### תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל הקודם:

* **Favorites**  
  Our application allows marking posts in user’s friends news feed as favorites. Those posts are displayed in additional tab in our application’s view. User can remove specific post from favorites later. In order to mark/unmark a post as a favorite user can use context menu (right-click around post area) or click on favorite’s sign – star (image) – that located on the left side of the post message.
* **Translation**  
  Our application allows instant translating of a specific post to desired language using one of the implemented translators. In this exercise we allowed selection of one of three languages – Hebrew, English, and Russian – but it could be extend to dynamic list supported by translators.  
  We have implemented real translator by integration with the Microsoft’s Bing translation service. Both translators and supported languages options are loaded dynamically to the applications menu (*Translation*). There are 2 additional translators implemented for demo purposes – Dummy (adds “translated” work to the text) and Base64 (translates text to base64).

### תבנית מס' 1 – Decorator

* סיבת הבחירה / שימוש בתבנית:

We implemented additional feature – option to display some user-defined information on the user picture box. We wanted to “add” this functionality to the existing windows control displaying images – **PictureBox**.

* אופן המימוש:

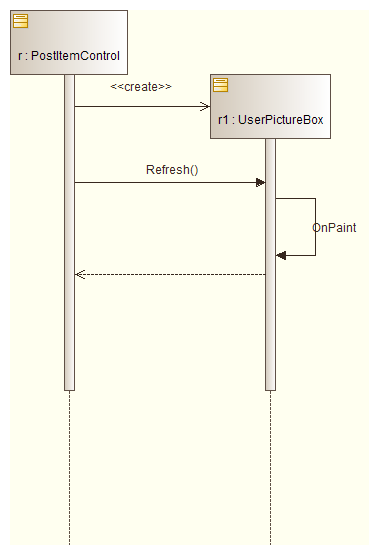
Since the **PictureBox** control doesn’t have an interface that would allow implementing it and be injected in the chain of painting, we had to implement the decoration by inheritance and add our custom additional logic on after the real (parent) picture box painted the image (override **OnPaint** protected method of **PictureBox**).

The implementation is located in the **FacebookWinApp** project, **Decorators\ UserpicDecorator\UserPictureBox.cs**.  
  
Roles:  
1. Client: **PostItemControl**

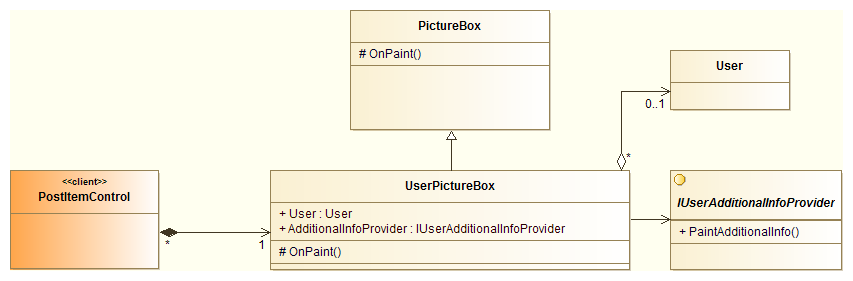
2. Component: **PictureBox**

3. Decorator: **UserPictureBox**

* **Sequence Diagram**



* **Class Diagram**



### תבנית מס' 2 – Strategy

* סיבת הבחירה / שימוש בתבנית:

In the new implemented feature (described in the previous part) we allow the user to decide at run-time (by choosing an appropriate item from the *Options->Additional User Info* menu) the type of data he would like to be displayed on the user’s picture. The options are: *None/Online Status/Favorites Count*.   
We wanted to use the same decorator we created. To achieve this goal we decided to pass to the decorator the logic of the info painting (strategy) dynamically.

* אופן המימוש:

We created an interface **IUserAdditionalInfoProvider** with one defined method:   
void PaintAdditionalInfo(Graphics i\_Graphics, User i\_User).

