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

* **Random friend liked pages:** From a list of friends, who are also using this application, pick randomly one friend and see 10 pages that this friend likes. The pages are shown as a grid of pictures 2 \* 5.
* **Find friends by city:** This feature helps the user to see in his list of a friends, who are also using this application, only the friends in a specific location (provided by the user).

### תבנית מס' 1 – Strategy

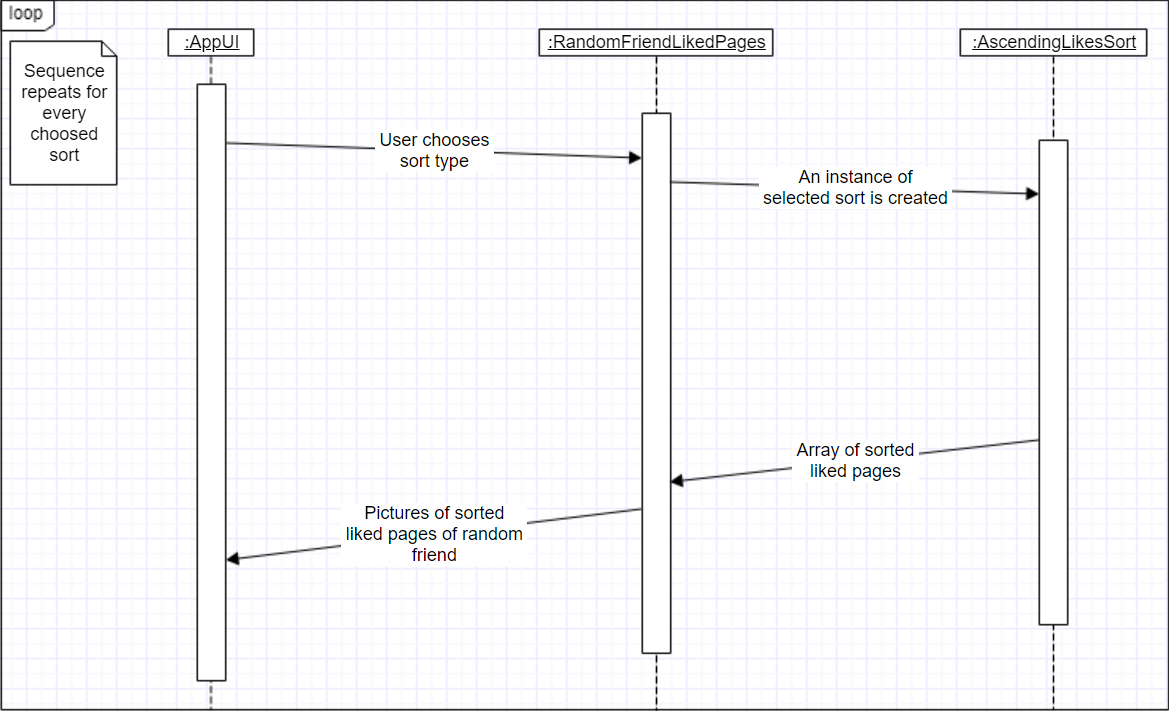
* סיבת הבחירה / שימוש בתבנית:

We’ve decided to implement Strategy Pattern for RandomFriendLikedPages feature.  
At this point the user sees first ten liked pages of random friend. We have added an option to sort these pages dynamically (currently in ascending or descending order). In the future amount of sorts can be increased by just implementing an interface and adding a string with sort name to comboBox at UI.

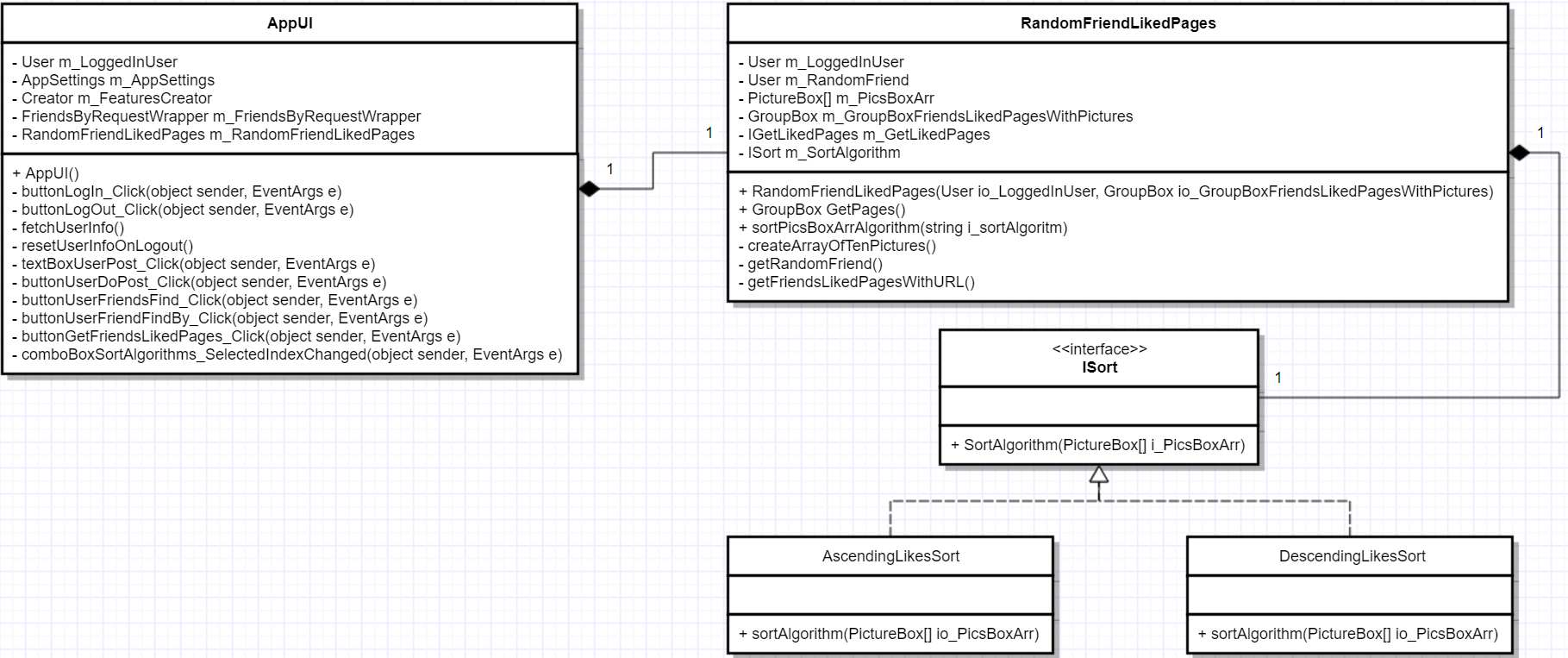
* אופן המימוש:

AscendingLikesSort and DescendingLikesSort are implementing an interface ISort. At runtime user chooses sort type (at AppUI), this sort type is given to RandomFriendLikedPages feature that dynamically creates an instance of needed sort class. **Relevant classes:** ISort, AscendingLikesSort, DescendingLikesSort, RandomFriendLikedPages, AppUI.

* Sequence Diagram



* Class Diagram

  
**Context:** RandomFriendLikedPages. **IStrategy:** ISort.  
**Concrete strategy:** AscendingLikesSort, DescendingLikesSort.

### תבנית מס' 2 – Observer

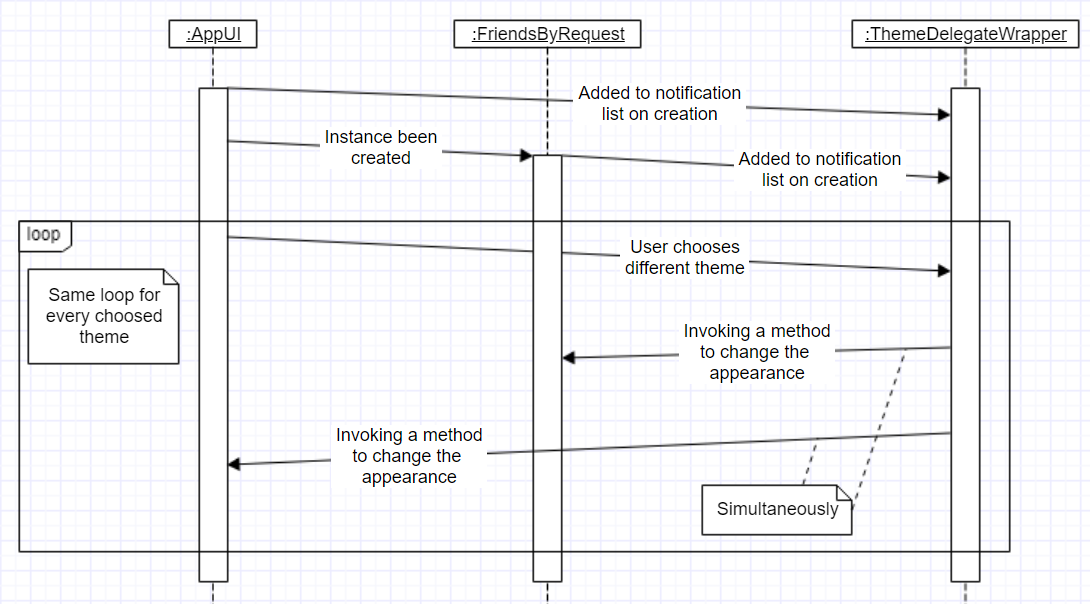
* סיבת הבחירה / שימוש בתבנית:

In this exercise we’ve decided to add an option to switch theme (light/dark/default) of the application. Pattern Observer is the best solution for this case: every current feature (and possible new features) just being notified by the delegate and it doesn’t matter how many features do we have. Also, every feature can implement unique solution for the theme switcher and easly add/remove himself from the notification list. One more point for this pattern is that we don’t have tightly coupled classes, that gives us reusability.

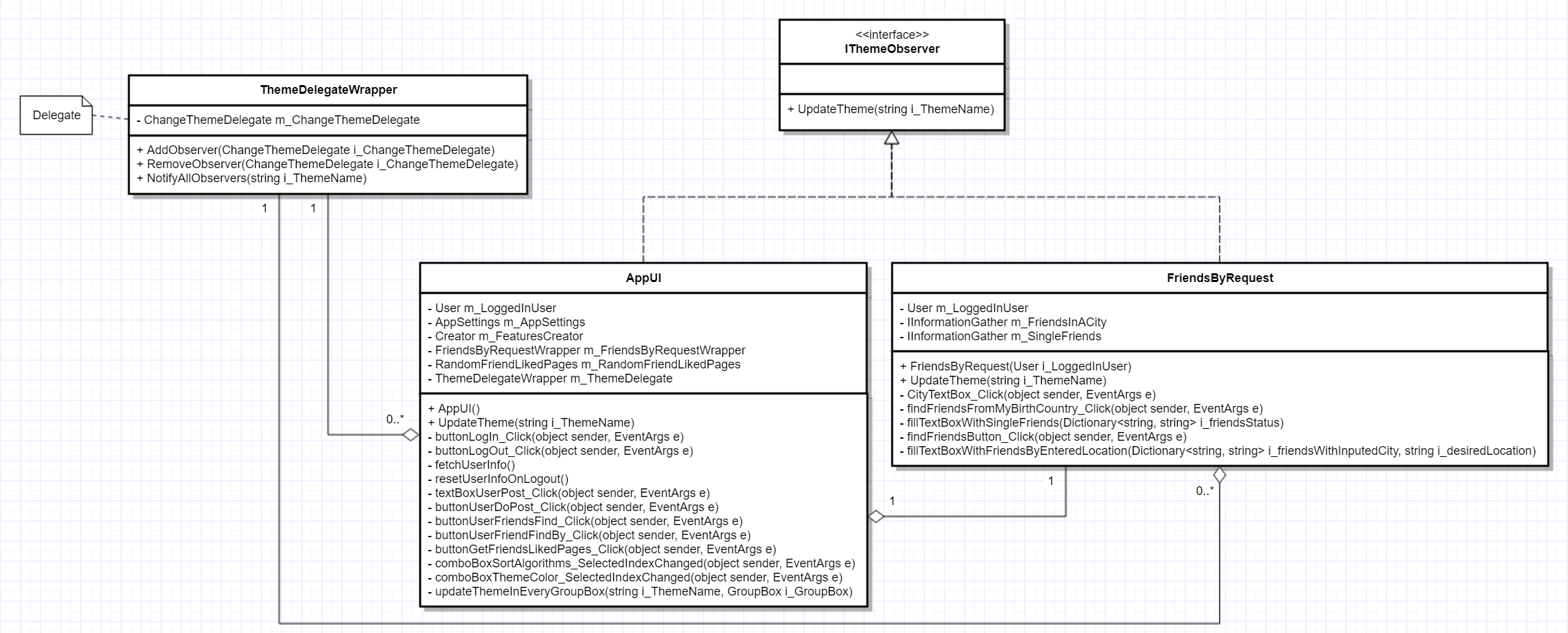
* אופן המימוש:

Every form (AppUI and FriendsByRequest) has a unique implementation of IThemeObserver interface. ThemeDelegateWrapper holds a delegate as a property (we’ve decided to create a wrapper because this class has also a couple of service methods). On creation every form been added to notification list and invocation been made after user chooses theme style. **Relevant classes:** ThemeDelegateWrapper, IThemeObserver, AppUI, FriendsByRequest.

* Sequence Diagram



* Class Diagram

  
  
**IObserver:** IThemeObserver.  
**Observers:** AppUI, FriendsByRequest.  
**Observable:** ThemeDelegateWrapper.

### תבנית מס' 3 – [שם התבנית]

* סיבת הבחירה / שימוש בתבנית:

[תיאור הסיבה / הצורך בשימוש בתבנית במערכת שלכם]

* אופן המימוש:

[תיאור המימוש והיכן ניתן למצוא אותו בקוד]

* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]

* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם