### תבנית מס' 1 – Adapter

* סיבת הבחירה / שימוש בתבנית:

בחרנו לממש Adapter ל- List על מנת לאפשר Lazy Creation ושמירה של נתונים לתוך סינגלטון מערכתי שיצרנו. השימוש ב-Adapter מאפשר יצירה אוטומטית שלו על ידי הסינגלטון רק בעת שקוראים לו.

* אופן המימוש:

ה-Adapter הוא המחלקה UserRankListAdapter<T> שמכיל בתוכו שדה List<UserRank<T>>. יש לו בנאי פרטי שמגדיר את השדה כ-null. יש לו מתודה סטטית פרטית initUserRankListAdapter שדרכה הסינגלטון קורא ליצירת האובייקט.  
בקוד שלנו אנחנו משתמשים באפשרות הזו כאשר מוצאים עבור המשתמש את רשימת האנשים שמתוייגים איתו הכי הרבה בתמונות, וכאשר מוצאים את רשימת האנשים שהיו איתו בהכי הרבה אירועים. ה-Adapter מאפשר לנו לשמור את המידע בסינגלטון ולהימנע מבדיקה חוזרת שתניב בדיוק את אותה תוצאה.

* Sequence Diagram

C:\Users\7\Downloads\Adapter-Seq (1).png

* Class Diagram

C:\Users\7\Downloads\UserRankListAdapter.png

### תבנית מס' 2 – Facade

* סיבת הבחירה / שימוש בתבנית:

רצינו אובייקט שישמור בתוכו כחבר אובייקט מסוג אחר ויספק סטטיסטיקה נוספת לגבי אותו אובייקט, ובנוסף יסתיר פעולות שאפשר לעשות על החבר בד"כ ויחשוף רק מתודות שרלוונטיות לשימוש שלנו לצורך פשטות.

* אופן המימוש:

ה-Facade אצלנו הוא מחלקה בשם UserRank<T> הנוצרת מאובייקט FacebookWrapper.ObjectModel.User ומכילה אותו כחבר פרטי ומסתירה את כל המתודות הקיימות שלו מלבד User.ImageSquare. בנוסף המחלקה מאפשר לשמור רשימת אובייקטים המשוייכים לאותו יוזר, לקבל את גודל הרשימה, ולהשוות בין אובייקטים מאותו סוג (קודם לפי גודל הרשימה ואז לפי שם המשתמש).  
בקוד שלנו אנחנו משתמשים באובייקט הזה כאשר מוצאים עבור המשתמש את רשימת האנשים שמתוייגים איתו הכי הרבה בתמונות, וכאשר מוצאים את רשימת האנשים שהיו איתו הכי הרבה אירועים. ה-Facade מאפשר לנו לשמור עבור כל משתמש שמצאנו את האירועים/תמונות שהוא נמצא בהן, לעשות השוואה בין המשתמשים כאשר נרצה למיין לפי הכמות שנמצאה, להוציא את התמונה של המשתמש, ולהציג שם + כמות ב-UI שלנו (תוך שימוש בשדה “NameCount”).

* Sequence Diagram

C:\Users\7\Downloads\Facade-Seq.png

* Class Diagram

C:\Users\7\Downloads\UserRank (1).png

### תבנית מס' 3 – Singleton (Universe)

* סיבת הבחירה / שימוש בתבנית:

רצינו שהתוצאה של השאילתות המתוארות בפי'צרים המיוחדים יווצרו פעם אחת בלבד (וכל בקשה נוספת תתן את התוצאה הקודמת), והשאילתא צריכה לקרות On Demand.

* אופן המימוש:

המחלקה CentralSingleton מרכזת את כל המופעים הסינגלטונים. במחלקה קיים משתנה פרטי לכל מחלקה סינגלטונית, וכדי לגשת אליו מימשנו Property get שגם בודק האם המשתנה אותחל, ואם לא מאתחל אותו.  
ממשק העבודה של מחלקה סינגלטונית מול CentralSingleton הוא קיומו של בנאי פרטי שלא מקבל פרמטרים, ומתודה סטטית פרטית ששמה מתחיל ב init שמחזירה מופע של המחלקה. ה CentralSingleton מאתחל את המחלקה באמצעות Reflection. ממשק עבודה זה מאפשר את אתחול המופעים בצורה גנרית ובכך מונע שכפול קוד.

* Sequence Diagram

C:\Users\7\Downloads\Singleton-Seq (1).png

* Class Diagram

C:\Users\7\Downloads\CentralSingleton.png

### שימוש ב Data Binding

השתמשנו ב Auto Data Binding בייצוג הפוסטים שעל קיר המשתמש. השתמשנו ב Post בתור ה Data Source והצגנו את הנתונים בטאב Wall

### עבודה אסינכרונית

כל פנייה לשרת החיצוני של פייסבוק נעשת בצורה אסינכרונית – בטעינה הראשונית של התוכנה, בפרסום סטאטוס, בשליחת תגובה לפוסט בWall ובשאילתות של הפיצ'רים המיוחדים.