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

* Internet checker  
  בדיקה אלו אנשים מתעניינים בפוסטים (סטטוסים וכו') שהיוזר מפרסם בפרק זמן ספציפי.   
  התעניינות נבחנת ע"י בדיקה מי מגיב (תגובה/LIKE) לפוסטים.  
  הפיצ'ר נמצא בכרטיסיית Interest Check באפליקציה.
* פילטור חברים ומיילים ע"י פילטרים מקונפגים מראש.  
  ע"י יצירת פילטר מתקדם (קריטריונים: גיל, מין ושייכות לקבוצת חברים)   
  ניתן לראות רשימה מוקטנת של חברים שמקיימים תנאי הפילטר  
  וכן לראות רק שרשורי הודעות ששותף בהן חבר שמקיים תנאי פילטר.  
  את הפילטר מוסיפים בכרטיסיית filters באפלקיציה.  
  את רשימת החברים המצומצמת (והמורחבת) ניתן לראות בכרטיסיית friends באפליקציה.  
  את שרשורי ההודעות המצומצמת (והמורחבת) ניתן לראות בכרטיסיית messages באפליקציה.

### תבנית מס' 1 – Observer

* סיבת הבחירה / שימוש בתבנית:

במערכת ישנם מספר אובייקטים שצריכים להתעדכן ברגע שעושים fetch מהפייסבוק.

על הfetch אחראי אובייקט אחד שכל שאר האובייקטים צריכים להתעדכן כאשר הוא מתעדכן.

* אופן המימוש:

בחרנו לממש בעזרת subject, observer ו-delegate (במקום interface) כפי שנלמד בכיתה.  
המחלקה FacebookApplication.FacebookFetchObject הינה מהווה הsubject של התבנית.  
מימשנו את התבנית 3 פעמים, לשלוש fetch שונים, שכן נדרשו 3 אפיונים שונים של עידכוני אובייקט fetch (חברים, רשימת חברים, הודעות).  
במסמך נתרכז בfetch friends שכן זהו המורכב ביותר בעל שני observer.   
המחלקות: FriendsFiltersManager, FriendsFetcher

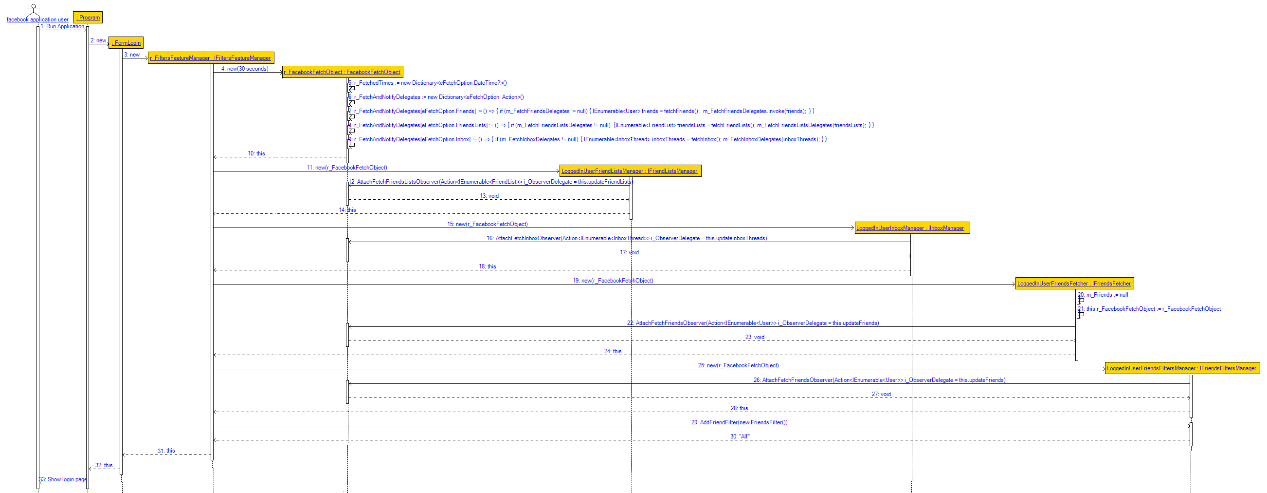
מהוות כל אחת את observer של התבנית.

הdelegate הגנרי המהווה את הIObserver של התבנית הינו:

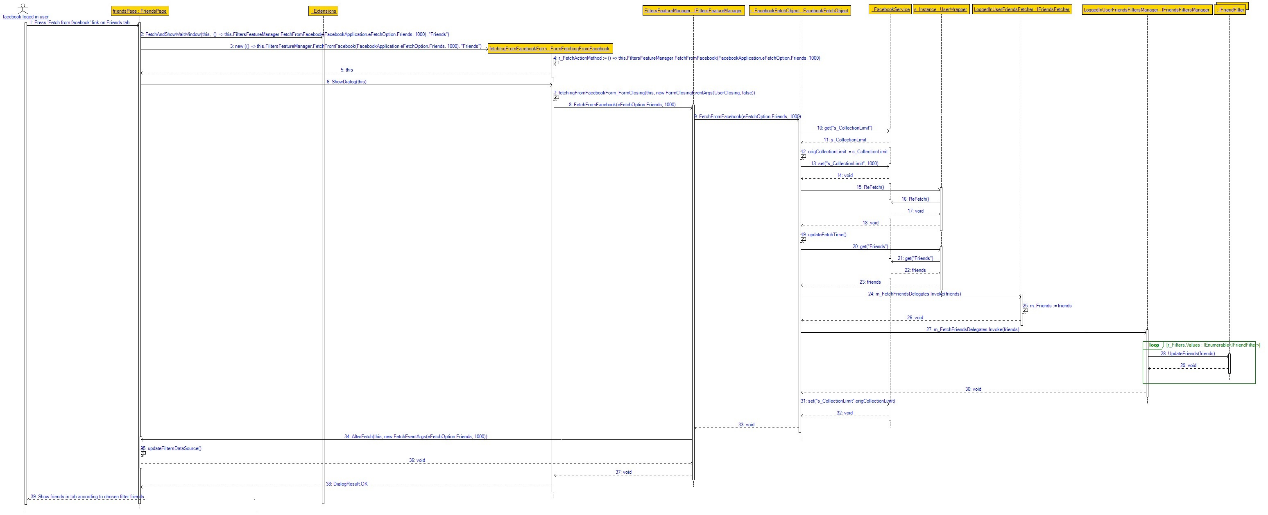
Action<IEnumerable<User>>

והevent שלו בsubject הינו: m\_FetchFriendsDelegates.

