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

* קבלת מידע על אירועים שהתרחשו בשנת הלידה של המשתמש.

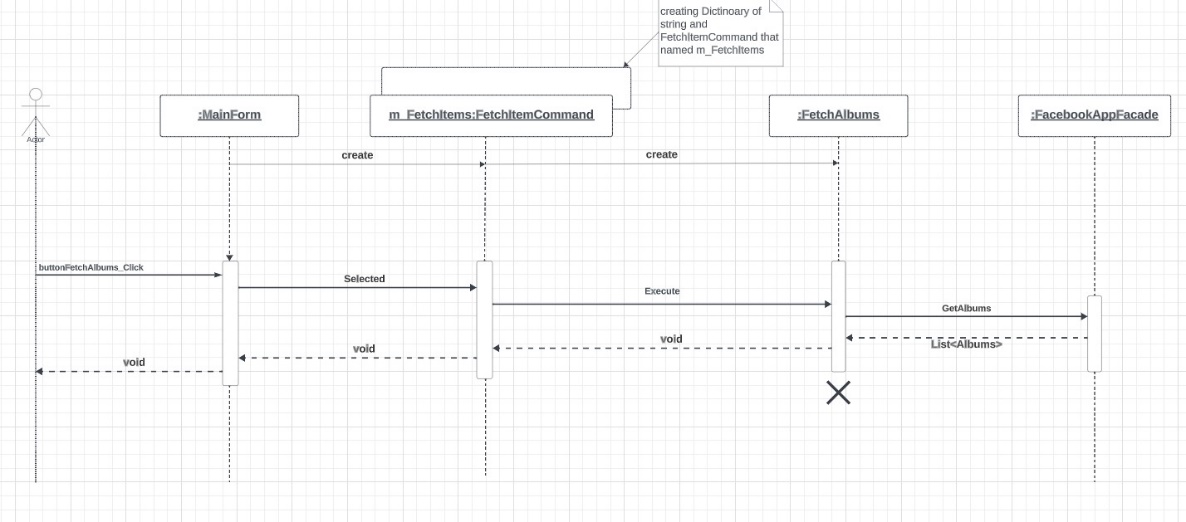
ע"י עבודה מול api חיצוני שמקבל את שנת הלידה של המשתמש ומחזיר לנו רשימה של אירועים שהארעו בשנת הלידה של המשתמש.

* בלחיצה על הכפתור Play Checkers נפתח חלון המשחק ובוא מופיע שם השחקן הראשון כשמו של המשתמש וניתן לו אופציה לשחק מול המחשב או לתת שם לשחק שני שיחק מולו.

