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

* קבלת רשימה של חברים שאהבו את אותו סרט\ספר\תוכנית טלוויזיה\זמר שאתה אוהב.

תחילה בוחרים קטגוריה מבין 4 הקטגוריות הבאות: סרטים, ספרים, תוכניות טלוויזיה ומוזיקה.

מתקבלת רשימה של כל הפריטים בקטגוריה שעשית להם לייק, למשל בקטגוריית סרטים מתקבלת רשימה של כל הסרטים שאהבת ולאחר בחירת סרט מהרשימה מתקבלת רשימת החברים שעשו לייק לאותו סרט*.*

* רשימה של כל החברים הרווקים עם הפרדה ל2 רשימות – בנים ובנות

ואפשרות לראות רשימת אירועים של חבר שנבחר.

* התווסף – פיצ'ר שמאפשר למשתמש לבחור קרטריון סינון, (גיל, עיר, מדינה או מין), ולאחר מכן מוצג למשתמש כלל חבריו (שאיפשרו הצגת מידע זה). המשתמש בוחר חבר מסויים מהרשימה ומוצג לו כל החברים האחרים שחולקים עם החבר הנבחר את אותו קריטריון.

### תבנית מס' 1 – Iterator

* סיבת הבחירה / שימוש בתבנית:

באפליקציה שלנו יש שימוש מהותי ברשימת האירועים של המשתמש.

בחרנו לעשות שימוש ב-Iterator מכיוון שרצינו שתהיה לנו שליטה נוחה על העצמים שאנחנו עוברים עליהם במבנה הנתונים שלנו (רשימת האירועים שמתקבלת). למשל: במידה ונרצה להוסיף איזשהו פרמטר נוסף לכל אירוע ברשימה או לעשות השמה נוכל לעשות זאת בקלות יותר עם ה-Iterator שלנו מאשר עם המימוש הרגיל. בנוסף יצרנו הפרדה בין ה-Client וההגדרה של האוסף וה-Iterator שלו.

* אופן המימוש:

המחלקות אותן יצרנו כדי לתמוך בתבנית זו הן:

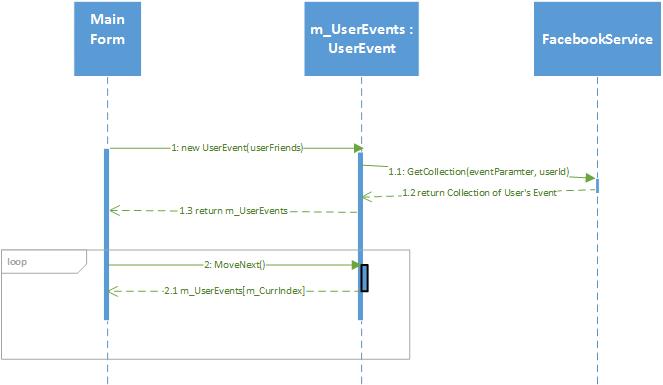
1. **UserEvent:**

לצורך מימוש ה-Iterator יצרנו מחלקה בשם UserEvent אשר תממש את IEnumerable ותחזיק רשימה לקריאה בלבד של Events.

