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

* פיצ'ר ראשון – **Friends Quiz** – פיצ'ר זה בוחר חבר רנדומלי ולאחר מכן יוצר לו חידון, בסוף החידון ניתן לבחור חבר חדש או להפסיק לשחק.
* פיצ'ר שני – **Photos Filter** – פיצ'ר זה מאפשר למשתמש לסנן ולהציג רשימת את רשימת התמונות שלו העונות לקריטריונים הבאים – Publish date, Amount of Likes, Album, Tagged.

### תבנית מס' 1 –Iterator

* סיבת הבחירה / שימוש בתבנית:

בחרנו להשתמש בתבנית iterator על מנת לבצע מעבר על מבנה הנתונים המחזיק את התמונות המפולטרות אשר מוצגות בממשק המשתמש ב AdvancedImageSearchFor.

הצורך למימוש תבנית זאת היא על מנת לאפשר מעבר על מבנה הנתונים מבלי שה client ידע את סוג מבנה הנתונים ודרך המעבר על המבנה הנתונים, על מנת שאם בעתיד נרצה לשנות את מבנה הנתונים נוכל לעשות זאת בצורה קלה מבלי לשנות את ה .client

* אופן המימוש:

במחלקה FacebookFilteredImages אשר מחזיקה באובייקט FilteredPhotos מסוג

List<Photo> מימשנו את IEnumrable<T> interface ומימשנו את המתודה GetEnumarator.

המתודה עוברת על כל Photo ב FilteredPhotosועבור כל Photo מבצעת yield return.

* Sequence DiagramDiagram

  Description automatically generated
* Class Diagram

Diagram

Description automatically generated

Client= AdvancedImageSearchForm

ConcreteAggregate = FacebookFilteredImages

ConcreteIterator = FilteredPhotosIterartor

### תבנית מס' 2 –Strategy

* סיבת הבחירה / שימוש בתבנית:

בחרנו להשתמש בתבנית strategy לצורך הצגת סוגי המידע השונים על התמונה שהמשתמש בחר. הצורך לכך הגיע מקיומה של יותר מסוג מידע אחד על התמונה שניתן להציג, וכיוון שרצינו שהסוג המידע יוכל להיבחר על ידי המשתמש בזמן ריצת התוכנית בצורה גמישה.

לכן החלפנו רק את החלק בו סוג המידע מתקבל לדינאמי שיבחר בזמן בריצה, בנוסף לכך אם בעתיד נרצה להוסיף סוג מידע אחר על התמונה, נוכל לעשות זו בצורה קלה יותר.

* אופן המימוש:

הגדרת interface בשם IPhotoData, אותו ממשות המחלקות PhotoTags, PhotoLikes ו

PhotoComments כל אחת מהן מהווה סוג מידע על התמונה שניתן להציג.

המתודה FetchSelectedData במחלקה SelectedImage מקבלת כפרמטר אובייקט מסוג IPhotoData ומחזירה כפלט רשימה של מחרוזות המייצגות את המידע הנבחר על התמונה של המשתמש לפי האובייקט מסוג IPhotoData שהמתודה קיבלה.

כל זאת עובד בצורה שבה המשתמש מסמן בטופס התמונות של האפליקציה איזה סוג מידע הוא מעוניין לראות ולפי זה שולחים את ה IphotoData המתאים אל המתודה FetchSelectedData.

* Diagram

  Description automatically generatedSequence Diagram
* Diagram

  Description automatically generatedClass Diagram

Context = SelectedImage

Strategy = IPhotoData

ConcreteStrategyA = PhotoLikes

ConcreteStrategyB = PhotoComments

ConcreteStrategyC = PhotoTags

### תבנית מס' 3 –Template Method

* סיבת הבחירה / שימוש בתבנית:

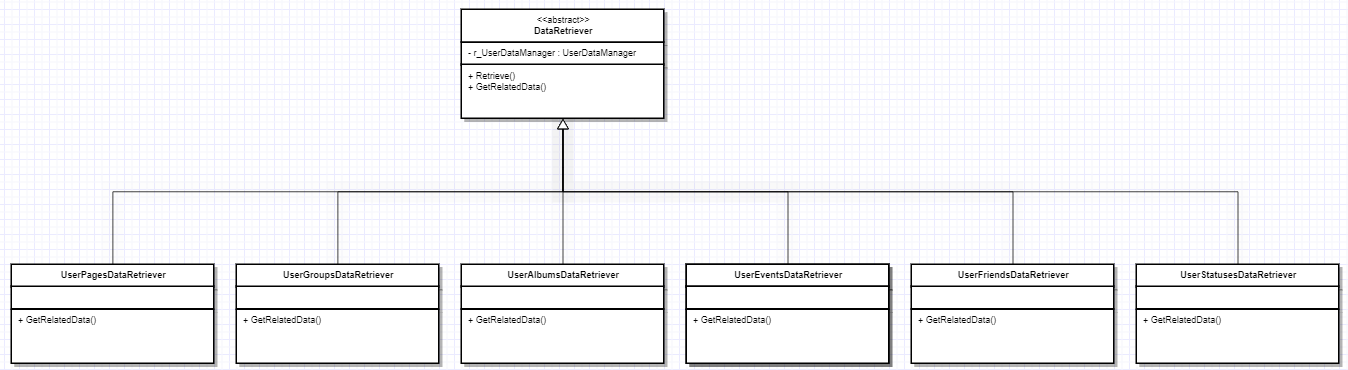
בחרנו להשתמש בתבנית Template Method לצורך מניעת שכפול קוד במחלקת ה Facade שלנו UserDataManger – אשר שלפה נתונים באופן זהה ליוזר המחובר לאפליקציה לכמה שדות שונים (קבוצות, עמודים וכדו'...). בנוסף לכך אם בעתיד נרצה לשלוף מידע נוסף אל האפליקציה שלנו ולהציג אותם נוכל לעשות זאת ללא שכפול קוד.

* אופן המימוש:

יצרנו את המחלקה '', מחלקה זו הינה אבסטרקטית ובה המתודה 'Retrieve' אשר מהווה Template Method ומתודה נוספת 'GetRelatedData' אשר היא אבסטרקטית גם ומהווה Injection Point. ויצרנו את המחלקות: UserAlbumsDataRetriever, UserEventsDataRetriever, UserFriendsDataRetriever, UserGroupsDataRetriever, UserPagesDataRetriever, UserStatusesDataRetriever אשר ממשות את המתודה 'GetRelatedData' באופן ספציפי המתאים לאופן שליפת המידע.

* Diagram, schematic

  Description automatically generatedSequence Diagram
* Class Diagram



להלן ההקבלות:

Abstract Class = DataRetriever

Template Method = Retrieve

Injection Point = GetRelatedData

ConcreteClassA = UserPagesDataRetriever

ConcreteClassB = UserGroupsDataRetriever

ConcreteClassC = UserAlbumsDataRetriever

ConcreteClassD = UserEventsDataRetriever

ConcreteClassE = UserFriendsDataRetriever

ConcreteClassF = UserStatusesDataRetriever