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

* הפיצ'ר הראשון הוא קבלת רשימה של פעילות חברים בפרופיל של המשתמש.
* הפיצ'ר השני הוא חיתוך של Checkins לפי המשתמש ו/או החברים שלו.
* הפיצ'ר השלישי הוא משחק של שמות.

### תבנית מס' 1 – Factory Method

* במערכת קיים פיצ'ר שמדמה משחק של יצירת כינויים ע"י בחירת הקשרים שונים.

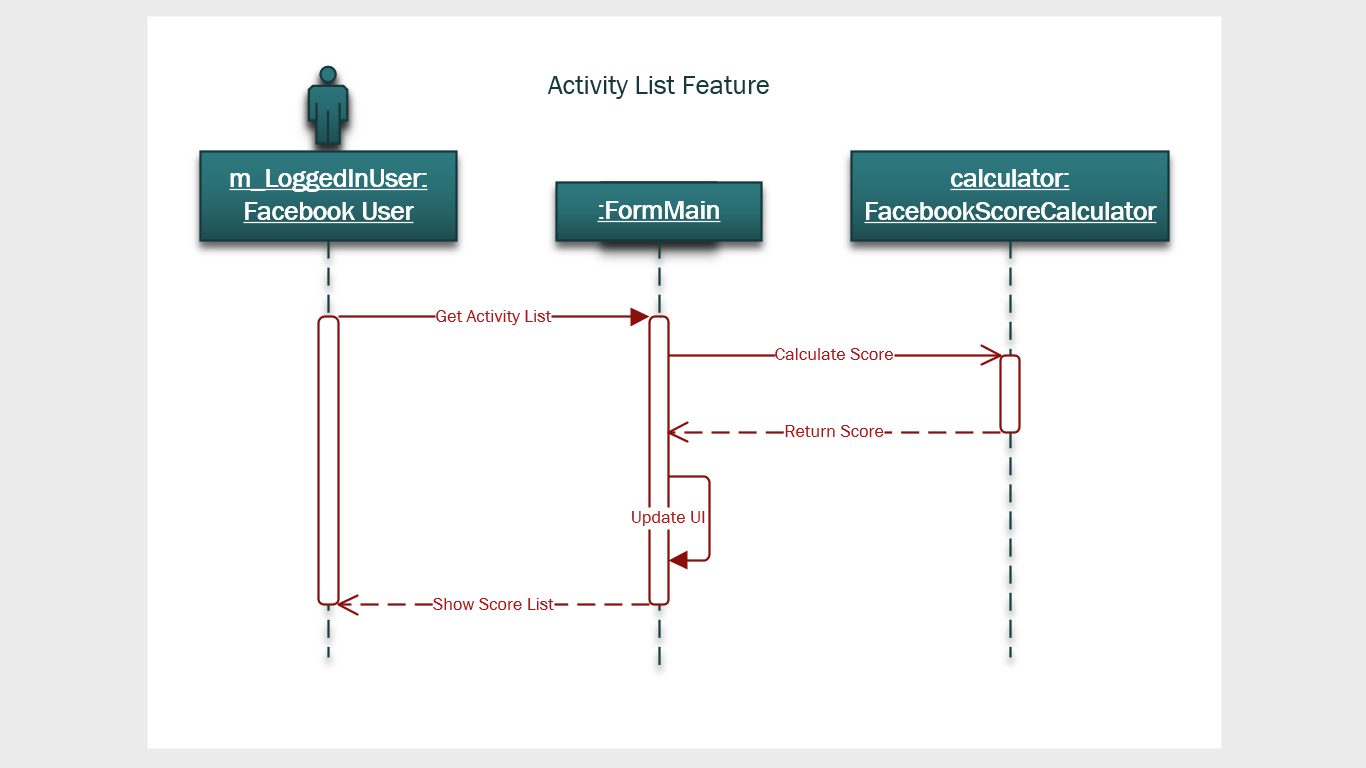
למעשה, זהו תיאור למעין רכיב שמקבל הקשר\קונטקסט והוא יוצר אובייקט קונקרטי שנותן את השירות הרצוי, ז"א Factory Method.

* אופן המימוש:

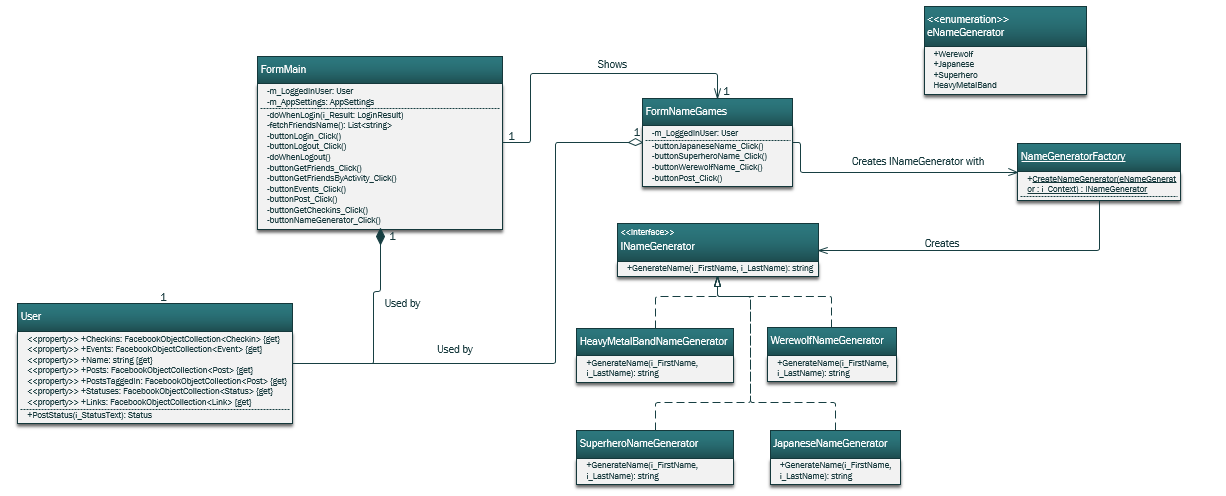
במקום מחלקה אחת NameGenerator, יש ממשק INameGenerator, ובו מתודה אחת GenerateName(), בנוסף יש 4 מחלקות שכל אחת מממשת את הממשק הזה באופן שונה.

כדי ליצור מחלקה כזאת יצרנו מחלקה סטטית עם מתודה סטטית שמקבלת enum(eNameGenerator), ועל פיו יוצרת מופע של אחת מן המחלקות המממשות את הממשק. ניתן למצוא את הקוד בFormNameGames.

* Sequence Diagram



* Class Diagram

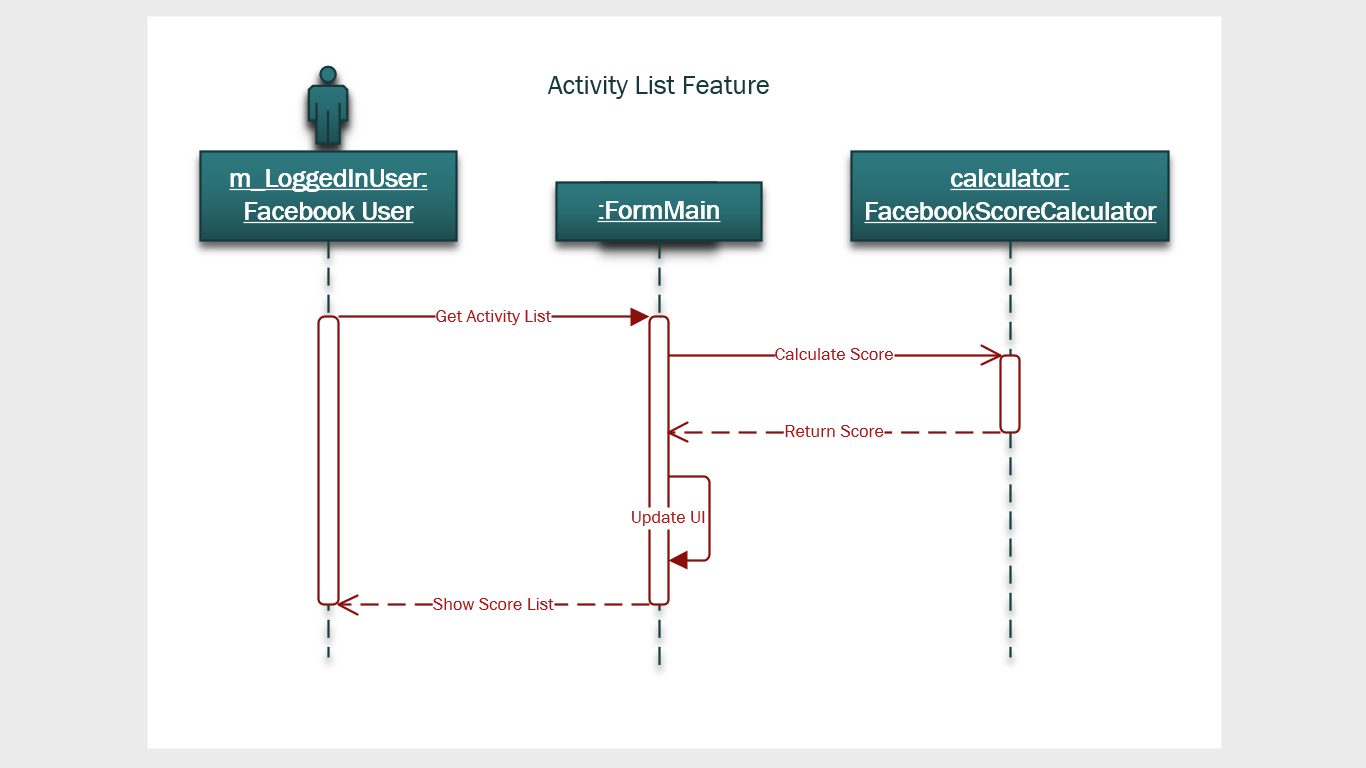


### תבנית מס' 2 – Facade

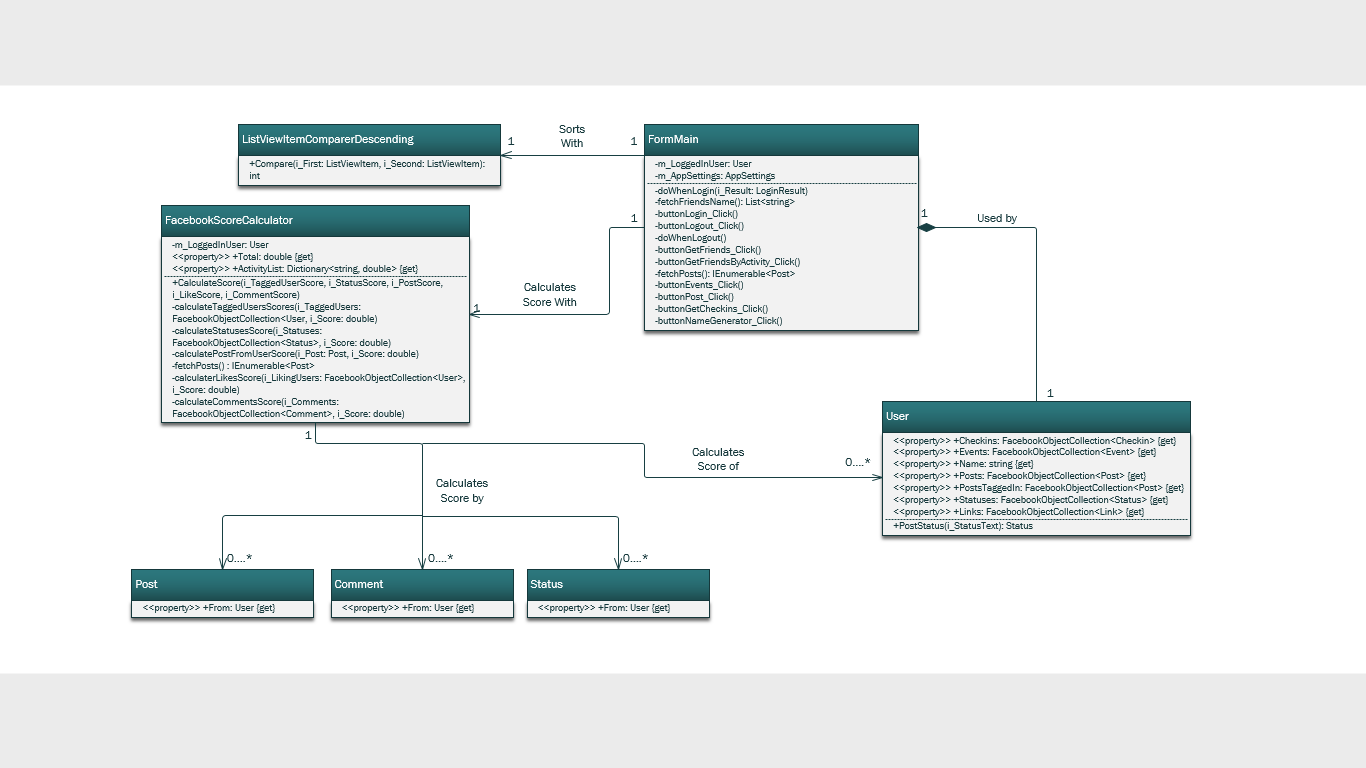
* במערכת קיים פיצ'ר שמבצע הרבה פעולות חישוב, ומצריך את הClient לבצע כל אחת מהן בנפרד. למעשה, על הClient להכיר את הרכיב, ולהפעיל את הפעולות בנפרד. נרצה שהשימוש ברכיב יהיה פשוט יותר, כך שבמקום שיבצע את כל הפעולות אחת אחרי השניה, יוכל לבקש באופן כללי את השירות, ויקבל את כולן. תיאור מקרה זה, למעשה מתאר את התבנית Facade.
* אופן המימוש:

במחלקה FacebookScoreCalculator הבנאי מקבל את ה User איתו התחברנו במקום השם. המחלקה כעת תהיה אחראית על השגת כל הפוסטים הרלוונטים ע"י fetchPosts(), וכעת בכדי לחשב את הניקוד עבור אותו היוזר, יש להפעיל מתודה אחת, CalculateScore() שמקבלת את הניקודים השונים שהמשתמש בחר, עבור כל מאפיין, וממלאת את המילון, Dictionary<string, double> ActivityList , וכך ניתן למעשה לקבל את כל הניקוד בפשטות, ללא צורך בהבנה של איך החישוב נעשה ע"י הפעלת ה getter property שניתן.

* Sequence Diagram



* Class Diagram

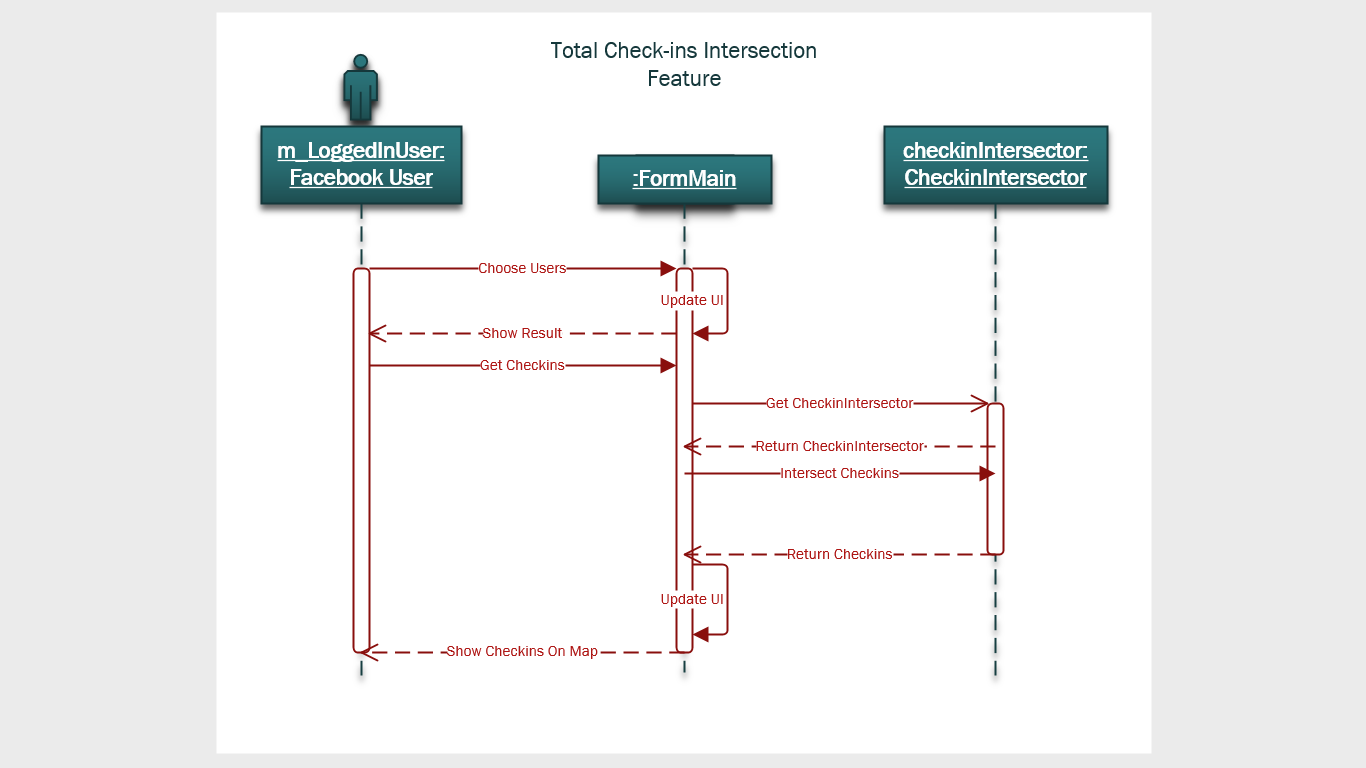


### תבנית מס' 3 – Singelton

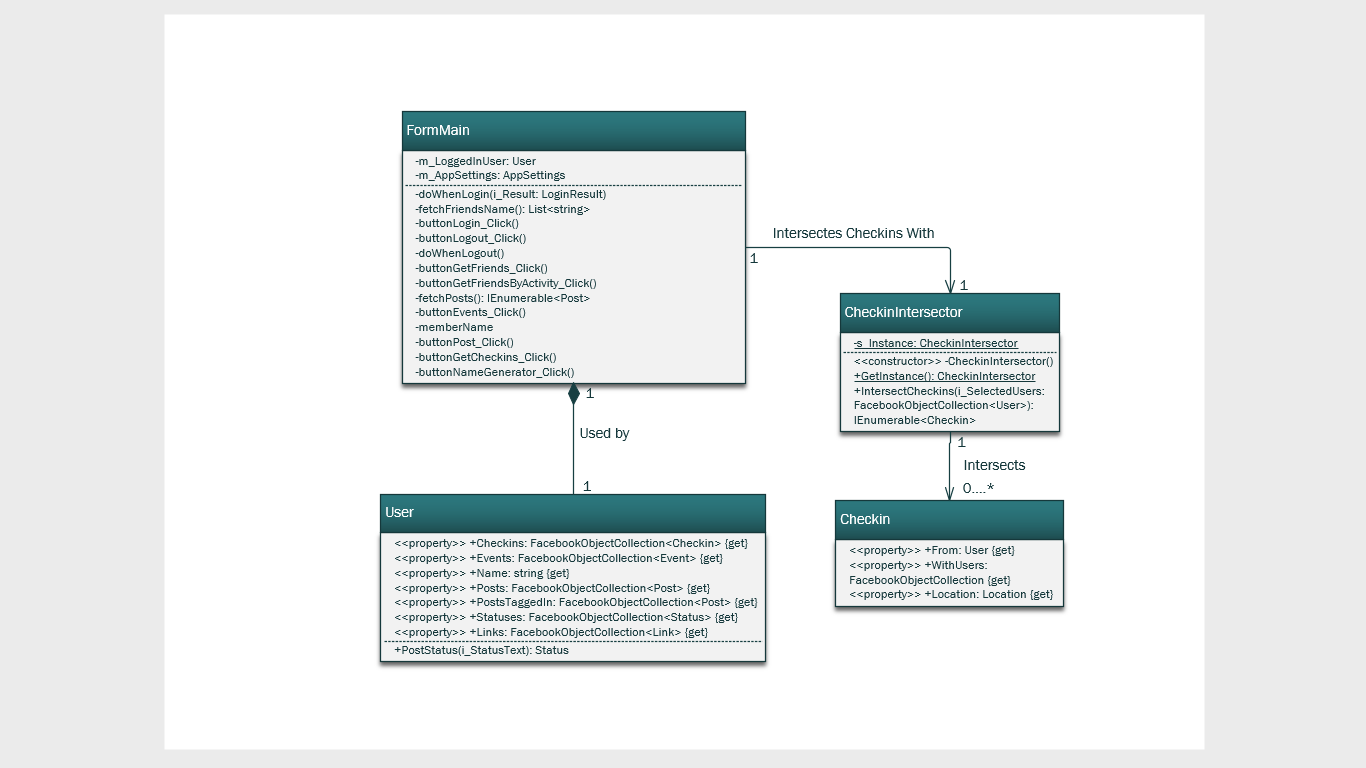
* במערכת קיים פיצ'ר שקיומו הוא יחיד, הפיצ'ר הוא חיתוך של Check-ins של היוזר עם/בלי חבריו והצגה על המפה ועל תפקיד זה מונה CheckinIntersector. הרכיב הנ"ל, הוא יחיד, ועל כן שימוש ב Singelton שומר על ייחודו.
* אופן המימוש:

במחלקה CheckinIntersector יש private constructor, static member (Instance), ומתודה סטטית GetInstance(). כל אלה ביחד נותנים את השירות של סינגלטון.

* Sequence Diagram



* Class Diagram



### השתמשתי בתיכנות אסינכרוני במגוון מקומות בתכנית:

* כאשר יש ביצוע של התחברות לפייסבוק – doWhenLogin() אני פותח Task חדש, שואב את החברים של המשתמש ואז מציג בתוך listBox.
* כאשר לוחצים על כפתור של פעילות החברים בפרופיל של המשתמש, שוב נפתח Task חדש ובו מחושבים כל אחוזי הפעילויות של החברים לפי ניקוד מסוים ולאחר מכן מוצג בUI
* כאשר לוחצים על כפתור Events פתחתי Task חדש ובו נפתח הטופס FormEvents.
* כאשר לוחצים על כפתור מצא צ'קאינז, אני פותח Task חדש ובו אני מחשב בעזרת מחלקה CheckinIntersector את הCheck-ins הרלוונטיים, ואז מציג על המפה בUI.

### שימוש בData Binding:

יצרתי Data Binding Source של Events והוא רלוונטי רק בטופס FormEvents. אפשר לשנות את התיאור של אירוע מסוים והוא ישאר ככה גם לאחר בחירה אחרת של אירוע.