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

**Friends With Common Interest**

פיצ'ר זה מציג את רשימת החברים איתם למשתמש יש עמודי פייסבוק, להם המשתמש והחבר שלו עשו "לייק". ברשימוה מוצגים כל החברים שיש להם לפחות עמוד אחד משותף.

הפונקציה שמבצעת את הפיצ'ר הנ"ל ממומשת (במטלה הראשונה) במחלקה AppLogic (במטלה הנוכחית ממומשת ב-FacebookUserProxy) והטופס הראשי (FormMain) משתמש בה בעת לחיצה על הכפתור "Click here to see your friends that like similar pages"

**MyEvents**

פיצ'ר זה מציג את רשימת האירועים להם המשתמש עשה Attend. למשתמש יש את האפשרות לסנן את האירועים שמוצגים לפי מיקום ההתרחשות שלהם – מתרחשים אונליין או מתרחשים במקום פיזי.

הפונקציה שמממשת את הפיצ'ר הנ"ל ממומשת (במטלה הראשונה) במחלקה AppLogic (במטלה הנוכחית ממומשת ב-FacebookUserProxy) והטופס הראשי (FormMain) משתמש בה בעת לחיצה על הכפתור "Fetch events". על הכפתור המשתמש ילחץ לאחר בחירה של אופציית ההצגה הרלוונטי.

**Get Nasa's Picture of the Day**

פיצ'ר זה מציג למשתמש תמונה מהחלל שנשלפת מתוך API של NASA. המשתמש יכול לצפות בתמונה של התאריך הנוכחי או בתמונה מתאריך אחר כלשהו שבו יבחר.

הפונקציה שמבצעת את הםיטצ'ר הנ"ל ממומשת במתודות fetchNasaPictureByDate ו-fetchNasaPicToday שבמחלקה FormMain ומשתמשות במתודות מתוך NasaFacade. בשתי המתודות הראשונות שהצגנו FormMain מתשמ בעת לחיצה על אחד מהלחצנים Get Today's Nasa Picture of the Day או Get Nasa's picture of the Day by date.

**שימוש ב-Data Binding**

השתמשנו בפיצ'ר זה בפיצ'ר MyEvents. כמו כן, קישרנו את שמות ה-events לחלק מהשדות שלו והם מוצגים בהתאם לאירוע הנוכחי שהמשתמש בוחר לראות מידע עליו.

השימוש בקוד הוא במחלקת ה-MainForm, במתודה buttonFetchEvents\_Click() שמופעלת בעקבות הלחיצה על הכפתור "Fetch events".

**שימוש בתכנות אסינכרוני**

בקוד יש 2 שימושים עיקריים בתכנות אסינכרוני:

1. בפיצ'ר Get Nasa's Picture Of the Day שהוספנו (בעקבות הערה שקיבלנו על המטלה הקודמת), במתודות fetchNasaPictureByDate ו-fetchNasaPicToday יש שהן מתודות אסינכורניות שמשתמות במתודות GetNasaPicBYDate, GetNasaPicOfTheDay שממומשות ב- Façadeשיוזכר בהמשך. המתודות הנ"ל משתמשות במתודה GetNasaPictureByDate שנמצאת במחלקה NasaApi.

השימוש במתודות האלה נעשה כך על מנת לאפשר למשתמש פונקציונליות של שימוש שוטף ורציף בזמן טעינת תמונות, שלוקח זמן היות והוא תלוי בתשובה של שרת חיצוני (של NASA API). השימוש נעשה באמצעות thread.

1. ב-FormMain, כל הפונקציות שתפקידן להביא מידע ולהציג אותו למשתמש עושות שימוש בתכנות אסינכרוני, מכיוון שהמידע מיובא משרת חיצוני של פייסבוק. זאת באמצעות שימוש ב-threads – כמובן עם שימוש בפעולות מהצורה Invoke new Action כדי שלא תהיה חפיפה בין ה-threads (Cross thread operation). השימוש בתכנות אסינכרוני, באופן דומה לנקודה הראשונה, נעשה כדי לאפשר למשתמש להינות משימוש שוטף באפליקציה בעת שהנתונים נטענים.

### תבנית מס' 1 – [Template Method]

* **סיבת הבחירה / שימוש בתבנית**:

רצינו לאפשר הצגת תמונות למשתמש מתוך אלבום בצורות שונות. לכן, בחרנו להתשמש ב-template method אשר מאפשר לנו להוסיף צורות וסדר הצגה שונים לאלבומים. זאת על ידי הגדרה מחדש של מקומות באלגוריתם ללא שינוי במבנה שלו.

באפליקציה שלנו, התמונות מהאלבום מוצגות בצורה רנדומלית – בכל לחיצה על הכפתור הרלוונטי אנחנו מגרילים תמונה חדשה מהרשימה שמחזיקה את כל התמונות באלבום.

בחרנו להשתמש ב-pattern הנ"ל היות ואם בעתיד נרצה להוסיף שיטת סידור נוספת לא נצטרך לשנות את הקוד ולכן הבחירה ב-pattern הנ" תורמת לנו בשיפור התחזוקתיות וההרחבה של הקוד.

* **אופן המימוש**:

על מנת לממש את ה-pattern יצרנו מחלקה אבסטרקטית בשם PhotoSelectionAlgorithmBase. במחלקה הנ"ל יצרנו מתודה אבסטרטית של סידור אשר בה יש injection point. המחלקה PhotoRandomizer שיורשת מהמחלקה האבסטרטית ממשמת את המתודה הנ"ל ומממשת את שיטת סידור התמונות. כתוצאה מכך המחלקה האחראית לסידור התמונות מחזירה ל-client את התוצאה בצורה הרצויה.

"שחקנים":

Client: FacebookUserProxy

AbstractClass: PhotoSelectionAlgoritmBase

Subclass: PhotoRandomizer

* Chart

  Description automatically generated**Sequence Diagram**
* **Diagram

  Description automatically generatedClass Diagram**

### תבנית מס' 2 – [Strategy]

* **סיבת הבחירה / שימוש בתבנית**:

רצינו לאפשר למשתמש בזמן ריצה לבחור לפי איזה שיטה (אלגוריתם) של מיון המשתמש רוצה לקבל את הרשימה של החברים שלו שעשו "לייקים" לאותם עמודים. התבנית מאפשרת למשתמש לבחור אחת משתי השיטות מיון של הנתונים. הראשון הוא סדר אלפבתית מהקטן לגדול (ascending) והשני הוא ההפך (descending).

בחרנו להשתמש בתבנית Strategy בגלל שאם בעתיד נרצה להוסיף עוד שיטת מיון נוספת אז נוכל לעשות את זה בלי לשנות את הקוד. לכן זה עושה את ה קוד יותר extensibily וגם reusable כי גם אפשר להשתמש בשיטת המיון במקומות אחרים בגלל שזה קיים בנפרד.

* **אופן המימוש**:

יצרנו interface בשם ISortListOfFriendsStrategy שממומש על ידי שתי מחלקות (שתי מחלקות עם אלגוריתם מיון שונה בכל אחת מהן):

1. SortListOfFriendsByAscendingStrategy
2. SortListOfFriendsByDescendingStrategy

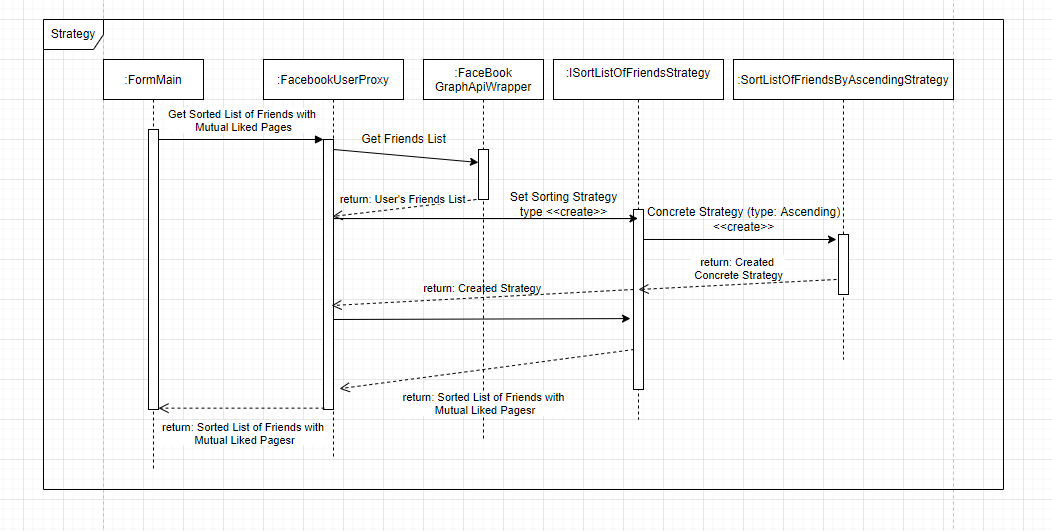
בנוסף, יש לנו את FacebookUserProxy שיש לו מופע של ISortListOfFriendsStrategy והמימוש שלו נבחר בזמן ריצה. זה קורה כאשר המשתמש לוחץ על ה-comboBox ובוחר איזו שיטת מיון הוא רוצה לפני שהוא לוחץ על שליפת רשימת החברים שעשו "לייקים" לאותם עמודים. לאחר שהמשתמש בחר ב-Ascending או Descending ולחץ על הכפתור הרלוונטי, מופעלת המתודה setSortStrategy שיוצרת מופע של מחלקת המיון המבוקשת ואז עושים מיון לפני השיטה (האלגוריתם) שלה בעזרת המתודה Sort שהפרמטר שלה הוא המופע של ה- concrete strategy).

"שחקנים":

Context (Client): FacebookUserProxy

Strategy: ISortListOfFriendsStrategy

Concrete strategies: SortListOfFriendsByAscendingStrategy, SortListOfFriendsByDescendingStrategy

* **Sequence Diagram**
* **Diagram

  Description automatically generatedClass Diagram**

### תבנית מס' 3 – [Observer]

* **סיבת הבחירה / שימוש בתבנית**:
* **אופן המימוש**:
* **Sequence Diagram**
* **Class Diagram**