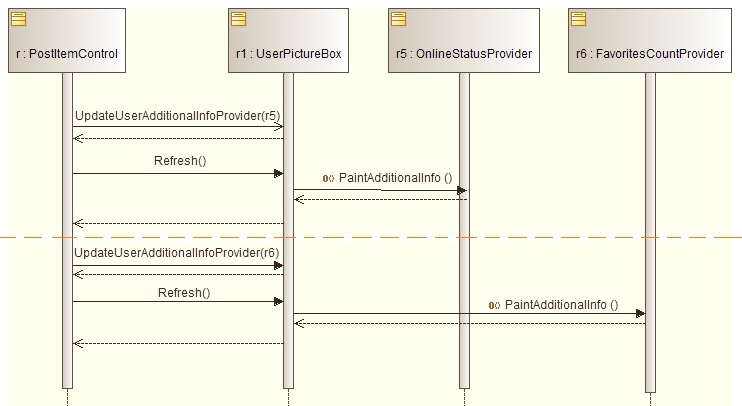
We have two implementation of this interface – painting online user info/favorites posts count using provided graphics object.  
Our decorator class has a “strategy” property of type **IUserAdditionalInfoProvider** which is set dynamically from **PostItemControl**. When the decorator should apply the decoration logic, it invokes the method in the strategy object to paint the desired information.  
  
The implementation is in the **FacebookWinApp** project, **Decorators\ UserpicDecorator\Strategy** folder.  
  
Roles:  
1. Client: **UserPictureBox**

2. Strategy: **IUserAdditionalInfoProvider**

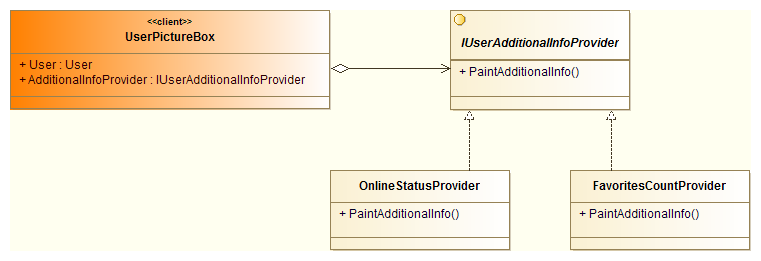
3. Concrete strategy 1: **OnlineStatusProvider**

4. Concrete strategy 2: **FavoritesCountProvider**

* **Sequence Diagram**



* **Class Diagram**



### תבנית מס' 3 – [שם התבנית]

* סיבת הבחירה / שימוש בתבנית:

We have a class – **FavoritesManager** – that is responsible for maintaining favorite posts. The post can be either added to or removed from the favorites. Some other collaborators might be interested in getting a notification when such event occurs. For example, our main form should update the title of the favorites posts tab when such event happens as we displayed the total number of favorite posts.

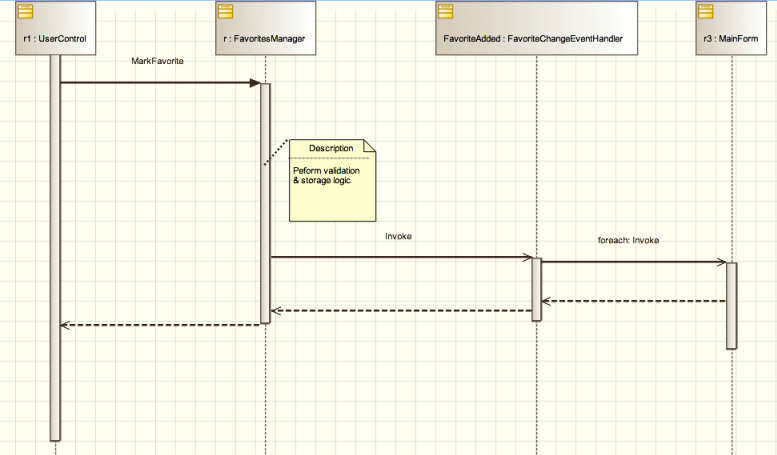
* אופן המימוש:

We used the mechanism provided by Microsoft.NET framework – delegates – in order to implement the solution. We defined new delegate type **FavoriteChangeEventHandler**. The **FavoritesManager** class defines two events of the **FavoriteChangeEventHandler** type – **FavoriteAdded** and **FavoriteRemoved**.  
**MainForm** has an instance of the **FavoritesManager** class and subscribes on those events.  
The implementation is in the **FacebookApp.Model** project, **Favorites\ FavoritesManager.cs**.  
  
Roles:

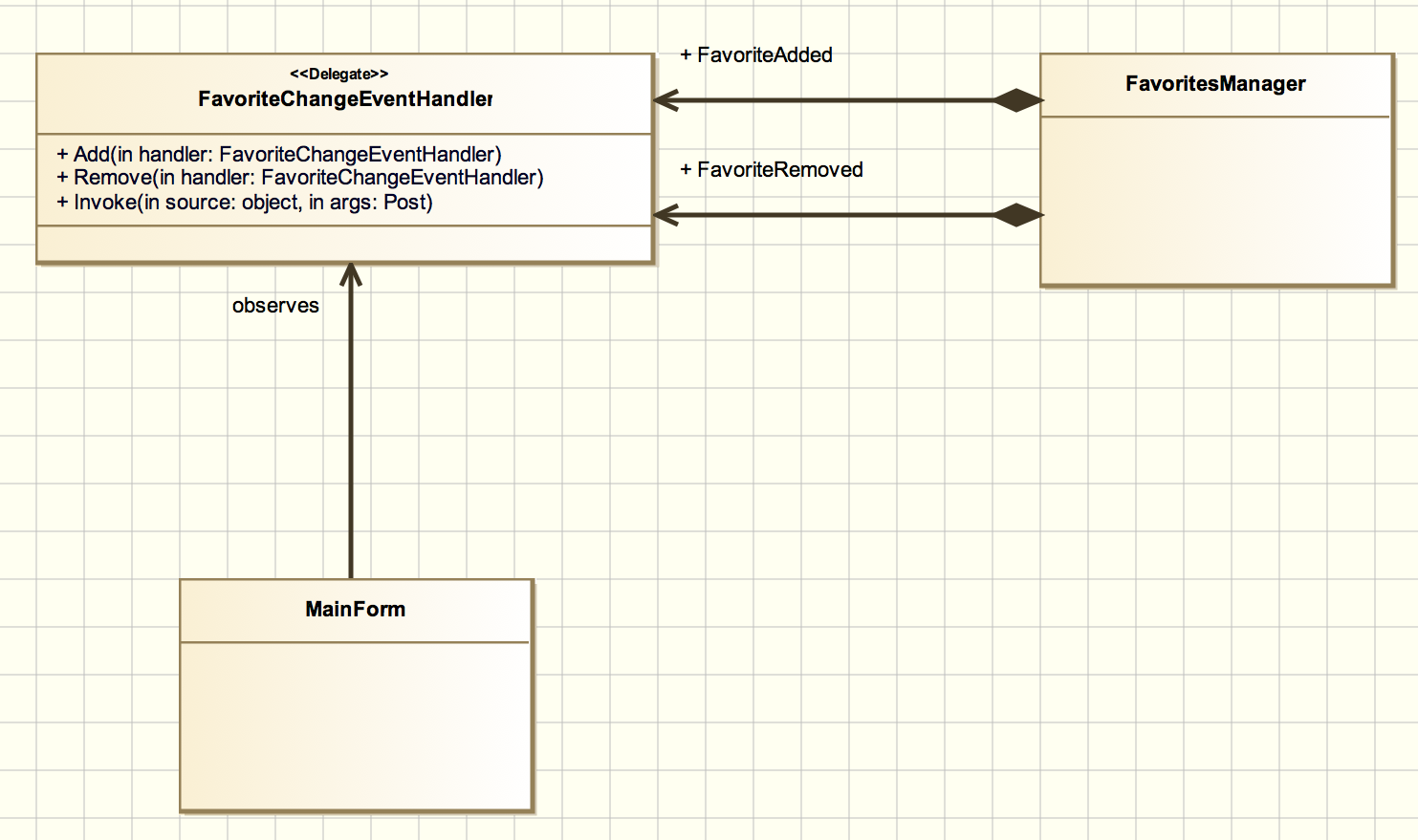
1. Observer: **MainForm**

2. Subject: **FavoriteChangeEventHandler**

* **Sequence Diagram**



* **Class Diagram**



### שימוש בתהליכונים (Threads)

We create our **PostItemControl** control dynamically on getting posts info from the Facebook server. As this operation could be time-consuming we want to do it in separate thread to prevent freezing of UI.   
In **MainFrom** we have this method which starts the loading posts and updating UI process in separate thread (using .NET Task class):   
 private void loadNewFeedAsync()

{

new Task(loadNewsFeed).Start();

}

The code is in **FacebookWinApp** project, **Forms\** **MainForm.cs**.  
  
  
In order to allow updating of UI from non-UI (main) thread we defined helper functions that perform appropriate update according to control’s **InvokeRequired** property value.  
The code is in the **FacebookWinApp** project, **Utils\Utils.cs**.

### שימוש בקישור לנתונים (Data Binding)

We provide an option to display detailed user information in separate form. We defined a data source from the User class and used the use binding source control it on the form (user control).

The code is in the **FacebookWinApp** project.  
Datasource: **DataSources\ FacebookWrapper.ObjectModel.User.datasource**

Usage: **UserControls\ UserViewControl.cs**