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

* הפיצ'ר הראשון שהוספנו הוא חיפוש חברים לפי המזל שלו כפי שמוגדר בגלגל המזלות – Zodiac Sign Manager.

המשתמש ילחץ על הכפתור ויקבל בחזרה רשימה של כל החברים שלו בעלי אותו מזל כמוהו.

עבור מימוש הפיצ'ר יצרנו Form חדש ובו ראשית יופיע ההורוסקופ של המשתמש המחובר. בלחיצת כפתור של: Find my Zodiac match נוצרת רשימה של כל חברי הפייסבוק של המשתמש הנוכחי שמחובר לאפליקציה (אם טרם נוצרה בשלב אחר של ריצת האפליקציה). יצרנו מחלקה חדשה המבצעת חישוב עבור כל חבר פייסבוק של המשתמש המחובר מה המזל שלו כפי שמוגדר בגלגל המזלות. המחלקה תוסיף כל חבר כזה לתיבת ההדפסה והמשתמש יקבל רשימה של כל חברי הפייסבוק שלו בעלי אותו מזל כמו שלו.

* הפיצ'ר השני שהוספנו הוא חיפוש המעריץ הטוב ביותר- FindYourBestFan

המשתמש ילחץ על הכפתור ויקבל בחזרה את השם המלא והתמונה של החבר הפייסבוקי שלו שלחץ "Like" על הפוסטים / התמונות של המשתמש לפי בחירתו.

עבור מימוש הפיצ'ר יצרנו Form חדש ובו checkbox בו המשתמש יוכל לסנן על פי איזה פרמטר הוא רוצה לבחון את המעריץ הטוב ביותר (תמונות/פוסטים/שניהם) ובעת לחיצה על הכפתור "Find Your Best Fan" הוא יראה את התמונה ואת השם של המעריץ הגדול ביותר (זה שעשה את כמות ה"Likes" הגדולה ביותר).  
הלוגיקה של הפרמטרים עובדת בצורה זהה ב2 מחלקות שונות.

יצרנו מחלקה חדשה המבצעת את החישוב עבור כל חבר פייסבוק של המשתמש המחובר. המחלקה משתמשת בשירותי המחלקה FacebookWrapper שכן יש לבצע בדיקה עבור כל משתמש פייסבוקי.

בלחיצת כפתור אם סומן הפרמטר של הפוסטים - נבדוק עבור כל הפוסטים שהמשמש המחובר פרסם- מי החבר שעשה לפוסטים אלה הכי הרבה לייקים.   
אם סומן הפרמטר של התמונות - נבדוק עבור כל התמונות שהמשתמש המחובר פרסם - מי החבר שעשה לתמונות אלה הכי הרבה לייקים.  
אם שני הפרמטרים מסומנים – נפעיל את שתי המחלקות לבדיקת החבר הכי טוב ובfacade נחבר את התוצאות של שתי המחלקות ונכריע מי החבר הכי טוב מבין שתיהן.

**עבודה עם Data Binding**

בחרנו להשתמש ב-Two way Data Binding כאשר ה-data object שלנו הוא ה-User המהווה Best fan של המשתמש המחובר לתוכנית.

כאשר משתמש של האפליקציה ילחץ על Find Your Best Fan הוא בעצם מבקש   
מה-data binding להביא לו מידע בנוגע לUser שמהווה best fan.

אנו מקבלים בחזרה אובייקט מסוג User והמידע עליו הוא השם המלא שלו והתמונה (הראשית) שלו.

את ה-Data Binding מימשנו במקום שורות הקוד שהיו במחלקת FacebookTabs וכן במקום חלק מהפונקציונליות שהייתה ב-Logic namespace.

כמו כן , ניתן לראות שיש הפרדה מלאה בין הלוגיקה והUI שכן הלוגיקה לא מחזירה לBestFan אובייקט מסוג User אלא מאתחלת את הפרמטרים לבדיקה ישר בForm היות ומחזיקה את הDataSource של אותו Form.

**עבודה עם Threads**

בחרנו להשתמש בתכנות א-סנכרוני עבור הפעולות אשר עובדות מול השרת של הפייסבוק ולוקחות זמן רב.

שימוש בthreads במהלך התכנית בכל פניה לשרת הפייסבוק מאפשרת שימוש מהיר יותר ומקבילי מול השרת.

אם לא היינו משתמשים בתכנות מקבילי כל פניה לשרת הייתה לוקחת זמן רב והיינו צריכים להמתין שתסתיים כדי לקרוא לפונקציות נוספות.

שימוש בthreads במהלך התכנית:

* לחיצה על כפתור "recent post" תפעיל thread שיריץ את המתודה "fetchPosts".
* לחיצה על כפתור "show friends" תפעיל thread שיריץ את המתודה "fetchFriends".
* לחיצה על כפתור "get your photos" תפעיל thread שיריץ את המתודה "fetchPhotos".
* לחיצה על כפתור "get your albums" תפעיל thread שיריץ את המתודה "fetchAlbums".
* לחיצה על כפתור "show check in" תפעיל thread שיריץ את המתודה "fetchCheckIns".
* לחיצה על כפתור "show liked pages" תפעיל thread שיריץ את המתודה "fetchPages".
* לחיצה על כפתור "get your videos" תפעיל thread שיריץ את המתודה "fetchVideos".

### תבנית מס' 1 – Facade

* סיבת הבחירה / שימוש בתבנית:

הפיצ'ר "מצא את המעריץ הטוב ביותר שלך" מבצע בדיקה של כל התמונות והפוסטים עליהם החברים של המשתמש המחובר עשו לייק ומחשב מי החבר שעשה הכי הרבה לייקים - והוא ייבחר להיות המעריץ הטוב ביותר.

ה- facade מספק רכיב המאפשר עבודה מולו וקבלת שירות פשוט יותר, כך שמאחרי הקלעים נמצאים שירותים מסובכים ומורכבים יותר. וכך אנו "מחביאים" מהמשתמש בצורה לוגית את חישוב המעריץ הטוב ביותר.

למעשה הfacade מסתיר את הלוגיקה המורכבת עבור הUI ובמקרה שלנו הוא מנותק לגמרי מהUI ומעדכן אותו ע"י data binding.

בחרנו להשתמש בFacade משום שהפיצ'ר של מציאת המעריץ הטוב ביותר הינו פיצ'ר מורכב שמשתמש במספר תתי מערכות בהן יש להשתמש על מנת להציג את האפשרות של מציאת המעריץ הטוב ביותר בהתאם לבחירת הפרמטרים ע"י המשתמש.

יצרנו ממשק פשוט למציאת המעריץ הטוב ביותר, שניתן להחליף את כולו או חלקים ממנו במידת הצורך בצורה שקופה למשתמש.

* אופן המימוש:

כדי לממש facade יצרנו מחלקה חדשה עבור חישובי המעריץ הטוב ביותר בשם Façade.

המחלקה המדוברת פונה (בהתאם לבחירת המשתמש) למחלקות :

