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

* הצגת חמשת התמונות של המשתמש שקיבלו הכי הרבה לייקים.
* הצגת רשימה של אנשים מפורסמים שנולדו באותו תאריך (בלי שנה) עם המשתמש.

### תבנית מס' 1 – Singelton

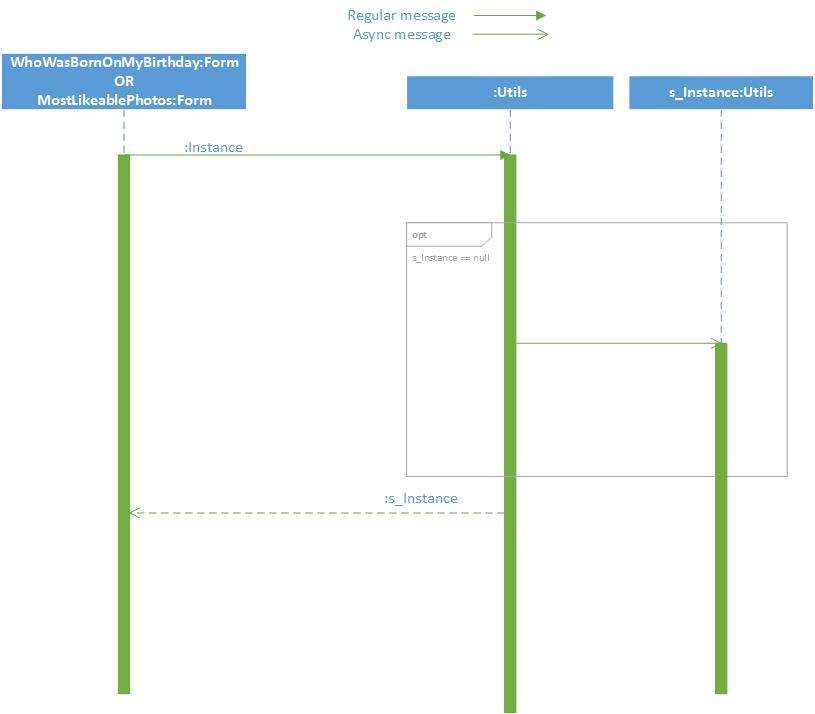
* סיבת הבחירה / שימוש בתבנית:

יש לנו מחלקה של שנותנת שירותים למשתמש (UTILS). נרצה לגשת למחלקה זו מכל מחלקה שהיא במערכת. נרצה לייצר אותה רק פעם אחת ולהתשתמש בשירותיה לפי דרישה. על כן בחרנו לממש אותה כסינגלטון.

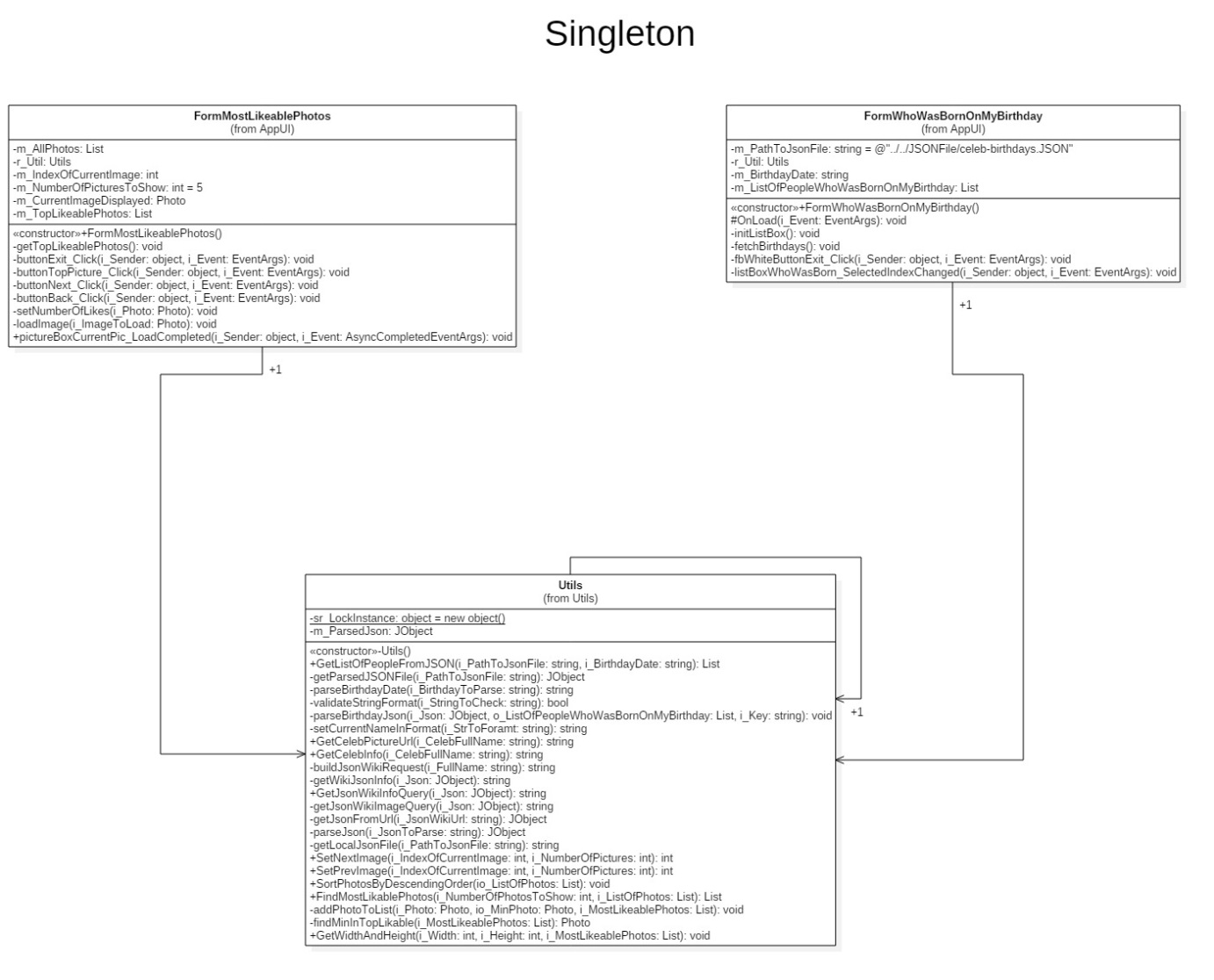
* אופן המימוש:

המחלקה Utils הוגדרה כ- public והיא בעלת constructor שהוא Private וכאשר רוצים להשתמש בשירותיה מבקשים את המופע שנוצר רק בפעם הראשונה שמבקשים את שירותי המחלקה.

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – Builder

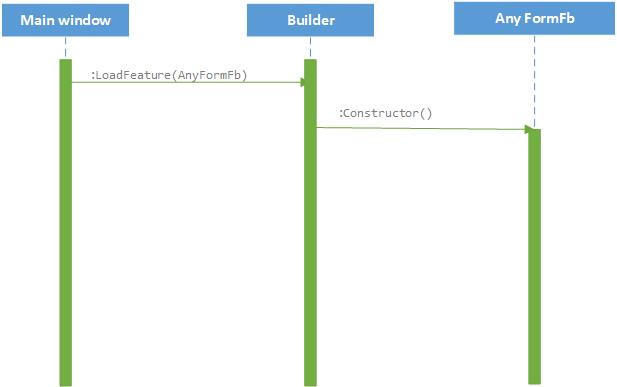
* סיבת הבחירה / שימוש בתבנית:

במידה ונרצה לייצר בעתיד פיצ'ר נוסף למערכת נרצה להקטין את כמות תוספת הקוד. לכן, יצרנו מחלקה שתיצור את הטופס עבור הפיצ'ר הנבחר ותציג אותו (ובמידה ונרצה פונקציונליות נוספות מהפיצ'ר אז נוסיף אותה לשם). בכך אנו מונעים מה- client את ה- new לפיצ'רים, אלא יפנה ל- builder שיצור אותם עבורו.

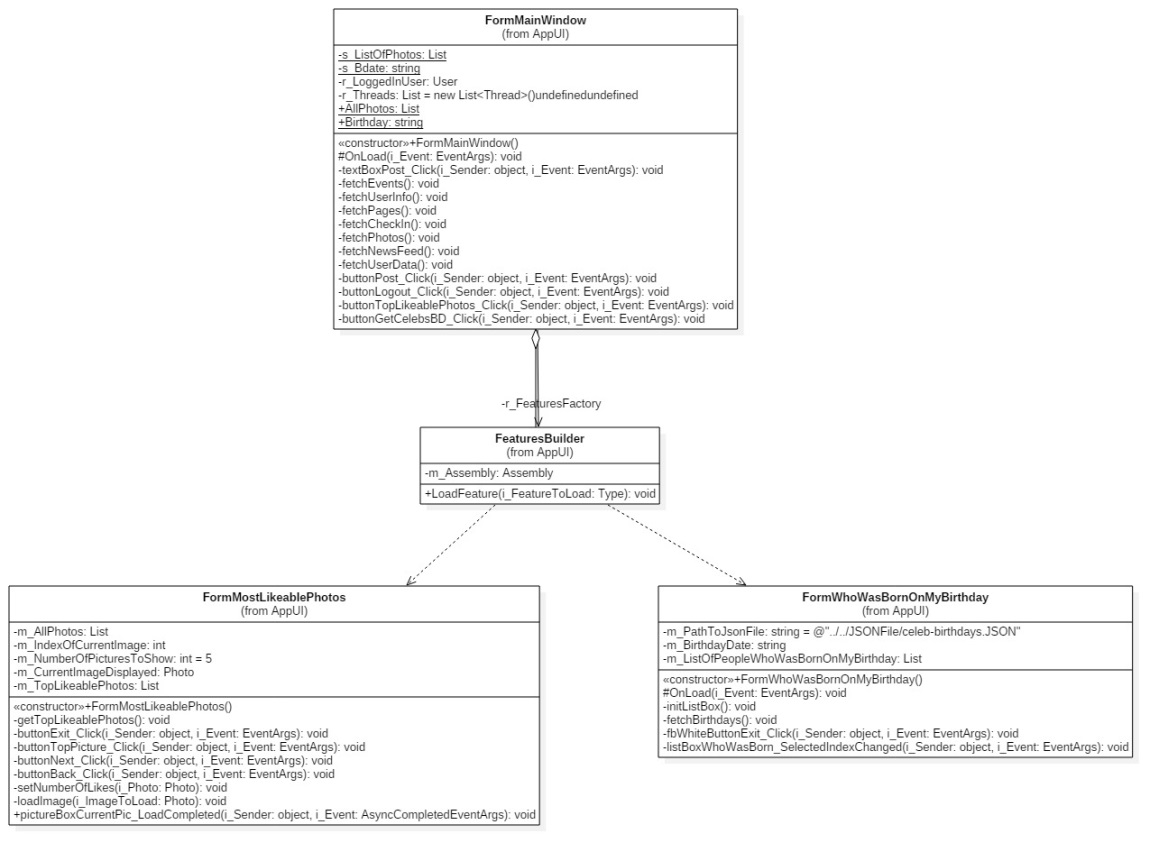
* אופן המימוש:

יצרנו מחלקה בשם FeatureBuilder שעל פי פניה אליה וה- type הנדרש תחפש באסמבלי הנוכחי את ה- type ובמידה וקיים תיצור את המופע שלו (ע"י reflection לאובייקט, מציאת ה- constructor שלו והפעלתו) ותציג את ה- FormFb. ה- builder יודע לעבוד רק עם FormFb ורק טפסים כאלו הוא ידע לייצר.

* Sequence Diagram



* Class Diagram



FeatureBuilder – מקבילה ל- Builder ב- Pattern

FormMainWindow – מקבילה ל- Director ב- Pattern

שני ה- Form התחתונים (הפיצ'רים) – מקבילים ל- Product ב- Pattern.

### תבנית מס' 3 – Proxy

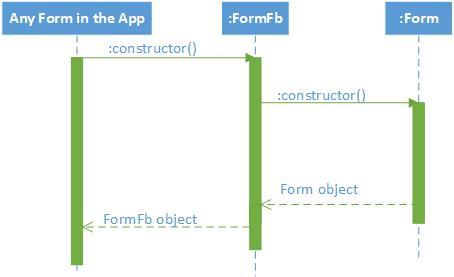
* סיבת הבחירה / שימוש בתבנית:

רצינו לייצר טופס אחיד עבור המערכת בעל עיצוב מתאים לפייסבוק ללא לשכפל קוד בין טפסים נוספים במערכת. לכן יצרנו טופס בשם FormFb עם העיצוב הנדרש שאותו שאר הטפסים הרלוונטיים במערכת מממשים. בכך כל טופס במערכת יהיה עם עיצוב אחיד ללא שורת קוד נוספת בו. באופן דומה, במידה ונרצה להוסיף עיצוב/פונקציונליות נוספת שרלוונטית לכל הטספים במערכת, נצטרך לשנות רק במקום אחד.

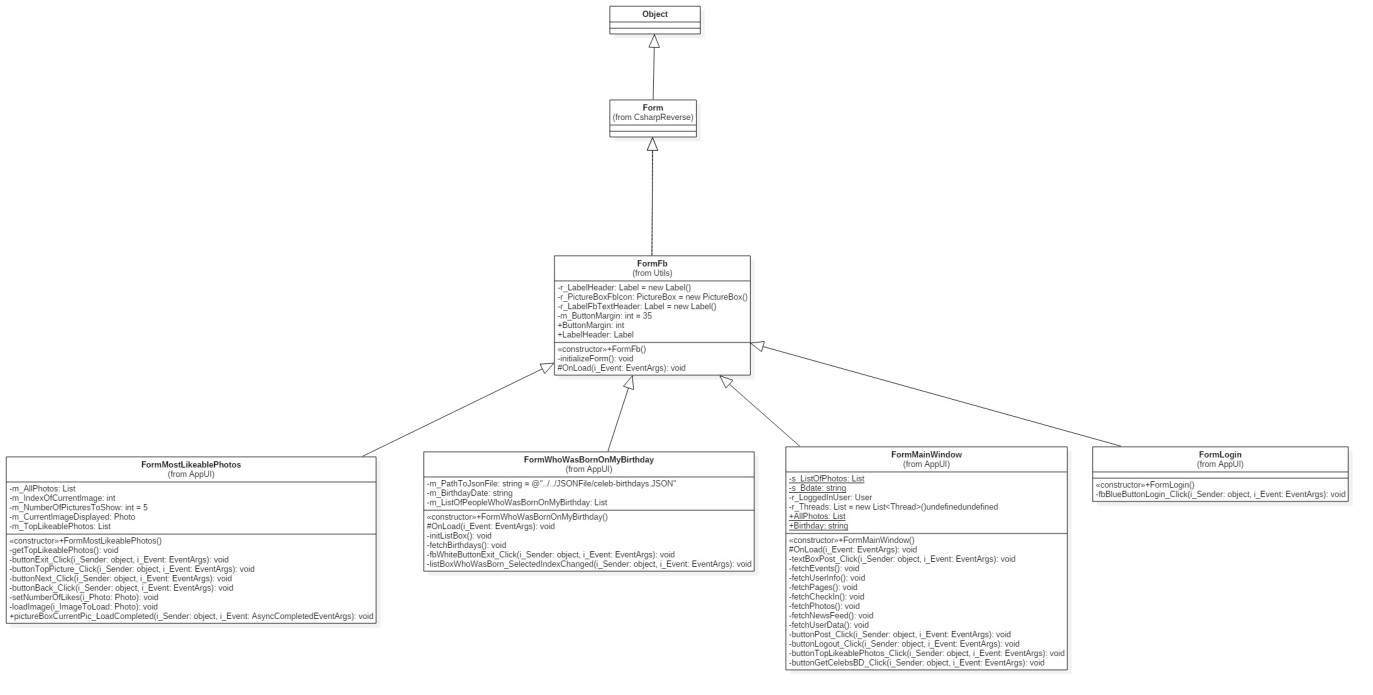
* אופן המימוש:

יצירת מחלקה FormFb שיורשת מ - Form רגיל ובה ממומש העיצוב הרצוי. כל טופס במערכת יורש מהטופס ה FormFb (נשים לב שקיימים proxy נוספים הם הכפתורים : ButtonFbWhite ו ButtonFbBlue שמהווים proxy ל button ועובדים באותו עקרון).

* Sequence Diagram



* Class Diagram



FormFb – מקבילה ל- Proxy ב- Pattern

Form – מקבילה ל- Real Subject ב- Pattern

Object – מקבילה ל- ISubject ב- Pattern.

**עבודה אסינכרונית**

בקוד ישנם מספר מקומות בהם מימשנו עבודה אסינכרונית. הסיבה לכך היא הרצון למנוע את קפיאת המערכת בעת טעינת הנתונים לפקדים. תחילה, בעת טעינת נתוני המשתמש כל פקד נטען בת'רד נפרד (שאינו UI ת'רד). מימשנו זאת ע"י יצירת ת'רד עבוד כל מתודת fetch (לדוגמה fetchEvents). לאחר מכן, כאשר מגיע המידע, הוא נטען לאפליקציה ע"י שימוש ב lambda expression לדוגמא

listBoxCheckIn.Invoke(new Action(() => listBoxCheckIn.Items.Add(fbCheckin)));

בנוסף עבור הפיצ'ר של מציאת 5 התמונות בעלות כמות הלייקים הגבוהה ביותר, מימשנו את טעינת כלל התמונות בת'רד נפרד והטופס לא יעלה עד שטעינה זו לא תסתיים (מומש ע"י פקודת join לת'רד הספציפי של טעינת התמונות).

**עבודה עם Data Bindings**

מימשנו עבודה עם Data Bindings עבור הפקד listBoxNewsFeed. הסיבה היא הרצון לערוך פוסט קיים ב - News feed דרך האפליקציה ללא פנייה מחודשת לשרתי פייסבוק (כמובן שהעדכון הוא לוקלי). המימוש נעשה באמצעות האובייקט postBindingSource שמתמלא בפוסטים המגיעים מהשרת ומוצגים ב- News feed. היות ואין Setter לpropery – Message של Post, אנו משתמשים ב MyPost (עוטף את הPost הרגיל ומוסיף לו פונקציונליות להצגת טקסט) כל פוסט שנסמן נראה אותו ב displayPostTextBox ונוכל לערוך אותו כבר בתוך שורת הטקסט והוא יעדכן באופן "אוטומטי" את הערך שלו ב news feed (2 way data binding). אם נרצה לפרסם סטטוס לשרתי פייסבוק נעשה זאת ע"י כתיבת טקסט ב postTextBox ולחיצה על הכפתור post נשלח הפוסט לשרתי הפייסבוק.