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

* הפיכת תמונת הפרופיל לרצף של תווים המדמים את התמונה המקורית.
* בחירת מסלול מכל מוקדי הצ'אקאינים של המשתמש, ויצירת מסלול בגוגל מפס של המסלול הנבחר.

### תבנית מס' 1 – Strategy

* סיבת הבחירה / שימוש בתבנית:

כחלק של הרחבת האפליקציה, רצינו להוסיף פיצ'ר שנותן אפשרות למשתמש למיין את החברים המוצגים באפליקציה בעזרת אלגוריתמים שונים באחת מ- 4 הדרכים הבאות:

1. אלפביתי
2. גיל
3. תאריך בשנה
4. באופן רנדומלי

לשם כך השתמשנו בpattern design – Strategy מהסיבות הבאות:

1. מניעת שכפול קוד – מכיוון שביצענו קומפוזיציה למחלקה שמבצעת את המיון, נוכל להשתמש בה בעתיד גם בחלקים אחרים בקוד. ובנוסף תחזוק הקוד יהיה קל יותר בעתיד.
2. הכנסת injection points לקוד – דיזיין פטרן Strategy מאפשר הכנסת injection points. כך שגם מימוש ארבעת האלגוריתמים בוצע באופן מינימאלי (כמות קוד).

וגם בעתיד נוכל להוסיף עוד אלגוריתמי מימוש בקלות רבה.

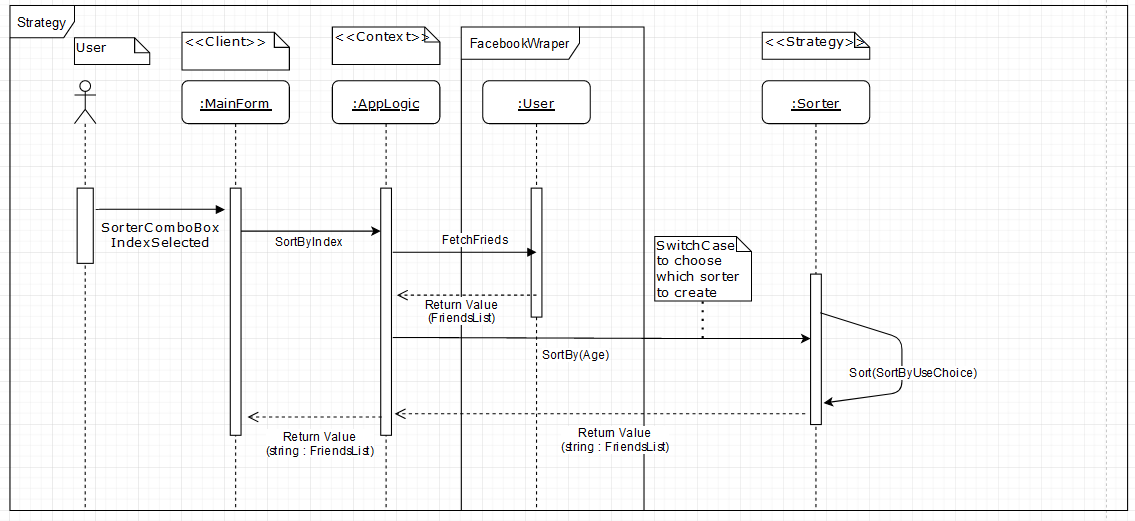
* אופן המימוש:

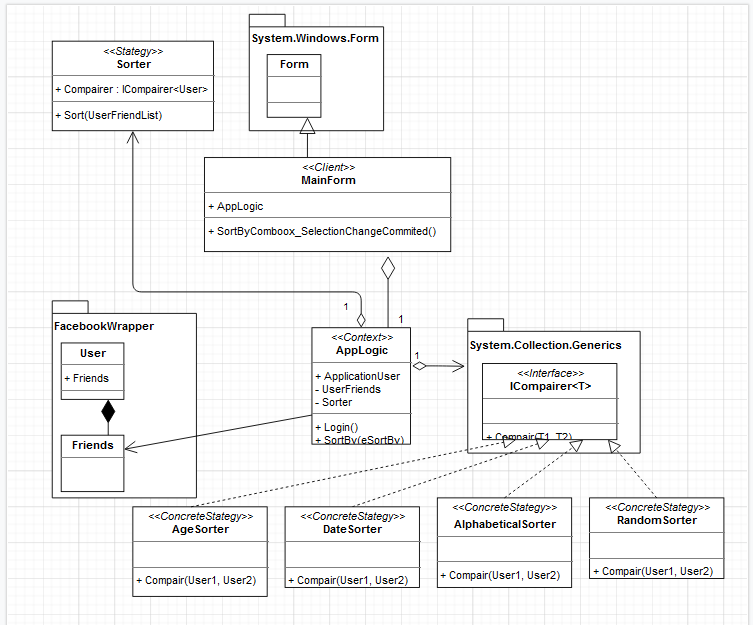
יצרנו מחקלת אב שנקראת Sorter , מקבלת כפרמט IComparer<User> ולפיו הוא ממש את המיון. בנוסף כל מחלקה שממשת את IComparer<User> מחליטה על אופן המיון.

כאשר המשתמש מגולל ובוחר את הcomboBox של אופן מיון החברים, נשלח לחלק הלוגי של האפליקציה את סוג המיון הרצוי, ומייצרים מופע חדש של ה-Sorter (שלנו) שהוא מממש את בחירת המשתמש. ששם מתבצע המיון באופן הרצוי והתוצאה החוזרת שהיא רשימת החברים הממויינת לממשק המשתמש.

המשתתפים בתבנית:

1. Strategy   : Class Sorter
2. AppLogic: Context
3. AgeSorter, AlphabeticalSorter, DateSorter, RandomSorter : ConcreteStrategy

* Sequence Diagram
* Class Diagram



### תבנית מס' 2 – Observer

* סיבת הבחירה / שימוש בתבנית:

הבעיה שרצינו לפתור:

רצינו לשכלל את האפליציה כך שיהיה ניתן לשנות סוג יוזר

(Free user, Business user, Premium user) תוך כדי ריצת התוכנית. (ולא רק בתחילת התוכנית)

כיצד Observer פתר את הבעיות:

Observer פתר לנו את הבעיה בכך, שהפיצ'רים שנתנו הרשאות שונות למשתמש יהיו האובזרברים של סוג המשתמש, כך שהם יודעו בכל פעם שהמשתמש החליף את סוגו.

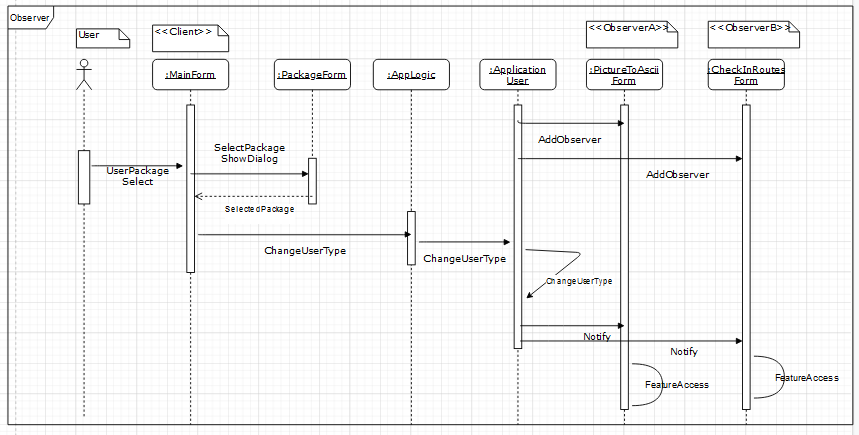
בעתיד אם נרצה להוסיף פיצ'רים חדשים, הם יוכלו להרשם לרשימת המאזינים של היוזר בקלות, דבר היקל על תחזוק הקוד בעתיד.