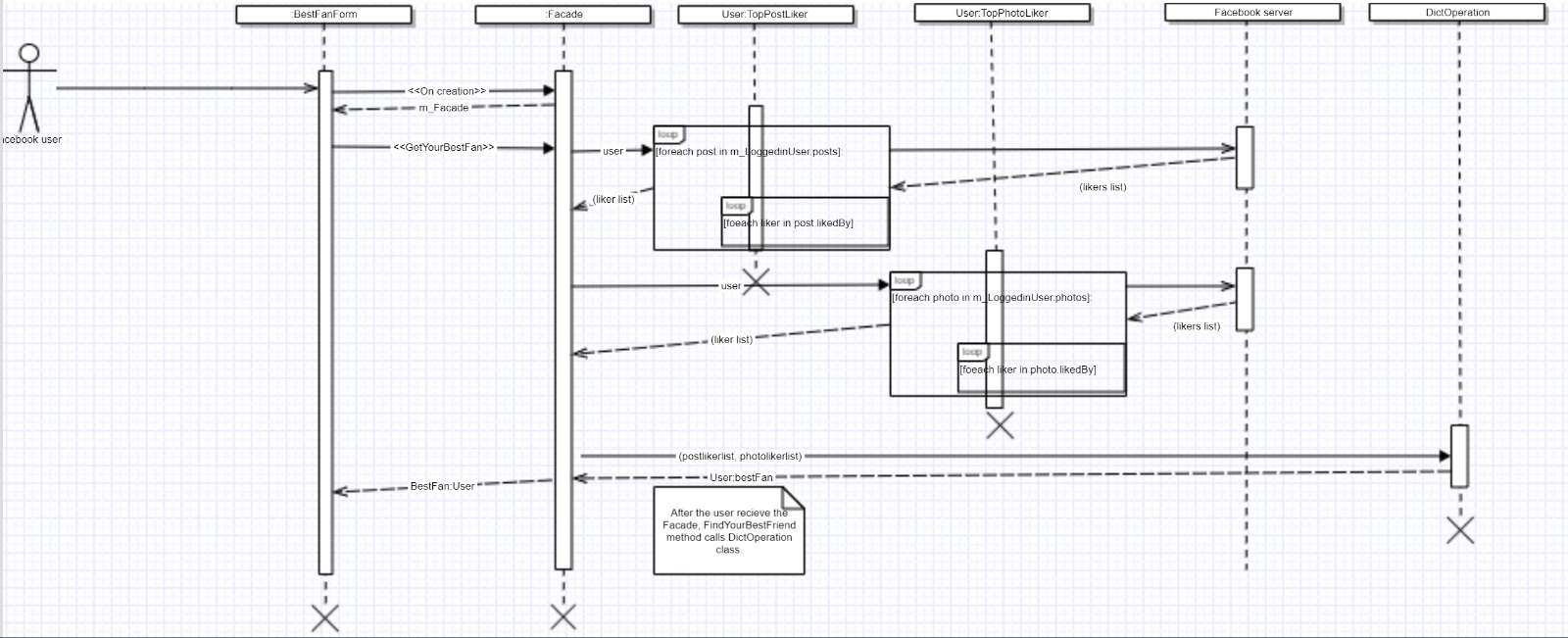
* TopPhotosLiker מציאת החבר שעשה הכי הרבה לייקים על התמונות של המשתמש המחובר.
* TopPhotosLiker מציאת החבר שעשה הכי הרבה לייקים על הפוסטים של המשתמש המחובר.
* DictOperations מחלקה זו מבצעת פעולות על המילונים המתקבלים משתי המחלקות המתוארות למעלה, כך שאנו מחברים את המידע שאנו מקבלים במילונים ע"מ לקבל את המעריץ הטוב ביותר.

מחלקת ה-Facade מסתירה את כל הלוגיקה של 3 המחלקות הללו.

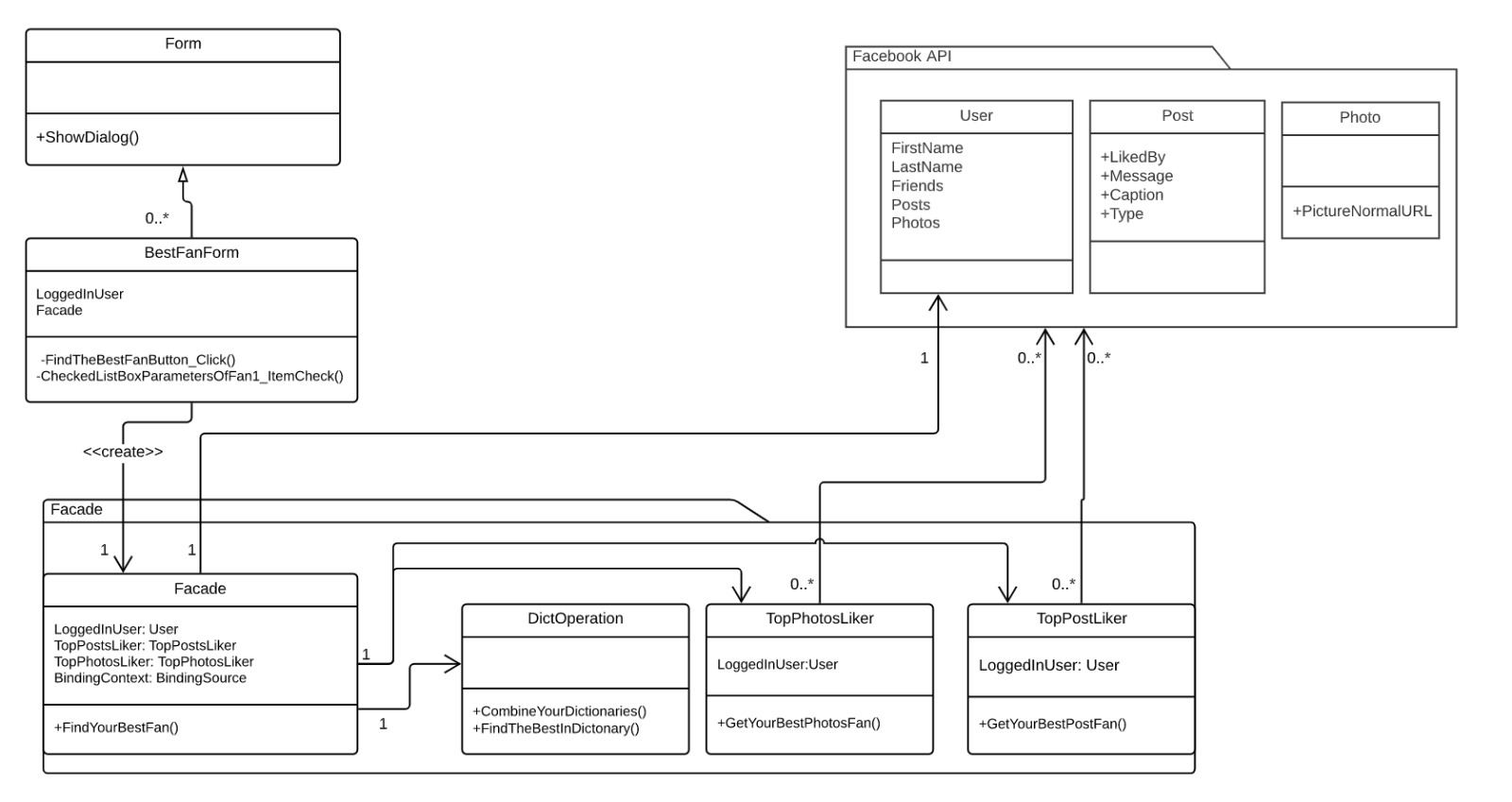
בעת ריצת התכנית בשלב יצירת ה-UI מתוך מחלקת: FacebookTabs אנו מייצרים אובייקט Facade שבעת הרצת הפונקציונליות של FindYourBestFan מממש את כל הלוגיקה המורכבת ומחזיר לUI, ל FacebookTabs בעזרת data binding את התוצאה של פעולותיו.

הערה: כפי שהסברנו בתחילת המסמך, השתמשנו ב-Data Binding כך שבעת לחיצה על כפתור Find Your Best Fan בטאב הרלונטי, אנו "נקשרים" לאובייקט הפאנל שמייצג את ה-User שמייצג את המעריץ הטוב ביותר של m\_LoggedInUser.

* Façade Sequence Diagram



* Façade Class Diagram



הקבלה ל-Façade Pattern:

Client: FacebookTabs

Façade: public class Façade

Sub- system classes:

* TopPhotosLiker
* TopPhotosLiker
* DictOperations

### תבנית מס' 2 – Singleton

* סיבת הבחירה / שימוש בתבנית:

תבנית זו עוזרת לנו להגביל את יצירת המופעים של מחלקה כלשהי למופע יחיד, במקרים בהם אנו זקוקים ל-instance אחד בדיוק המערכת שלנו.

באפליקציה שלנו אנו מייצרים אובייקט של משתמש פייסבוקי- m\_LoggedInUser ואנו זקוקים למופע אחד בדיוק שלו במהלך כל ריצת התכנית היות ובכל זמן נתון יש רק משתמש אחד שמחובר לאפליקציה. יצירה של מופע נוסף הינה בזבזנית ועלולה אף לייצר באגים בריצת התכנית.ְ

* אופן המימוש:

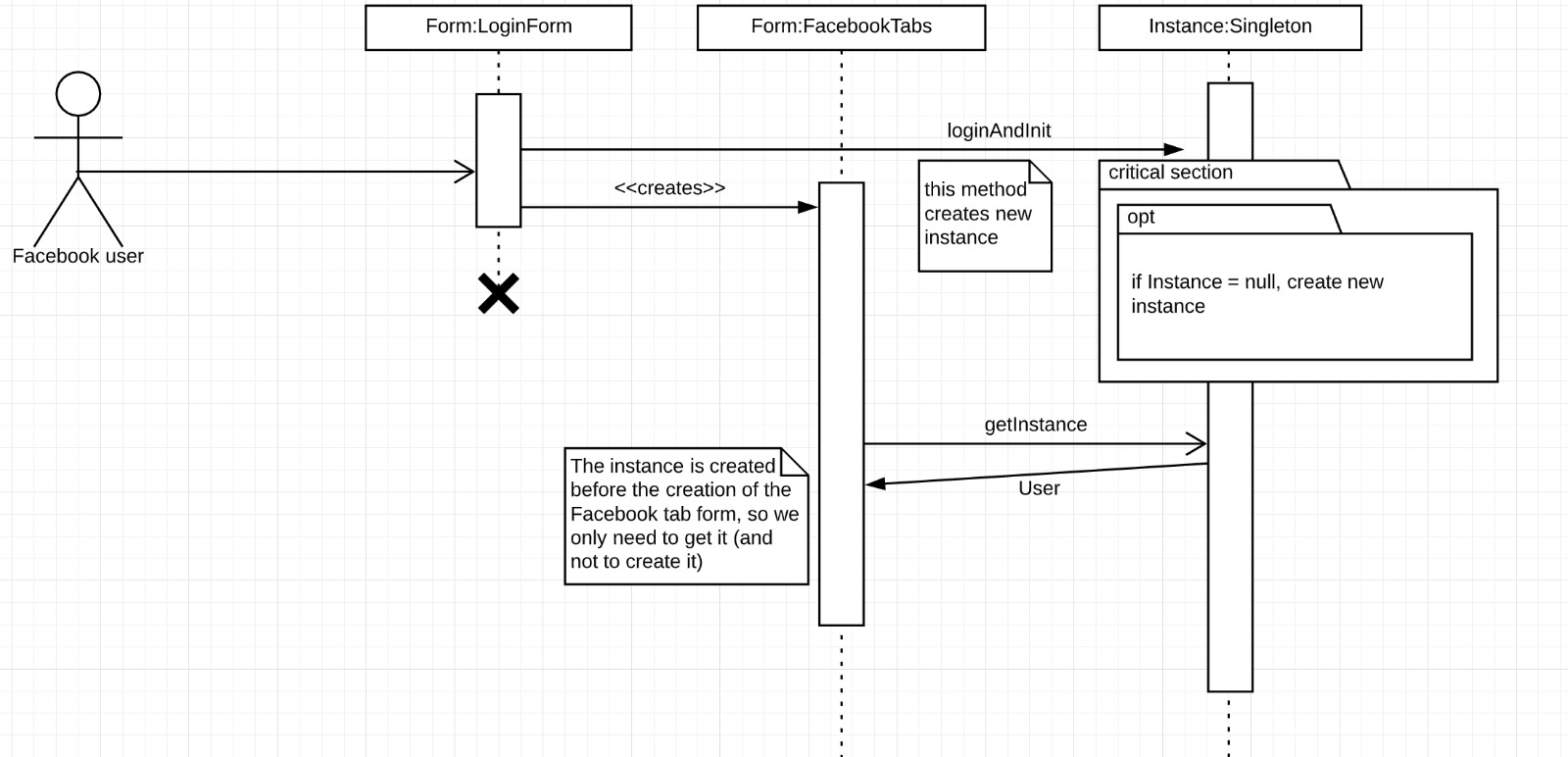
בעת הפניה למחלקת singleton שלנו, אחד מ-2 הדברים הבאים יתרחשו:

1. ה-instance היחיד לא קיים- במקרה כזה אנו מייצרות אותו
2. ה-instance היחיד קיים- במקרה כזה אנו פונות לאובייקט הקיים

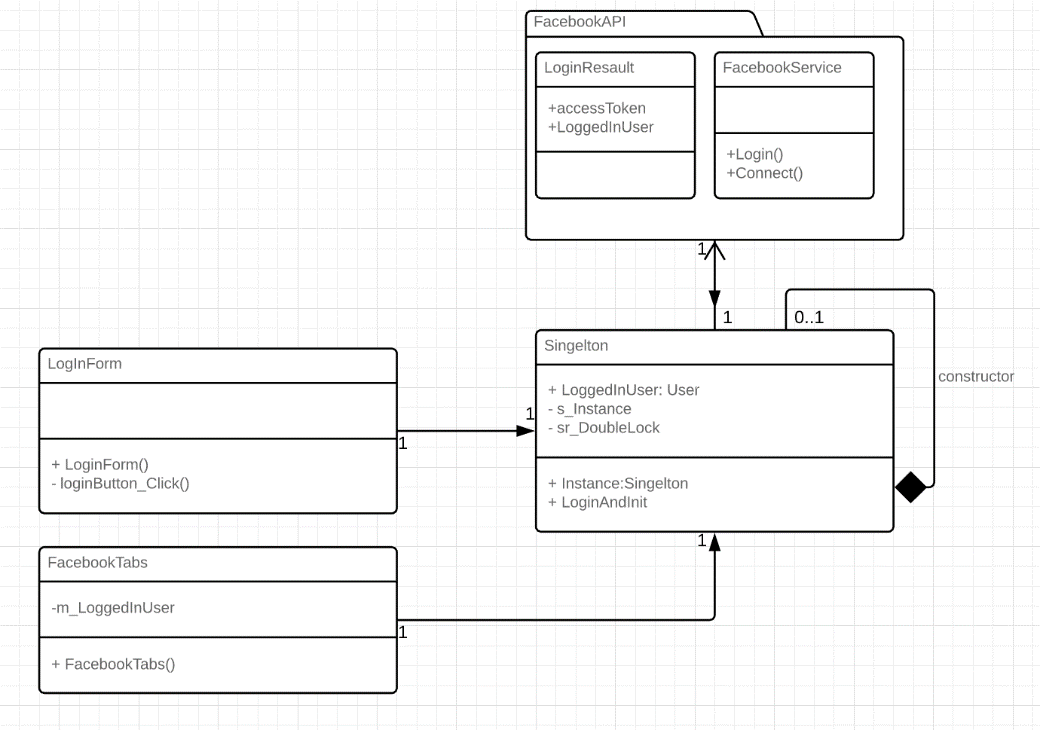
מימשנו את הSingelton באמצעות double check lock כדי שלא יקרה מצב של החלפת משתמשים שעלולה לגרום לבאג במערכת בתכנות מקבילי.

* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]



* Class Diagram



הקבלה לSingleton Pattern:

Client: FacebookTabs , LoginForm

Creator: sealed class Singleton

Product: Instance

### תבנית מס' 3 – Factory Method

* **סיבת הבחירה / שימוש בתבנית:**

המטרה של factory method היא ליצור אובייקטים החולקים ממשק אחיד מבלי להכיר את המחלקות שלהם – ללא תלות בclient.

ראינו כי יצירת הforms השונים של כל פיצר מתבצעת באותה צורה ולכן נממש מחלקה אבסטרקטית שתרכז את יצירת הforms השונים ותנתק אותם מהclient בצורה שאם בעתיד נרצה להוסיף פיצרים נוספים או להוסיף קיימים נצטרך לשנות רק חלק מזערי בfactoryMethod ולא בכל הclient וההפעלה של כל הform זהה ע"י פעולת showDialog().

