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

* **דרג את החברים שלך:**

הפיצ'ר מאפשר למשתמש לדרג את החברים שלו לפי כמות האהדה שלהם כלפיו בפייסבוק. הדירוג מחושב לפי מספר הלייקים והתגובות של החברים על פוסטים של המשתמש, וכן על מספר הדפים, הקבוצות והצ'ק אינים שיש להם במשותף.

* **שלושת התמונות האהובות ביותר:**

הפיצ'ר מאפשר למשתמש לצפות ב3 התמונות שזכו למספר הגבוה ביותר של לייקים, מבין כל התמונות שהוא פרסם בפייסבוק.

### תבנית מס' 1 – Facade

* סיבת הבחירה / שימוש בתבנית:

התבנית מאפשרת עבודה נוחה של הForm אל מול ממשק אחד פשוט, שמספק את כל הדרישות שלו.

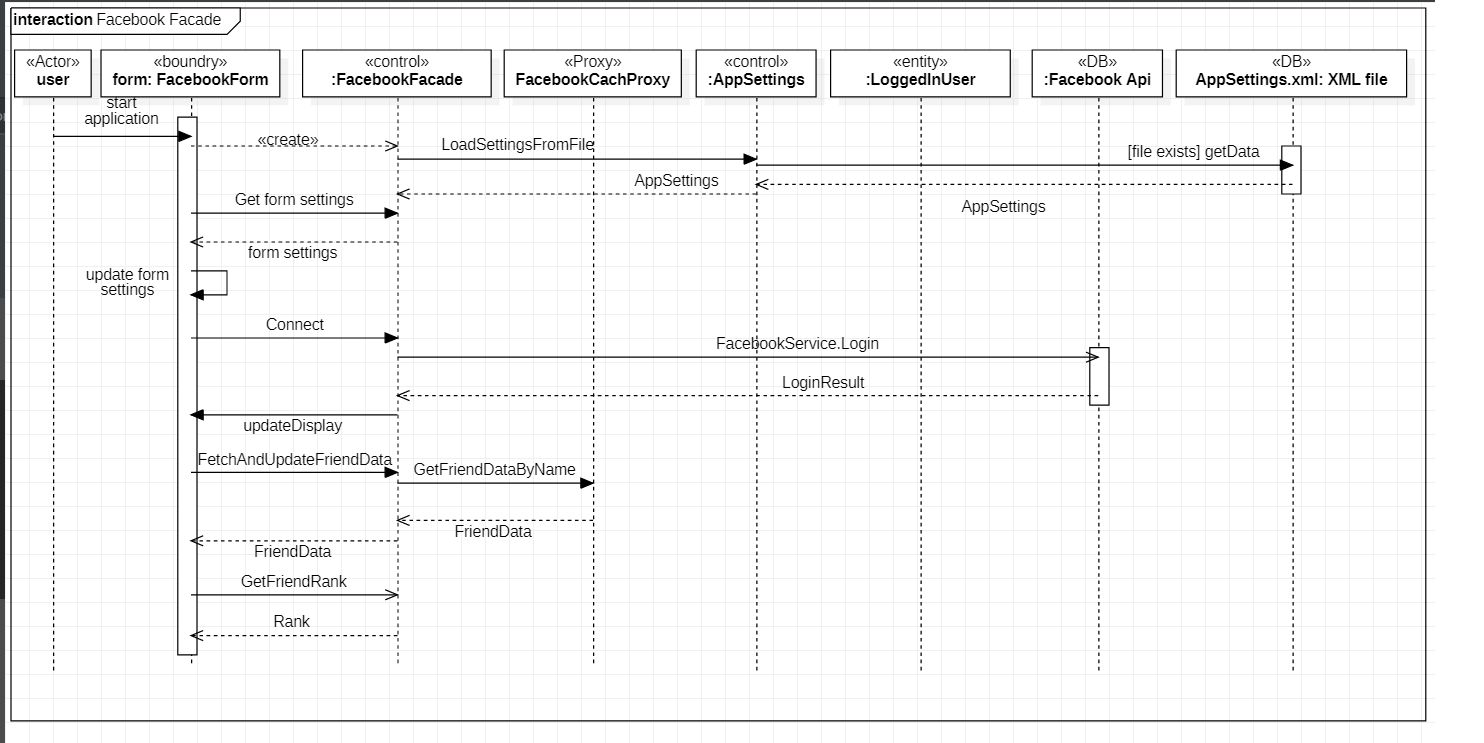
התבנית אחראית למעשה להפעלת כל רכיבי המערכת הנדרשים לביצוע פקודה מסויימת של הForm, באופן שמסתיר ממנו את כלל הרכיבים הנדרשים לשם ביצוע הפקודה.

* אופן המימוש:

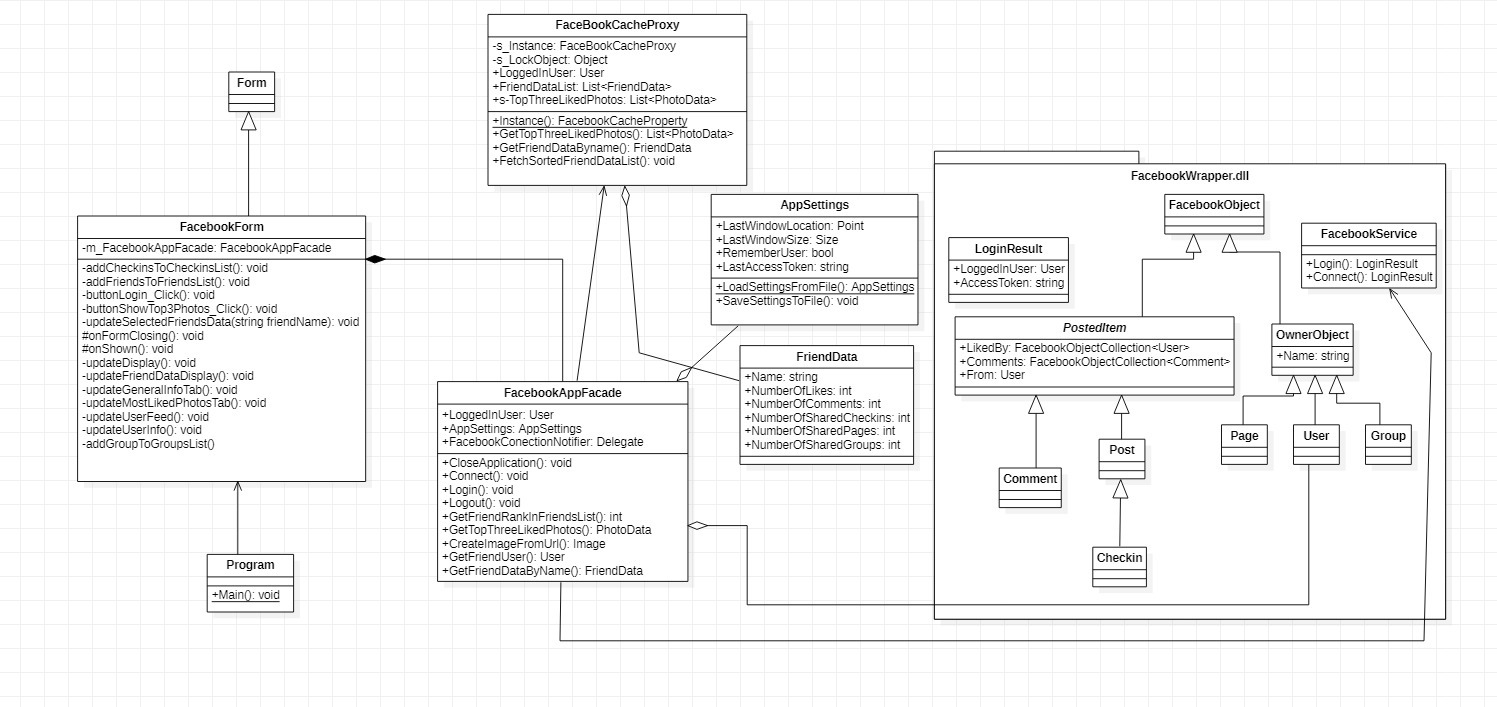
את התבנית מימשנו במחלקה FacebookAppFacade, והמחלקה שמשתמשת בה היא המחלקה FacebookForm. בפועל, הForm משתמש רק בFacade ולא מכיר את שאר הרכיבים במערכת.

הFacade אחראי על העבודה למול FacebookWraper, Proxy וכל שאר הרכיבים במערכת.

* Sequence Diagram - התרחיש המתואר הוא עליית המערכת ושימוש בפיצ'ר " דרג את החברים שלך":



* Class Diagram



### תבנית מס' 2 – cache Proxy

* סיבת הבחירה / שימוש בתבנית:

בחרנו להשתמש ב cache proxy בשביל להימנע מביצוע חישובים כפולים.

לדוגמא, בפיצ'ר שמאפשר להשוות בין חברי המשתמש מבחינת דירוג:

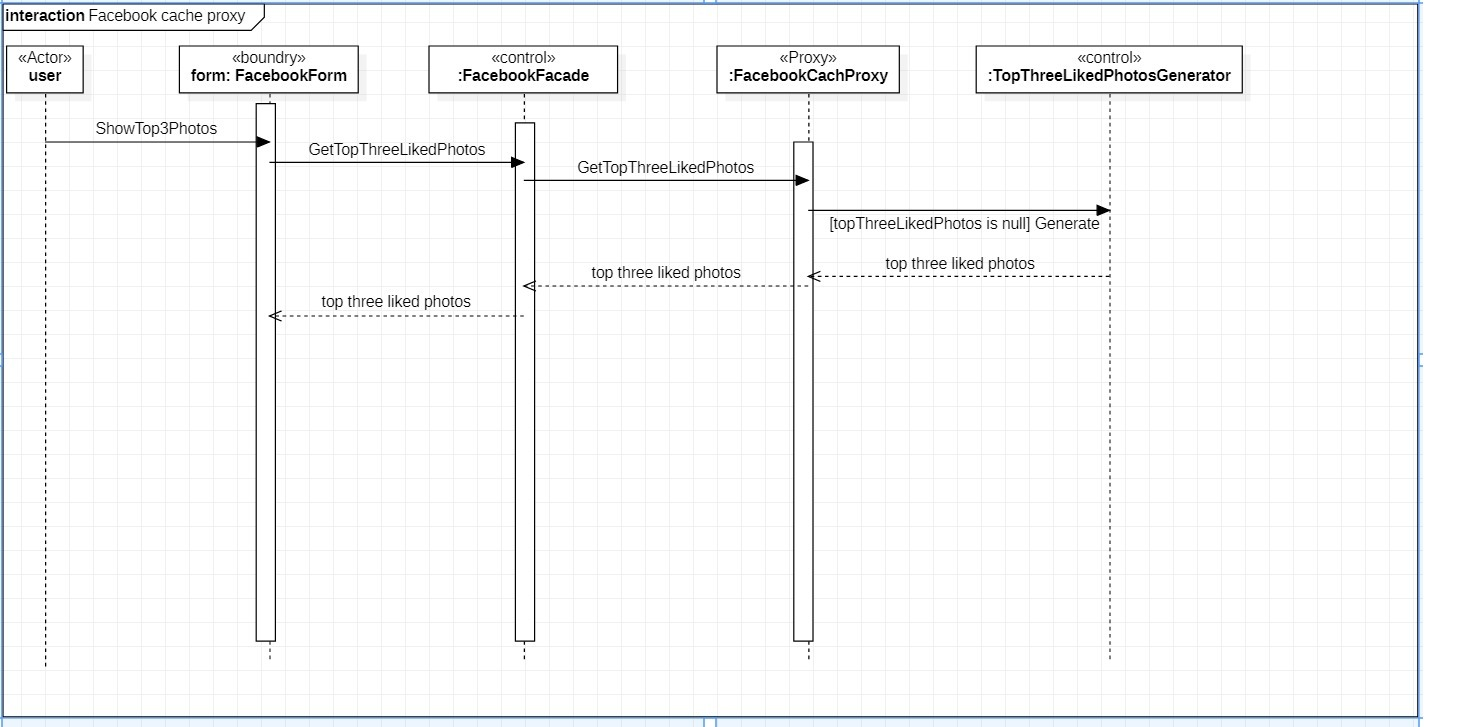
על מנת לחשב את הדירוג של חבר מסויים, צריך לחשב את הניקוד שלו ואז לדרג אותו ביחס לניקוד של כל שאר החברים של המשתמש. באמצעות הProxy, חישוב הניקוד של כל חבר ומיון רשימת החברים לפי ניקודם מתבצע בפעם הראשונה שהפיצ'ר בשימוש, ותוצאותו נשמרת בproxy. בכל השימושים הבאים בפיצ'ר, הרשימה כבר מעודכנת וממויינת ושמורה בproxy ובכך, נמנעים חישובים כפולים במהלך ריצת האפליקציה.

