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

* Most In Common: פיצר שמאפשר למשתמש לבדוק עם מי מהחברים שלו יש לו הכי הרבה במשותף (דפים אהובים, קבוצות, וכו).
* Facebook Match: פיצר שמאפשר למשתמש למצוא התאמה זוגית מרשימת החברים שלו (מוצא רק את החברים שפנויים לקשר), תוך הפעלת פילטרים שיעזרו לו למצוא את הזיווג המתאים.

### תבנית מס' 1 – [Visitor]

* סיבת הבחירה / שימוש בתבנית:

ישנם כמה סוגי רכיבים (TextBox, Button) שנרצה לבצע עבורם Enabled.

במקום ש"נלמד" כל רכיב באופן קונקרטי כיצד לעשות Enabled,

נממש תבנית Visitor שידע לעשות את אותה פעולה, תוך היכרות של הרמה העליונה המשותפת  
(ללא היכרות ציקלית) בין הרכיבים(Control) ויבצע את אותן פעולות עבורנו.

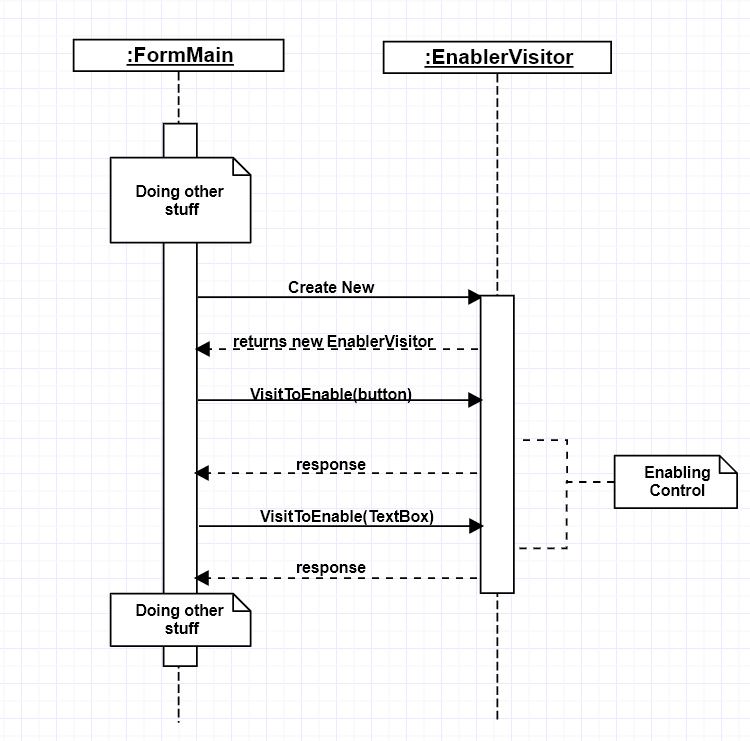
לכן אנו יוצרים EnablerVisitor ועושים שימוש בפונקציוליות שלו.   
כמו כן, אם בעתיד נרצה לעשות שימוש נוסף כזה עבור רכיבים ממשפחת Control נוכל לעשות זאת בקלות. בנוסף, אם בעתיד נרצה לממש Visitor שעושה פעולות אחרות, נוכל פשוט לממש את  
 .interface IControlVisitor

* אופן המימוש:   
  הרכיב FormMain מחזיק את EnablerVisitor, וכמובן רכיבי Button, TextBox.  
  בהתאם לשימוש בעבודה אסינכרונית, תחילה כאשר יופיע FormMain, כל הכפתורים בו יהיו Disabled ע"מ שלא יהיה ניתן להפעיל אותם לפני שכל הנתונים הובאו בבטחה. כמו כן גם הTextBox המאפשר כתיבת טקסט לצורך פרסום סטטוס.  
  במהלך ריצת התוכנית, כאשר הנתונים יובאו בהתאם לכל רכיב, נרצה שרק אז הכתפורים  
  והTextBox יהיו Enabled, כאן אנו עושים שימוש בEnablerVisitor שיבצע enable.