* Sequence Diagram  
  Attach Observers

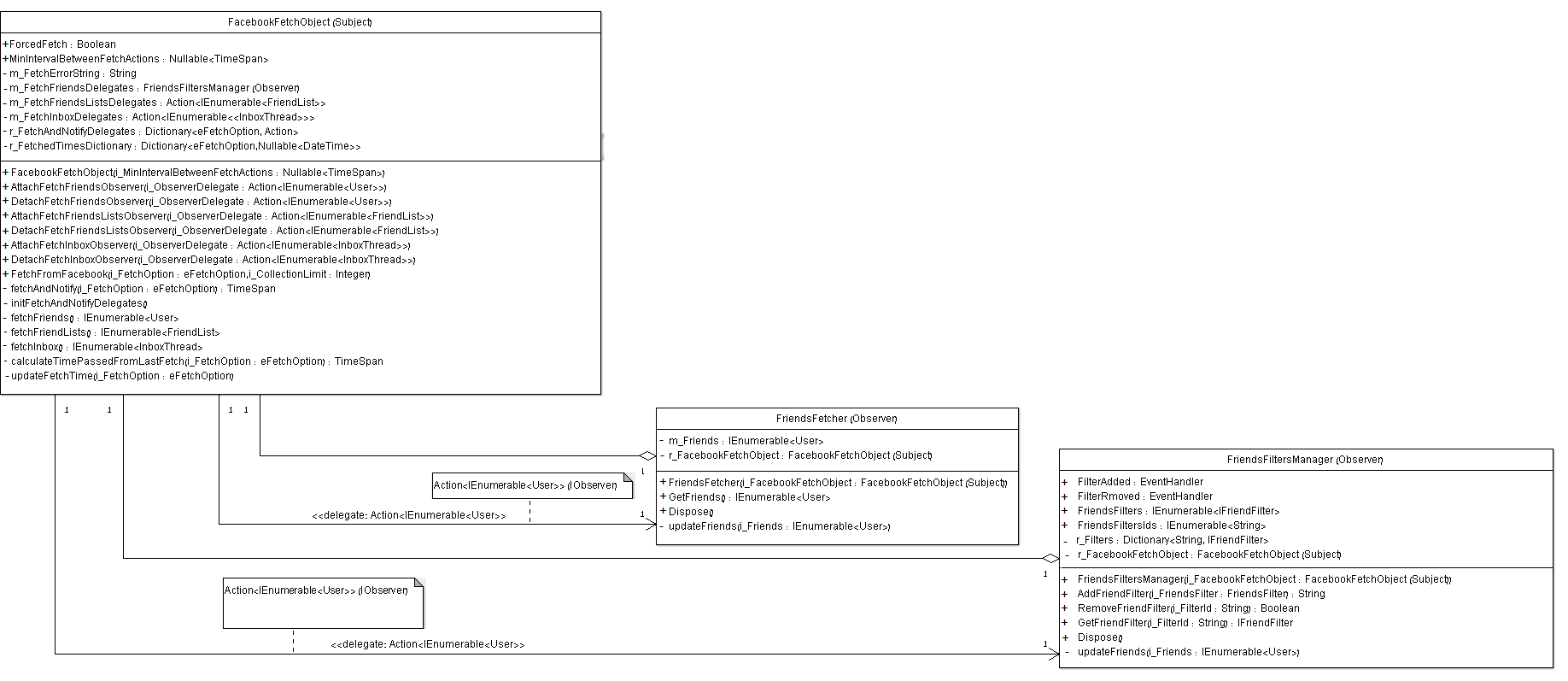


Update subject and notify observers  
 |  
 V

  
יש קבצים מצורפים:

ObserverSequenceDiagram.jpg, AttachObserversSequenceDiagram.jpg

* Class Diagram



יש קובץ מצורף: ObserverClassDiagram.jpg

### תבנית מס' 2 – Iterator

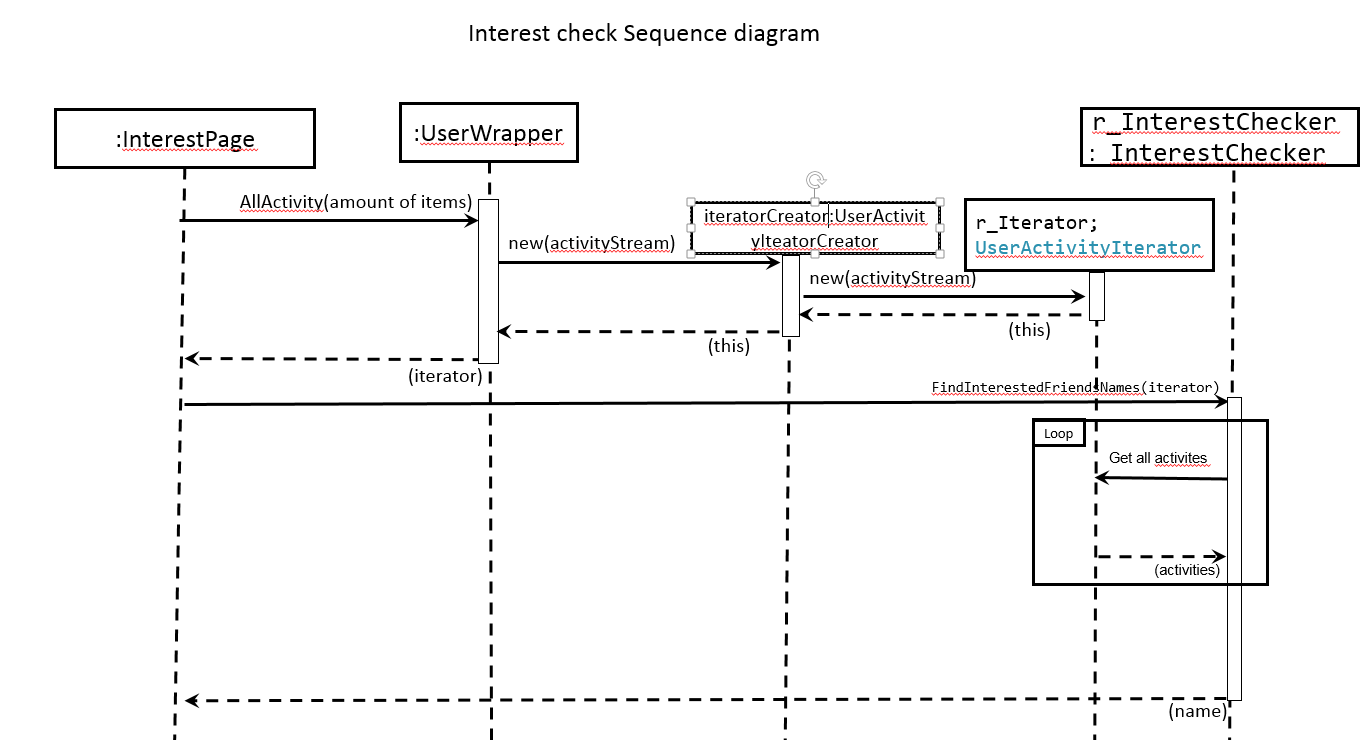
* סיבת הבחירה / שימוש בתבנית:

