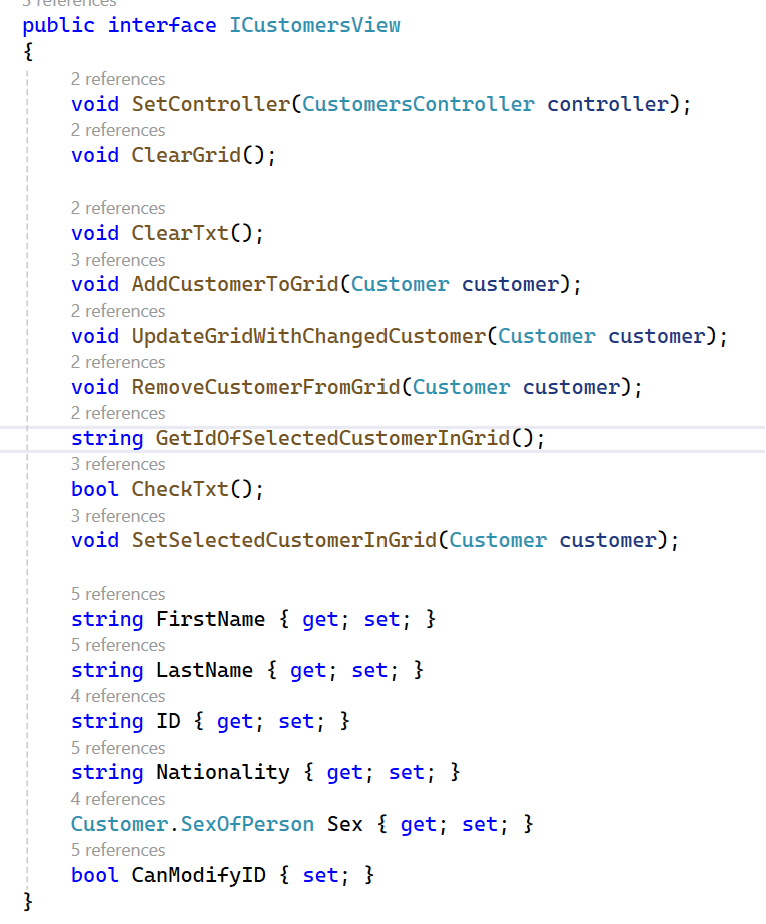
The “Customer Information Management” application show a list of customers and allow the client to add, update, and delete customer information via the Controller class, which is organized according to the Model-View-Controller pattern. The purpose is to separate the user interface into View (produce the display, call the Model to get information as necessary), and Controller (in respond to user request as well as interact with both View and Model as necessary).

MVC, a well-proven design pattern, is to solve the problem of separating data (model) and user interface (view) concerns in order not to affect the data handling when there are changes to the user interface. In the meantime, the changes to the data set can be done with impacting/ changing the UI. The MVC pattern resolves this problem by decoupling data access and business logic layer from UI and user interaction by introducing the controller, an intermediate component. The MVC architecture allows the creation of reusable components within a flexible program design in which components can be easily modified.

The Description of three Components in the design as below:

* Controller: to separate the logic from the View, we need to make a view be as helpless as possible, and the controller do all the hard work and just give the View some simple commands not requiring any further processing. Therefore, the application design has defined an Interface, called ICustomersView, which the View must implement. This interface contains only the signatures of properties/methods we need to use as the screenshot below:



In the design application, subscribe to the events and then delegate the handling to the Controller which connect the Model (Customer class) with View (CustomerView class). The use of interfaces makes the test more robust to changes in the system, and the CustomersController depends on abstractions (interfaces). It is easier than ever to change the behaviour of a concrete class. instead of creating concrete objects in CustomersController class, we only pass the objects to the constructor of CustomersController. The controller class is very important and crucial to the application. Therefore, we need to keep the controller class light, agile, and loosely couple to other components of the program. Below are methods used in the controller class.