בנוסף, אם בעתיד נידרש להרחיב את פונקציונליות המערכת שלנו עם פיצ'רים נוספים הדורשים חישובים כבדים שעשויים לחזור על עצמם, תבנית הproxy תוכל לשמור גם את תוצאותיהם ובכך שוב ימנעו חישובים כפולים מיותרים.

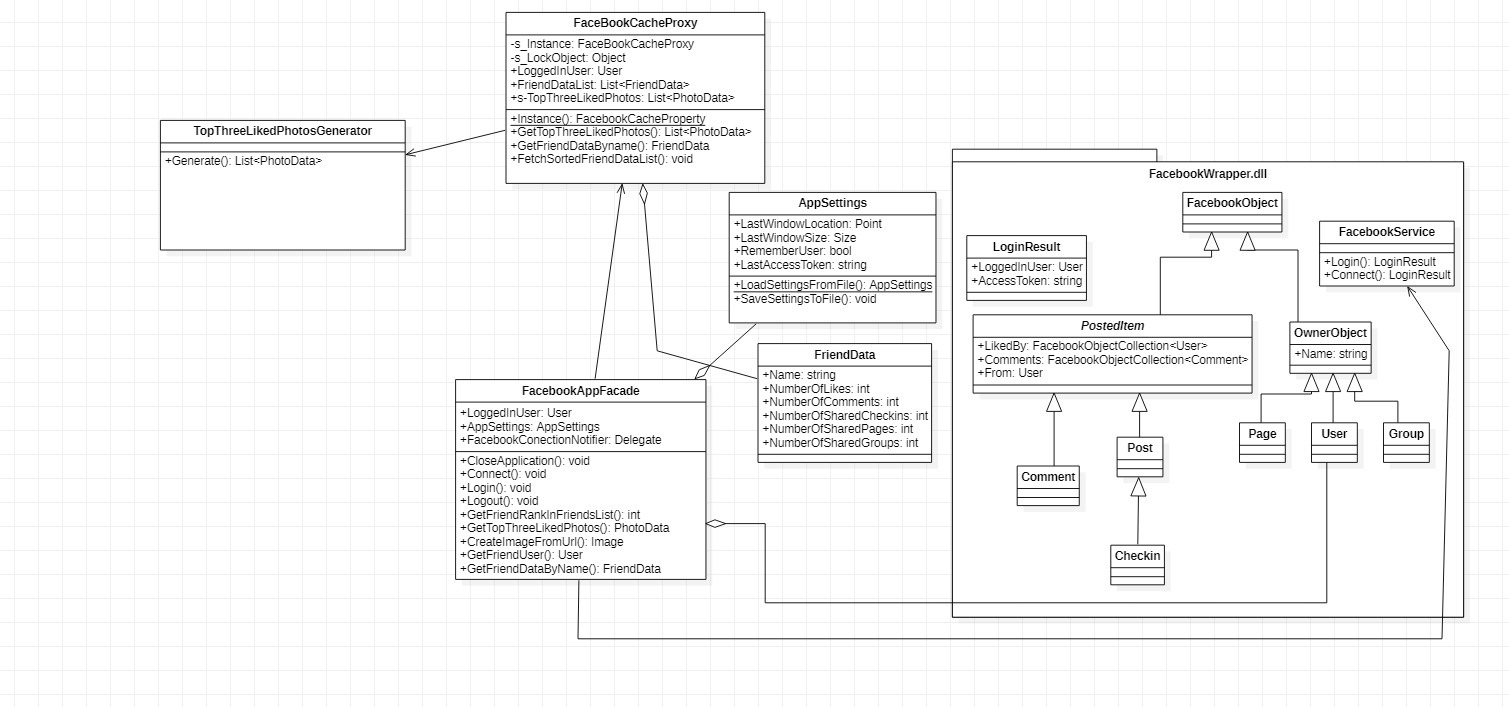
* אופן המימוש:

התבנית ממומשת במחלקה FacebookProxy, והמחלקה המשתמשת בה היא FacebookFacade. כאשר הFacebookFacade נדרש על ידי הFacebookForm לחשב את התוצאות של פיצ'ר מסויים, הוא מעביר בקשה אל הproxy, והproxy בתורו בודק אם תוצאת החישוב הנדרשת כבר קיימת אצלו – אם כן, הוא מחזיר אותה מבלי לבצע חישוב נוסף, ורק במידה שזאת הפעם הראשונה שהוא נדרש לחישוב הזה הוא אכן מחשב אותו, שומר את התוצאה ומחזיר אותה לFacade.

* Sequence Diagram - התרחיש המתואר הוא שימוש בפיצ'ר 3 התמונות האהובות ביותר



* Class Diagram



### תבנית מס' 3 – Singleton

* סיבת הבחירה / שימוש בתבנית:

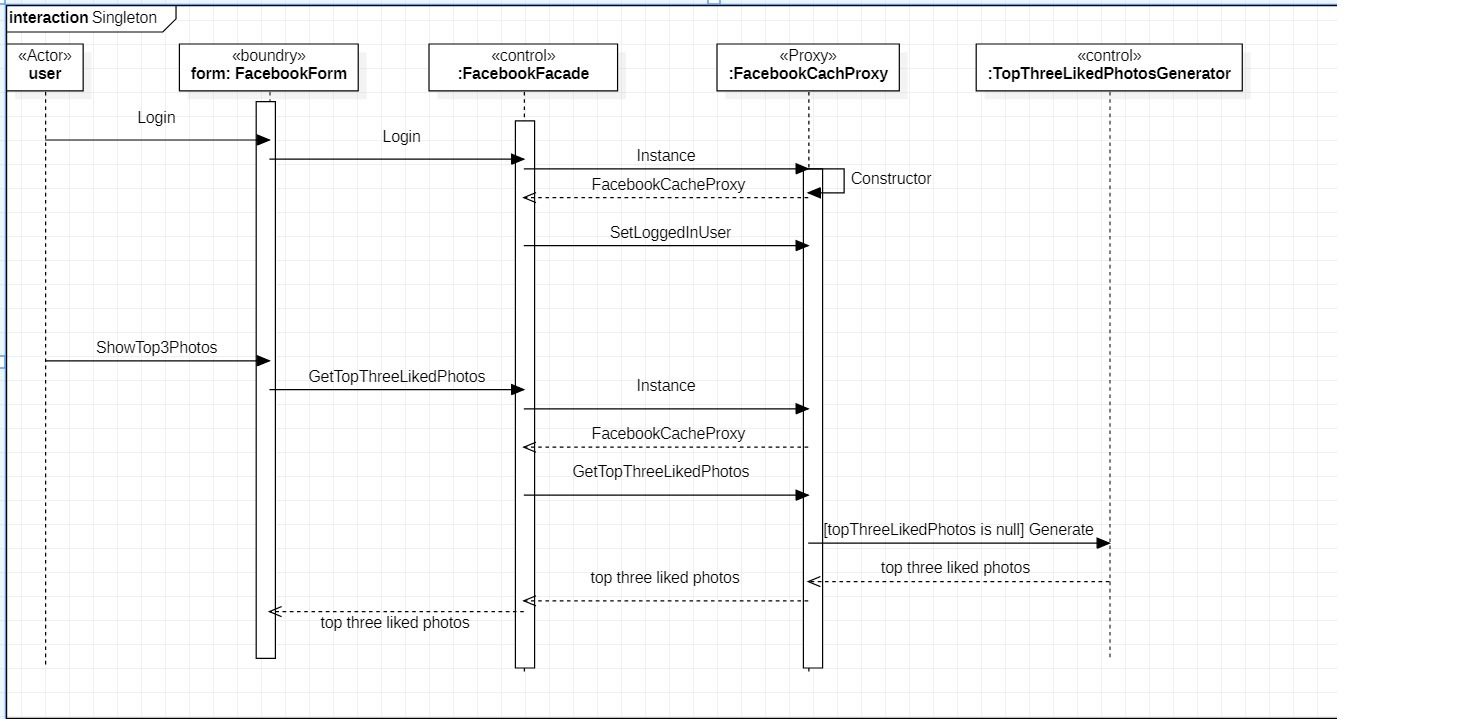
בחרנו להשתמש בתבנית singleton על המחלקה FacebookProxy מכיוון שרצינו למנוע מצב שבו קיים יותר ממופע אחד של הProxy. הסיבה לכך היא שכפי שציינו, המחלקה proxy אחראית על החזקת תוצאות החישוב של הפיצ'רים השונים של המערכת, ולכן, קיום יותר ממופע אחד שלה עלול לגרום למצב שחישוב מסויים אכן יבוצע יותר מפעם אחת, וזאת בסתירה לשאיפה שלנו שכל חישוב יבוצע לכל היותר פעם אחת בריצה של התוכנית.

* אופן המימוש:

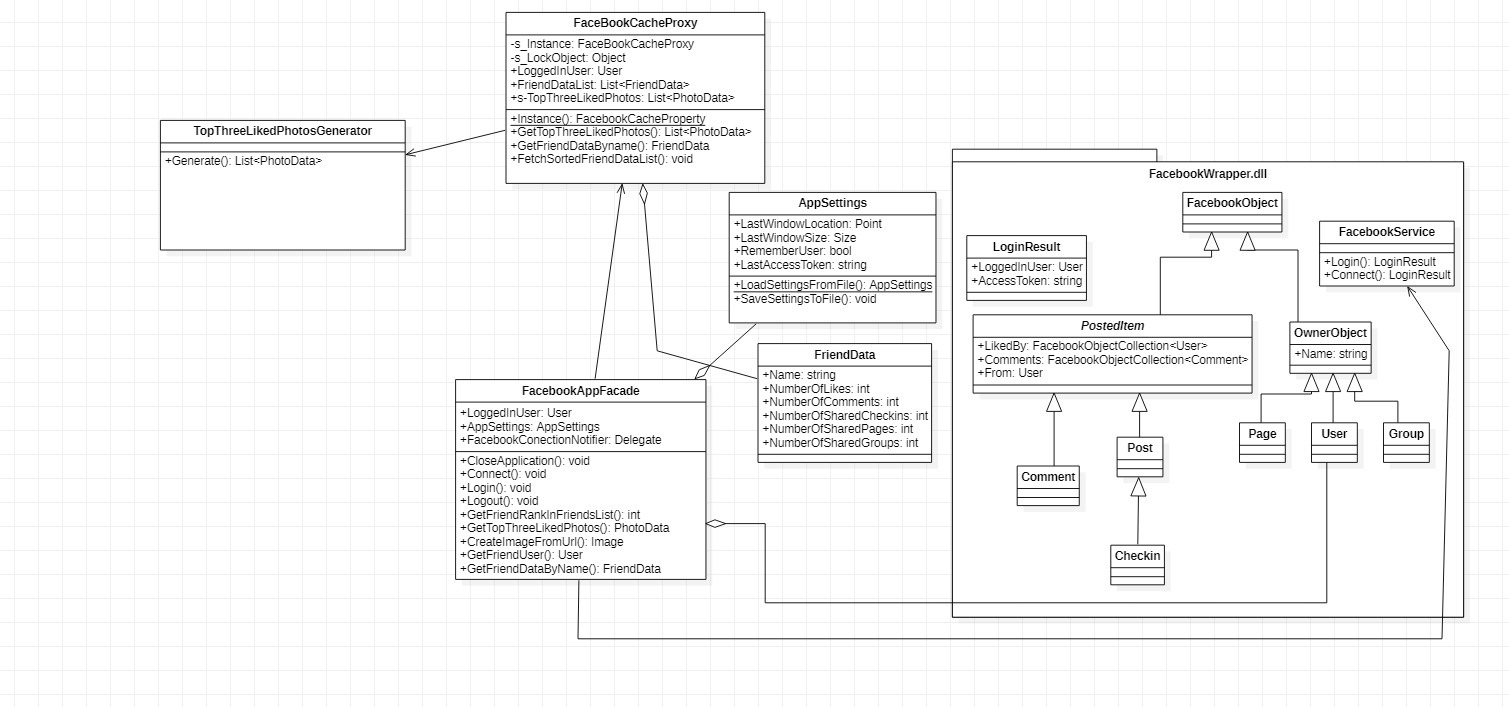
מימשנו את התבנית במחלקה FacebookProxy. עשינו זאת כפי שלמדנו בכיתה, על ידי הסתרת הconstructor, חשיפת מתודה סטטית Instance והגדרת המחלקה כpublic sealed.

בנוסף, מימשנו את הproxy באופן thread safe על ידי שימוש בdouble check lock.

* Sequence Diagram - התרחיש המתואר הוא התחברות ויצירת הFacebookCacheProxy instance בפעם הראשונה ולאחר מכן שימוש בפיצ'ר 3 התמונות האהובות ביותר



* Class Diagram



### שימוש בעבודה אסינכרונית

בתרחישי הlogin והconnect מופעלים thread-ים חדשים במחלקה FacebookAppFacade שמבצע את ההתחברות מול פייסבוק. בסוף פעולות אלה מופעל delegate שמעדכן את התצוגה בForm. לכן בthread הראשי של הUI נעשה העדכון של התצוגה במקום בthread הנפרד שהתחיל את הפעולה.

בנוסף בתרחיש RateYourFriends כאשר נבחר חבר לדירוג, נוצר thread חדש במחלקה FacebookForm וגם כאן התצוגה מתעדכנת על ידי הthread הראשי.

### שימוש ב – Data Binding

ב UI של ה – Form בטאב שנקרא Your Groups ישנו מימוש של Data Binding בין הGroupDataSource לפאנל שנמצא בטאב זה.

אופן המימוש בקוד נמצא ב FacebookForm.cs בשורה 98 בה אנו מקשרים את ה groupDataBindingSource.DataSource לרשימת הGroupData שהForm מקבל בחזרה מהFacebookAppFacade.