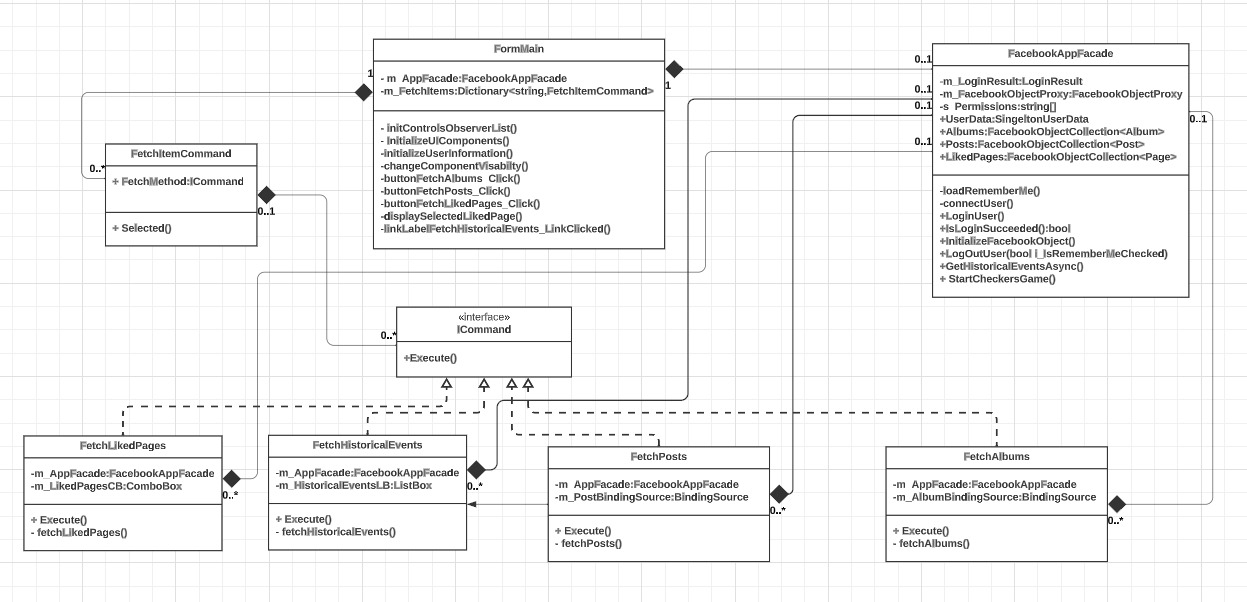
### תבנית מס' 1 – Command Pattern

* סיבת הבחירה / שימוש בתבנית: שמנו לב שבמערכת שלנו יש מגוון פעולות על מגוון המידעים אותו מספק ה Facebook API שלצורך הצגתו אנו צריכים לבצע פעולות fetch עבור כל אחד מסוגי המידע, בשל כך ראינו צורך בלממש את הפטרן הנ"ל על מנת שלכל פעולת מידע שנצטרך לבצע עליה fetch נייצר מחלקה חדשה שתכיל את הפונקציה fetch שמבצעת את הפעולה. כל זה לצורך כך שב FormMain כל מה שנצטרך לעשות זה להוסיף שורה אחת שבה אנו יוצרים fetchItem עם המחלקה החדשה הרלוונטית ומפעילים אותה במקום המתאים.

* אופן המימוש: יצרנו מחקלה של FetchItemCommand אשר מחזיקה FetchMethod, וכל מה שהיא עושה זה לקרוא ל Execute של המחלקה שמממשת את ה ICommand ואת פעולת ה fetch עצמה שהיא רוצה לבצע בעזרת data members שמועברים למחלקה ביצירתה מה ,FormMain ומפעילה את המתודה הרלוונטית להצגת המידע.
* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – Observer pattern

* סיבת הבחירה / שימוש בתבנית:

במערכת שלנו הפקדים אינם פעילים עד אשר מתבצעת התחברות לאפליקציה, ולכן יש פה תלות של רכיבים רבים (הפקדים) לבין הכפתור הLogin .

התבנית Observer עוזר לנו בתחזוקה עתידית של כלל הפקדים, כאשר נוכל להגדיר פעולות בצורה תחזוקתית פשוטה בעתיד (כמו למשל להוסיף פנל עם מידע חדש של תיאור מידע נוסף למערכת).

כך שלא נצטרך לשנות את הקוד להגיב לשינוי של מידע הקשור לעובדה עם הנתונים בform הקיים.

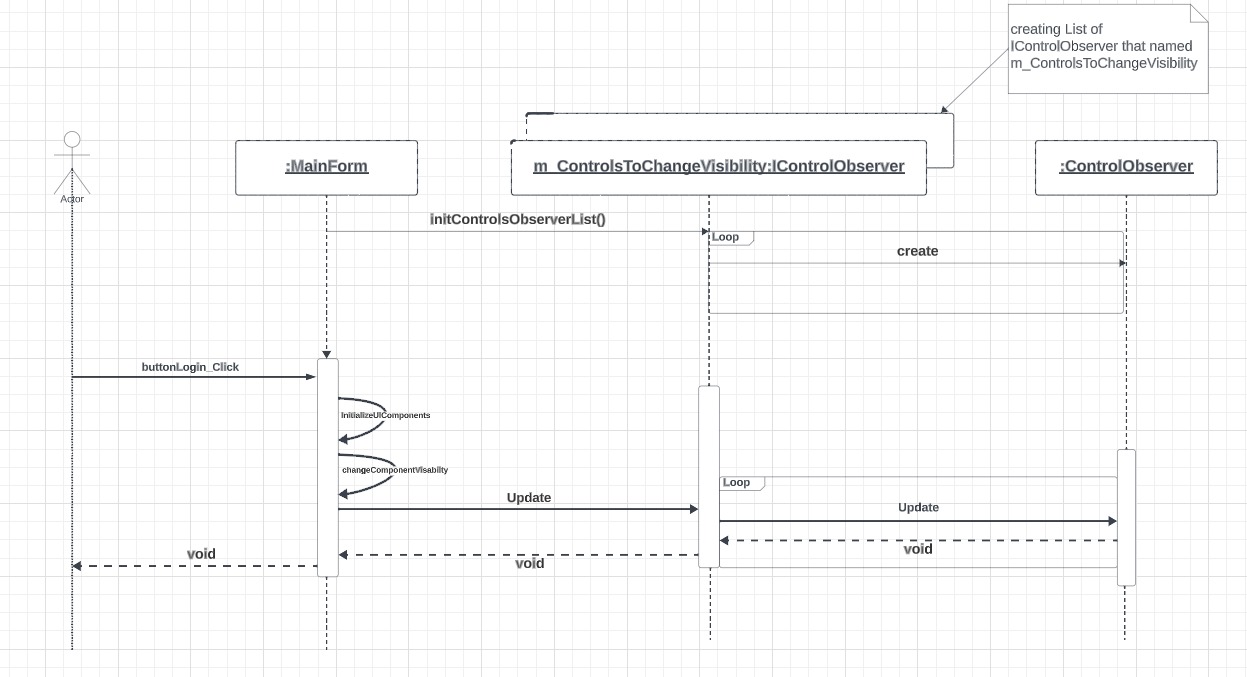
* אופן המימוש:

המחלקה FromMian.cs

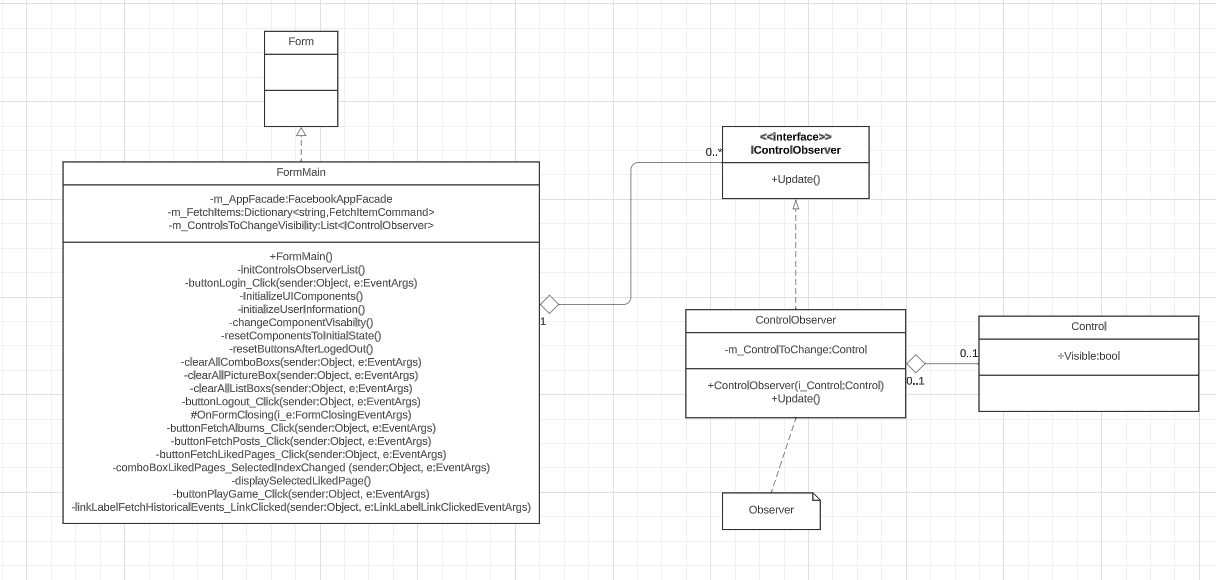
יצרנו רשימה של IObserverControl מתמלאת על ידי מעבר על כל הפקדים שיש לנו בtab הראשי, פרט לכפתור הLogin . ברגע שההתחברות בוצע בהצלחה מתבצע הקוד של המתודה changeComponentVisabilty שמשנה את המצב של כל בפקדים להיות visible .

במתודה עצמה הופעלת הפקודה Update שהיא המתודה של הinterface-.

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – Strategy pattern

* סיבת הבחירה / שימוש בתבנית:

ראינו שיש הרבה אי סדר בנתונים שמגיעים מהשרת של פייסבוק ורצינו לסדר אותם.

פרט לשימוש של הproxy כדי לשמור את המידע החלטנו גם למיין את הנתונים שמגיעים ולא להציג מידע באופן רנדומלי אלא על פי סדר מסוים.

ולשם המיון נדרשנו ליצור מתודה למיון של האוביקטים.

בחרנו בstrategy עם מימוש של delegateמכיון שאם בעתיד נרצה להוציא מידע נוסף מהשרת שלא עובר דרך הproxy נוכל למיין אותו כראוי כדי להציג אותו למשתמש.

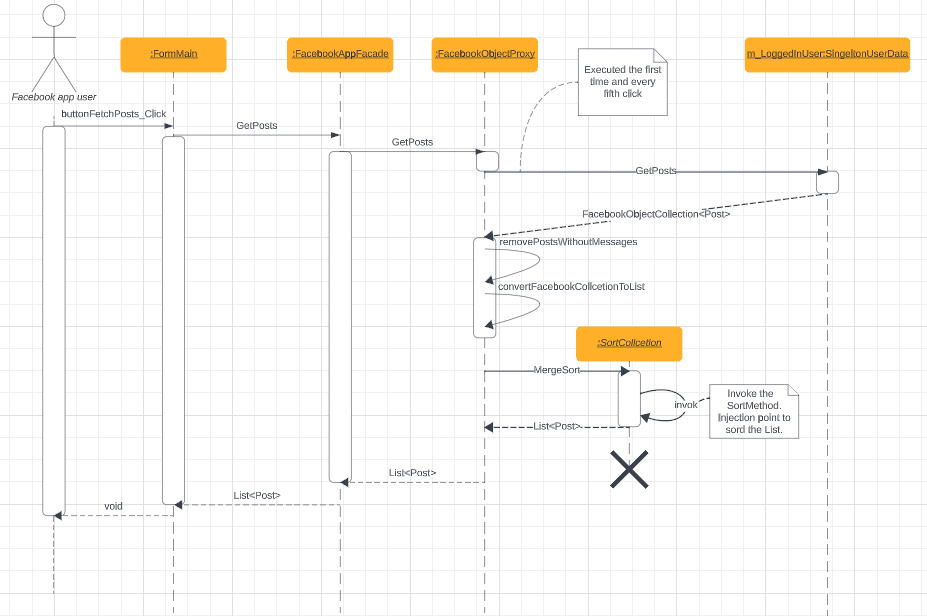
* אופן המימוש:

יצרנו מחלקה שנקראת SortCollcetion<T> אשר מקבלת delegate שלפיו ממיינים את הList שיקבל.

המיון עצמו שממשנו הוא merge sort כיוון שהוא יעיל מבחנית מיון O(nlogn)

כאשר מתבצעת משיכת מידע מהשרת של facebook על ידי הproxy נוצר אוביקט של המיון, הוא מקבל את האינדקטור למיון של המידע שמגיע מהשרת, המידע שמגיע כ FacebookObjectCollection מומר ל Listיעודי, עליו אנחנו מבצעים את המיון ואותו אנחנו מחזירים לFacade, שיחזיר לממשק והוא יציג אותו למשתמש.

* Sequence Diagram



* Class Diagram