היה צורך לעבור על כל מקורות התוכן של המשתמש ולאחד אותם למקור יחיד של PostedItem, באופן שביעילות יסנן את כל הפוסטים שלא זכו לתגובה או ללייק, באופן שמנותק ממהמבנה של המחלקה User.

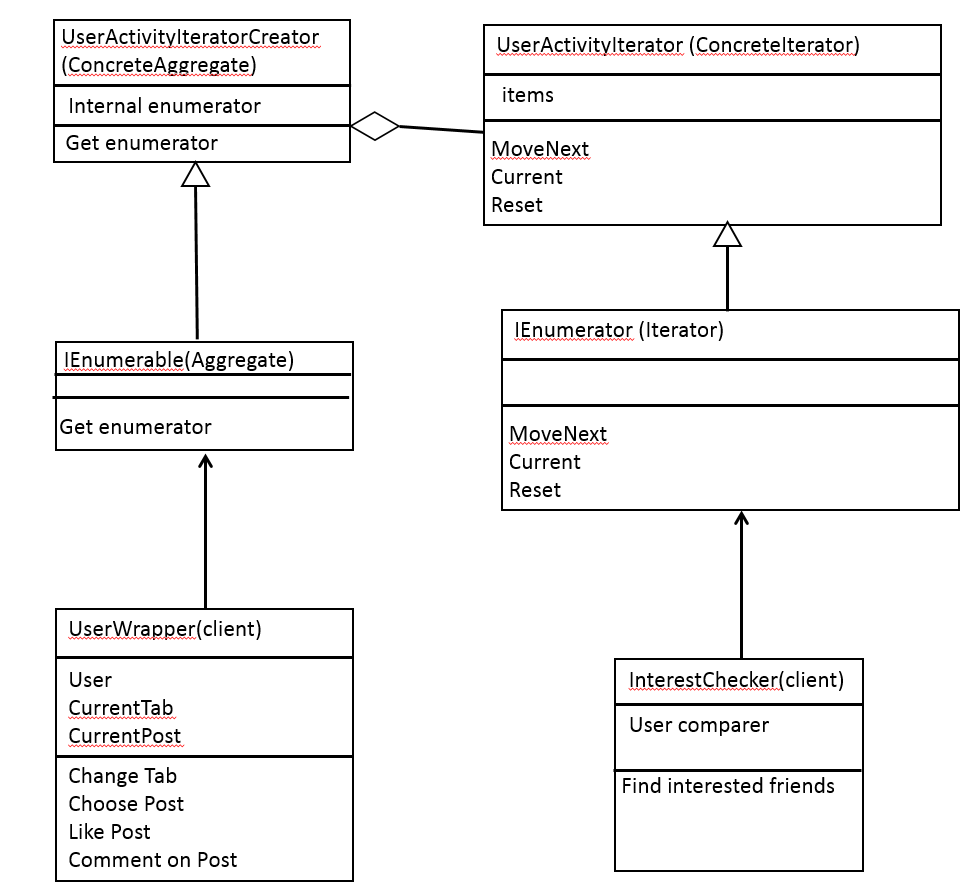
* אופן המימוש:

המימוש נמצא בקובץ UserActivityIterator.cs. המחלקה UserActivityIterator נוצרת כשהיא מקבלת מספר מקורות מידע מסוג IEnumerable<PostedItem>, ויוצרת בתוכה איטרטור פנימי מסוג UserActivityInternalIterator, אותו היא מחזירה כשמבקשים ממנה איטרטור.   
האיטרטור הפנימי שומר איטרטור מעל כל מקורות המידע שלו, ועובר עליהם אחד-אחד. בכל פעם הוא שומר איטרטור מעל מקור המידע הנוכחי ועובר עליו עד ש-MoveNext() מחזיר ערך שלילי, ואז הוא מחליף ועובר למקור המידע הבא.