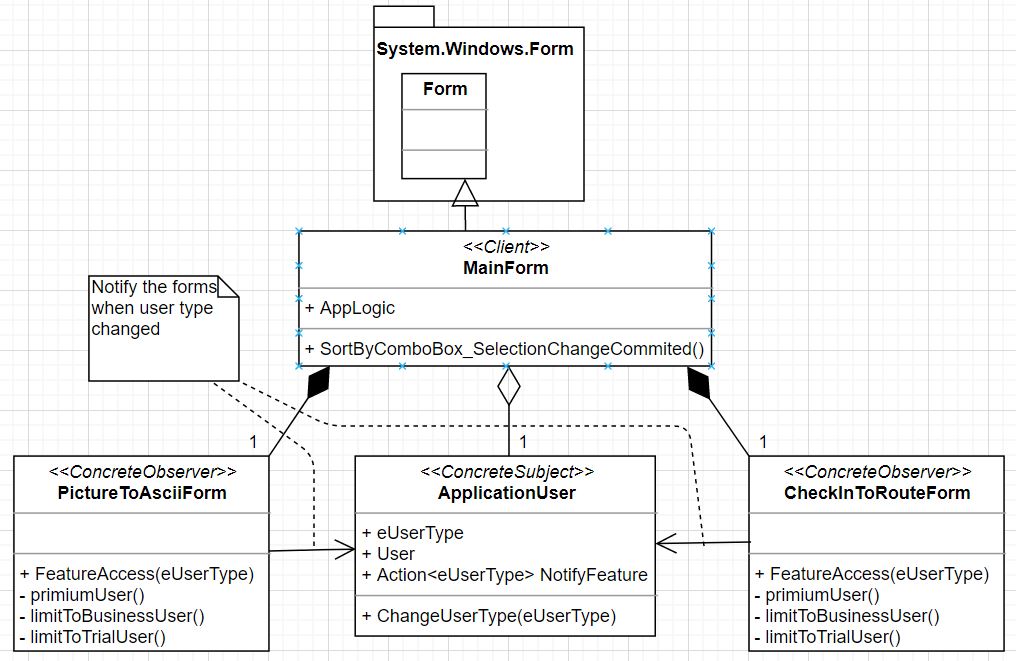
* אופן המימוש:

היוזר מחזיק Action<eUserType>, שבאיתחול התוכנית כל הפיצ'רים נרשמים לרשימת המאזינים. כאשר המשתמש בוחר או משנה את סוג חבילת השימוש, המשתמש מיידע את כל המאזינים אליו על בחירתו.

המשתתפים בתבנית:

1. ApplicationUser : ConcreteSubject
2. PictureToAsciiForm, CheckInToRouteForm: ConcreteObserver

* Sequence Diagram
* Class Diagram



### תבנית מס' 3 – Visitor

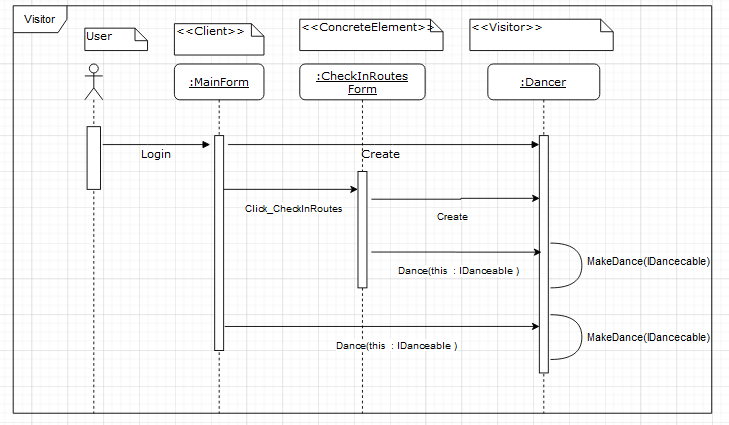
* סיבת הבחירה / שימוש בתבנית:

בהמשך הרחבת האפליקציה רצינו לגרום לכך שכל טופס יוכל "לרקוד", לשם כך היינו צריכים שמספר אובייקטים שונים יוכלו לבצע פעולה זהה. סיבת הבחירה העיקרית הייתה מניעת שכפול קוד כלומר שכל אחד המאובייקטים שנרצה ש"ירקוד" יחזיק אובייקט שיגרום לו לרקוד. ואם בעתיד נוסיף פיצ'רים שנרצה שגם הם "ירקדו" נוכל לממש זאת בקלות.

* אופן המימוש:

ישנו ממשק IDanceable שאובייקט צריך לממש על מנת "לרקוד" הממשק צריך לממש את התכונות: Location, BackColor, ובנוסף אותו אובייקט מחזיק Dancer שהוא מסוגל להרקיד IDanceable. כאשר רוצים להרקיד את האובייקט אז מבצעים את הפעולה של Dance.MakeDance(this).

המשתתפים בתבנית:

* Dancer : Visitor
* IDanceable : Element
* MainForm, CheckInToRouteForm, PictureToAsciiForm : ConcreteElements
* Sequence Diagram
* Class Diagram 