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

* **פיצ'ר ראשון – הצגת מידע מהפייסבוק בצורה טבלאית עם יכולות מיון:**

פיצ'ר זה מאפשר למשתמש לבחור אוסף מידע מסוג כלשהו (תמונות, חברים, פוסטים, עמודים שאהב וכו'...) ולהציגו בצורה טבלאית עם יכולות מיון לפי ערך רצוי מבין הערכים המוצגים.  
כרגע מומשו רק האפשרויות הבאות:

* הצגת תמונות שהמשתמש העלה
* החברים של המשתמש
* העמודים שהמשתמש אהב
* כל הפוסטים בהם המשתמש מתוייג

אך אופן כתיבת הקוד נעשה כך שניתן בעתיד להוסיף אוספים אחרים של מידע בקלות רבה   
(לדוגמא: תמונות שהמשתש תוייג בהן ויש להן יותר מ20 לייקים)

\*פירוט נוסף תחת תבנית מס' 3 – Static Factory Class

מיקום בקוד: תיקיית DataTables וקובץ TabDataTables.cs.

* **פיצ'ר שני – ניתוח החברות של המשתמש עם אחד מחבריו לפי בחירה:**

פיצ'ר זה מאפשר למשתמש לבחור את אחד החברים שלו מרשימת החברים (שגם משתמשים באפליקציה)

לאחר מכן יוכל המשתמש להציג סטטיסטיקות כגון:

* כמה לייקים החבר עשה על תמונות של המשתמש
* רשימת כל התגובות של החבר על תמונות של המשתמש (בלחיצה כפולה תוצג התמונה עם פירוט נוסף על אותה התמונה)
* מהי התמונה האחרונה שהמשתמש והחבר תויגו בה יחדיו

בנוסף יוכל המשתמש למצוא את אוספי התמונות הבאים:

* תמונות שהמשתמש העלה והחבר מתוייג בהן (מקובץ לפי אלבום)
* תמונות שגם המשתמש וגם החבר מתוייגים בהן (מקובץ לפי שם המשתמש שהעלה את התמונה)
* תמונות של החבר שהמשתמש מתוייג בהן (מקובץ לפי אלבום)

\*\* בעת ניסיון להצגת התמונות המשתמש יכול לבחור באיזה אלבומים לחפש במקום לסרוק את כל האלבומים.

מיקום בקוד: קובץ TabFriendshipAnalyzer.cs (החלק הלוגי נמצע בקובץ FriendshipAnalyzer.cs).

### תבנית מס' 1 – [Visitor]

* סיבת הבחירה / שימוש בתבנית:

לטובת שדרוג חווית המשתמש בכל הקשור לתצוגה טבלאית של נתוני המשתמש, כאשר המשתמש לוחץ לחיצה כפולה על שורה בטבלה בחרנו להציג טופס עם יותר פרטים אודות אותו אובייקט.

כל אובייקט מיוצג ע"י הצגת טופס על המסך, אך בכדי לשמור על הפרדה בין לוגיקה לUI רצינו שהאובייקטים הלוגיים (concrete FacebookDataTable) לא יכירו את האופן בו הטופס מוצג אלא יחזיקו רכיב (אותו ניתן להחליף) שעושה זאת.

ברצוניינו לייצר אובייקט אשר יציג מידע על אובייקט כלשהו, כך שהאובייקט אשר מציג את המידע לא יכיר את האובייקט בשלמותו, אלא רק את התכונות שיש לו צורך אליהם. באמצעות כך, נוכל להציג מידע מסוג זהה של אובייקטים שונים באותה הדרך.

* אופן המימוש:

FacebookDataTable – המחלקה שנרצה לייצג מידע ממנה.

IDisplayableObjectHolder – ממשק אשר דורש למממש את התכונה הספציפית אותה נרצה להציג.

FacebookObjectDisplayer – אובייקט אשר מציג מידע של אובייקטים מסוג IDisplayableObjectHolder.

נשים לב, שכאשר FacebookObjectDisplayer מציג מידע מאובייקט מסוג FacebookDataTable, הוא אינו יודע שמדובר באובייקט מסוג FacebookDataTable.

* Sequence Diagram

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

* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

### תבנית מס' 2 – [Strategy]

* סיבת הבחירה / שימוש בתבנית:

ישנו צורך חוזר ונשנה לקבל תמונות מסויימות מאוסף תמונות כלשהו. כאשר בכל מקום יש דרישה שונה לסוג התמונות הדרוש. בכדי להמנע משכפול קוד ותחזוקה של מעבר על כל התמונות ובדיקה האם התמונה מקיימת את הדרישה, נמממש מחלקה אחת שמקבלת מאיתנו את סוג התמונות הדרוש אשר נוכל להחליף סוג זה בכל שלב. לפיכך, אם בעתיד נרצה לבצע סינון מסוג אחר לאוסף תמונות, לא נצטרך לשנות כלום בקוד מלבד להגדיר את המטודה אשר יודעת האם תמונה כלשהי עונה על דרישת המשתמש.

* אופן המימוש:

FilterPhotos – ישנה מטודה אשר מבצעת את הסינון בעזרת מטודה שמבחינה האם התמונה עונה על דרישת המשתמש.

FriendshipAnalyzer - מחזיקה אובייקט מסוג FilterPhotos ומשתמשת ביכולות שלו במטודות PhotosTaggedTogether, GetPhotosFromAlbumsUserIsTaggedIn וכל מטודה שנרצה להוסיף בעתיד אשר דורש תמונות ספציפיות מאוסף תמונות או אלבומים.

TabFriendshipAnalyzer – מחזיקה אובייקט מסוג FriendshipAnalyzer, ומציגה את הנתונים עפ"י דרישת המשתמש.

* Sequence Diagram

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

* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

### תבנית מס' 3 – [Observer]

* סיבת הבחירה / שימוש בתבנית:

כחלק מאופן המימוש של הפיצ'ר בו אנו מציגים מידע מהפייסבוק בצורה טבלאית רצינו שטעינת השורות לטבלה תיעשה באופן אסינכרוני.

כתוצאה מכך עלה הצורך להודיע למשתמש ברגע שהטבלה מסיימת להיטען.

לכן, בחרנו לעשות שימוש בתבנית observer.

* אופן המימוש:

יצרנו ממשק, IRowsPopulatedObserver, המגדיר את המתודה AllRowsUpdated, אותה כל מחלקה המעוניינת "להאזין" צריכה לממש.

הרכיב הלוגי, FacebookDataTable, האחראי על ביצוע פעולת עדכון השורות מחזיק רשימה של מאזינים וכן מממש שתי מתודות המאפשרות להוסיף או להסיר מאזין מהרשימה.

כאשר פעולות טעינת השורות לטבלה מסתיימת, מופעלת המתודה notifyAllRowsPopulatedObservers אשר מודיעה לכל מאזין ברשימת המאזינים להריץ אתAllRowsUpdated שלהם.

\*\* כמובן שניתן היה לממש את התכנית בעזרת Action, וכך גם עשינו במקומות אחרים בקוד, אך בחרנו לעשות זאת כדי לתרגל עבודה עם Interface (וגם כדי לפשט את הדיאגרמות 😊)

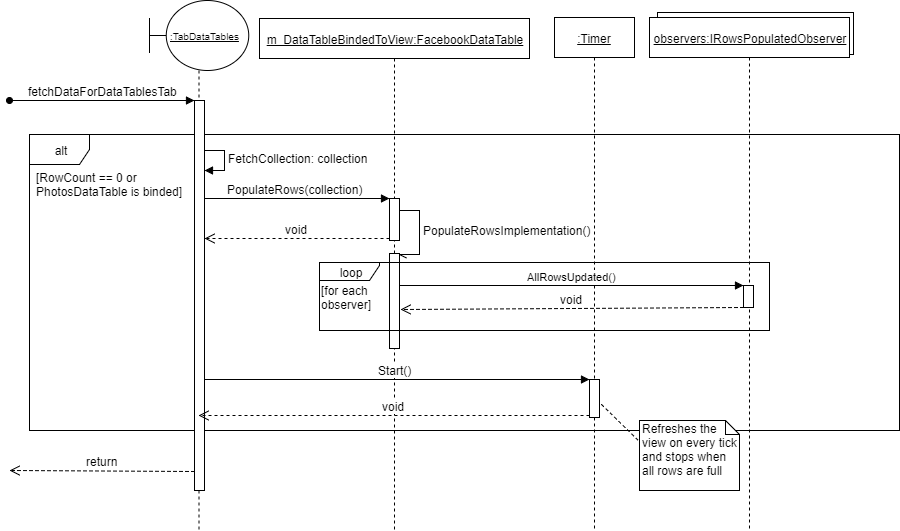
מיקום בקוד:

הפטרן ממומש בקבצים הבאים:

1. DataTables.FacebookDataTable.cs
2. DataTables.IRowsPopulatedObserver.cs
3. Forms.Tabs.TabDataTables.cs

* Sequence Diagram

\*\*\* צריך גם להראות את החלק שבוא נרשמים כמאזין???\*\*\*\*



* Class Diagram