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – Strategy

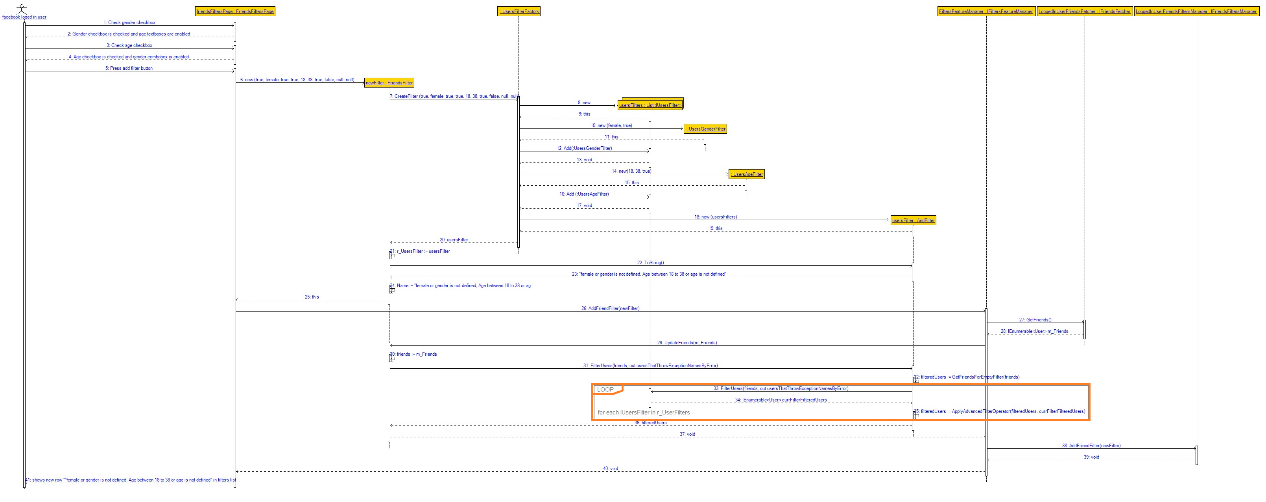
* סיבת הבחירה / שימוש בתבנית:

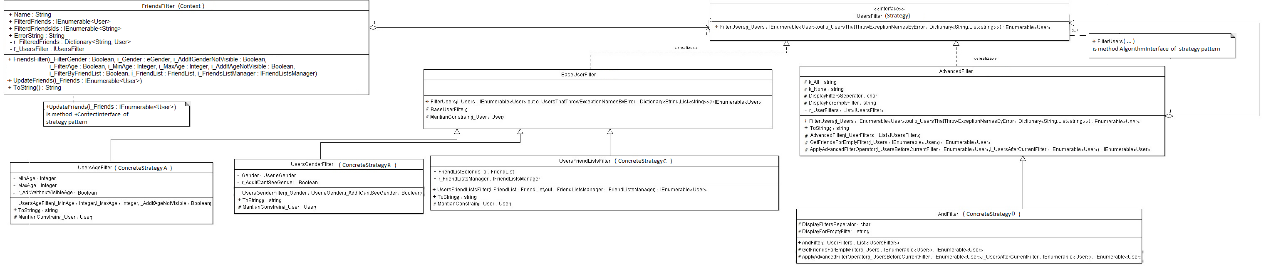
בפיצר של הפילטרים מדובר במשפחה של אלגוריתמים לפילטור   
(בין גילאים, ממין מסויים, שייך לרשימת חברים, משולב)  
נרצה שהלוגיקה של הפילטור לא תיהיה דרך הclient בהפעלת הפילטר, על כן בחרנו בתבנית זו.

* אופן המימוש:

הממשק IUsersFilter הינו ה Strategyשל התבנית.  
המחלקות UsersAgeFilter , UsersFriendListsFilter , UsersGenderFilter ו- AndFilter הינם הConcreteStrategyA, ConcreteStrategyB, ConcreteStrategyC, ConcreteStrategyD של התבנית.  
הFriendsFilter מחזיק IUsersFilter ומהווה ה-Context של התבנית.  
לגבי המתודות:  
ContextInterface() של הContext זוהי המתודה

* Sequence Diagram

  
מצורף קובץ: StrategySequenceDiagram.jpg  
הערה: בחרנו להראות sequence diagram בהוספת פילטר חברים, שכן זוהי המורכבת יותר.  
הקריאה למתודה של הStrategy הינה בלולאה.  
חוץ מהsquence diagram המופיע מעל גם זה של הObserver טוב גם כאן שם בלולאה הירוקה הקריאה למתודה של הStrategy.

* Class Diagram   
    
  מצורף קובץ: StrategyClassDiagram.jpg