The class Store.cs is like a manager. It keeps lists of all types of Product that there are in the Store: Book, Movie, Music, Toy, StudentAccessory. Also here in this class there is a method named **CheckAvailability()**, which method checks if givven by name type product exists in the store or no. If there is not such a product the method throws **ProductNotExistsExeption**.

In the project we use Singleton Design Pattern. For that purpose we create class Logger.cs. The idea of a Singleton Pattern is to have only one available instance of a class. There for the constructor is set to private and there is a static method in the class, that either calls an existing instance or creates a new one for the executing program. In this class we have method **Log()** which if an exception is thrown make a record of the exception message in a text file – ErrorLog.txt. There is a class ErrorType.cs which holds an enum of error types(error, warning or info). To test the **CheckAvailability()**method, in the MainClass.cs we create a product type Book with givven name. After that we create a new list of books - storeBooks and the product of type Book calls the **CheckAvailability(),** if the givven name of the book is not found in the list storeBook the only one instance of the Logger class calls the **Log()** method and the messageof**ProductNotExistsExeption**is written in the text file.

We have two evens named:

1. **AlmostEmptyInventory**- fires when the quantity of any product is less than 5
2. **EmptyInventory**- fires when the quantity of a certain item is zero.

These events are included in SellOneQuantity method, so we check what is going on only when a customer requires to buy some product.

In case his buy request leads to change in quality, required in any of the two events, the appropriate one fires. The subscribing methods are:

* The delegate of EmptyInventory method subscribes to EmptyInventory Event.
* The delegate of AlmostEmptyInventory method subscribes to AlmostEmptyInventory Event.

**Description of the interface IDiscountable and its application for the Client class.**

The class Client contains some informmation about the clients – name and ID number as well as the shopping card (the current purchase amount) and the client “History”, i.e. previous purchases. The class Clients inherits the interface IDiscountable. The implementation of this interface is as follows. If the current purchase price is over 200 and previously the client has bought something and the last purchase was within the last 3 months, give 20 % discount over the current purchase. If this is not the case, then if the current purchase price is over 500 and the previous purchase amount is over 500, give 10 % discount over the current price. If the above conditions are not satisfied, then the client cannot get any discount.

The structure History, used inside the Client class contains the information for the previous client purchases. It has 2 properties – purchase and purchase date and time; The property purchase is of type Product and has properties - bought product, its price and quantity and the name of the item which was bought.

**Testing the functunality of IDiscountable & Client class:**

The test for the functionality of the above implemented items is – create an instance of a client. The client has previously bought one movie – “The last samourai” and this purchase was realized last January (within the last 3 months). Checking the discount – the client can get 60$ discount. Discount amount is shown in the Console.

string name = "Test name";

string clientID = "Test ID";

Product movie = new Movie(500, 1, "Last Emperor"); // the client bought the movie "Last Samourai", price = 500, quantity = 1

DateTime date = new DateTime(2013,01,07); //this movie was bought on the 7th of January

History hist = new History(movie, date); //create the history for the above

decimal shoppingCard = 300; //the present shoping card amount

Client clientTest = new Client(name, clientID, hist, shoppingCard); // instance of Client

decimal disc= clientTest.Discount(); //check if discount can be provided

Console.WriteLine("Discount {0:c}", disc); //show the discount

