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

* TODO – חלון שמטרתו היא הצגה למשתמש את כל האירועים שהוא צריך להגיב אליהם בזמן הקרוב או שמחכים לו. לאחר החיבור יופיעו למשתמש כלל האירועים שהוא צריך להגיד אם הוא מגיע/לא אליהם.

בנוסף, יופיעו לו כלל הפוסטים בהם תוייג ועדיין לא הגיב, כדי להימנע מתקריות חברתיות.

* Feed מועדפים – מאפשר בחירת החברים המועדפים עליך ולראות את ה-posts שלהם בלבד. כך המשתמש מייצר לו feed המתאים לו בדיוק. המשתמש בוחר את חבריו אותם רוצה לראות ב-feed האלטרנטיבי והוא מקבל את כלל הפוסטים שלהם.

**תבנית מס' 1 – Observer**

* סיבת הבחירה / שימוש בתבנית:

לאחר המשתמש מבצע Login למערכת מתבצעת טעינה של נתונים אל המסך וכך גם לאחר שהמשתמש מתנתק אז מתבצע ניקוי של הנתונים. אם בעתיד נראה להוסיף/להוריד אנחנו צריכים בכל פעם להוסיף ולהוריד את המהלכים ממחלקת ה-Form המרכזית באופן פרטני וספציפי לכל נתון שנטען. ועל כן אנחנו משתמשים ב-observers שמחכים לאירוע ההתחברות/ההתנתקות והם יעדכנו את המסך בהתאם לאירוע שעולה.

* אופן המימוש

ILoadObserver – interface שמי שממש אותו הוא Observer והוא מבצע את פעולות הnotify אשר במקרה שלנו הם התחברות והתנתקות.

המחלקות ListBoxLoadObserver<T> ו-ListViewLoadObserver<T> הינם המחלקות הממשות את הממשק, הם מחזיקות את אובייקט ה-UI המתאים אליו צריך לטעון את הנתונים ואת פעולה (FUNC<>) הטעינה. T מייצג את המחלקה הנטענת אל הפקד, במקהר שלנו כולם Adapters.

ב-Form יש רשימה של ILoadObservers וכאשר יש אירוע מתאים אז מודיעים לכלל ה-observer לבצעו.

* Sequence Diagram



* Class Diagram



**תבנית מס' 2 – Decorator**

* סיבת הבחירה / שימוש בתבנית:

Facebook Post הוא אובייקט שניתן להציגו בצורות שונות – (עם משתמשים מתויגים/רק כותרת/ רק הודעה וכו'). אם בעתיד נרצה לשלב ולתת לתצוגה (ToString במקרה שלנו) מותאמת אז נוכל לעשות זאת בקלות ע"י מימוש decorator חדש ולתמוך בכמה הצגות שונות במקביל ותוך שינוי קוד פשוט. במקרה הקונקרטי שלנו יש שימוש בשינוי הצגה של הפוסט על מנת שיראה גם את ההודעה שב-POST.

* אופן המימוש:

יצרנו interface חדש בשם IFacebookPostAdapter אשר אותו ממשים ה-Adapter של Post שהיה כבר ממומש ובנוסף הוספנו מחלקה אבסטרקטית שנקראת FacebookPostAdapterDecorator שכל מי שרוצה לעשות דקורציה ל-ToString של ה-PostAdapter יכול לרשת וממש.

המחלקה האבסטקרטית מחזיקה PostAdapter כך שבמקרה של בקשה לאחד המתודות של ה-Post המקורי אז ניגשים ישירות למקורי אך כאשר נרצה את המאפיין לו עשינו דקורציה (ToString) אז נקבל את הדקורציה המתאימה.

* Sequence Diagram



* Class Diagram



**תבנית מס' 3 – Iterator**

* סיבת הבחירה / שימוש בתבנית:

במערכת שלנו יש שימושים באלבומי התמונות של המשתמש, ואלה מיוצגים ע''י המחלקה FacebookAlbum, ובתוכם נמצא מבנה הנתונים שמחזיק את כל התמונות. אין אנו רוצים שהמע' שמשתמשת ב-FacebookAlbum תדע מהו מבנה הנתונים, ואיך לרוץ עליו קונקרטית, וכן לא נרצה שהמערכת תוכל לשנות אותו. לכן נשתמש בתבנית ה-Iterator כך שסריקת מבנה הנתונים שקופה עבור כל מבנה ש-FacebookAlbum יחזיק.

* אופן המימוש:

המע' שעושה שימוש באלבומי התמונות היא AppManager, אשר זה מריץ תמונות שונות מאלבום נבחר בטופס בסריקה מחזורית ע''י דפדוף קדימה ואחורה. ממישנו ממשק חדש אשר יורש מ-IEnumerator, וקראנו לו ICyclicEnumerator. את הממשק הזה ממשת המחלקה AlbumIterator, וזה נשלח לכל אחד שירצה לסרוק אלבום ע''י מימוש-IEnumerable ב-FacebookAlbum. המימוש עושה שימושים בפיצ'רים של C# 3.0, ובמימוש הפונ' הדרושות מה-Interface של המערכת, וכן ממימוש-ICyclicEnumerator והוספת המטודה MovePrev().

* Sequence Diagram



* Class Diagram