נשים לב שבעת יצירת factory method הclient לא צריך ליצור כל פעם form חדש – כלומר, אין new() והוא פונה לcreator אשר יוצר את הform לפי הכפתור עליו לחץ

* **אופן המימוש:**

יצרנו 2 מחלקות נוספות לקוד:

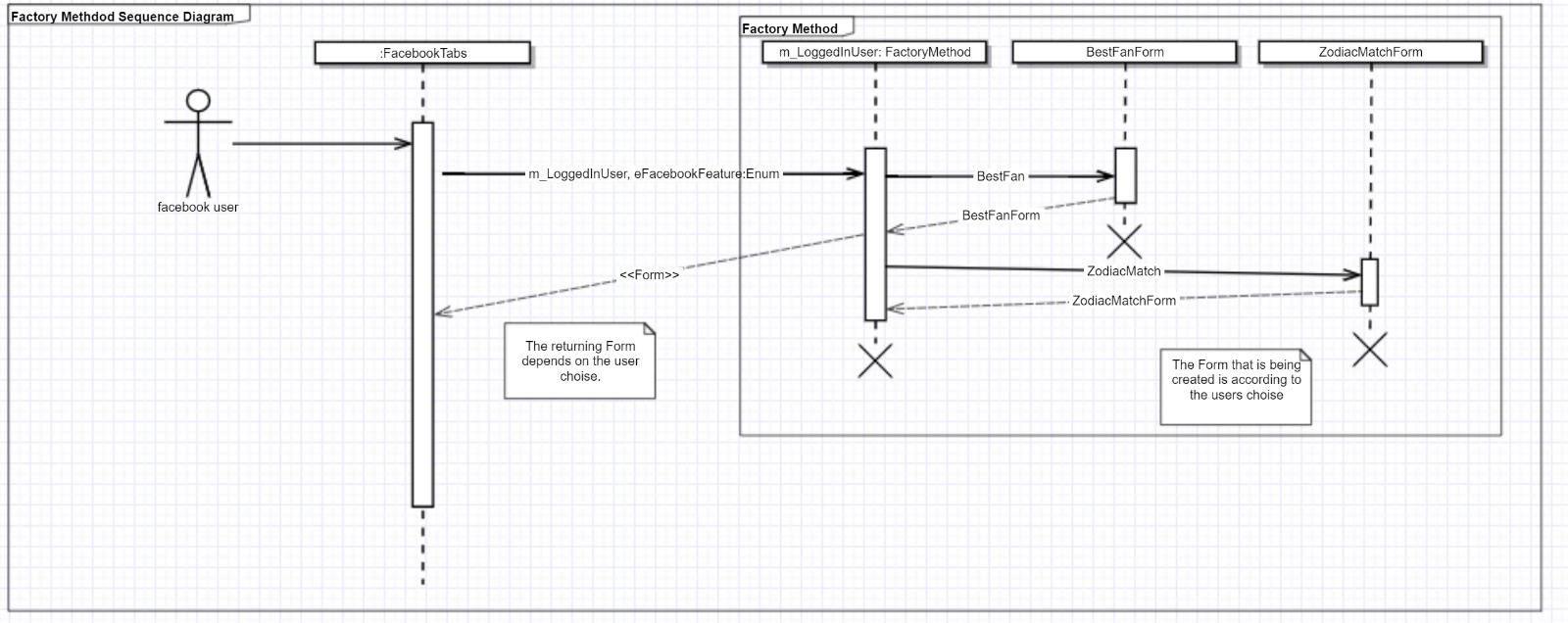
* eFacebookFeature – מחלקת eNum אשר מגדירה את סוגי הform אותם ניתן לייצר.

