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

* cityFriends – מביא את רשימת הערים של החברים וכמות החברים שלך שגרים בכל אחד מהם
* unfriendFriends – מביא את כל החברים שלך שאתה צריך להפסיק חברות איתם שאוהבים את "ניקלבק"

### תבנית מס' 1 – Template Method

* סיבת הבחירה / שימוש בתבנית:

הפעולה של סינון חברים של משתמש לפי תכונה מסויימת יכולה לחזור על עצמה (אם בהמשך לדוגמה נרצה להוסיף פילטר שמסנן חברים לפי המזל שלהם) לכן כדי להימנע משיכפול קוד בעתיד השתמשנו בtemplate method ככה שאם בהמשך נרצה להוסיף פיצ'ר כזה כל מה שיהיה עלינו לעשות זה לממש את מטודת הFilter

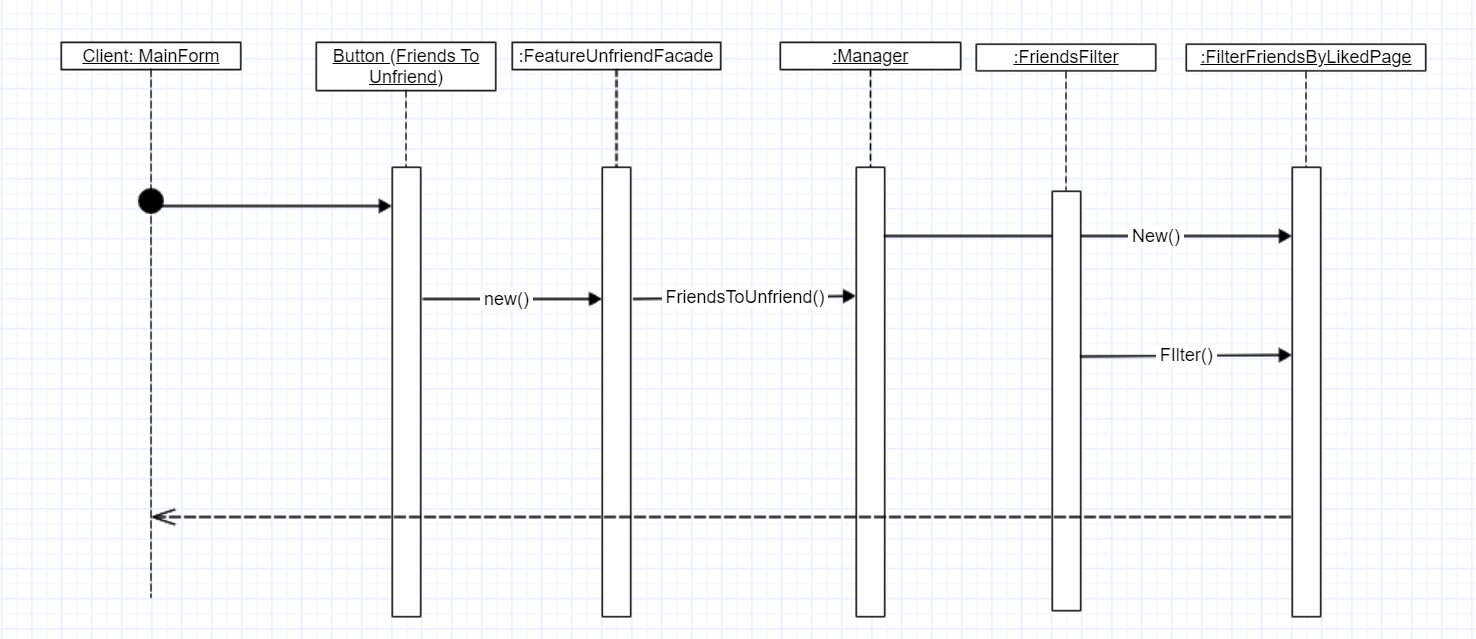
* אופן המימוש:

המחלקותFriendsFilter, FilterFriendsByLikedPage נמצאות בתיקיית LOGIC

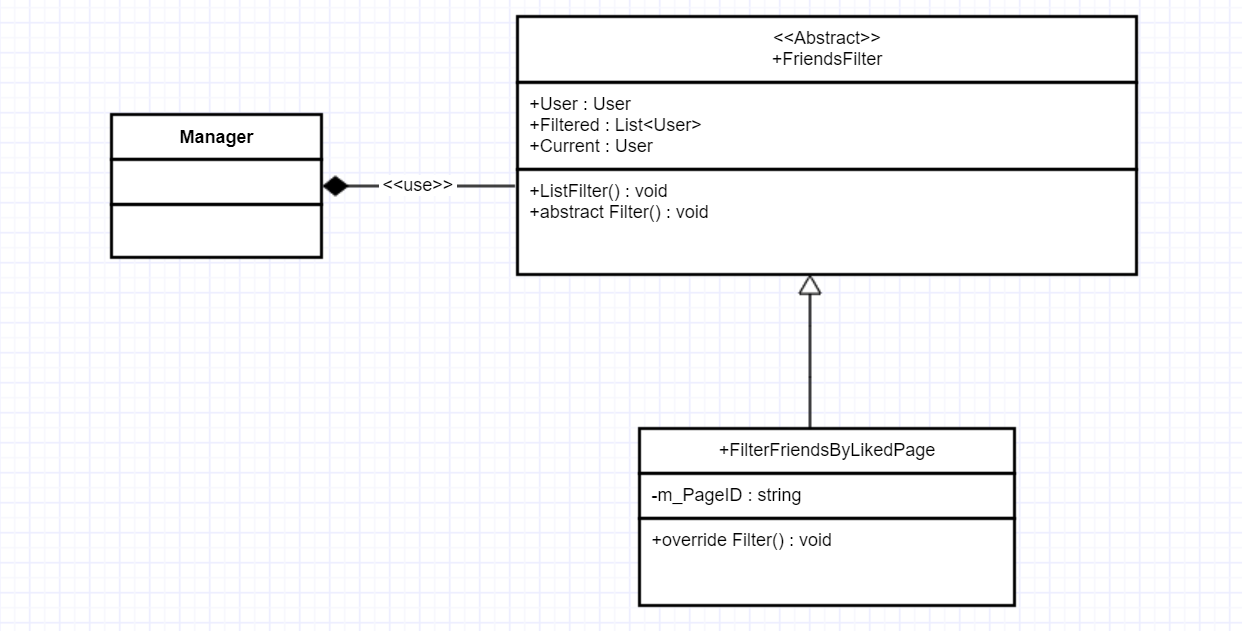
FriendsFilter היא מחלקה אבסטרקטית לסינון חברים לפי תכונה מסויימת.

FilterFriendsByLikedPage יורשת ממנה יורשת מ FriendsFilterוממשת את Filter()

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – Observer

* סיבת הבחירה / שימוש בתבנית:

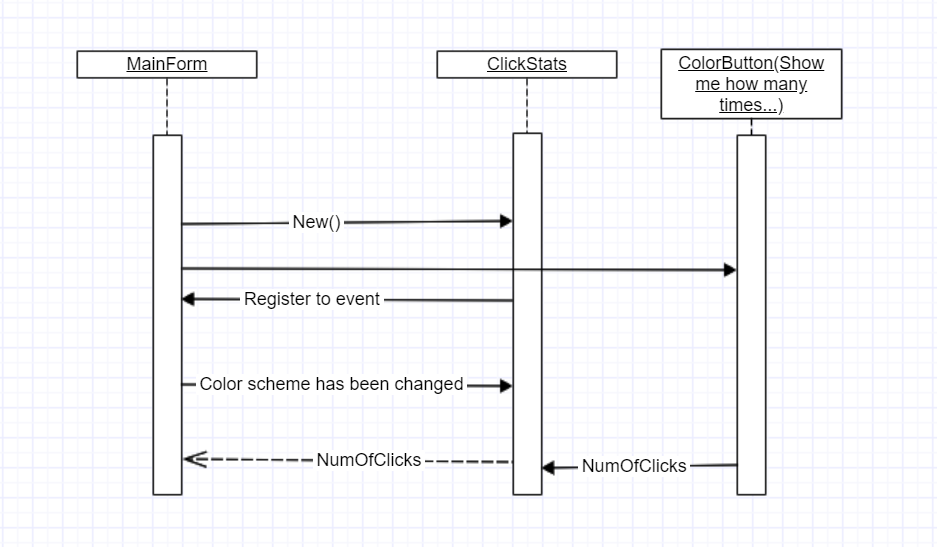
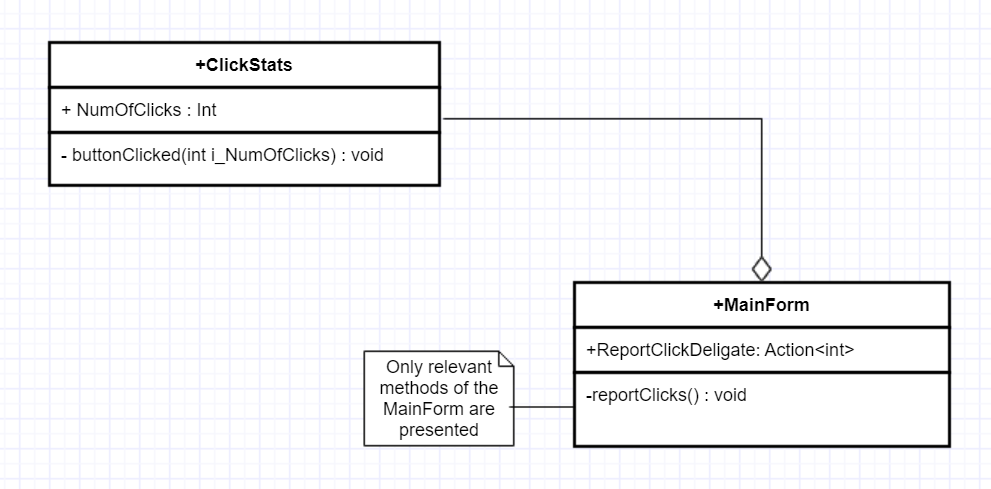
הוספתנו פיצ'ר חדש, שברגע שלוחצים על הכפתור Show me how many times color scheme has been changed, text box מציג את סך השינויים. הייתי צריך מחלקה חיצונית שתאזין (Observe) למה שקורה במחלקה אחרת (MainForm).

* אופן המימוש:

על מנת לדמות מצב של Observer ועל מנת שתהיה הפרדה נכונה של מחלקות, יצרתי מחלקה נוספת בשם ClickStats (Logic) שהיא ה –Observer למחלקה של MainForm.

כתבתי מטודוה כ subscribe ב delegate שיצרתי ב MainForm.

ברגע שיש לחיצה על כפתורי הColorScheme, האירוע מופעל.

* Sequence Diagram
* Class Diagram
* 

### תבנית מס' 3 – Command

* סיבת הבחירה / שימוש בתבנית:

רצינו להוסיף אפשרויות לגבי סידור הערים בהפעלת הפיצ'ר של קיבוץ החברים ע"י הערים, אפשרות כזו שנוכל להציג את החברים ל]י הערים בסדר עולה או יורד לדוגמה, ואם בעתיד נרצה להוסיף אפשרות שתציג לנו את החברים לפי הערים סדר אחר כפי שנרצה נוכל לעשות זאת מבלי לשנות קוד קיים

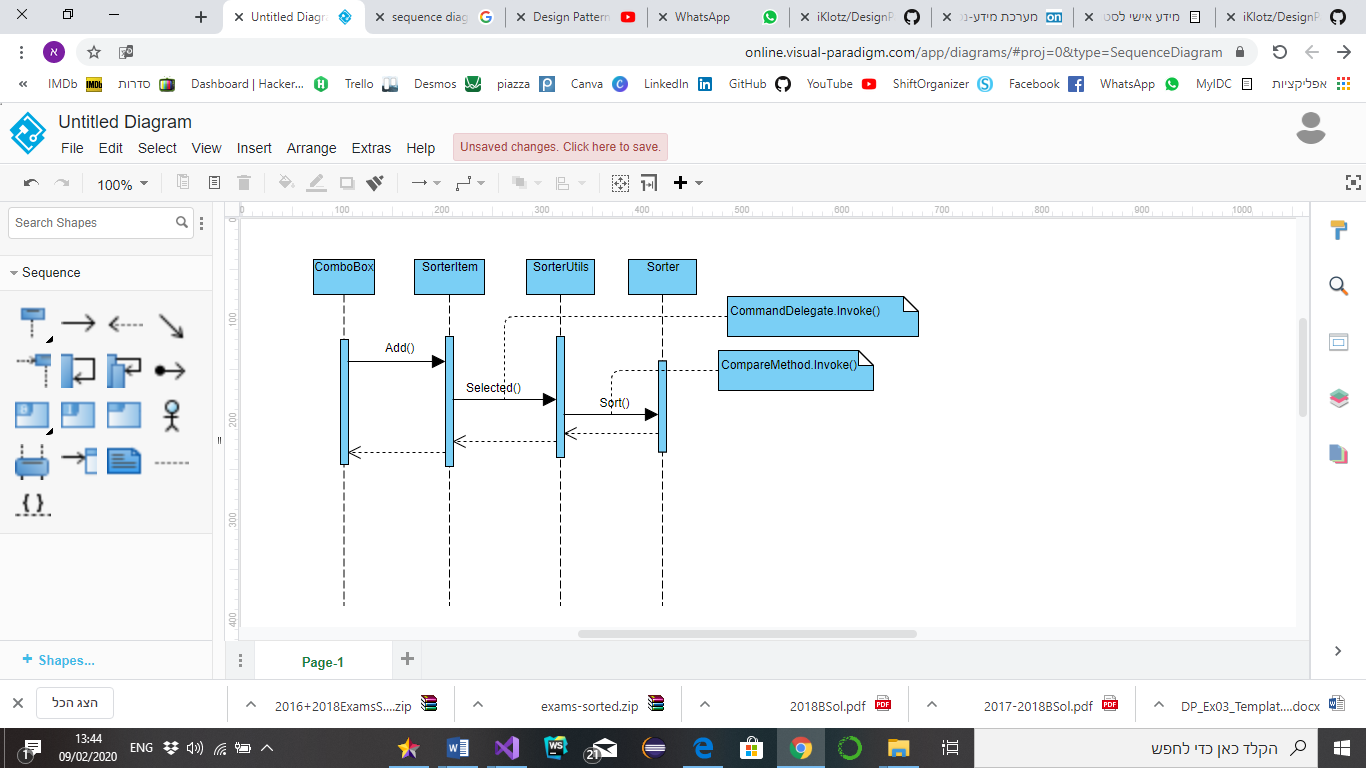
אופן השימוש הוא כזה שכאשר נלחץ על אפשרות בFORM שלנו לפי סדר עולה/ יורד הוא יציג לנו את החברים בהתאם.

* אופן המימוש:

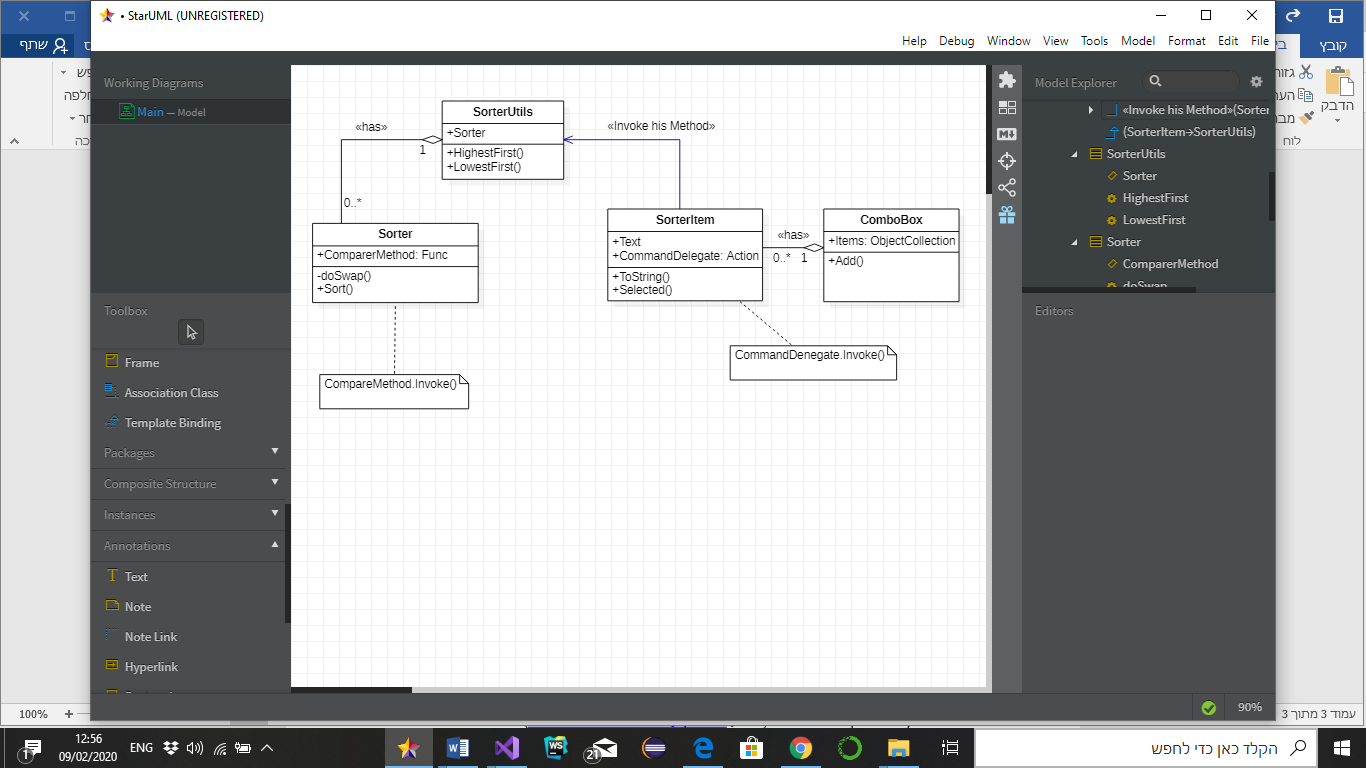
הוספנו מחלקןת בשם SorterItem,Sorter , SortUtils שמרו member מסוג הSortUtils שהוא מחבר בין כל SortItem שכזה (עולה/יורד) לבין המפעיל Sorter שמבצע את תהליך המיון לפי איזה סדר שנבחר בעזרת הDelegate ששלחנו.

אם בעתיד נרצה להוסיף עוד אפשרות כזו נוכל לעשות את זה בהוספת שורה אחת בלבד כמו שעשינו בשורה 243 בMainForm, נוסיף SorterItem נוסף עם איזה טקסט שנרצה ונשלח לו Action שיפעיל את הSorter ששלחנו, סוג המיון עצמו.

* Sequence Diagram



* Class Diagram



הComboBox(Menu) מוסיף לItems שלו אובייקטים מסוג SortItem(UIItems) שכאשר בוחרים בו ספציפית מפעיל את Selected() שמפעיל את הCommandDelegate שלו, מתוך הSorterUtil(Client) שמפעיל את הSorter שלו בעזרת פעולת Sort

(ניתן להוסיף עוד מתודות שכאלה ע"י הוספת SorterItem ולתת לו ל CommandDelegate את המתודה שנרצה בתור Sorter)

### תבנית מס' 4 – Strategy

* סיבת הבחירה / שימוש בתבנית:

רצינו שנוכל לשנות את ההתנהגות של הSorter בלי לשנות קוד קיים, אז השתמשנו בFunc שכל מי שרוצה להשתמש בשירותיו של הSorter יספק לו איך הוא ירצה למיין את האיברים שלו, ואם בעתיד נרצה להוסיף אפשרות שונה לא נצטרך לשנות את הSorter אלא רק לספק מתודה שתתאים לFunc שהSorter מבקש

