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

* Most In Common: פיצר שמאפשר למשתמש לבדוק עם מי מהחברים שלו יש לו הכי הרבה במשותף (דפים אהובים, קבוצות, וכו).
* Facebook Match: פיצר שמאפשר למשתמש למצוא התאמה זוגית מרשימת החברים שלו (מוצא רק את החברים שפנויים לקשר), תוך הפעלת פילטרים שיעזרו לו למצוא את הזיווג המתאים.

### עבודה אסינכרונית:

היכן: במחלקה FormMain הCtor מפעיל מתודה createNewFormsWithThreads שתיצור ותנהל תהליכונים.  
כיצד: כאשר עולה הטופס הראשי (MainForm), ישנה מתודה שתפקידה ליצור תהליכונים, שמטרת כל אחד מהם היא לעשות fetch לנתונים הספיציפיים הדרושים לאותו הטופס באופן אסינכרוני. לדוגמא עבור טופס FormEvents תתבצע הבאת נתוני האירועים של המתמש לתוך הטופס.   
מדוע: ע"מ שכאשר הטופס הראשי מתחיל להיטען, נרצה שהאפליקציה תהיה רספונסיבילית ונוכל להנות מתוצאות הביניים. במקום להמתין שכל התתי טפסים יטענו נתונים ורק אז האפליקציה תעלה, כעת האפליקציה קודם תעלה, ואז כל פעם שטופס יהיה מוכן לשימוש נוכל להשתמש בו.

### עבודה עם Data Binding:

היכן: במחלקות: PanelNGB, PanelCustomFull, PanelCustom  
כיצד: על מנת לבצע את הקישור בין הנתונים לבין ה-UI, יצרנו פאנל שבו יושב ה-Binding Source, וקישרנו אותו לנתונים המתאימים.

### תבנית מס' 1 – [Singleton]

* *סיבת הבחירה / שימוש בתבנית:*

בכל טופס חדש שנוצר, נרצה לעשות שימוש בFacebookUser ובנתונים שלו, משום שכל טופס מיוצג ע"י מחלקה נפרדת, נצטרך למסור את הFacebookUser מהFormMain לכל אחד מהטפסים נידרש למסור את הFacebookUser ע"מ שנוכל לבצע fetch לנתונים.  
נרצה שכל אחד מהטפסים יוכל לגשת לFacebookUser מבלי תלות במסירתו ע"י   
רכיב חיצוני (FormMain). בנוסף בחרנו לממש סינגלטון כך שבכל המערכת קיים רק User יחיד שמחובר למערכת ורק איתו עובדים.

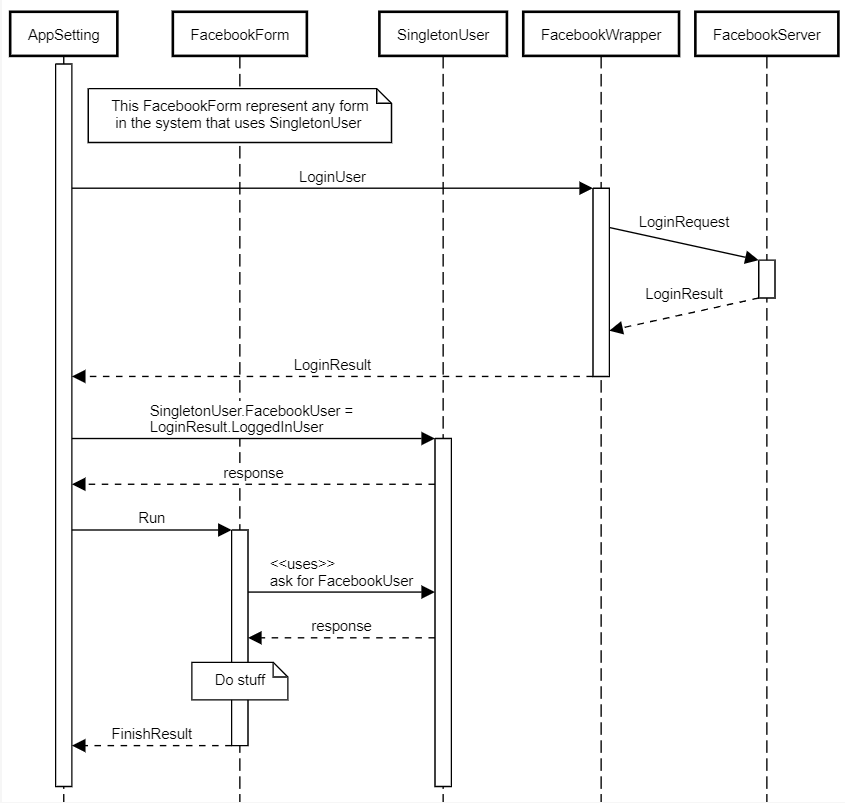
* אופן המימוש:

(תפקידי הרכיבים מפורטים בדיאגרמת המחלקות)

בשלב הLogin לשרתי הפייסבוק, FormLogin אחראי על הבאת נתוני הFacebookUser, והAccessToken שלו. כך שהוא מוסר את הנתונים למחלקה SingletoneUser שיוצרת את המופע היחיד. המופע סטטי, כך שכל מי שרוצה להשתמש בו(כל אחד מהטפסים), יוכל לעשות זאת ע"י SingletonUser.FacebookUser. בנוסף, ניתן להירשם לאירוע של המחלקה SingletonUser, המודיע כאשר מופע נוצר בהצלחה.

ניתן למצוא בקוד: בפרוייקט Logic ישנה מחלקה הקרויה SingletonUser, בנוסף כל Forms עושים שימוש בFacebookUser ולכן גם בתבנית עצמה.

* Sequence Diagram



FacebookServer:

FacebookWrapper:

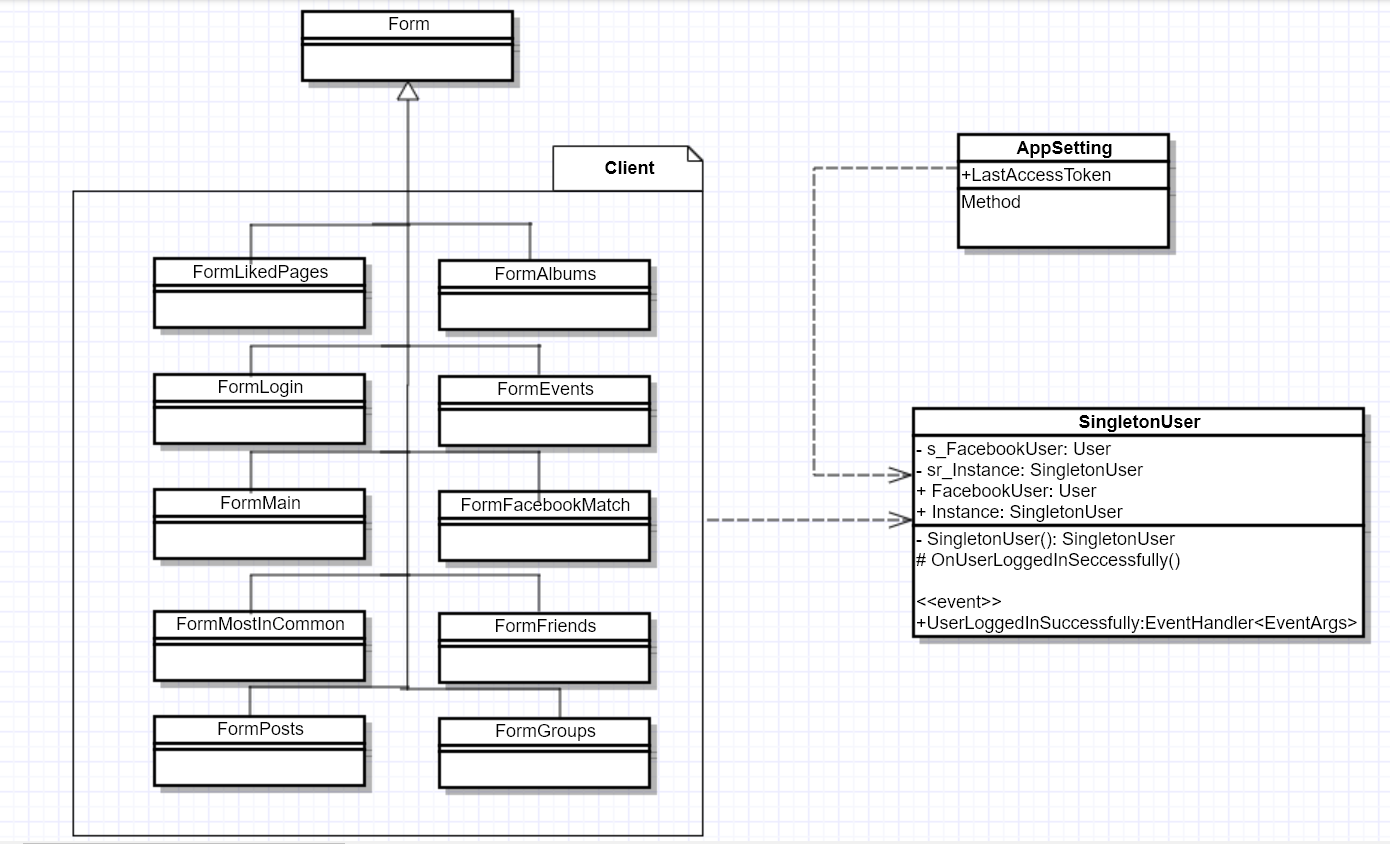
:

SingletonUser:

FacebookForm:

AppSetting:

* Class Diagram



### תבנית מס' 2 – [Factory Method]

* סיבת הבחירה / שימוש בתבנית:

בחרנו להשתמש בFactory Method ע"מ שנוכל לייצר סוגי Panel מותאמים היורשים מPanel ויוצרים משפחה פולימורפית, ובכך לעבוד איתם באופן פולימורפי. כל זאת מבלי שהמשתמש יצטרך להכיר את לוגיקת היצירה של הפאנלים. בנוסף, אם בעתיד נרצה להוסיף פאנלים במקומות נוספים, כבר קיימת לנו מחלקה שיודעת לעשות זאת עבורנו. הצורך נבע משימוש ועיצוב באופן שונה של כל אחד מהPanelים. לדוגמא FormMain צריך להשתמש בPanelCustomFull, וFormMostInCommon- צריך להשתמש בPanelNG.

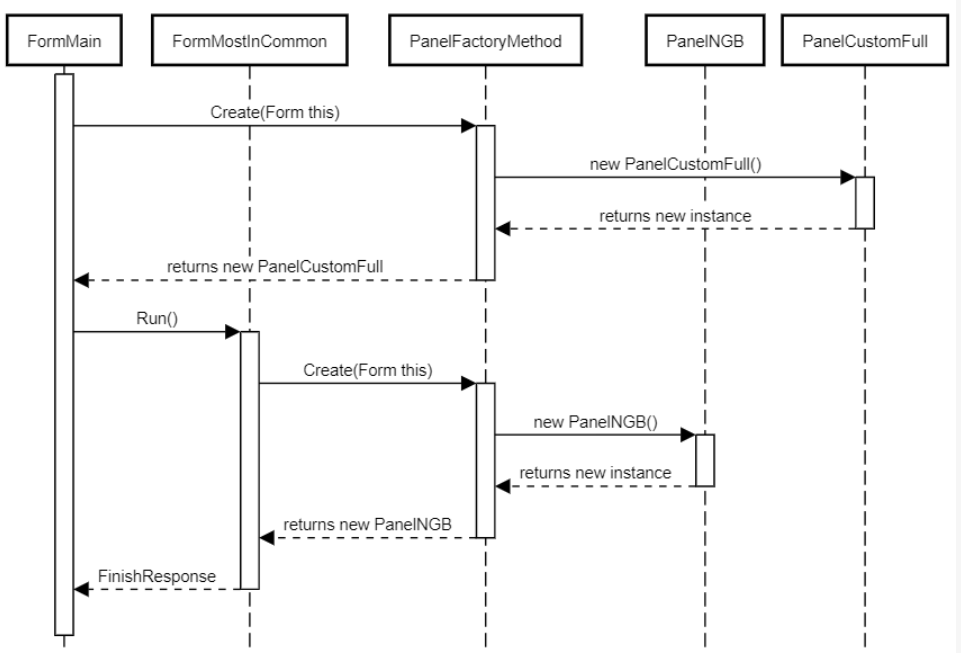
* אופן המימוש:

(תפקידי הרכיבים מפורטים בדיאגרמת המחלקות)

ישנם שני סוגי פאנלים PanelNGB וPanelCosutmFull ששניהם יורשים מפאנל נוסף, PanelCustom ובכך יוצרים משפחה פולימורפית. למחלקה PanelFactoryMethod ישנה מתודה Create שמקבלת Form, כך שהטופס שירצה לקבל פאנל לשימושו האישי, ישלח את עצמו (this), ובכך הFactoryMehod ידע איזה פאנל ליצור עבורו.

ניתן למצוא בקוד: בפרוייקט FacebookApp בתיקייה שנקראת PanelsFactory נמצאים הפאנלים השונים והFactoryMethod. המחלקות העושות שימוש בתבנית הן: FormMain וFormMostInCommon.

* Sequence Diagram



PanelCustomFull:

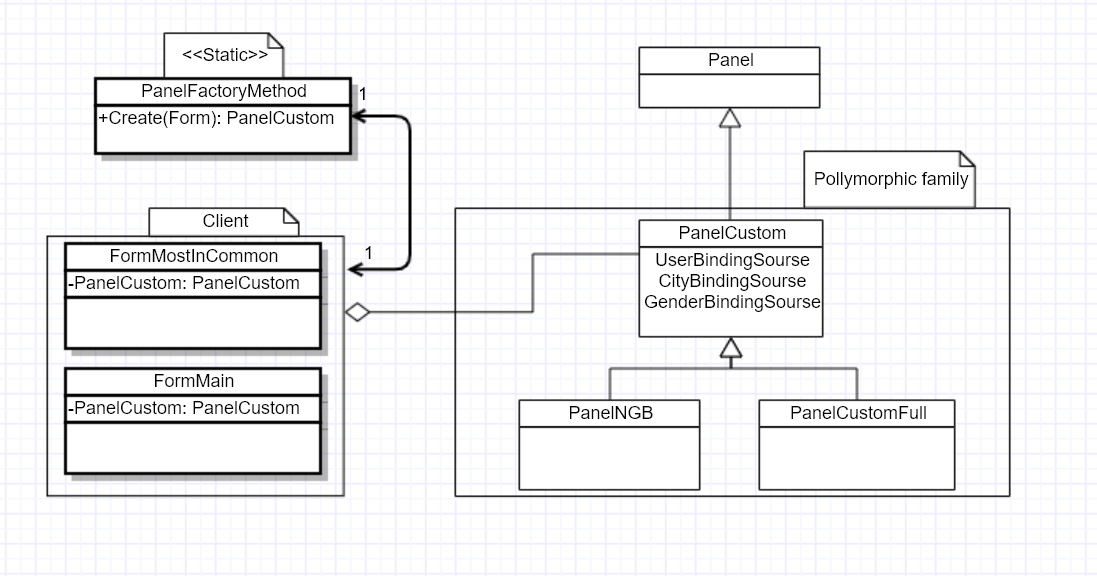
PanelNGB:

PanelFactoryMethod:

FormMostInCommon:

FormMain:

* Class Diagram



### תבנית מס' 3 – [Composite]

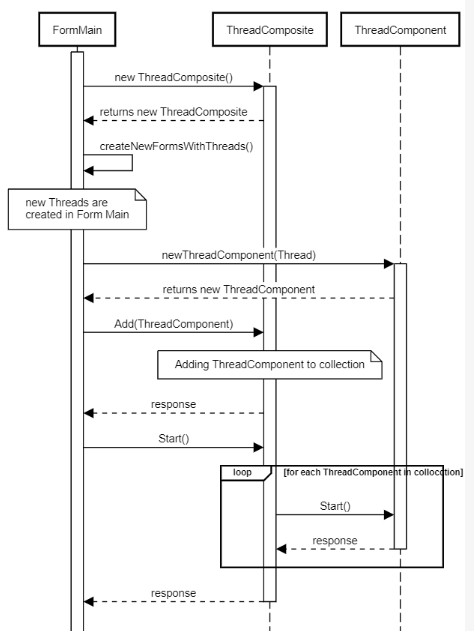
* סיבת הבחירה / שימוש בתבנית:

המטרה היא לאגד Threadים ולהתייחס אליהם כקולקציה הפועלת יחדיו כאילו מדובר ברכיב אחד, לכן מימשנו תבנית Composite שתאפשר הפעלת פעולות על האוסף כולו כאילו מדובר ב ThreadComponent יחיד. פעולות שנרצה להפעיל: Start, Abort, Add, Remove, Join.

(תפקידי הרכיבים מפורטים בדיאגרמת המחלקות)

ניתן למצוא בקוד:  
בפרוייקט FacebookApp ניתן לראות תיקיית מחלקות בשם ThreadsComposite המכילה את הרכיבים המשתתפים בתבנית. בנוסף הקליינט הוא FormMain.  
אופן המימוש: הclient (FormMain) מחזיק אובייקט מסוג IThread, הקרוי r\_ThreadComposite – אשר מקבל ערך של אובייקט ממחלקת .ThreadComposite וכאשר הclient יוצר Threadים הוא תחילה יוצר אובייקט מסוג ThreadComponent המקבל Thread בctor, ולאחר מכן מאגד אותם לתוך r\_ThreadComposite באמצעות פעולת Add. כאשר נרצה להתחיל את התהליך נפעיל על האוסף פעולת Start. באופן דומה נפעיל פעולות Remove, Join, Abort.

* Sequence Diagram



ThreadComponent:

ThreadComposite:

FormMain:

* Class Diagram