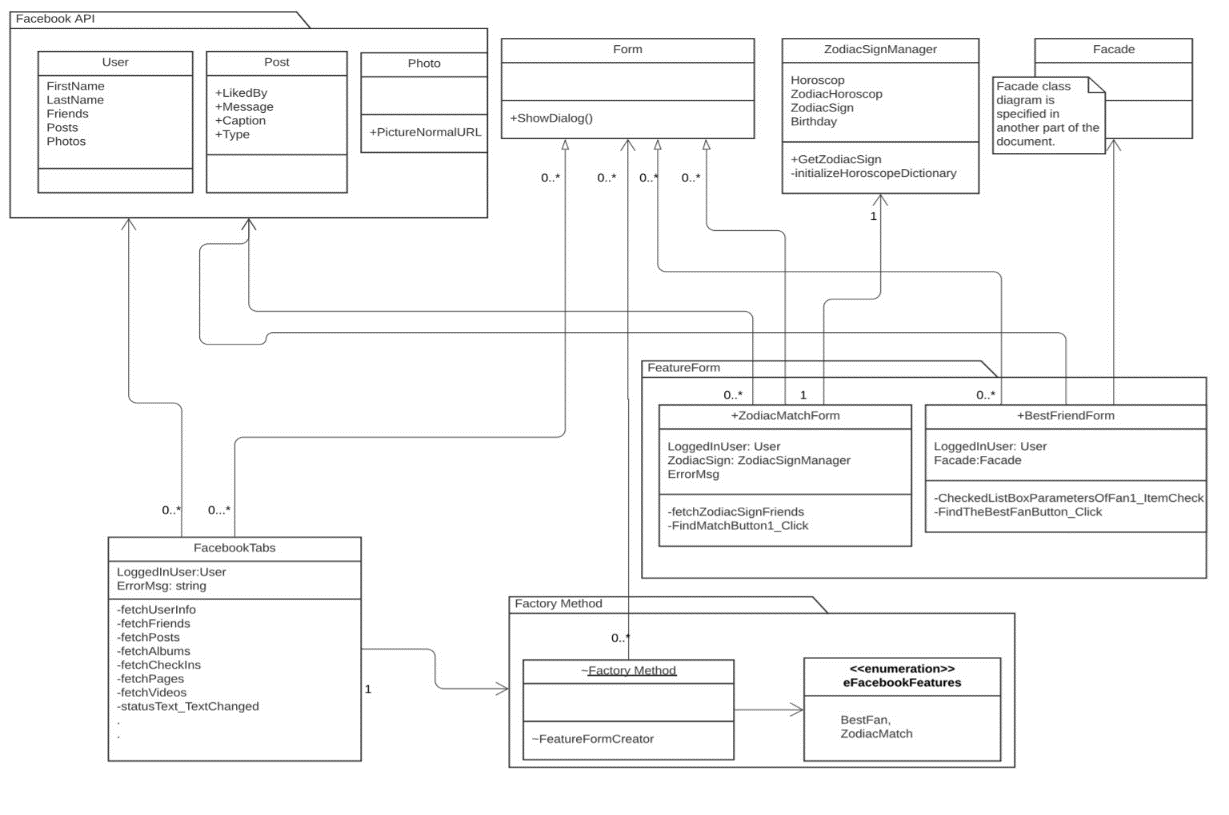
אם בעתיד נרצה להוסיף form נוסף נעדכן זאת במחלקה זו.

* FactoryMethod – מחלקה בה מופעלת המתודה FeatureFormCreator אשר מקבלת את המשתמש שהפעיל את המתודה ואת סוג הform אותו הוא ירצה לייצר ע"י eFacebookFeature ומחזירה את הform הנדרש.

בקוד עצמו , הקריאה למחלקת FactoryMethod תהיה בעת לחיצה על הפיצרים , FindYourBestFanForm\_Click ו ZodiacMatchForm\_Click אשר יקראו למחלקה כדי ליצור את הforms.

* Factory Method Sequence Diagram



* Factory Method Class Diagrm

הקבלה ל-Factory Method Pattern:

Client: FacebookTabs

Creator: internal class FactoryMethod

Product: BestFanForm , ZodiacMatchForm