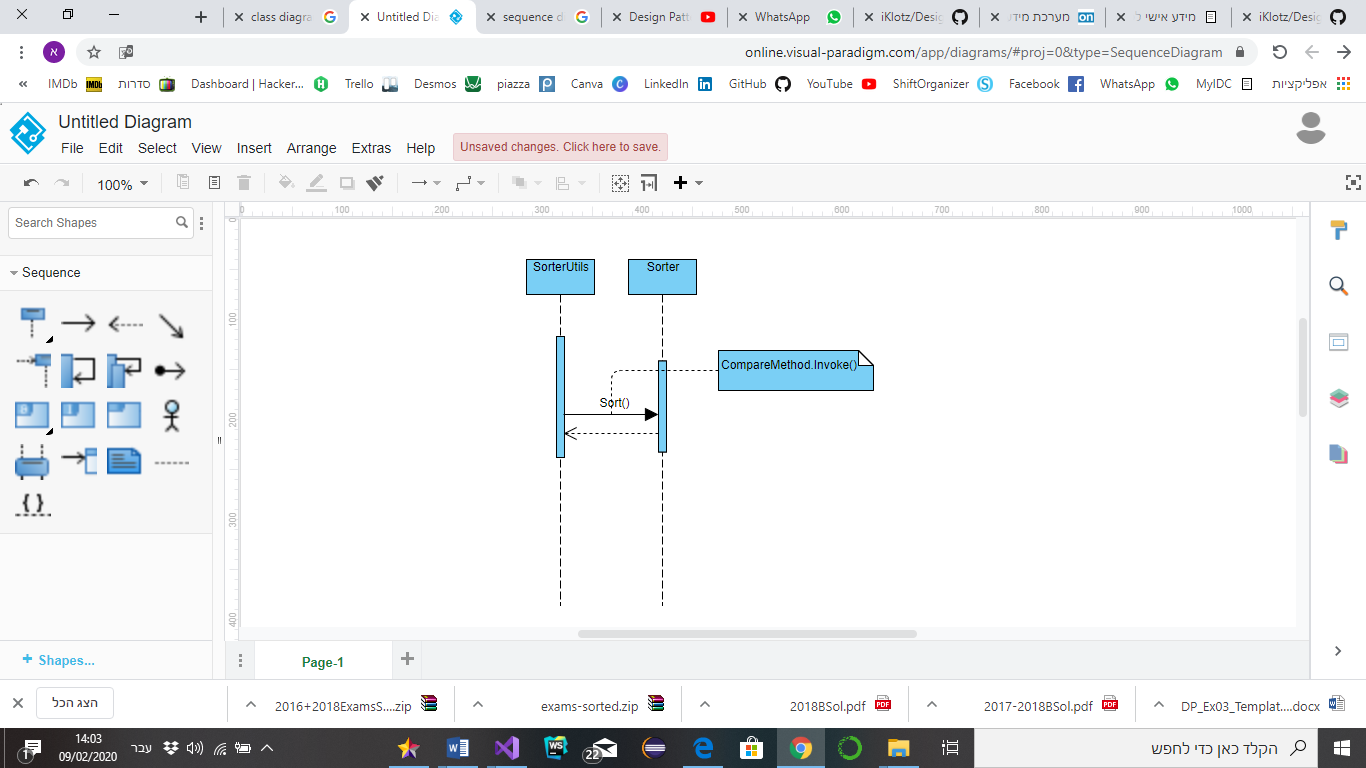
אופן השימוש הוא הכך שאנו רוצים למיין את החברים שלנו לפי הערים שלהם באיזהו סדר שנרצה.

* אופן המימוש:

הוספנו מחלקה Sorter שיש לה Property מסוג Func<int, int, bool> שמקבלת שני מספרים שלמים ומחזירה ערך בוליאני, הSorter ממיין את הקולקציה הנתונה שלנו ע"י המתודה שניתנה לנו בProperty.

הלקוח מחליט איזה מיון יהיה בעזרת הSorterUtils שמחזיק Sorter בתור Property, ואם בעתיד נרצה להוסיף מתודה אנו נוכל אם ורק אם היא מסוג שמקבלת שני מספרים שלמים ומחזירה ערך בוליאני

* Sequence Diagram



* Class Diagram

