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

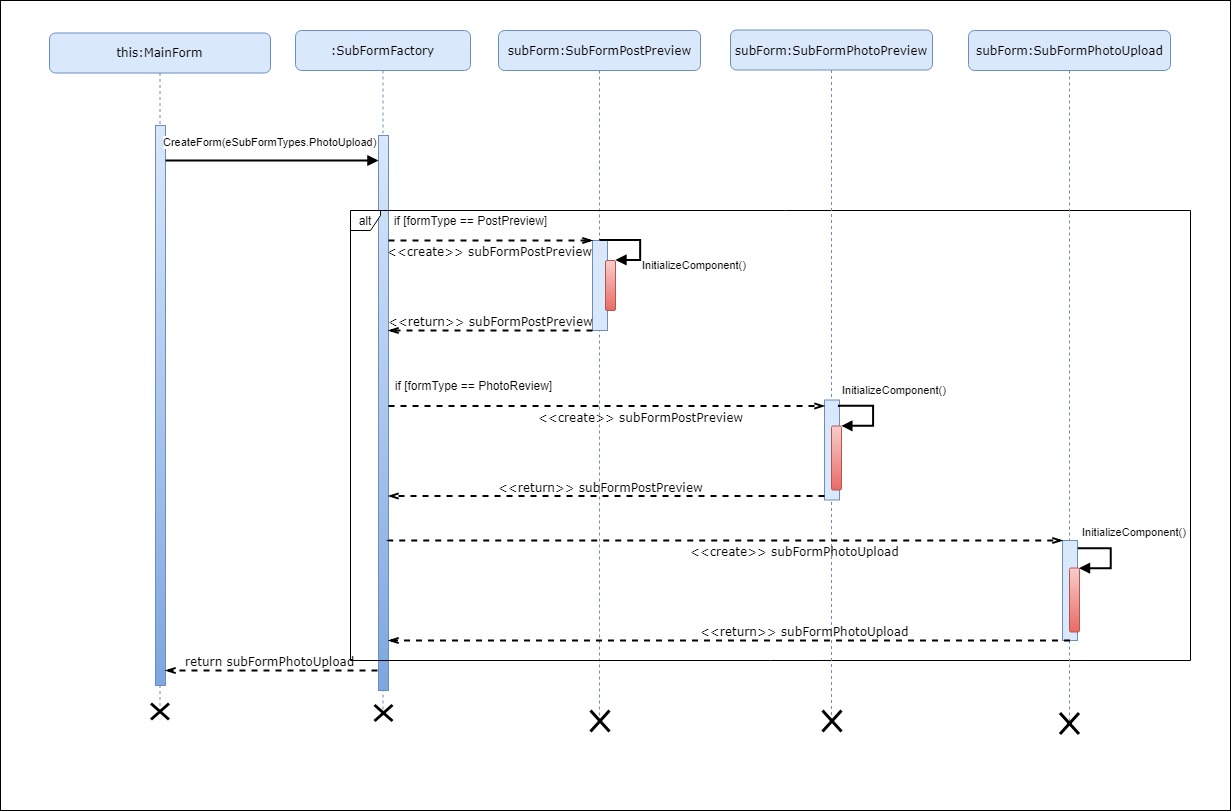
* Tagged Places – מפה שעל גביה מוצגים מיקומים של המשתמש על פי סוג המידע המחובר למיקום.
* Top Liked Photo Feature – פרמידה של 6 תמונות שזכו למירב ה"לייקים" של המשתמש, ורשימת 5 החברים המפרגנים ביותר ב"לייקים".

### תבנית מס' 1 – Factory Method

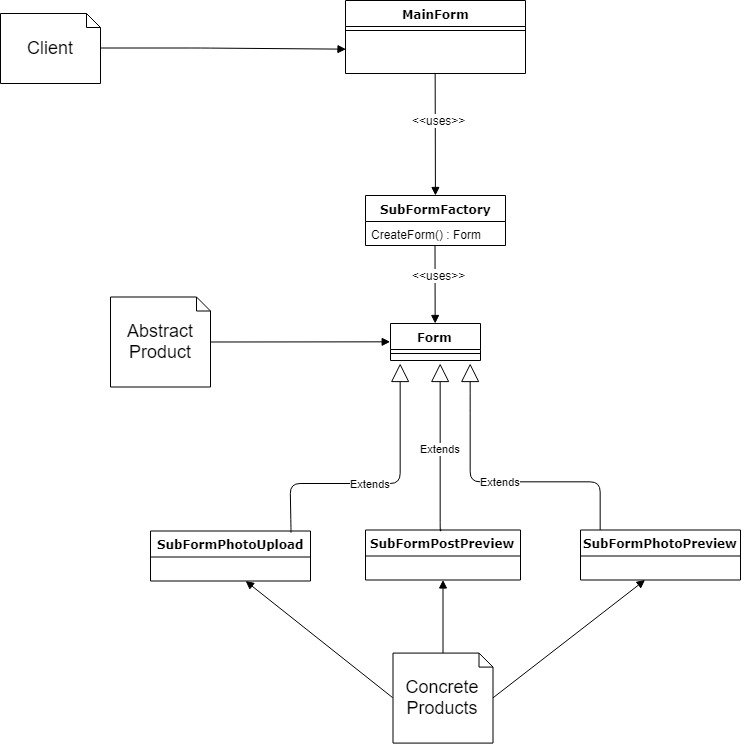
* סיבת הבחירה / שימוש בתבנית

השימוש בתבנית זו נעשה על ידי ניהול יצירת טפסים (Forms) נוספים במערכת שכן הם מהווים משפחה פולימורפית. בחרנו בתבנית זאת כדי למנוע מצב בו חלק יצירת טפסים במערכת מפוזר וחוזר על עצמו בקוד של ה-Client. מכאן שאם בעתיד נרצה להוסיף טפסים נוספים במערכת, נוכל להשתמש בתבנית זאת בהתאם ל-Context מסויים.

* אופן המימוש:
  + יצרנו את המחלקה "SubFormFacroty" אשר מכילה Enum שמציין את סוגי הטפסים שניתן לייצר להם מופע. כמו כן במחלקה קיימת המטודה הסטטית "CreateForm".
  + המטודה הסטטית "CreateForm" מקבלת בתור ה-"Context" את סוג הטופס אותו המערכת רוצה ליצור, והיא מחזירה מופע חדש של הטופס המבוקש.
  + המחלקה "SubFormFacroty" משמשת כ-Factory Method, והמחלקות "SubFormPhotoPreview", "SubFormPhotoUpload" ו-"SubFormPostPreview" הינן הטפסים שניתן ליצור להן מופע באמצעות ה-Factory Method.
* Sequence Diagram

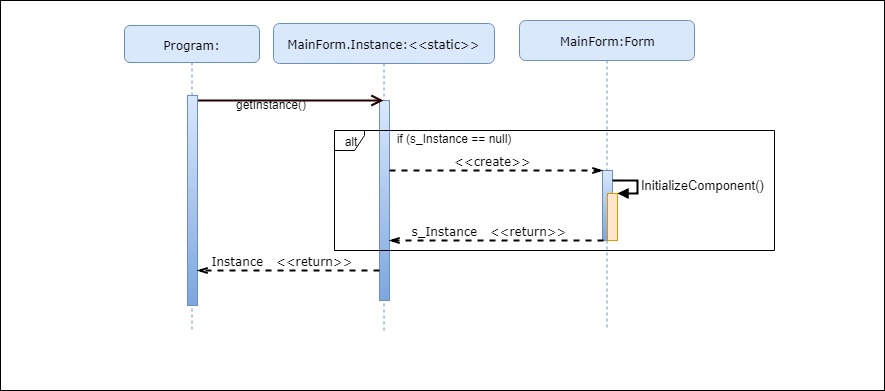


* Class Diagram

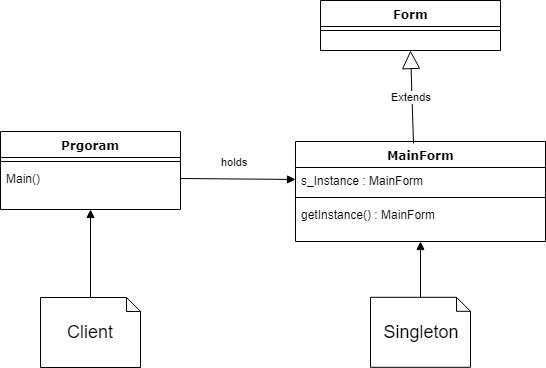


### תבנית מס' 2 – Singelton

* סיבת הבחירה / שימוש בתבנית: ממשנו את המחלקה "MainForm", אשר משמשת כטופס הראשי, כ-Singleton. הסיבה לבחירה היא שבמערכת זו קיים טופס אחד בלבד שתפקוד המערכת נקבע לפיו. כמו כן יכול לקרות מצב בו ניתן ליצור מספר טפסים (Forms), דבר שעלול לגרום נזק לתפקוד ולשימוש במערכת.
* אופן המימוש:
  + המחלקה "MainForm" כעת מוגדרת כ-"sealed", זאת על מנת שלא יהיה ניתן לרשת ממנה (אם יהיה ניתן לרשת ממחלקה זו, אזי יהיה ניתן ליצור object שהוא "MainForm").
  + ה-"constractor" מוגדר כעת כ-"private", זאת על מנת שלא יהיה ניתן ליצור מופע של ה-MainForm מחוץ למחלקה.
  + הוספת "property" (s\_Instance) שמוגדר כ-"static" ואתחולו כ-"null".
  + הוספת "method" (Instance) שמוגדרת כ-"static", אשר מחזירה את המופע של המחלקה. מתבצעת בדיקה האם התבצע אתחול של המחלקה, אם כן, מוחזר המופע של המחלקה שכבר נוצר טרם הקריאה, אחרת נבדוק ע"י s\_LockObj אם מישהו נוסף מנסה ליצור אובייקט ואז ניצור מופע חדש של המחלקה באמצעות ה-"constractor" הprivate של המחלקה, שהגישה אליו ניתנת אך ורק מתוך המחלקה.
* Sequence Diagram



* Class Diagram

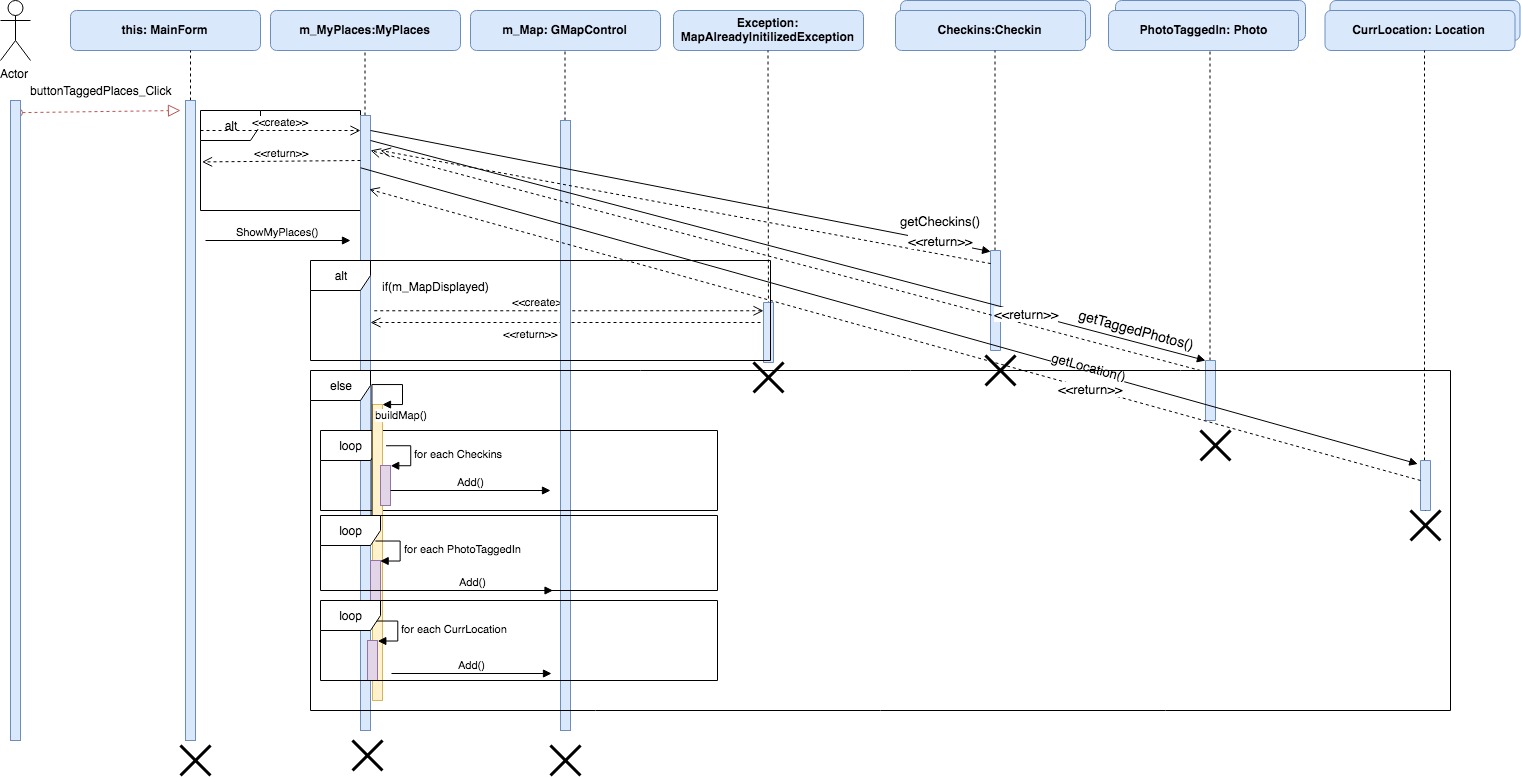


### תבנית מס' 3 – Façade

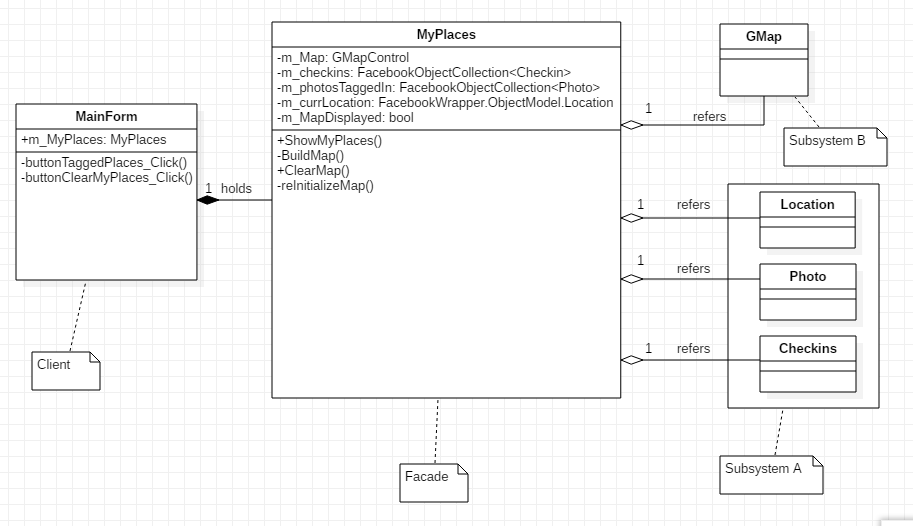
* סיבת הבחירה / שימוש בתבנית:

הסיבה שבחרנו להשתמש ב-facade היא משום שרצינו לפשט את המערכת בין המשתמש לממשק, כך שאם בעתיד נרצה לשכלל את הפונקציונליות של הפיצ'ר, היינו מתקשים יותר לעשות זאת כשהקוד (לוגיקה) מעורבב בתוך ה-MainForm. בנוסף, אם בעתיד נרצה לעשות reuse לפיצ'ר שלנו, זה לא היה אפשרי כשכל הקוד מעורבב ב-MainForm. ואילו כעת אפשר לעשות זאת בלחיצת כפתור. בנוסף, כשכל הלוגיקה של הפיצ'ר שלנו היתה כתובה ב-MainForm זה מקשה על הקריאות של הקוד שלנו.

* אופן המימוש:
  + יצרנו מחלקה חדשה MyPlaces שה-MainForm מאתחל (lazy) ומחזיק.
  + MyPlaces מחזיקה ref ל-GMapControl ול-Checkins, PhotosTaggedIn, Location
  + העברנו את כל המתודות הלוגיות למחלקה החדשה והצענו ממשק פשוט מאד עבור ה-MainForm – ShowMyPlaces(), CleanMap().
  + יצרנו מחלקת Exception חדשה: (MapAlreadyInitializedException) שה-MyPlaces זורקת ל-MainForm במידה והמפה מלאה ובכך יצרנו הפרדה מלאה יותר בין ה-UI ללוגיקה.
* Sequence Diagram



* Class Diagram



### עבודה אסינכרונית

במערכת ביצענו שימוש בעבודה אסינכרונית ע"י שימוש ב-threads נוספים אשר כל אחד מהם מפעיל טופס משני (תצוגה מקדימה ומפורטת של פוסטים, תצוגה מוגדלת של תמונות והעלאת תמונה חדשה לאלבום).

העבודה האסינכרונית במערכת נמצאת במחלקה "MainForm", בה באמצעות threads נעשת הפעלה של הטפסים המשניים, וכמו כן במחלקה "SubFormPhotoUpload" על מנת להפעיל File Dialog. השימוש בעבודה אסינכרונית מעניק ממשק עשיר יותר לשימוש במערכת ויכולות נוספות עבור המשתמשים.

### Data Binding

במערכת ביצענו שימוש ב-Data Binding בהצגת הפרטים של המשתמש אשר מחובר למערכת. הצגת פרטי המשתמש נמצאת בטופס הראשי של המערכת במחלקה "MainForm".

באמצעות השימוש ב-Data Binding ניתן לאגד את הנתונים המוצגים על המשתמש ביחידת לוגית דינמית אשר מתעדכנת במידה ומשתנים פרטים מסויימים.