GUI Tier

Here is the front end code, which simply populates a repeater with all of the customer records:

```
  <asp:Repeater ID="rptCustomers" runat="server"
        DataKeyField="CustomerID" >
<HeaderTemplate>

                    <tr>
                      <th>
                          ID
                      </th>
                      <th>
                          Name
                      </th>

                      <th>
                          Edit
                      </th>
                      <th>
                          Delete
                      </th>
                    </tr>

</HeaderTemplate>
 <ItemTemplate>

                <tr>
                <td>
                    <%#Eval("CustomerID")%>
                </td>
                <td>
                    <%#Eval("Name")%>
                </td>

                <td align="right">
                    <asp:Button ID="btnEdit" runat="server"
                         Text="Edit"
                         CommandArgument='
                         <%#Eval("ProductID")%>'
                         CommandName="edit"/>
                </td>

                <td align="right">
```

```
                        <asp:Button ID="btnDelete" runat="server"
                             Text="Delete"
                             CommandArgument='
                             <%#Eval("ProductID")%>'
                             CommandName="delete"/>
                    </td>

                </tr>

        </ItemTemplate>

    </asp:Repeater>
```

This is a simple repeater control which gets populated with a list of `Customer` objects from the database, using the following code in the `CustomerList.aspx.cs` file:

```
private void FillCustomers()
        {
            CustomerCollection list=new CustomerCollection();
            rptCustomers.DataSource = list.FindAll();
            rptCustomers.DataBind();
        }
```

The `CustomerCollection` class, which is defined in the next section, simply returns a collection of `Customer` objects. So the GUI tier is completely independent of the Data tier, and talks to the BL tier via a one-way reference (we have added a reference to the BL in the GUI tier, and not the other way round). So our system is loosely-coupled.

We can bind the `Customer` object properties in the ASPX using a declarative syntax, and if we need to edit a particular customer, we just need to directly use the *Customer* object's properties in the `Editcustomer` form, as in:

```
txtCustomerEmail.text = customer.Email;
```

When this property is called, the `Load()` method defined in the property will check if the `Customer` object is fully loaded or not; if not, it will load all of the properties. So this is called **load on demand**— the core principle of the lazy loading design pattern.