

Mid-Term Part 2 – Programming Part

1. GDC Inc. has two types of customers: wholesale customers and retail customers. Both customer types are derived from a Customer base class, both derived classes extend the Customer class by adding a property and separate forms are used to add two types of customers. See the following diagram to help create your classes and Add Customer forms.

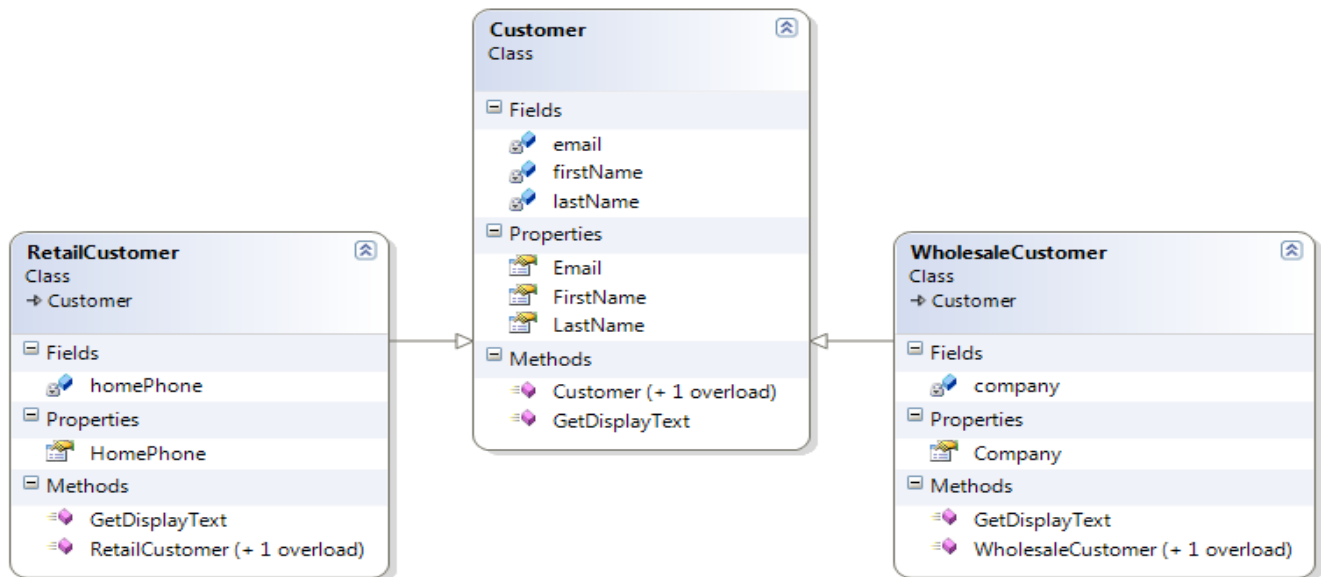
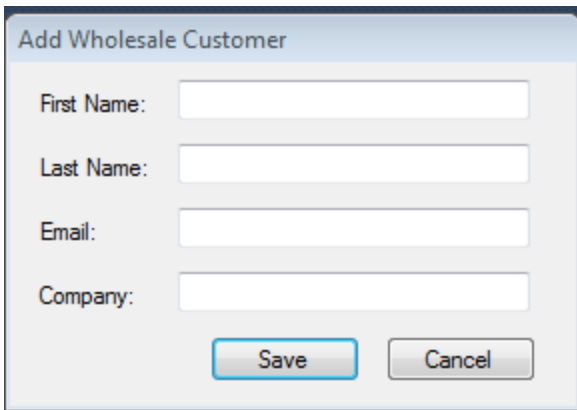


Figure 1 - Class Diagram with Class Details

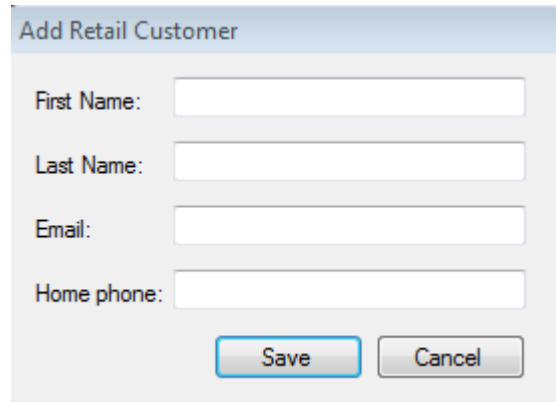
The screenshot shows a window titled "Customer Maintenance". Inside, there is a text area labeled "Customers:" containing the following text: "John Smith, john.smith@GSC.com (GSC Inc.)" and "Mary Jones, mary.jones@gmail.com ph: 123-234-678". To the right of the text area are four buttons: "Add Wholesale", "Add Retail", "Delete", and "Exit". The "Add Retail" button is highlighted with a blue border.

