```
            //log error
               throw;
               }
         }
         #endregion

         #endregion Public Methods

      }//end class
   }//end namespace
```

Let us now analyze this business class. First of all we wrap the class attributes in the `CustomerDTO` class and aggregate it as a private member in this `Customer business` class as:

```
private CustomerDTO _customerDTO;
```

We then create class properties by using DTO fields in the get or set methods as:

```
public System.String  Password
      {
         get
          {
           Load();
            return  _customerDTO.Password;
          }
        set { _customerDTO.Password  = value;}
      }
```

In the `get` properties, note the `Load()` method being called. This is the implementation of the Lazy Loading design pattern. When the class object is first loaded from the database, we fill only the most needed properties such as `ID`, `Name` and so on. For all other properties, we load them on demand. This `Load` method in the properties will be called whenever the consumer of this business class tries to access that particular property value:

```
private void Load()
      {
        try
         {
            if( _customerDTO.loadStatus == LoadStatus.Ghost)
            {
              _customerDTO=CustomerDAL.LoadCustomer( _customerDTO.ID);
              _customerDTO.loadStatus = LoadStatus.Loaded ;
            }
         }

      }
```

Note that in this `Load` method above, we are first checking if the load status is `Ghost` (partially loaded) and if yes, then we are loading the customer DTO from the DAL and then changing the `Load Status` to `Loaded` (because now all of the properties would need to be loaded). This will make sure that we don't load the properties' values again when some other property value is accessed. In the DAL method, we check for the load status value, and then load partially or fully, based on the status value that is passed. Here is the code from the `Customer DAL` class (for clear understanding, I have removed the code unrelated to the current topic of discussion):

```
public static Collection<CustomerDTO> GetAllCustomers(LoadStatus
loadStatus …)
      {
       try
         {
       string strCommandText = "GetAllCustomers";
         Collection<CustomerDTO> list;
          using(SqlConnection cn = new SqlConnection(SQLHelper.
                                        GetConnectionString()))
           {
               SqlCommand cmd = new SqlCommand(strCommandText, cn);
               cmd.CommandType = CommandType.StoredProcedure;

         SqlDataReader reader = null;
         list = new Collection<CustomerDTO>();

         cn.Open();
          reader = cmd.ExecuteReader();
         if(reader.HasRows)
           {
             while(reader.Read())
            {
              //Create collection and fill
              CustomerDTO c = new CustomerDTO();

                  c.ID = (System.Int32)reader["ID"];
                  c.Name = reader["Name"];

               if(loadStatus==LoadStatus.Loaded)
                {
                  c.Address = reader["Address"];
```