Assignment -3: Properties and Indexer

Objectives:

O[1]. To learn properties and apply them in real-world problem-solving.
O[2]. To learn Indexers and apply them in real-world problem-solving.

In this works, students are going to learn program writing in the C# console-based platform with the use of properties, and indexers.

Task1: Implement a class named Account that contains the following members:

- A private int data field named Id for the account (default 0).
- A private double data field named Balance for the account (default 0.0).
- A private double data field named minBalance for the account (default 1000.0).
- A no-arg constructor that creates a default account with Id=0 and Balance=0.0
- An argument constructor that creates an account with the specified id, initial balance
- A method named setBalance() that sets a given amount to an instance variable Balance if the given amount is more than minBalance.
- A method named getBalance() that returns the value of the current balance.

a) Create a **class library (.dll)** of the class and execute the dll from another project in Visual Studio 2019 console Editor.
b) Check that the private field members are not visible from your driver project.

Property in C# is a member of a class that provides a flexible mechanism for classes to expose private fields. Internally, C# properties are special methods called accessors. The set accessor automatically receives a parameter called value and assigns it to property. Get accessor automatically returns the property value. The value keyword represents the value of a property. Property is accessed as a public field. We can also use a property instead of a method that has one parameter and a return value because we can define custom logic in the get and set accessors.

- A property with get and set accessor is a Read/Write property(Id, Balance).
- A property with get accessor is a Read-only property (minBalance).
- A property with set accessor is a Write only property.

C# also supports static properties, which belong to the class rather than to the objects of the class. All the rules applicable to a static member apply to static properties also. Remember that set/get accessor of static property can access only other static members of the class. Also , static properties are invoking by using the class name.

Task2: Change the problem in Task1 so that we can add a property for balance private variable and call set and get accessors automatically.

a) Create a class library (.dll) of the class and execute the dll from another project in Visual Studio 2019 console Editor.
b) Check that the private field members are not visible from your driver project and you do not need to share to the driver project.
c) Change the property to be static. Thus Balance also needs to be static. Explain Why?
d) Create a Read Only property for minBalance and try to change the value of minBalance property. You will get Compile error. Explain- Why?
e) Show auto-implemented properties(get; set;). Auto-implemented properties in C# make code more readable and clean if there is no additional calculation needed. Compiler creates a private anonymous field that can only be accessed through the get and set accessors.

This feature in C# allows you to index as class or struct as you would do it with an array. When we define an indexer for a class, we force it to behave like a virtual array. The array access operator, or [], can be used to access instances of a class that implements the indexer. The user can get or set the indexed value without pointing to an instance or a type member. The indexers are very similar to properties, but the main difference is that accessors to the indexers will take parameters, while properties cannot.

Task3: Implement a class named Account that contains the following members:

– A private double data field array named Balance. Which will hold the account balance of customers against their customer id.
– An argument constructor that allocates the size of the Balance and the size will be given during the Account class instantiation.
– An indexer with set and get accessors will be added in the Account class.
    - set accessor will check the valid id (0 to size) and set the value to the Balance array corresponding to customer id location, if and only if the id is valid.
    - get accessor will return the Balance array value of a customer only if the customer id is valid, otherwise, return the -2 value.

a) Create a class library (.dll) of the class and execute the dll from another project in Visual Studio 2019 console Editor.
b) Test the driver class with the creation of an Account object
c) Now use the object reference as a virtual array with [ ]operator to create 5 balance values and store them using indexer.
d) Trace the result of how the indexer is used to store and retrieve balance values.
e) Show overloaded indexer to use an index as string

```
String[] array= new string[] {“zero”,“one”, “two”, “three”, “four”, “five”}
public int this[string name]
 {
     get
     {
       for(int i=0, i<6; i++){
            if(array[i].ToLower() == name.ToLower())
              return Balance[i];
```

```
                }

            return 0;
        }
    }
```

f) Show multi-dimensional indexers.

C# includes specialized classes that store series of values or objects are called collections. There are two types of collections available in C#: non-generic collections and generic collections. The System.Collections namespace contains the non-generic collection types and System.Collections.Generic namespace includes generic collection types. In most cases, it is recommended to use generic collections because they perform faster than non-generic collections and also minimize exceptions by giving compile-time errors.

Generic List<T> contains elements of the specified type. It grows automatically as you add elements to it. The List<T> is a collection of strongly typed objects that can be accessed by index and having methods for sorting, searching, and modifying the list.

Task4: Create a student list using List collection and access the values using Indexers.

a) Implement a student class and create property for Id, Name, and Address.
b) Implement a department class and create a student list using List collection
c) Add few student objects into the List
d) Create an indexer into the department class and use the instance of the class to behave a virtual array.
e) Indexer get will return the student name whose id is given as index.
f) Indexer set will change the student name whose id is given as index.
g) Show the working strategies of the indexer.