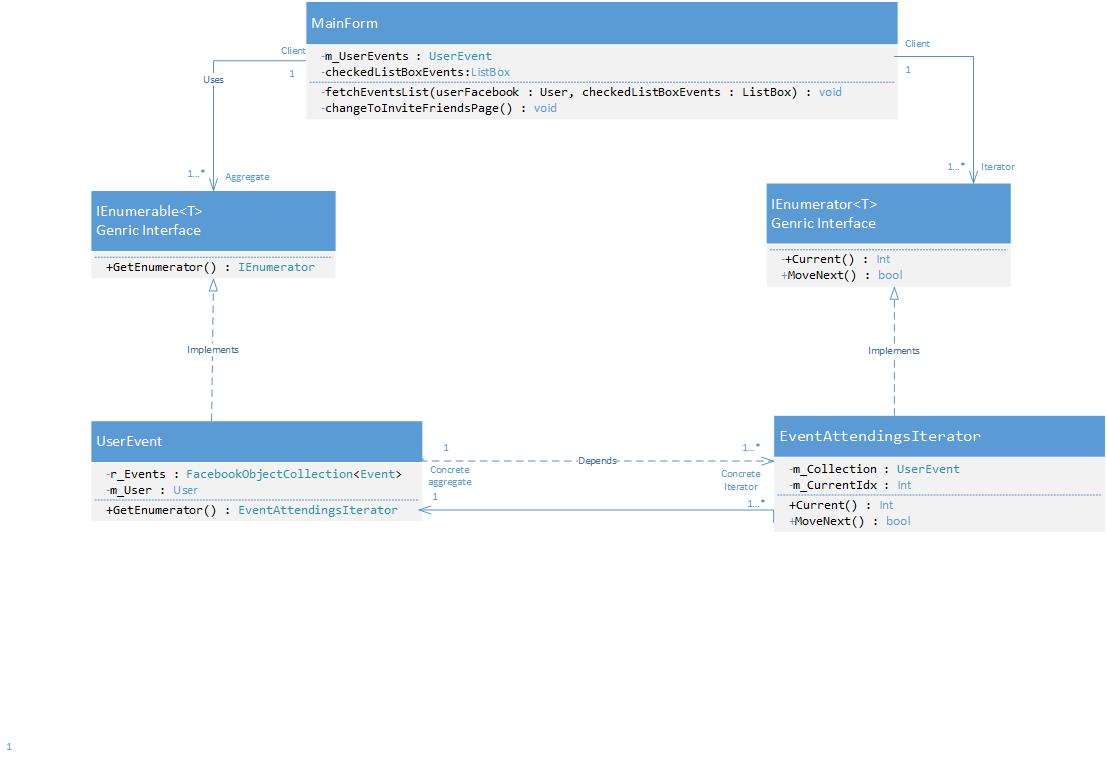
1. **EventAttendingsIterator:**

ה-Iterator הוא מחלקה מקוננת בתוך ה-UserEvent והיא מממשת את כל הפונקציונאליות הבסיסיות של האיטרטור (IEnumerator).

* Sequence Diagram:



* Class Diagram:

המקביל ל-client הוא כמובן ה-mainForm שלנו. בנוסף, בתפקיד ה-Aggregate משחק הממשק של IEnumerable, כאשר ה-UserEvent ממלא בתפקיד ה- ConcreteAggregate. ה-UserEvent מכיל מחלקה פנימית שהיא ה-EventAttendingIterator אשר ממלאת בתפקיד ConcreteIterator. ברור כי מחלקה זו מממשת את הממשק IEnumerator שהוא למעשה ה-iterator.

### תבנית מס' 2 – Strategy

* סיבת הבחירה / שימוש בתבנית:

הסיבה בגינה בחרנו להשתמש בתבנית זו נעוצה בעובדה שהפיצ'ר החדש שהוספנו מאפשר להשוות בין חברים לפי מספר קריטריונים. הקריטריונים נכון להרגע הם : מין (Gender), עיר (City), מדינה (Country) ו-גיל (Age). כאשר המשתמש בוחר קריטריון מיון, מוצג לו רשימה של חברים שסיפקו מידע זה למערכת, ומתוכם כאשר הוא יבחר אחד מהם, תוצג לו רשימה נוספת של כל החברים בעלי אותו מכנה כמו המשתמש שנבחר.

הסיבה שראינו לנכון לעשות שימוש בתבנית זו כי היא מאפשרת לנו להוסיף שיטות סינון חדשות בצורה מודולרית ונוחה לתחזוקה בעתיד (maintainability). אם נרצה להוסיף פרמטר חיפוש חדש, לא נצטרך לשנות ולסרבל את הקוד.

* אופן המימוש:

המחלקות אותן יצרנו / איתן השתמשנו כדי לתמוך בתבנית זו הן:

1. **IMatchFriendBehavior:**

זהו ה-interface שאותו כל התנהגות חיפוש חדשה תצטרך לממש. אם ממשק זה אנו נעבוד באפליקציה, כך שנוכל להחליף בקלות סוגי חיפוש שונים במידה והם מימשו את הממשק.

