### 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 –

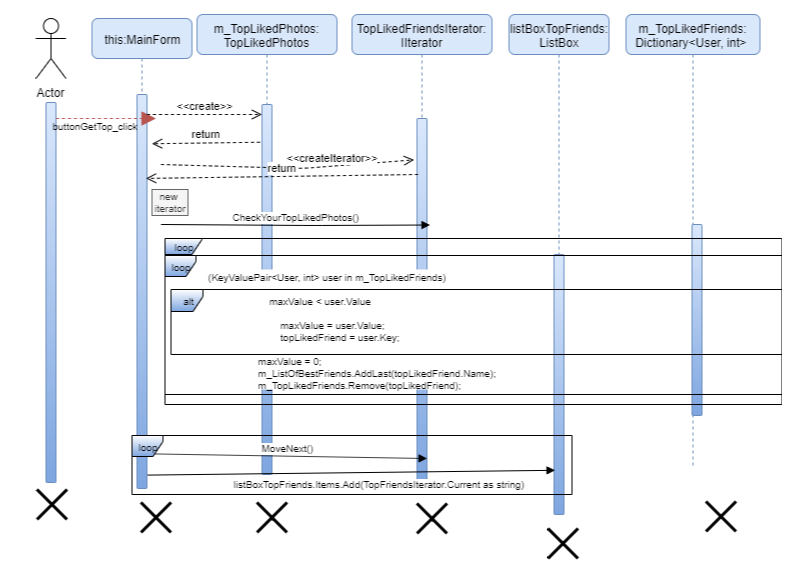
### תבנית מס' 2 – Iterator

* סיבת הבחירה / שימוש בתבנית:

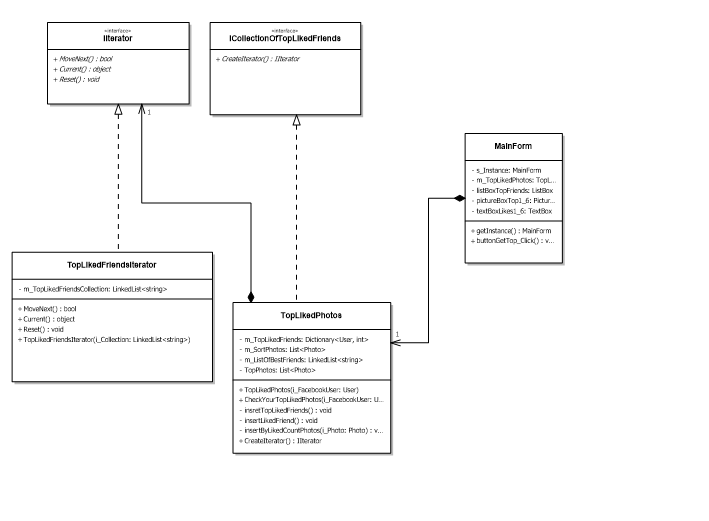
בחרנו להשתמש בתבנית זו במימוש הפיצ׳ר Top Liked Photos על מנת לספק ממשק למעבר על אוסף פריטים (החברים "המפרגנים ביותר" של משתמש) אשר נמצא באובייקט שמאגד אותו (המחלקה שמממשת את הפיצ׳ר TopLikedPhotos), ללא תלות במבנה הנתונים שמשמש לשמירתם.

בעזרת מימוש תבנית זו, אם בעתיד נרצה להחליף את מבנה הנתונים שבו נשמרת רשימת החברים הטובים, נוכל לעשות זאת על ידי שינוי מבנה הנתונים בלבד וללא שום שינוי נוסף בקוד שנכתב קודם לכן.

* אופן המימוש:
  + בנינו ממשק IIterator בעל שלוש חתימות:
    - bool MoveNext() – האם ניתן להתקדם במעבר על מבנה הנתונים, תוך התקדמות על מבנה הנתונים.
    - object Current { get; } – מחזיר את הפריט הנוכחי תוך כדי המעבר על מבנה הנתונים.
    - Void Reset() – איפוס המעבר על מבנה הנתונים.
  + בנינו את המחלקה TopLikedFriendsIterator שמממשת את הממשק IIterator ומכילה Data Member שמשמש כמבנה הנתונים שמכיל את רשימת החברים המפרגנים ביותר במסגרת הפיצ׳ר.
  + בנינו ממשק ICollectionOfTopLikedPhotos בעל חתימה של מתודה אחת IIterator CreateIterator().
  + המחלקה TopLikedPhotos שמממשת את הפיצ׳ר, מממשת את הממשק ICollectionOfTopLikedPhotos, כלומר מאתחלת מופע חדש של האובייקט TopLikedFriendsIterator אשר מאפשר לבצע מעבר על מבנה הנתונים שמכיל את רשימת החברים המפרגנים של המשתמש (שקשורה לפיצ׳ר (TopLikedPhotos , ללא תלות במבנה הנתונים שנבחר.
* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – Strategy

* סיבת הבחירה / שימוש בתבנית:

בחרנו להשתמש ב-Strategy במימוש הפיצ'ר Tagged Places שלנו, שכעת נרצה להציג את המקומות בהם היה המשתמש לפי סנן מסויים.

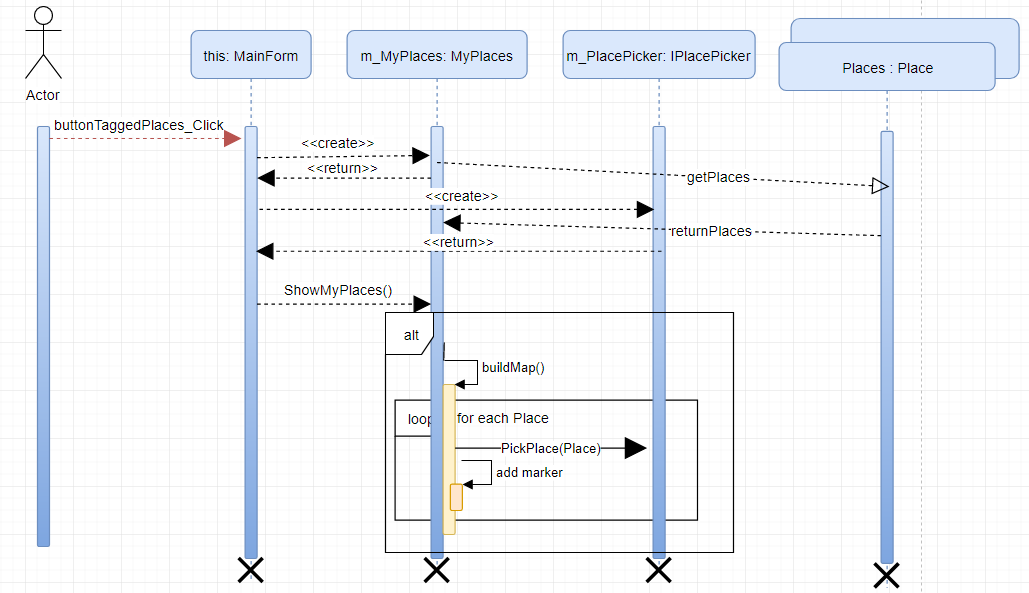
את התבנית הכנסנו לשלב שבו אנחנו סורקים כל מקום שבו המשתמש ביקר (לפי הבחירה שלו), ואז לפי הבחירה שלו בlistBox () אנו מסננים את המקומות בהתאם. כעת, אם בעתיד נרצה להחליף את הדרך בה אנחנו מסננים את המקומות בהם ביקר או להוסיף דרך סינון חדשה (למשל, נרצה להראות את המקומות בהם היה בעיר מסויימת או רחוב מסויים), נוכל לממש IPlacePicker ולבחור האם המקום יבחר בהתאם לפרמטרים של המקום בו היה המשתמש.

* אופן המימוש:
  + בנינו ממשק IPlacePicker בעל חתימה של מטודה אחת

bool pickPlace(Page i\_place).

* + בנינו עוד 4 מחלקות שממשות את הממשק הנ"ל:
    - PlacePickerAll – לוקח את כל המקומות בהם היה המשתמש.

(זה במקרה והמשתמש לא מעוניין לסנן את המקומות)

* + - PlacePickerFromIsrael – רק מקומות בישראל בהם היה המשתמש.
    - PlacePickerFromJerusalem - רק מקומות בירושלים בהם היה המשתמש.
    - PlacePickerFromTelAviv - רק מקומות בתל אביב בהם היה המשתמש.
  + הממשק משמש Data Member במחלקת MyPlaces שהיא ה- façade של הפיצ'ר.
  + בשלב הסריקה של הפוסטים, מתבצעת קריאה למטודה pickPlace אשר מחזירה ערך בוליאני אם לקחת את המקום הזה או לא.
* Sequence Diagram
* Class Diagram

