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

* הצגת "הצעדים הראשונים" של המשתמש בפייסבוק: הפוסט הראשון שהוא העלה, תמונת הפרופיל הראשונה, תמונת הCover- הראשונה, התמונה הראשונה שהוא העלה בקיר, התמונה הראשונה באלבום הראשון, הצ'ק-אין הראשון.
* הצגת המקומות האהובים שנבחרו על ידי המשתמש.

המידע מחושב על ידי כמות הפעמים שהמשתמש ביקר במקום מסוים, וממוין בסדר יורד, כך שהמקום המופיע ראשון ברשימה הינו "המקום הפוטנציאלי האהוב ביותר" של המשתמש.

### תבנית מס' 1 – Factory Method (Class Version)

* סיבת הבחירה / שימוש בתבנית:

מהיות שבאפליקציה שבנינו יש שימוש נרחב במגוון Forms, כמו FormFavouritePlaces, FormFirstSteps, FormLogin, FormMain, אזי נעדיף לפנות למחלקה כללית בשם FormFactory. נפנה אליה בכל פעם שנרצה ליצור טופס חדש, במקום שנפנה בכל פעם לForm הרלוונטי שאותו נרצה ליצור.

בנוסף, ניתן לראות שכל הטפסים שנעשה בהם שימוש באפליקציה הם פולימורפיים ולכן מדובר על משפחות פולימורפיות ולכן השימוש ב- Factory Class הינו שימוש נכון למקרה זה.

* אופן המימוש:

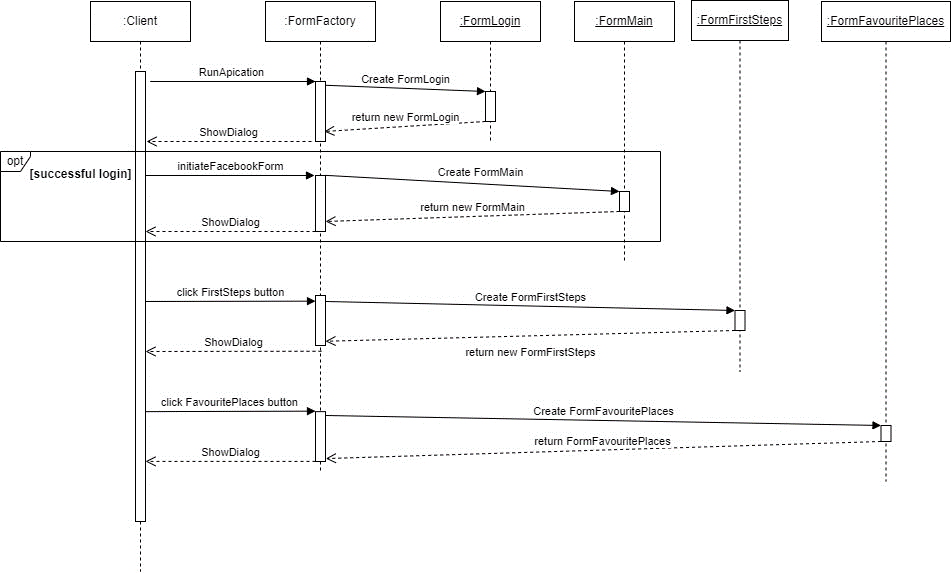
את ה Factory Method Pattern ניתן למצוא בקובץ FormFactory. במחלקה יש מתודה בשם CreateForm() שמקבלת כ string את שם הForm ומחזירה את ה Form המתאים.  
קריאה למתודה CreateForm() ניתן למצוא ב-

1. מחלקת Program, במתודה Main(). שם קוראים ל CreateForm()תוך העברת הString Login"" למתודה שבתורה תיצור טופס הנקרא FormLogin.

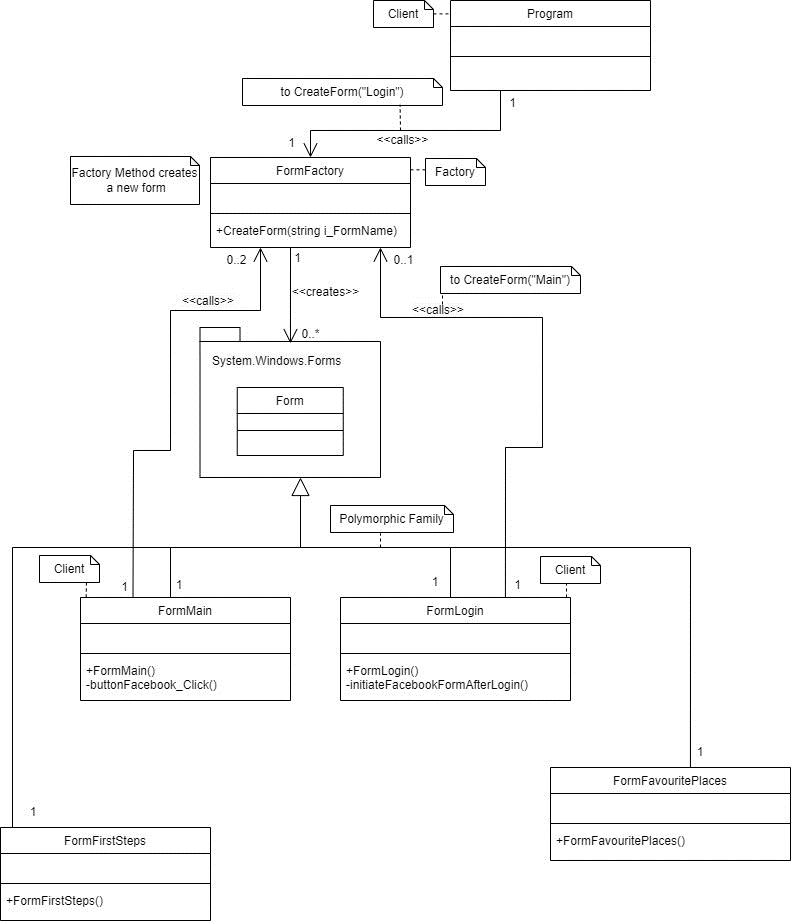
2. מחלקת FormLogin שם שוב נקרא למתודה CreateForm() תחת המתודה initiateFacebookFormAfterLogin – ונבקש ליצור את הטופס שנקרא FormMain באמצעות העברת הString Main"" למתודה CreateForm().

3. מחלקת FormMain, בפונקציה שנקראת buttonFacebook\_Click – היוצרת את הטפסים הבאים FormFavouritePlaces, FormFirstSteps בהתאם ל- Sender (כפתור) עליו לחצנו ובאמצעות העברת הStrings FirstSteps"" או "FavouritePlaces" למתודה CreateForm() שב - FormFactory.

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – Facade

* סיבת הבחירה / שימוש בתבנית:

התוכנה שלנו מכילה הרבה מחלקות שכל אחת מספקת לנו מידע כלשהו. לדוגמה צעדים ראשונים בפייסבוק ורשימה של מקומות אהובים

באמצעות הdesign pattern- Facade בנינו מחלקה שנקראת FacebookUserData, שמטרתה היא להסתיר למשתמש את כל "מאחורי הקלעים" של המחלקות האחרות בקוד, ובאמצעות המופע של המחלקה FacebookUserData לקרוא למתודות ולפרופרטיז שלו, וכך רק באמצעותו לחשוף לנו את כל המידע שאנחנו צריכים.

* אופן המימוש:

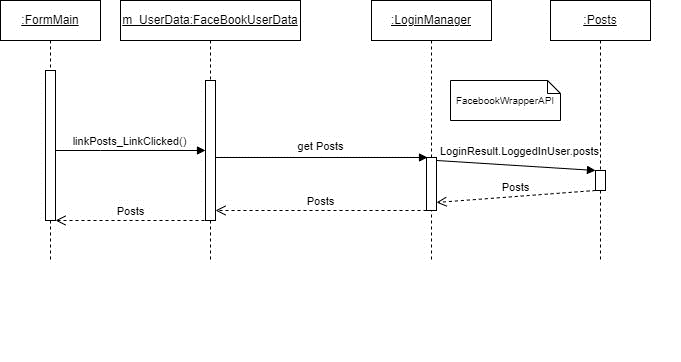
בקובץ FacebookUserData.cs אנו מאחדות את כל המידע שהטפסים FormMain, FormsFirstSteps, FormFavouritePlaces ו FormLogin צריכים. לדוגמה: FormMain מכיל את חבר המחלקה m\_UserData שהוא מופע של FacebookUserData. כדי לבצע פעולות לוגיות, תתבצע קריאה ל m\_UserData על שלל ה properties הרלוונטים.

* תיאורי התפקידים במערכת:  
  ה client – FormMain, FormsFirstSteps, FormFavouritePlaces ו FormLogin.

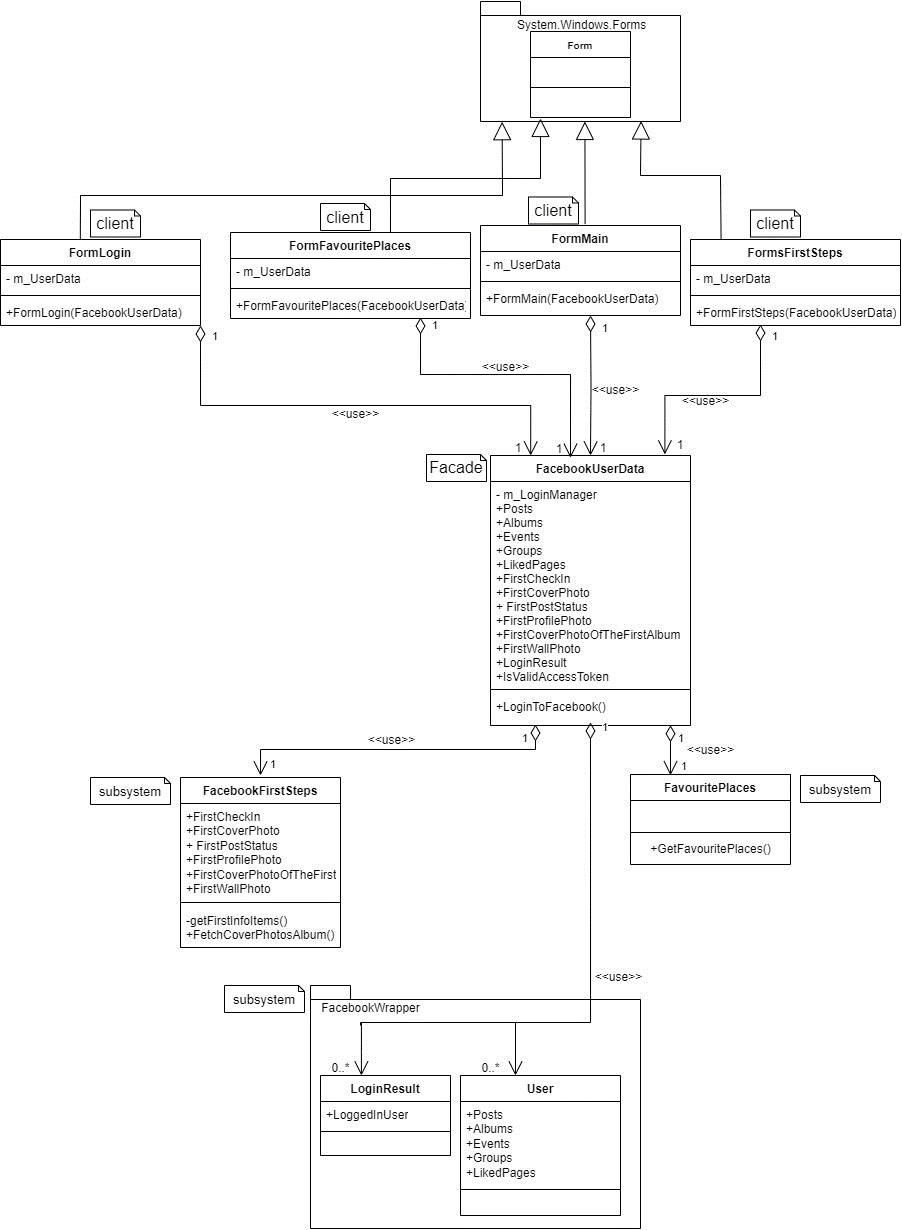
ה Facade – FacebookUserData

תתי המערכות –("מאחורי הקלעים") - Posts, Albums, Events, Groups, LikedPages, FacebookFirstSteps, PictureNormalURL.

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – Singleton

* סיבת הבחירה / שימוש בתבנית:

במערכת שלנו יש מחלקת FacebookFirstSteps. המטרה שלה היא להציג את הצעדים הראשונים של המשתמש מבחינת העלאת מידע בפעם הראשונה במספר קטגוריות. האובייקט של מחלקת FacebookFirstSteps מכיל את כל השדות המבוקשים לצעדים הראשונים. אובייקט זה נקרא ע"י מספר properties במחלקת FacebookUserData. היותו של FacebookFirstSteps כ pattern singleton מאפשר גישה אליו (אותו המופע האחד), מכל הproperties המבקשים אותו. על כן הוא נוצר פעם אחת, on demandולאחר מכן הרכיבים ישתמשו באותו האובייקט האחד שנוצר.

* אופן המימוש:

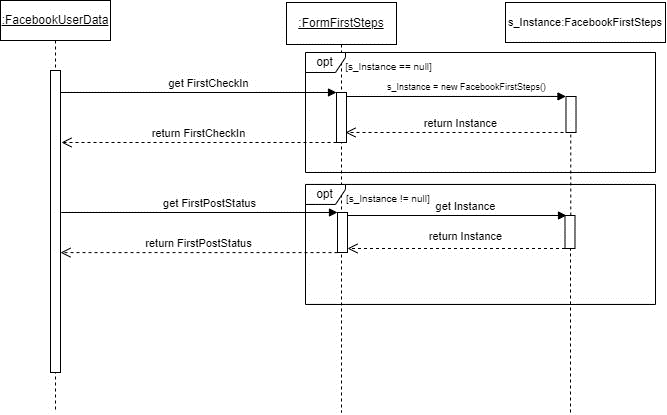
מחלקת FacebookFirstSteps היא בתפקיד ה singleton (מדובר ב sealed class), שם נמצא ה public static property: Instance. תפקידו של הפרופרטי הוא ליצור את האובייקט בפעם הראשונה, (בעזרת ה private c’tor), ולהחזיר את אותו אחד אם קוראים לפרופרטי שוב.

אנו משתמשים ב private static reference למופע שיווצר אם יווצר.

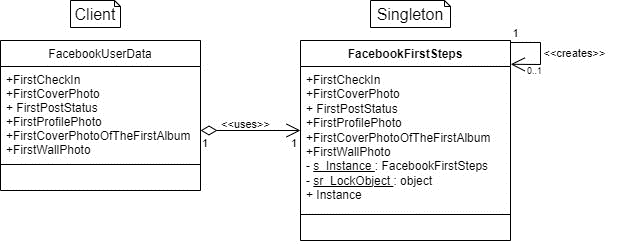
הproperty של ה instance עושה בדיקת double check lock במימוש שלו.

מחלקת FacebookUserData היא בתפקיד הclient שכן הרכיבים בה (מספר properties) קוראים ל FacebookFirstSteps.Instance.

* Sequence Diagram



* Class Diagram



**עבודה אסינכרונית עם ממשק משתמש**

* מקום בקוד בו עשינו שימוש בתכנות אסינכרוני:
* FormMain, FormFirstSteps, FormFavouritePlaces בקבצים האלו השתמשנו בתכנות אסיכנרוני כדי להשתמש בכמה threadים נפרדים שישיגו לנו את המידע הדרוש בזמן קצר יותר מאשר בsingle threading.
* כיצד: עטפנו את הקריאה למתודות בthread-ים ויצרנו Lambda Expressions בתוך מימושי המתודות שיפעילו את פונקציות ההוספה והטעינה של האובייקטים כגון: Event/Group/Page/Post/CheckIn, עטפנו אותן ב action ושלחנו ל Invoke.

**עבודה עם Data Binding**

* מקום בקוד בו עשינו שימוש ב Data Binding:

בקובץ FormMain.cs בפונקציה בשם loadAlbums(). שם עשינו קישור בין האלבום ותמונת הנושא שלו.