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

* MatchFinder – פיצ'ר זה מקנה למשתמש את האפשרות למצוא זיווג מתאים עפ"י קריטריונים של מין וטווח גילאים, מבין חבריו בפייסבוק.
* LikeSorter – פיצ'ר זה מקנה למשתמש את האפשרות למיין בסדר עולה את רשימת חבריו בפייסבוק לפי כמות הלייקים שניתנו מכל חבר עבור כלל הפוסטים של המשתמש.

### תבנית מס' 1 – Strategy

* סיבת הבחירה / שימוש בתבנית:

תבנית זו מציעה פתרון לבעיה הבאה: קיים מנגון במערכת אשר אנו מעוניינים שבעתיד נוכל לעשות איתו שימושים אחרים מבלי לשנות את הקוד הקיים. שמנו לב כי הclient (FacebookAppFacade) משתמש באותה המתודה, בה הוא עובר על סוגים שונים של אוספים בהתאם לסוג האובייקט (פיצ'רים מאותה המשפחה הפולימורפית IFeature).

מכאן ראינו לנכון ליצור אב פולימורפי (IReturnableCollection) בעל מתודה אשר מחזירה ICollection ותמומש ע"י מי שמעוניין לספק שירות זה. כעת, מי שיבקש אוסף מאובייקט שמספק שירות זה, יקרא למתודה של האב הפולימורפי וכך בזמן ריצה יקבע סוג האוסף המוחזר.

כעת, אם בעתיד נרצה לאפשר לשאר הפיצ'רים (או אובייקטים חדשים) לספק שירות זה, כך שיידרש הוא לממש את האב הפולימורפי מבלי לשנות את הקוד בclient.

* תיאור המימוש והיכן ניתן למצוא אותו בקוד:

הclient שלנו הוא הFacebookAppFacade שבו ממומשת המתודה GetEnumerator() אשר מחזירה IEnumerator<User>.

במתודה זו יש צורך בבדיקה של סוג האבוייקט על מנת להבחין האם הוא מסוגל להחזיר טיפוס מסוג IEnumerator<User>.

בנוסף, יש צורך לעבור על האוסף המוחזר במידה ומקיים את הנ"ל.

במערכת שלנו MatchFinder מחזיק אוסף מסוג List<User> ואילו LikeSorter מחזיק אוסף מסוג Dictionary<User,int>.

כאמור מדובר באוספים מסוג שונה שמוחזקים ע"י אובייקטים שונים, ולכן מכיוון שמוחזק בclient אובייקט מסוג IFeature (האב הפולימורפי שהם מממשים) יש צורך בבדיקה על איזה אובייקט קונקרטי מדובר כדי להגיע לאוסף שבו בהתאם.

ראינו כי ניתן לחסוך את הבדיקה ע"י השמת מתודה שתהווה את הinjection point .

יצרנו interface IReturnableCollection בעל מתודה ReturnCollection()שמחזירה ICollection.

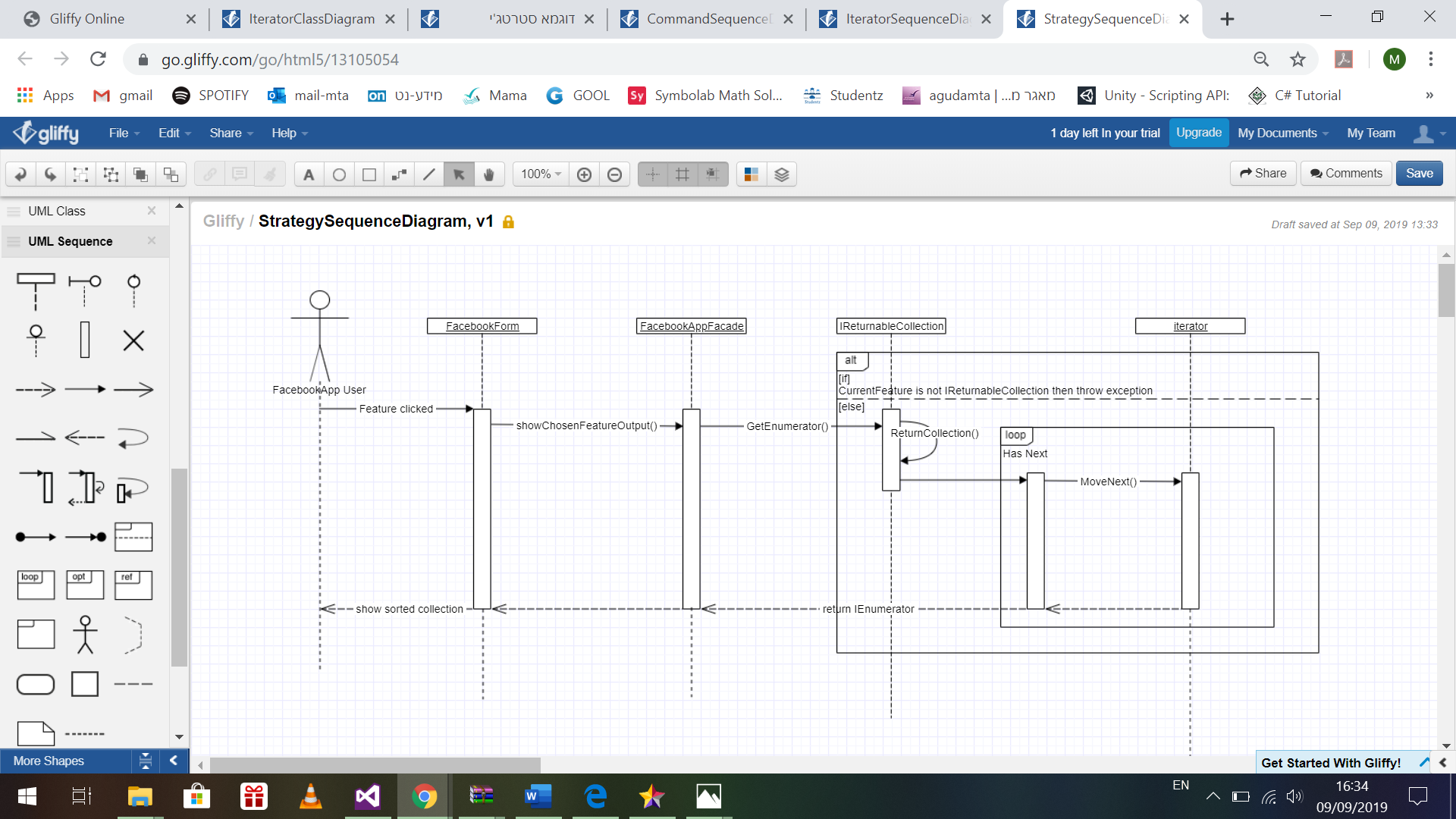
כעת, MatchFinder ו LikeSorter מממשות אב פולימורפי זה ובכך חוסכות את הבדיקה על סוג הIFeature ומאפשרות החזרת האוסף הרלוונטי בזמן ריצה לפי האובייקט המפעיל.

כעת, במידה ונרצה כי פיצ'רים עתידיים יחזיקו אוסף עליו יהיה ניתן לעבור בצורה שלא תשנה את הקוד הנוכחי, הם יוכלו לממש את האב הפולימורפי הנ"ל.

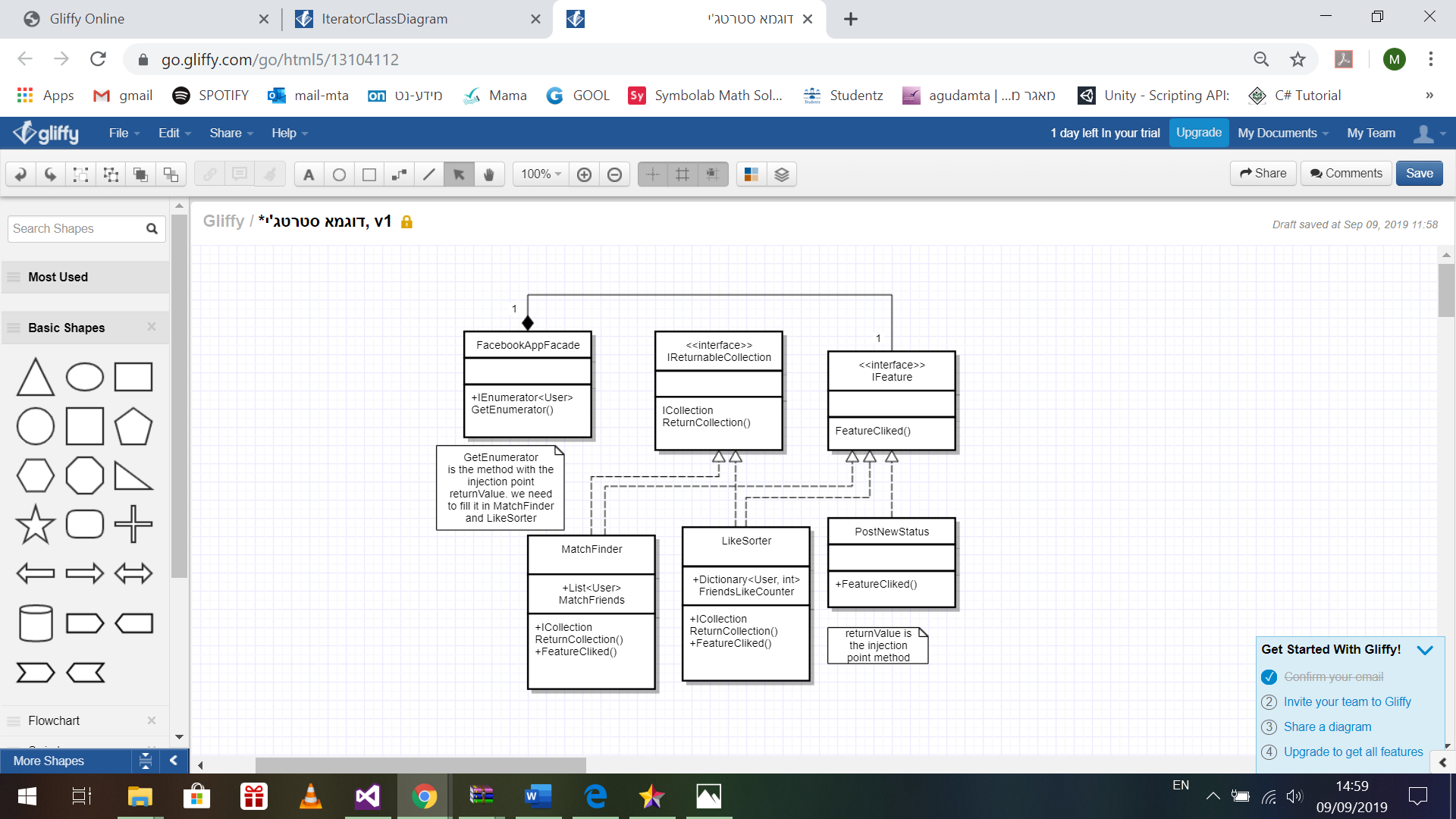
יש לציין כי הinjection point בא לידי ביטוי בשורת הforeach:

foreach (User user in (CurrentFeature as IReturnableCollection).ReturnCollection())

Sequence Diagram



* Class Diagram



Context: FacebookAppFacade

Strategy: IReturnableCollection

ConcreteStrategy: MatchFinder, LikeSorter

### תבנית מס' 2 – Iterator

* סיבת הבחירה / שימוש בתבנית:

במערכת שלנו קיימות מחלקות (שמממשות IFeature) אשר מחזיקות אוסף כprivate member.

כמו כן, אנו רוצים לאפשר מעבר על אוספים אלו מבלי להכיר את מבנה הנתונים ,ביעילות של o(n) ובאופן כזה שאם בעתיד נרצה לשנות את מבנה הנתונים לא נצטרך ליידע את הclient (הFacebookAppFacade).

* אופן המימוש:

במערכת שלנו הFacebookAppFacade מהווה את הclient עבור תבנית זו.

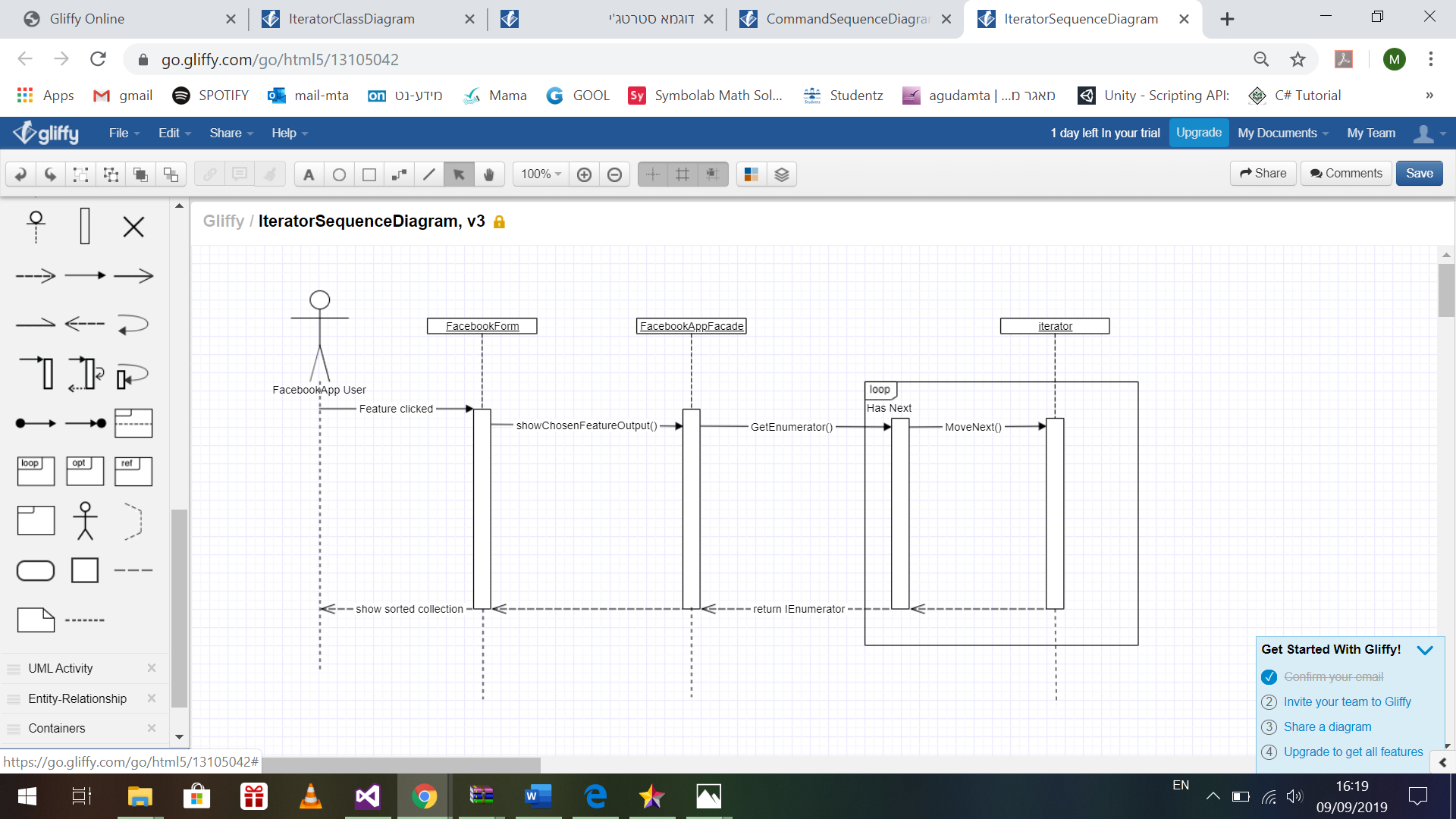
בתוך מחלקה זו מוחזק בין היתר IFeature אשר מטרתו להחזיק את הפיצ'ר המופעל באותו רגע.

שמנו לב כי על מנת להציג את הפלט שמוחזר מפיצ'רים מסויימים כאוסף, יש צורך במעבר על אותו האוסף, דבר הדורש ליידע את הקומפיילר על איזה סוג אוסף מדובר ומי הוא האובייקט הקונקרטי המממש IFeature.

כדי לגרום לכך שלא יהיה צורך ליידע את הקומפיילר בסוג האוסף השתמשנו גם בתבנית Strategy (תבנית מס' 1).

לכן, ראינו לנכון לממש את תבנית Iterator במתודה IEnumerator<User> GetEnumerator() , כדי שנוכל לספק שירות של מעבר על האוסף בזמן ריצה מבלי שה client יכיר את מבנה הנתונים.

* Sequence Diagram



* Class Diagram

### aggregate: IEnumerable

concrete aggregate: FacebookAppFacade

client: FacebookForm

Iterator: IEnumerator<T>

Concrete iterator: IEnumerator<User>

### תבנית מס' 3 – Command

* סיבת הבחירה / שימוש בתבנית:

החלטנו להרחיב את הפיצ'ר המאפשר למיין את רשימת החברים לפי כמות הלייקים שניתנו עבור אוסף הפוסטים של המשתמש, ולאפשר בחירה של סוג האוסף אותו רוצים למיין (תמונות,סטטוסים ופוסטים).

בשל כך, נוצר הצורך לבדוק מי האוסף שנבחר ולהפעיל את מתודת המיון הרלוונטית.

לכן, ראינו לנכון לממש את תבנית command אשר מציעה פתרון reusable וmaintainable במקום מימוש של switch ביעילות של o(n) ובעל פוטנציאל לבאגים (אין נוחות תחזוקה ובטיחות שימוש במידה ובעתיד נרצה להוסיף אוספים נוספים למיון).

* אופן המימוש:

הגדרנו 2 מחלקות חדשות:

* CollectionMenuItem אשר מחזיקה Delegate ואת שם האוסף שאותו רוצים למיין.

בנוסף, היא מממשת מתודה Selected() אשר תפקידה להפעיל את הDelegate

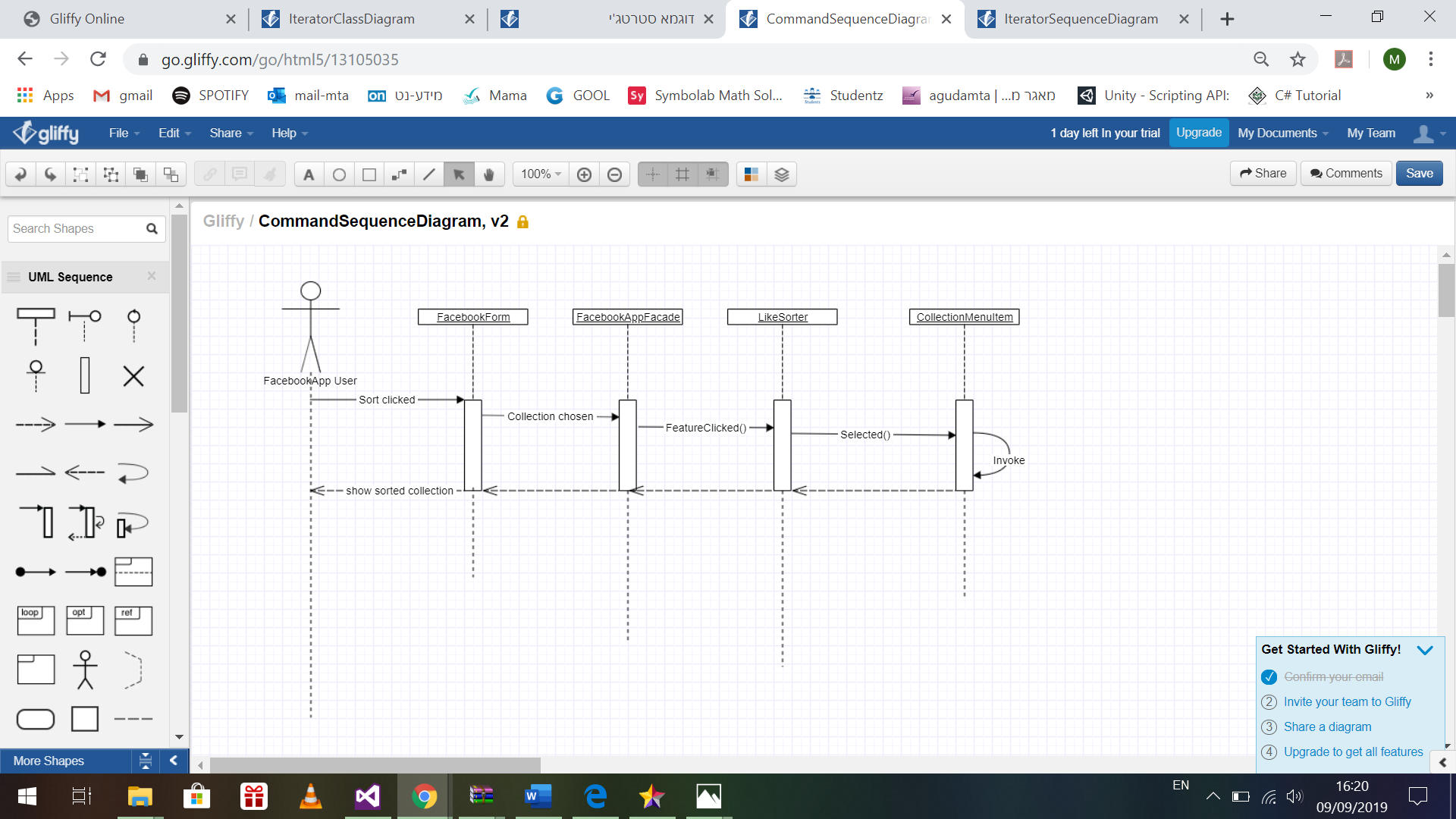
שקיבלה.

* CollectionMenu אשר יורשת מ List<CollectionMenuItem>.

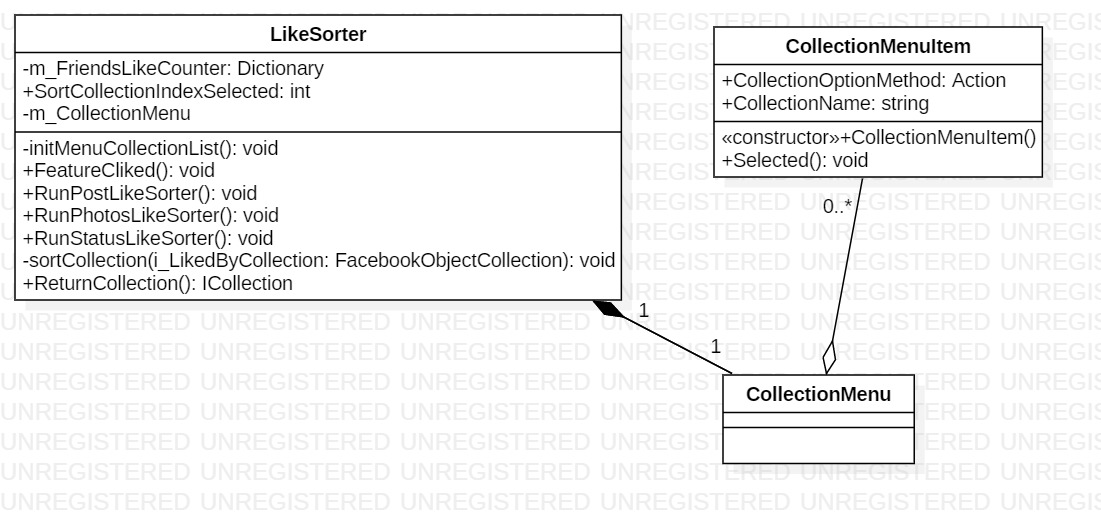
הclient (LikeSorter) מחזיק מופע של CollectionMenu ומתודת אתחול עבור הרשימה.

במתודה FeatureCliked() אשר קיבל דרך האב הפולימורפי IFeature מופעלת מתודת הselected() אשר מפעילה את הdelegate הרלוונטי בהתאם לאינדקס אשר מוחזק כmember במחלקת הclient.

* Sequence Diagram



* Class Diagram



Client: LikeSorter

Concrete command: CollectionMenuItem