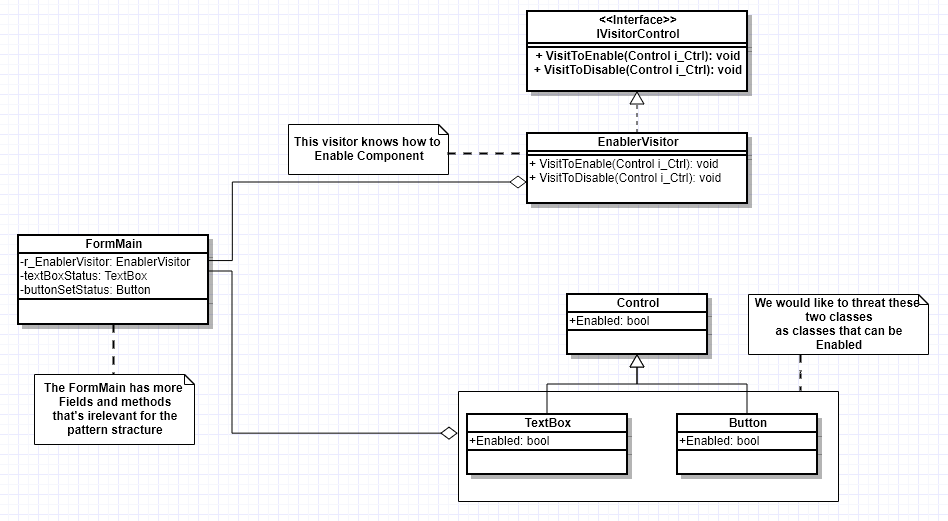
ניתן למצוא בקוד:  
בפרוייקט FacebookApp בתוך תיקיה ControlVisitor ישנן המחלקות הבאות:

IControlVisitor – Interface עבור הVisitor הקונקרטי.  
EnablerVisitor – מחלקה שיודעת לקבל רכיב Contol ולעשות עבורו Enable.

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – [Observer]

* סיבת הבחירה / שימוש בתבנית:

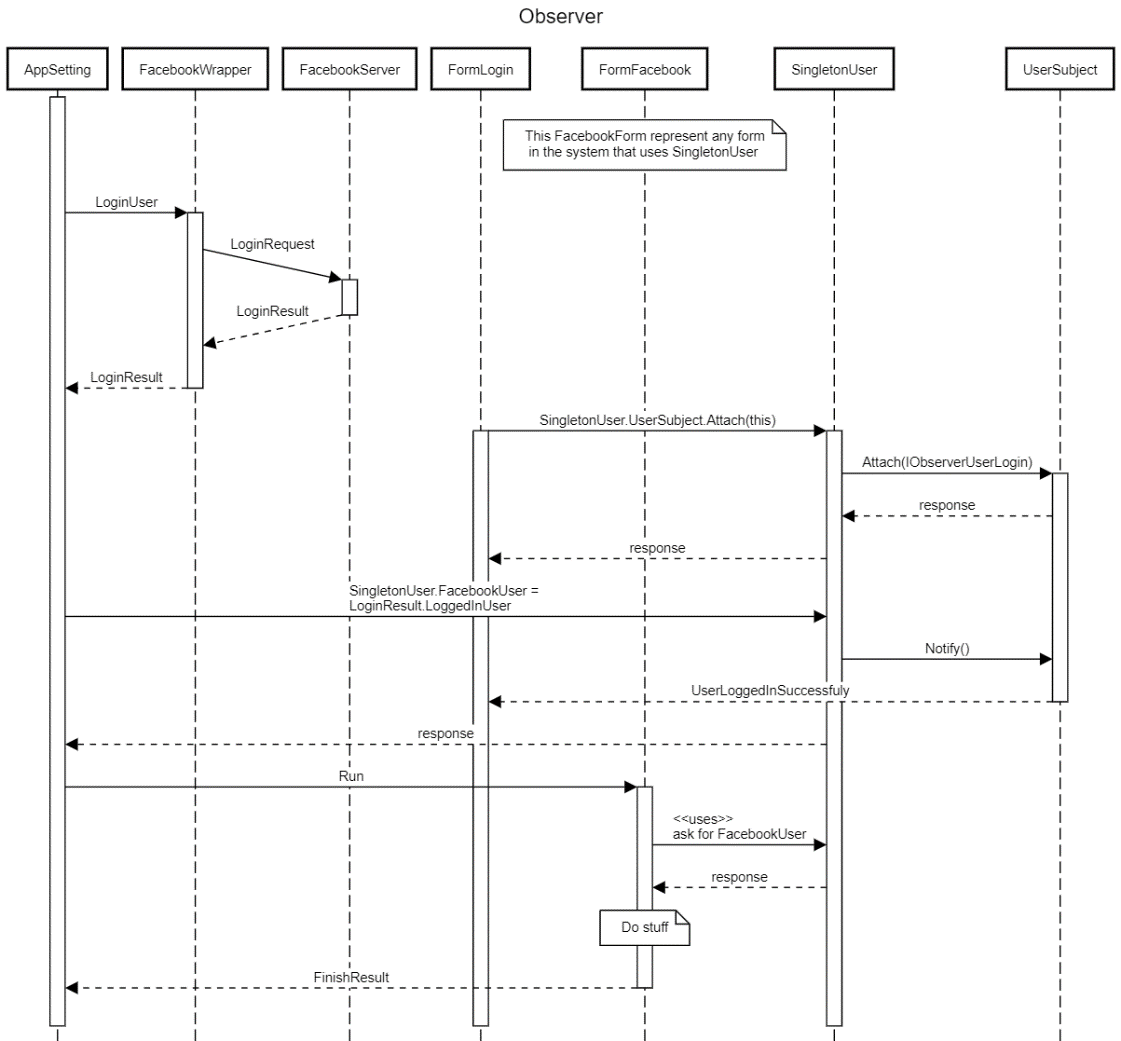
כאשר ישנו חיבור מוצלח בין User לשרתי הפייסבוק בשכבה הלוגית, נרצה להודיע על כך לרכיבים בשכבת המשתמש (FormLogin) ע"מ שיוכלו להתקדם ולאפשר למשתמש להשתמש בשירותי הפייסבוק. בחרנו לממש Pattern Observer כדי שנוכל להירשם כעת ובעתיד לאירוע זה כמאזינים, ללא קשר ציקלי בין ישויות, כמתאפשר ע"י loose coupling בעזרת שימוש בdelegate.   
בנוסף, אם בעתיד נרצה להירשם ממקומות שונים נוספים, המערכת הנוכחית הינה אקסטנסיבילית, כך שלא נצטרך לשנות קוד קיים אלא רק להירשם כמאזינים לאירוע של UserSubject.

* אופן המימוש:

ישנו interface IObserverUserLogin המהווה observer שצריך לממש ולהרחיב אותו ע"י מימוש הinterface, ישנה מתודה Update המתבצעת כאשר האירוע מתרחש. הFormLogin הוא למעשה הconcreteObserver הרשום לאירוע של המחלקה UserSubject וכאשר האירוע מתרחש הוא מטפל בערך של משתנה בוליאני UserLoggedIn בהתאם. המחלקה UserSubject מייצגת את המודיע, ויש לה מתודות שיודעות לקבל IObserverUserLogin ולרשום\ להסיר כמודיע, וכמובן גם לעדכן כשאירוע התרחש. המחלקה SingletonUser היא זו שתבצע תקשורת מול שרתי הפייסבוק, ובעת אירוע מוצלח, תפנה ל UserSubjectותבקש ממנו לעדכן את המאזינים.

ניתן למצוא בקוד:בפרוייקט Logic ישנה תיקיה בשם Observer המכילה את המחלקות: IObserverUserLogin, UserSubject, SingletonUser. ובפרוייקט FacebookApp את FormLogin.

* Sequence Diagram



FacebookServer:

FormLogin:

FormFacebook:

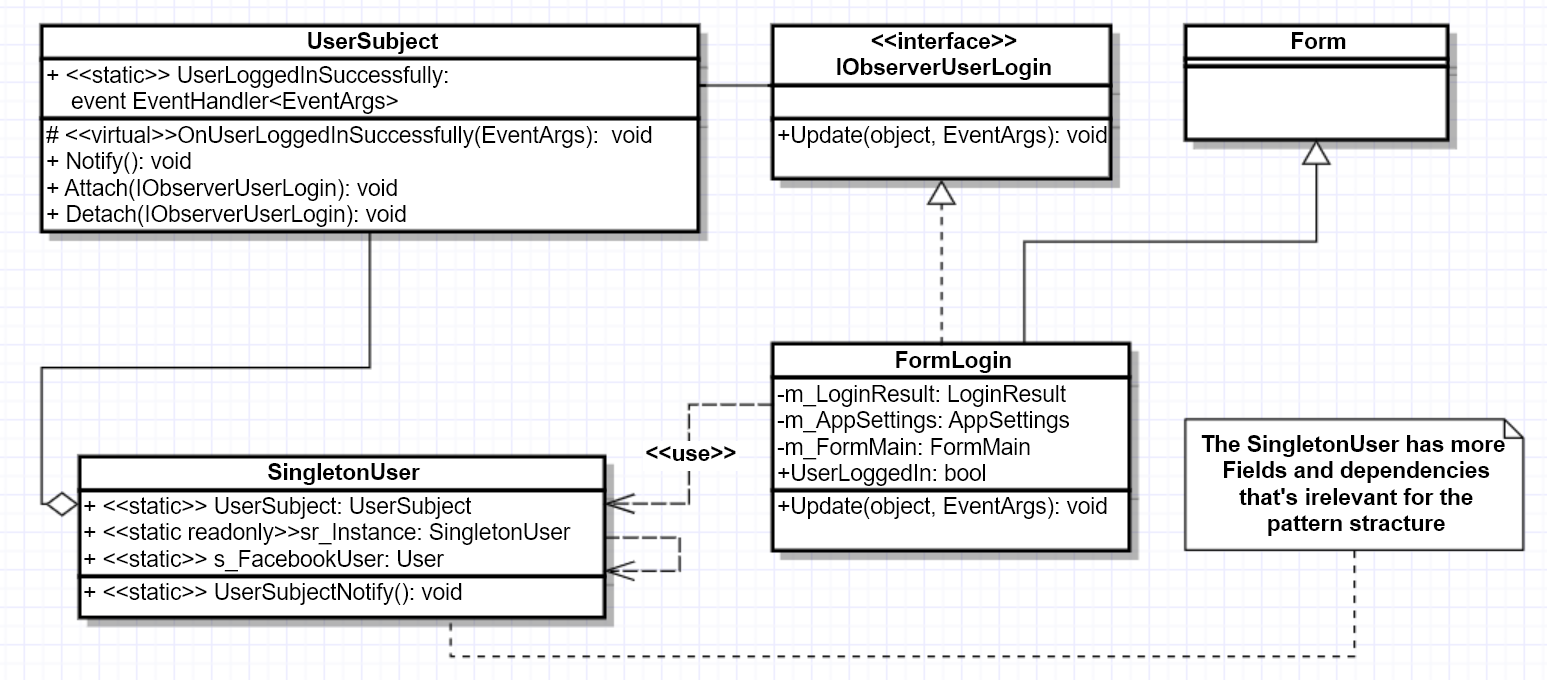
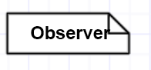
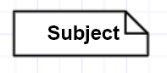
UserSubject:

SingletonUser:

FacebookWrapper:

AppSetting:

* Class Diagram



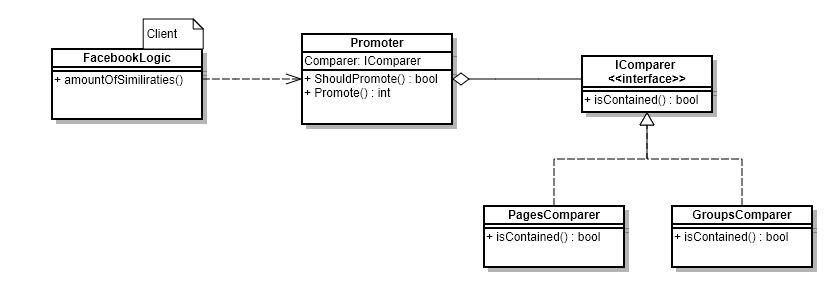
**תבנית מס' 3 – [Strategy]**

* **סיבת הבחירה / שימוש בתבנית:**

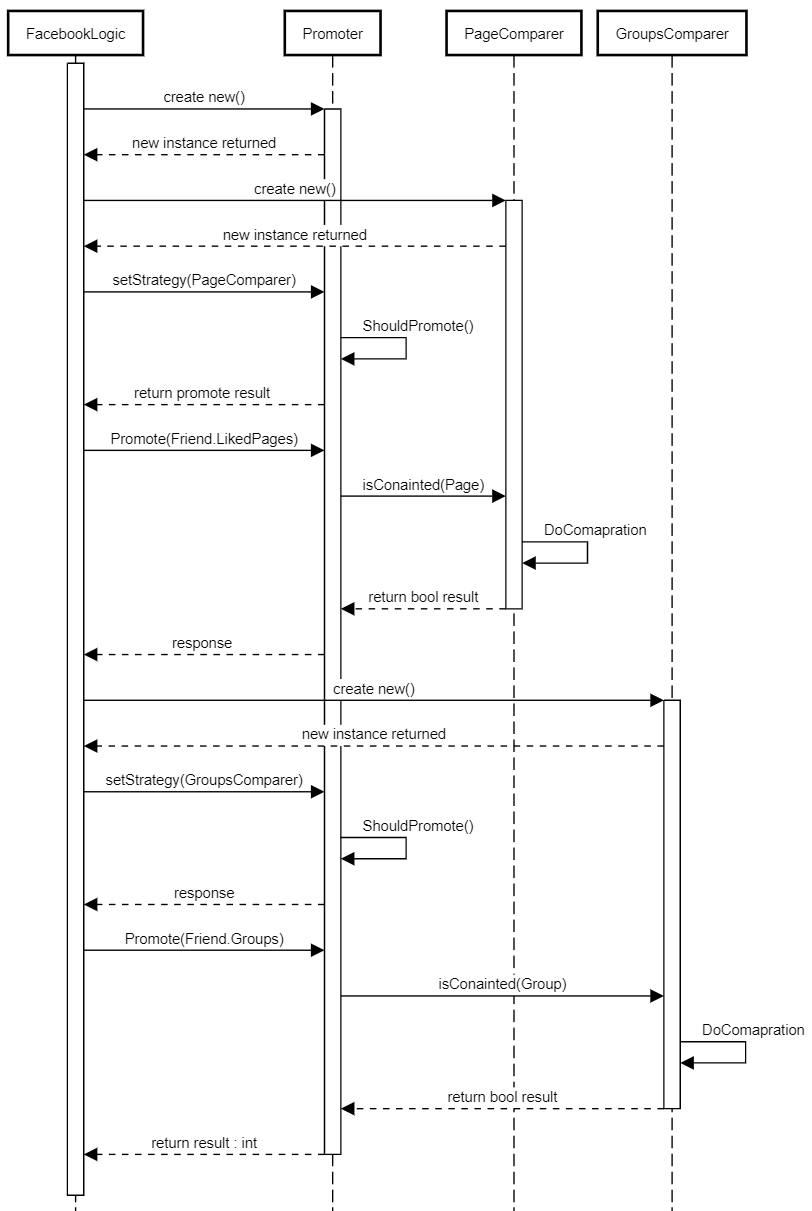
הסיבה בגללה בחרנו לממש Strategy Pattern במערכת שלנו נבעה מהצורך לדרג את החברים בפיצ'ר MostInCommon. ישנם כמה סוגי דירוגים(לפי דפים משותפים, עמודים שסומנו לייק וכו'). מאחר והדירוג נעשה באותו אופן ורק אסטרטגיית המיון השתנה בהתאם לבחירת המשתמש, בחרנו להשתמש ב-Strategy Pattern כך שבכל פעם שנרצה להשתמש באסטרטגיית דירוג נבחר רק באסטרטגיה הרצויה ולפי בחירת המשתמש נדרג את החברים ללא שכפול קוד. בנוסף, פטרן זה הופך את המערכת לאקסטנסיבילית וקלה יותר לתחזוקה. אם בעתיד נרצה להוסיף עוד פיצ'רים שמממשים דירוג – נוכל לעשות זאת בצורה קלה ביותר – הזרקת אסטרטגיית מימוש ללא הצורך בשינוי בקוד למעט הוספת המחלקה המתאימה.

* **ניתן למצוא בקוד:**  
  בפרויקט Logic ניתן לראות תיקיית מחלקות בשם Strategy המכילה את הרכיבים המשתתפים בתבנית. בנוסף ה - client הוא FacebookLogic.
* **אופן המימוש:**
  + **טכני:**
    - מחלקת ה “strategy”: *ICompare*
    - מחלקת ה “context”: *Promoter*
    - מחלקות “concrete”: *PagesComparer*, *GroupsComparer*
  + **כללי:**
    - לאחר שה - user מבקש לראות את חבריו הקרובים ב - “MostInCommon” פיצ'ר המתודה "buttonCheckFriends\_Click" ניגשת למתודה "AmountOfSimilarties" שנמצאת ב - client (מחלקת FacebookLogic). שם אנחנו מייצרים אובייקט של המחלקה Promoter שבתוכו ניתן לאתחל את ה instance לאחת המחלקות ה concrete לפי איזה סוג השוואה אנחנו רוצים. לאחר האתחול ניתן לעשות את ההשוואה ואז לשנות את ה instance לאחד אחר, מה שיוצר שימוש reusable וגם maintainable.

**Class Diagram:**



**Sequence Diagram:**



Promoter:

GroupsComparer:

PagesComparer:

FacebookLogic: