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

* הפיצ'ר הראשון FindOptionalForRelationship מוצא על נתוני החברים שלי בפייסבוק את הרשימה של האנשים האופציונליים להתכבתבות ברגע זה אם עברו את הבדיקה לפי הנתונים שלהם ויצאו שהם אופציונליים לקשר כלשהו בשבילי.
* ברגע שנלחץ על הכפתור Google Selected Friend הדפדפן של האינטרנט יבצע בדיקה בגוגל על שם החבר המסומן ברשימת החברים

### תבנית מס' 1 – Façade Pattern

* סיבת הבחירה / שימוש בתבנית:

בחרנו בתבנית זו כדי לעשות את ההפרדה בין הForm(UI) לבין הלוגיקה של התכנית והשתמשנו בתבנית זו כדי שאם בעתיד נרצה למשל להחליף את ממשק המשתמש נוכל לבצע זאת בקלות מבלי להצטרך לשנות כלום, אלא רק להשתמש בתבנית הFACADE שלנו שהוא מעיין Controller/Manager

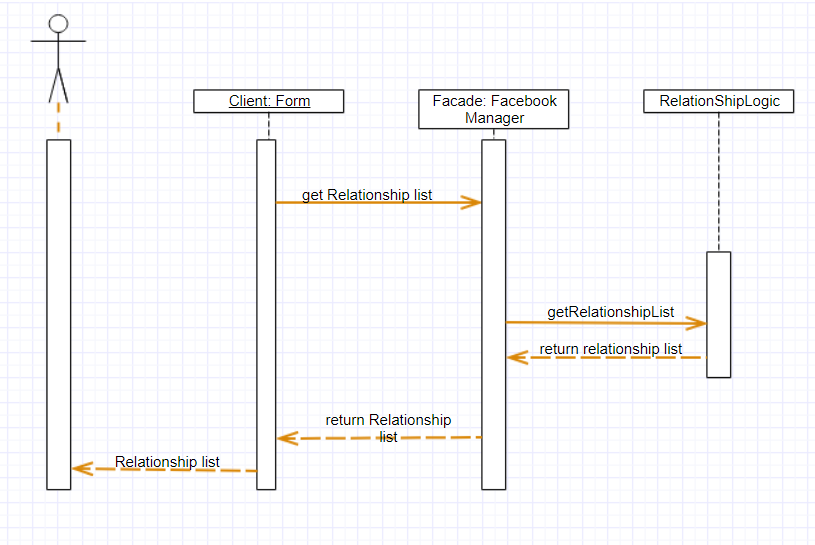
* אופן המימוש:

ממישנו את תבנית זו במחלקה שנקראת FacebookManager שהיא מעיין Controller בין ממשק המשתמש לבין הלוגיקה של התכנית ובנוסף תנהל לנו באפליקציה את

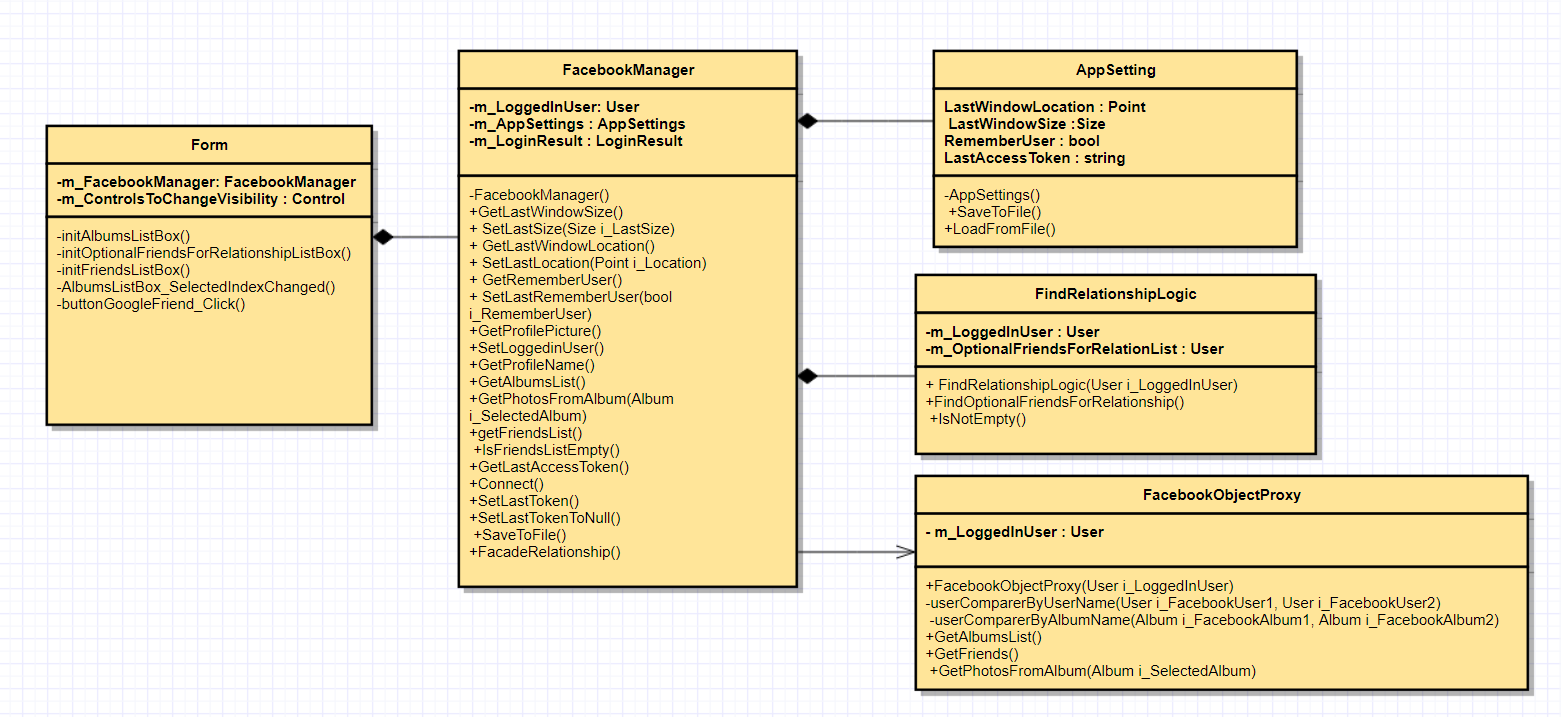
ההתחברות / הבקשות / והרצונות של ממשק המשתמש מבלי לדעת כלום על לוגיקת התכנית, ועשינו זזאת באמצעות קומוזיציה – כאשר ממשק המשתמש (FORM) מחזיק כMEMBER את המנהל ומשתמש בו כדי "לעבור" הלאה לבקשות מהלוגיקה המורכבת יותר.

ניתן למצוא אותו בקוד במחלקת FacebookManager

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – Singelton Pattern

* סיבת הבחירה / שימוש בתבנית:

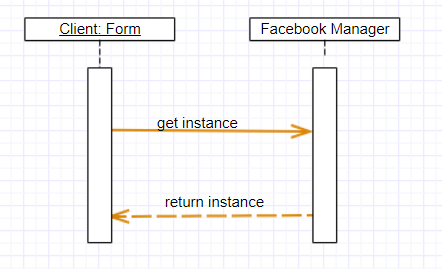
הסיבה שבה השתמשנו בתבנית זו היא כי יצרנו FacebookManager שהוא בעצם המנהל של האפליקציה שלנו ואנחנו רוצים שיהיה רק מנהל אחד כזה שישלוט בבקשות מלוגיקת התכנית, והכי חשוב שאם בעתיד נרצה לגשת או מישהו אחר ירצה לגשת למנהל זה נוכל לגשת אליו ולבקש ממנו את מה שנצטרך.

* אופן המימוש:

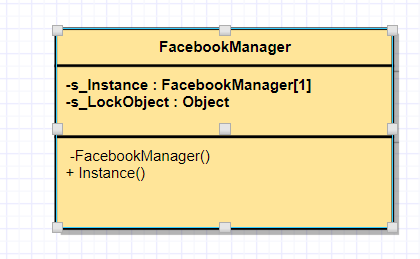
ניתן למצוא אותו בקוד במחלקת FacebookManager,

את בנאי המחלקה הסתרנו בעזרת Access Modifier- Private ויצרנו מתודה סטטית שמחזירה תמיד את מופי המחלקה כאשר- אם עוד לא בנו אותה היא בונה אותה (רק בפעם הראשונה) ואם כן בנו אותה בעבר אז היא מחזירה את המופי הקיים ע"י החזקת MEMBER של המופע.

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – Proxy Pattern

* סיבת הבחירה / שימוש בתבנית:

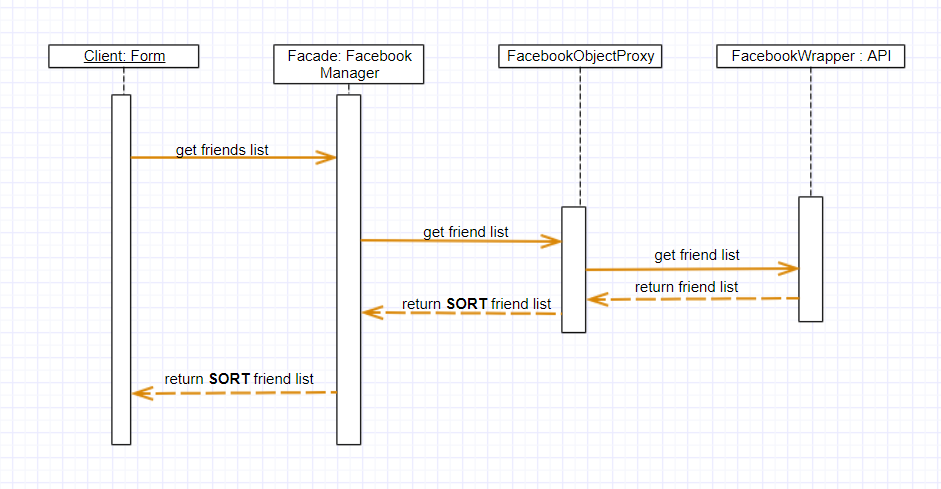
השתמשנו בתבנית זו כדי שנוכל לבצע מיון של האובייקטים USER ו ALBUM עפ"י שם ולהחזירם כאובייקטים מה שבעצם נותן לנו את האפשרות להוסיף פונקציונליות נוספת שאצלנו באפליקציה באה לידי ביטוי .

* אופן המימוש:

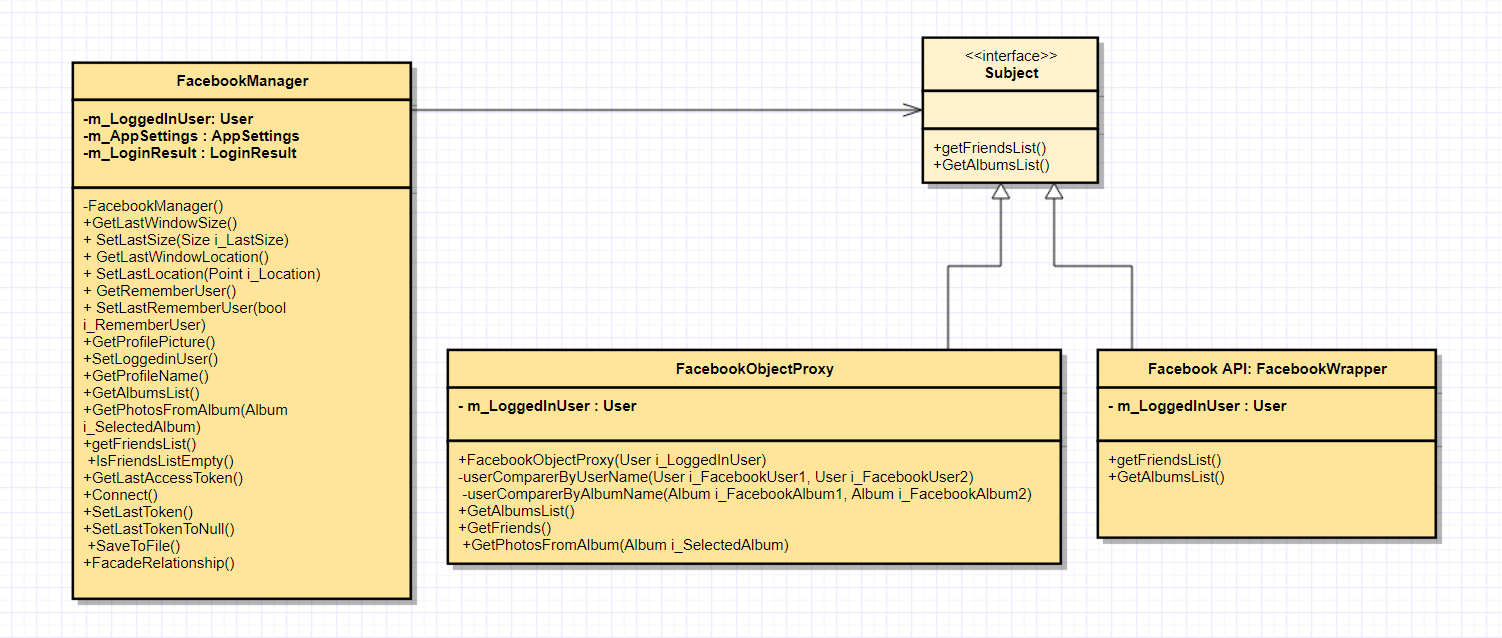
ניצן למצוא אותו במחלקת FacebookObjectProxy,

המחלקה מתחזה למי שנותנת את המידע בעיני ממשק המשתמש ומנהל האפליקציה בכך שההתנהגות של מנהל האפליקציה (FacebookManager) כלפי מחלקת הפרוקסי דומה מאוד להתנהגות שהייתה לפני זה עוד כשהיה לה גישה ישירה לAPI של פייסבוק מה שמאפשר לנו לבצע Sorting ורק אז להעביר את המידע למנהל האפליקציה ועשינו זאת באמצעות קומפוזיציה והחזקת Member של הUSER במחלקה שלנו והתחזות למחלקת הAPI.

* Sequence Diagram



* Class Diagram



### עבודה אסינכרונית –

את העבודה האסינכרונית באפליקציה שלנו עשינו בממשק המשתמש כאשר אנחנו מעלים את האפליקציה, כל פונקציונליות משתמשת בTHREAD נפרד כי מדובר בבקשות לשרת והבאת מידע מה שיכול לגרום ליעילות טובה יותר , אבל בעצם יצר לנו בעיה משום שהחיבור בין רכיבים בUI והשימוש בTHREADS יכול לגרום להרבה בעיות של סנכרון המידע, ולכן השתמשנו בANONYMUS METHOD ביחד עם Lambda Expression וDelegate כשי ליצור את הסנכרון בין התהליכונים לבין ממשק המשתמש.

### עבודה עם Data Bindings

את השימוש בData Bindings בתכנית שלנו אפשר למצוא בחיבור בין Friend List שנמצא בממשק המשתמש לבין Friend Data במתודה בשם initFriendsListBox כשבעצם כל פעם נבחר שם אחר מרשימת החברים הוא יציג לנו את הData על אותו חבר באמצעות שימוש בData Bindings והאובייקט User.