Figure 2 Customer Maintenance Form



The 'Add Wholesale Customer' form is a light blue dialog box with a title bar. It contains four text input fields labeled 'First Name:', 'Last Name:', 'Email:', and 'Company:'. At the bottom right, there are two buttons: 'Save' (highlighted with a blue border) and 'Cancel'.

Figure 3 Add Wholesale Customer Form



The 'Add Retail Customer' form is a light blue dialog box with a title bar. It contains four text input fields labeled 'First Name:', 'Last Name:', 'Email:', and 'Home phone:'. At the bottom right, there are two buttons: 'Save' (highlighted with a blue border) and 'Cancel'.

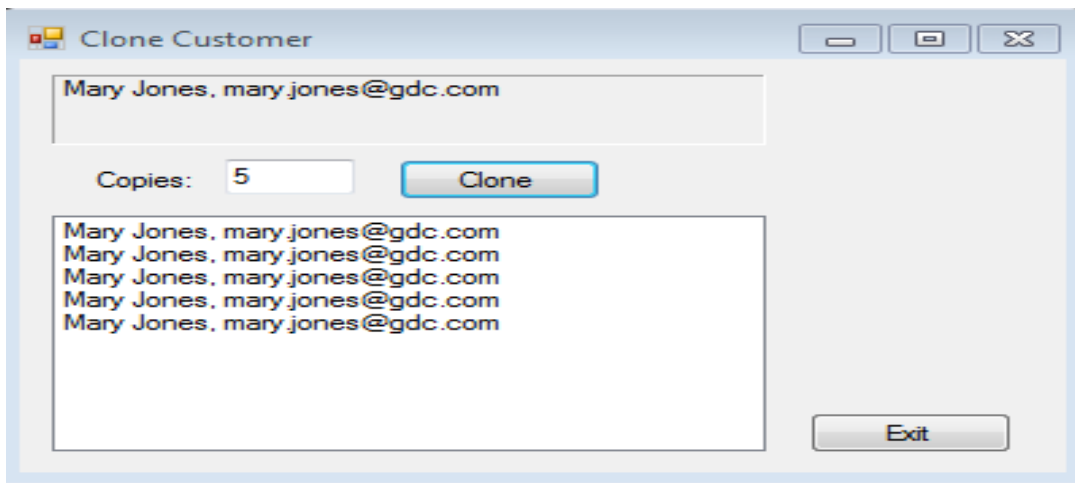
Figure 4 Add Retail Customer Form

- Create a customer maintenance application which uses the above class diagram to create Customer base class, and WholesaleCustomer and RetailCustomer derived classes. The base class Customer has three fields, firstName, lastName, and email and they are all string data type. The Customer class also contains a virtual method named GetDisplayText(). Wholesale customer has one extra field named company with string data type and the Retail customer also has one extra field named homePhone. Both derived classes will override the method GetDisplayText() to display info from base class plus their own individual fields.
- Create Add Wholesale Form and Add Retail Form which allow users to enter information for new customer from each type and add Click event for Save and Cancel buttons. The event handler for the Cancel button should just close the form. In your event handler from the Save button Click event, you should validate the data and then create a new instance for that customer type using the data entered by the user.
- In your Customer Maintenance form, write event handler for Click event of Add Wholesale and Add Retail buttons. These methods should create an instance of the appropriate Add customer form and then call its GetNewCustomer() method. (GetNewCustomer() method is created in Add Wholesale Customer form and Add Retail Customer form to display the form instance and return its new customer, here is the code for this method.)

```
public Customer GetNewCustomer()
{
    this.ShowDialog();
    return customer;
}
```

The Customer object that is returned should be saved in a local variable of type Customer. Then, if the returned value is not null, the customer should be added to the customer list. Since this changes the list, so you should update your list box which displays the customers.

2. Create a Windows forms application that includes a Customer class. This class is identical to the Customer base class from the previous question. (You can just copy the code for this class.) You need to modify this class so that it implements the ICloneable interface. You can also remove virtual keyword from the method GetDisplayText() since Customer class is no longer a base class in this project. (It is fine if you keep it.) This project uses List<T> that contains clones of a pre-defined Customer object and displays the cloned customers in a list box as shown below.



- a. In your form load event handler, you will create a Customer object, stores it in a variable named **customer**, and displays the **customer** in the label at the top of the form.
- b. Modify the Customer class, so it implements the ICloneable interface.
- c. In your event handler for Clone button, Click event, you should first validate the data. Then create a List<> object that contains the required number of clones of the Customer object. Finally, it should display the cloned customers in the list box.

Zip your two projects and submit them in Eagle online for grading.