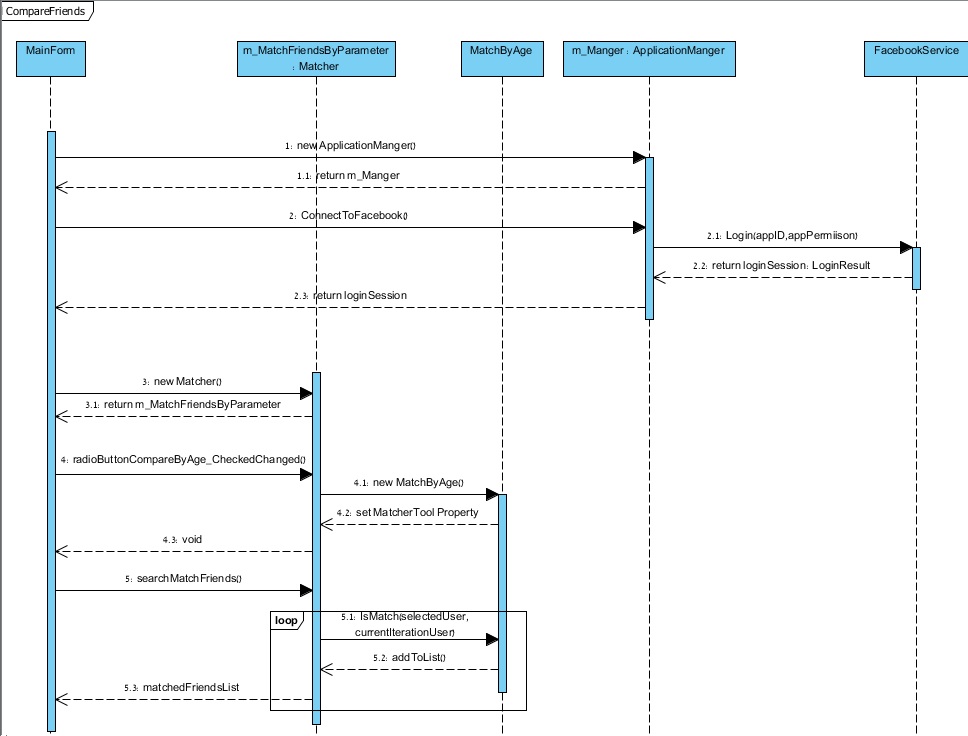
1. **Matcher:**

אובייקט זה מחזיק בתוכו התנהגות חיפוש (IMatchFriendBehavior) ומאפשר החלפה של התנהגות תוך כדי זמן ריצה. בנוסף, אובייקט זה מנצל את יכולות הממשק שמומש ומאפשר לבצע השוואה בין משתמש אחד לנגד שאר המשתמשים לפי אותו קריטריון חיפוש.

1. **MatchByGender, MatchByCountry, MatchByCity, MatchByAge:**

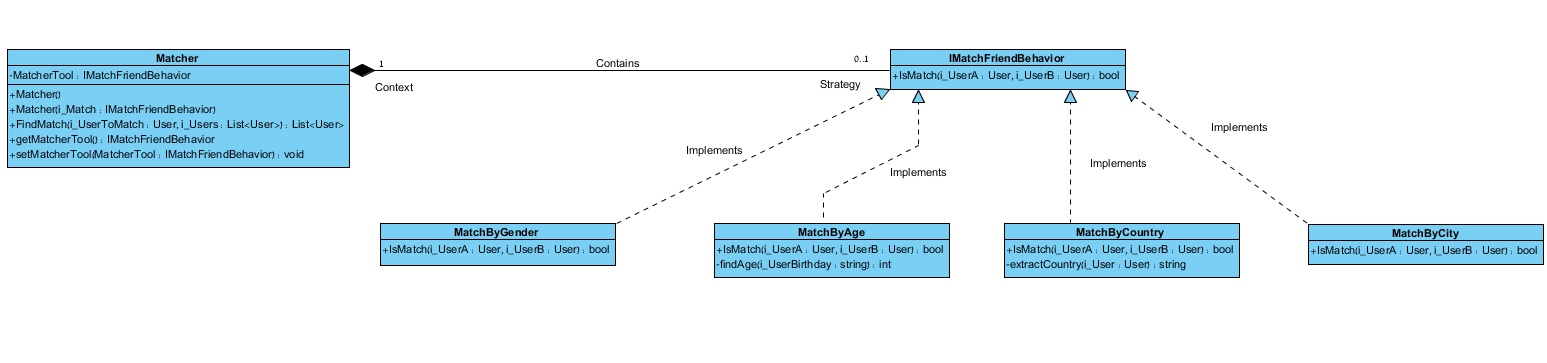
אלו ממימושי ההתנהגות ל-Matcher, שים לב שהם מממשים את הממשק IMatchFriendBehavior. אובייקטי התנהגות אלו מבצעים השוואה בין שני משתמשים ע"פ מינם, מדינתם, עירם וגילם.

* :Sequence Diagram



הערה: לא הוכנס התהליך למקריי סינון אחרים, כיוון שהם חוזרים על עצמם.

* Class Diagram:



המקביל אל ה-Context הוא ה-Matcher. ה-Matcher מכיל בתוכו (ב-composition) את ה-IMatchFriendBehaior שהוא מקביל ל – Strategy. ה-ConcreteStrategy שלנו הם כל מימושו, כלומר – MatchByGender, MatchByCity, MatchByCountry, MatchByAge.

### תבנית מס' 3 – Decorator

* סיבת הבחירה / שימוש בתבנית:

הסיבה בגינה בחרנו להשתמש בתבנית זו הינה כדי לשכלל אצל הuser את האפשרות של פרסום הפוסט ולהוסיף לה גיבוי של הפוסטים בקובץ.

מכיוון שה User לא מאפשר שמירת Post בקובץ log, עלינו לעטוף את ה-User המקורי ולהוסיף לו פונקציונאליות נוספת.

* אופן המימוש:

המחלקות אותן יצרנו / איתן השתמשנו כדי לתמוך בתבנית זו הן:

1. **IPostStatus:**

זהו ה-interface איתו אנו רוצים לעבוד עבור ביצוע של –PostStatus. ה- Decorator שנבנה יממש את ה-interface הנ"ל וירחיב לו את הפונקציונאליות.

1. **User:**

זוהי מחלקה פנימית של FacebookWrapper והיא ה Real Legacy Subject

1. **CoreUser:**

מחלקה זו מחזיקה בתוכה את האוביקט User ומממשת את הממשק IPostStatus. כדי לממש decorator ה Real Legacy Subject צריך למממש Interface ולכן CoreUser יתחזה ל User.

1. **DecoratedUser:**

מחלקה זו מממשת את הממשק IPostStatus , המתודה PostStatus ממומשת במחלקה זו כvirtual כדי שDecoratedUserBackupPost שתירש מהמחלקה תוכל לעשות override למתודה כדי להוסיף לה פונקציונאליות נוספת.

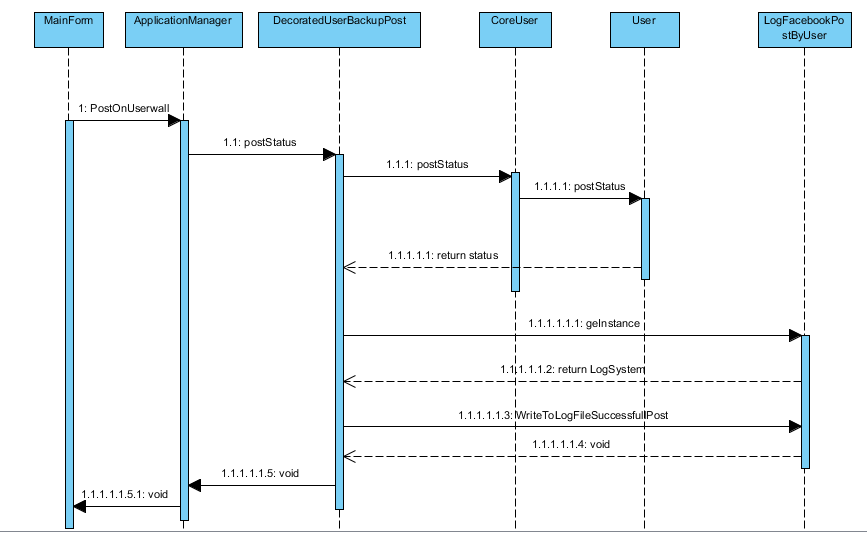
1. **DecoratedUserBackupPost:**

מחלקה זו יורשת מ DecoratedUser שהיא מסוג IPostStatus ומוסיפה בעזרתה את אופציה לשמירת פוסט בקובץ למתודה PostStatus.

1. **LogFacebookPostByUser:**

מחלקה זו נועדה לצורך עדכון קובץ ה-Log, בכל פעם שמתבצע PostStatus.

* :Sequence Diagram



* Class Diagram:

