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

* הצגת פעילות פוסטים – מציג למשתמש כמה זמן עבר מאז העלה פוסט, העלה תמונה או העלה סירטון.
* הצגת רשימה של חברים על פי שני פרמטרים: חודש או שנת לידה, ומין. מציג למשתמש רשימה של חברים על פי הפרמטרים האלו, והיוזר יכול לבחור איזה
* פיצ'ר חדש לתרגיל הזה – הצגת החברים שעשו לייק לסטטוסים שלי עם פילטר.

### תבנית מס' 1 – [שם תבנית]

* סיבת הבחירה / שימוש בתבנית:
* אופן המימוש:
* Sequence Diagram
* Class Diagram

### תבנית מס' 2 – Chain of Responsibility

* סיבת הבחירה / שימוש בתבנית:

על מנת לממש פילטר על רשימת משתמשים (חברים שעשו לייק) שמורכב ממספר תנאים, היה צורך בפתרון שלא מתייחס למספר הקומבינציות האפשריות של התנאים ועובד בצורה שיטתית, תנאי אחר תנאי בכל פעם.

* אופן המימוש:

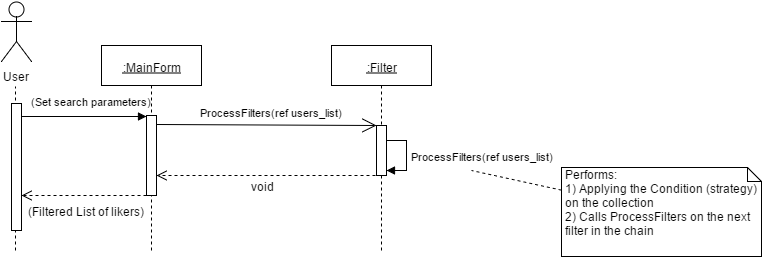
המחלקה Filter ממומשת בקובץ Filter.cs.

את המחלקה מימשנו בצורה גנרית כך שאין דרישה לאובייקט או קולקציה ספציפית.

Filter – מכילה תנאי סינון על הקולקציה ואת הפילטר הבא בשרשרת.

בעת הפעלה של פילטר זה (אם בידי המשתמש או ע"י פילטר אחר) תסונן הקולקציה שהתקבלה כקלט ע"י תנאי הסינון ויקרא הפילטר הבא עם הרשימה המסוננת.

* Sequence Diagram -



* Class Diagram -

Red – תפקידי ה-CoR

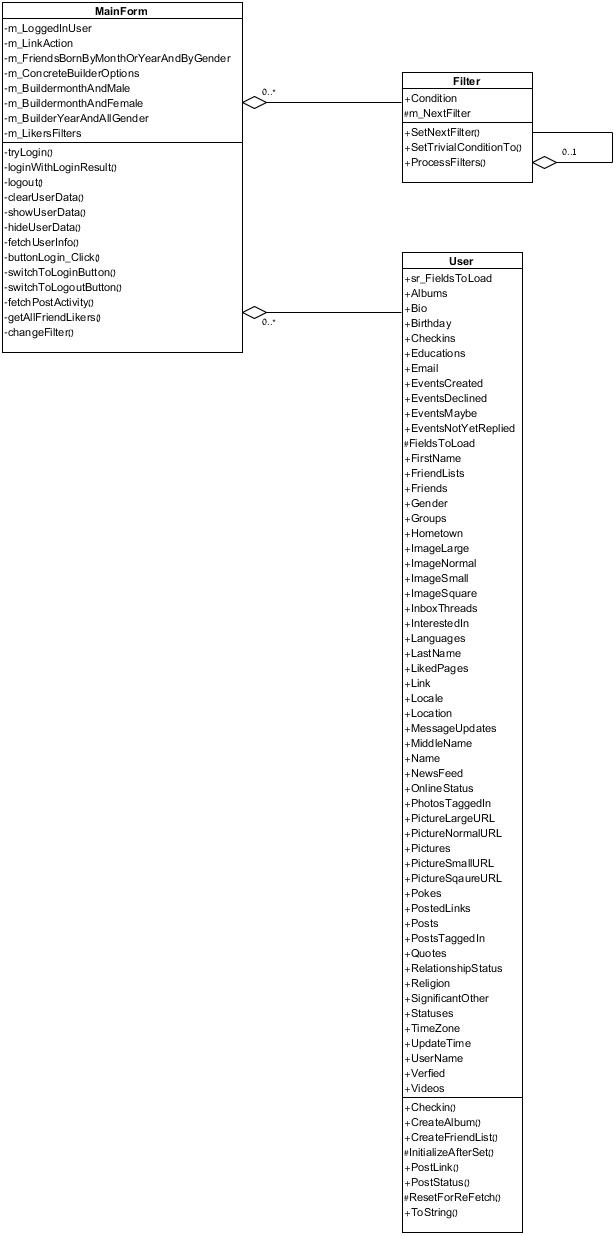
Green – תפקידי ה-Strategy

Client

Context

Handler / Concrete Handler

Strategy



### תבנית מס' 3 – Strategy

* סיבת הבחירה / שימוש בתבנית:

כל פילטר מכיל אסטרטגיית סינון ייחודית משל עצמו. כיוון שחוץ מאסטרטגיית הסינון, כל הפילטרים זהים בהתנהגותם, נידרש פתרון שימנע את שכפול הקוד הפוטנציאלי.

* אופן המימוש:

המימוש נמצא במחלקה Filter בקובץ Filter.cs.

בנוסף, המימוש עושה שימוש ביכולת של C# להחזיק מצביע לפונקציה (Delegate) ובכך אנו נחזיק את אסטרטגיית הסינון כמשתנה בתוך הפילטר במקום ליצור מחלקה שתירש מ-Filter ותממש את האסטרטגיה שם. בנוסף, דרך מימוש זו מאפשרת לנו לשנות את האסטרטגיה של פילטר מסוים בזמן ריצה (יקרה כאשר נשנה את הקריטריונים של החיפוש).

השדה Condition הוא שדה האסטרטגיה.

* דיאגרמות:

כיוון שהתבנית מוכלת במחלקה Filter, גם הדיארגמות מוכלות באופן דומה.

עם זאת, ציינו את התפקידים השונים ב-Class Diagram בצבעים שונים בהתאם לתבנית.