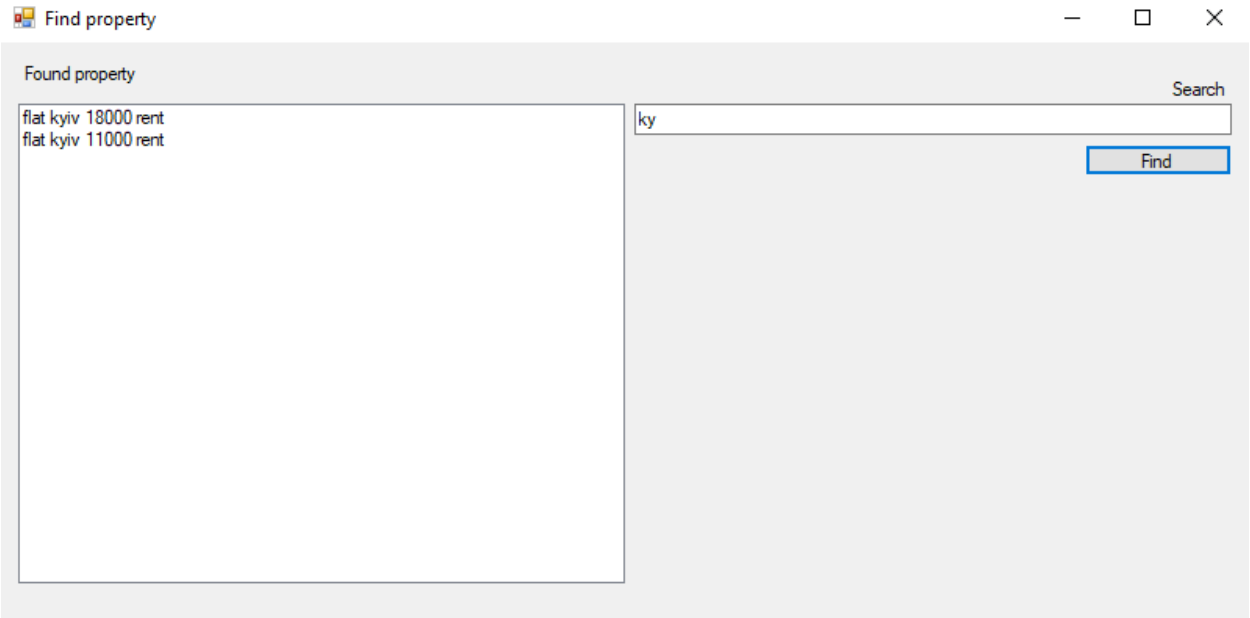
8. Sorting property:



9. Find customer:



10. Find property:



11. Find property from preferences: