### 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 – Adapter

* סיבת הבחירה / שימוש בתבנית:

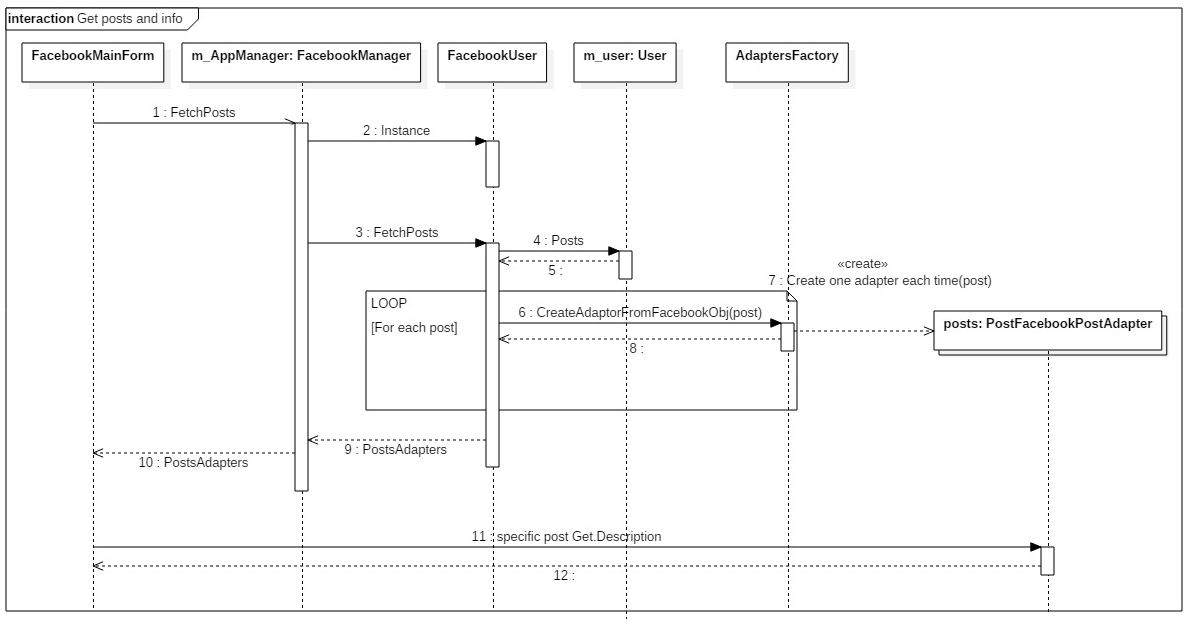
לא ניתן לשנות את המחלקה post של ה-API ולכן אם בעתיד יבוצע בו שינוי נרצה לשנות רק חלק מהקוד שלנו אשר נמצא רק ב-adapter ולא בכלל מחלקות המערכת. בנוסף, ה-UI אינו מכיר את האובייקט post ונרצה לייצר ממשק נוח לעבוד (אין צורך בכלל הנתונים שלו של ה-post המקורי). השתמשנו בתבנית adapter על מנת להתאים את האובייקט מה-API אל הדרישות של הUI (FacebookPost מתרגיל 1).

התבנית ממומשת מספר פעמים למחלקות שונות (Post, Event ו-User בתור friend).

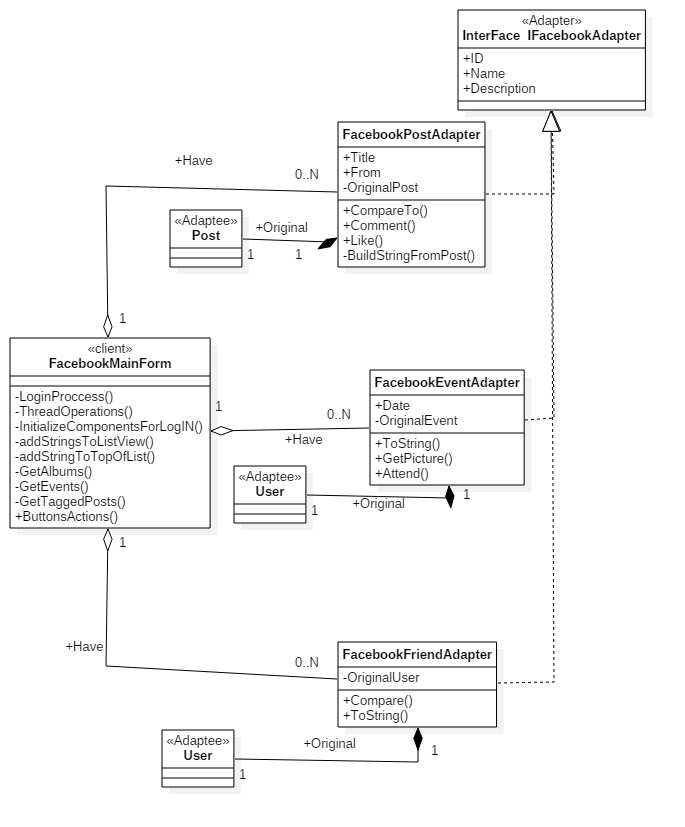
* אופן המימוש:

קיים Interface בשם IAdapter אשר כלל ה-Adapters יורשים ממנה. המימוש של כל Adapter נמצא במחלקה אותו מייצג. המימוש מבוצע ע"י כך שה-Adpter מחזיק אובייקט מתאים ובכל פנייה/בקשה הוא מביא את הנתון בעזרה ה-API המתאים לדרישות ההמערכת, במקרה שלנו ה-UI.

* Sequence Diagram



* Class Diagram



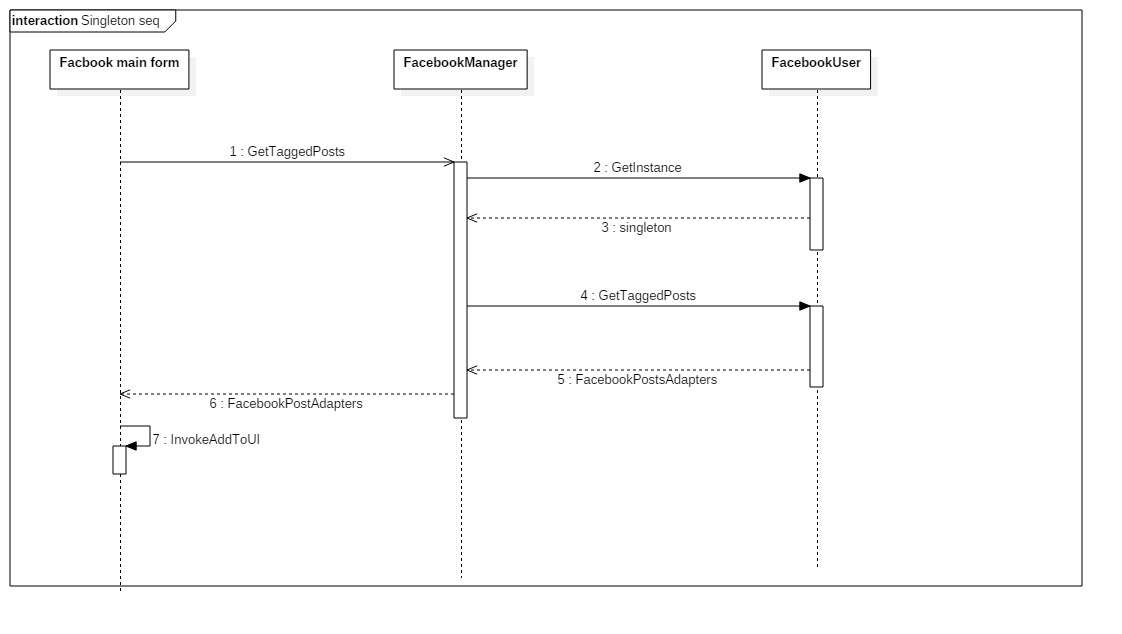
### תבנית מס' 2 – Singletone

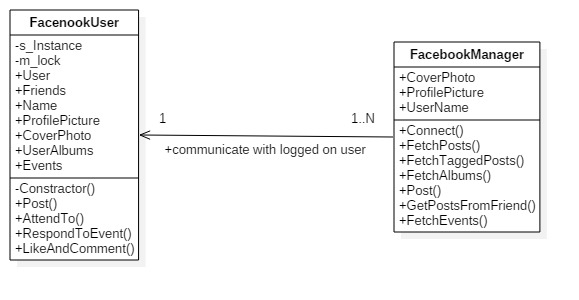
* סיבת הבחירה / שימוש בתבנית:

בתרגיל מספר 1 יצרנו מחלקה לייצוג המשתמש שמחובר כרגע במערכת, המחלקה נקראת FacebookUser, יש בה את פרטי המשתמש הנוכחי. מכיוון שבכל רגע נתון אנחנו רוצים לוודא כי יש משתמש אחד מחובר במערכת וכל בקשה שמבוצעת היא דרכו בלבד, נדרש שימוש ב-singlton. בעתיד כאשר נרצה לבצע שינויים במערכת, כלל המחלקות והשירותים במערכת יעבדו עם אותו משתמש מחובר.

* אופן המימוש:

המימוש נמצא במחלקה FacebookUser ואופי מימוש הsingleton הוא בעזרת נעילה משולשת, בדומה לדוגמא שהוצגה בכיתה (שני בדיקות שהאובייקט שונה מ-null ובנוסף נעילה בעזרת הפקודה lock ואובייקט סטאטי במחלקה. כמובן שCTOR המחלקה הוגדר כ-private.

* Sequence Diagram
* Class Diagram

E

### תבנית מס' 3 – Factory

* סיבת הבחירה / שימוש בתבנית:

בהמשך לתבנית הקודמת, אנו רוצים להעביר את מלאכת יצירת ה-Adapter למחקה מרוכזת אשר יודעת לייצר לכל מחלקה נדרשת את ה-Adapter המתאים שלה. אם בעתיד יצרטפו עוד Adapters ויהיו שינוי באופי יצירת ה-Adapters נוכל לבצע את כלל השינויים במחלקת ה-Factory כאשר ה-client, עובד באותה צורה של יצירת אובייקטים.

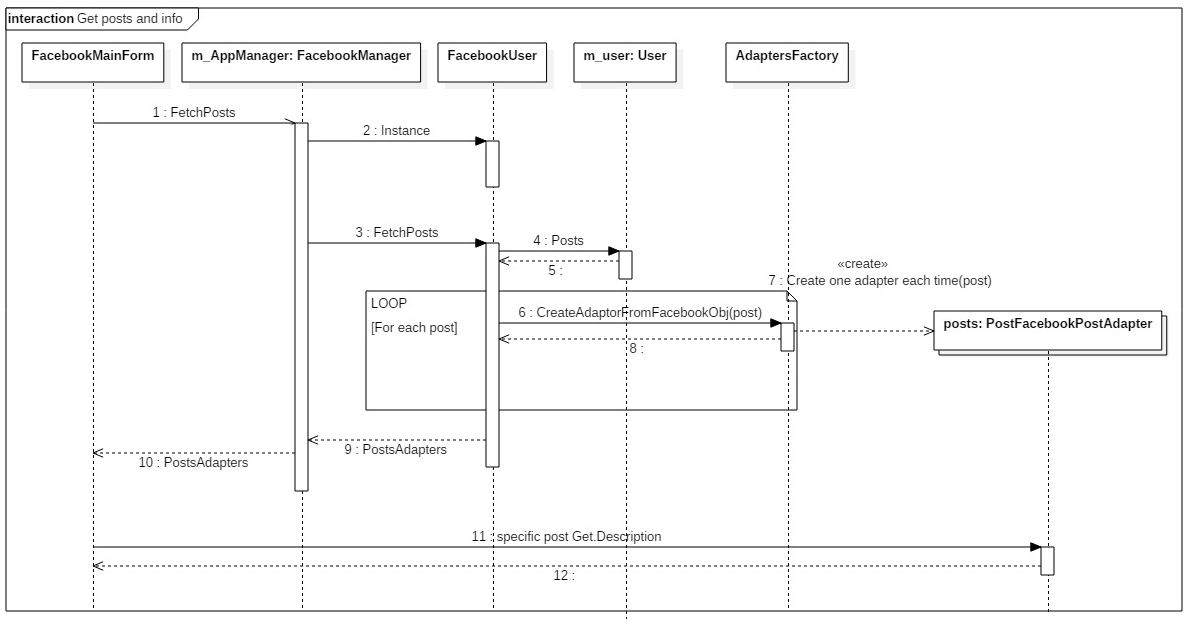
* אופן המימוש:

על מנת לממש את התבנית השתמשנו בתבנית הקודמת – Adapter. יש מחלקה שנקראת AdaptersFactory שהיא מחלקה סטטית עם מתודה סטטית CreateAdapterFromFacebookObj אשר מקבל אובייקט ולפיו מייצרת את הAdapter המתאים, במידה והאובייקט לא מתאים אז מוציאה Exception.

Face

* Sequence Diagram

התהליך בתבנית ה-Adapter מתאים בדיוק לתהליך האינטרקציה בין המחלקות ב-Factory, התהליך בה ליידי ביטוי בצעדים 3-9. הclient – שהוא FacebookUser יודע לשלוף מחלקות מסוג שנתקבל ב-DLL (User, Post וכו') על מנת לייצר את ה-Adapter המתאים כך שמחלקות המערכת ידעו לעבוד איתם, הוא נותן ל-Factory לייצר לו את המחלקה המתאימה.



* Class Diagram 