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

1. Like Rated:

הפיצ'ר הוא לגבי תמונות באלבומים.

אנו נציג למשתמש את התמונה הכי פופולרית שקיבלה את הכי הרבה לייקים ואת הכוזבת ביותר שקיבלה הכי פחות לייקים.

וכן את התמונה הכי מדוברת שעליה יש את מספר התגובות הרב ביותר, והתמונה עם הכי פחות תגובות.

בנוסף נציג לו סטטיסטיקה לגבי מספר הלייקים והתגובות שהוא קיבל בממוצע עבור התמונות, ואת תאריך ההעלאה שלהן.

כמו כן, ניתן בפיצ'ר לבצע סינון לפי השנים בה המשתמש העלה את התמונות לFacebook לטווח מסויים להשוות לפיו.

2. Posts Counter:

הפיצ'ר הוא לגבי פוסטים שהמשתמש שלח.

נציג למשתמש גרף בו מופיע מספר הפוסטים שהמשתמש פרסם בכל חודש בשנה, בגרף יש את חודשי השנה על ציר ה-X ועל צריך ה-Y מספר פוסטים.

ניתן לבצע סינון לפי השנה בו פורסמו הפוסטים או וגם לפי החודש שבו הם פורסמו.

בנוסף אפשר לצפות בכמות פוסטים שנתית לפי חודשים בלבד.

\*את הפיצ'רים ניתן למצוא בקוד בפרויקט UI -> Forms -> Features.

### תבנית מס' 1 – Singleton

* סיבת הבחירה / שימוש בתבנית:

סיבת הבחירה בתבנית זו היא כיוון שישנה מחלקה FacebookUserFetcher שאחראית על משיכת נתונים מהמחלקה FacebookService.

אין צורך שכמה מחלקות יימשכו מידע מה-FacebookService, כלומר אין צורך ליותר ממופע יחיד עבור FacebookUserFetcher, לכן ניצור ממנו מופע יחיד.

* אופן המימוש:

המחלקה FacebookUserFetcher מיצרת מופע יחיד של עצמה עם הגנה מפני thread-ים ומנגנון של double lock.

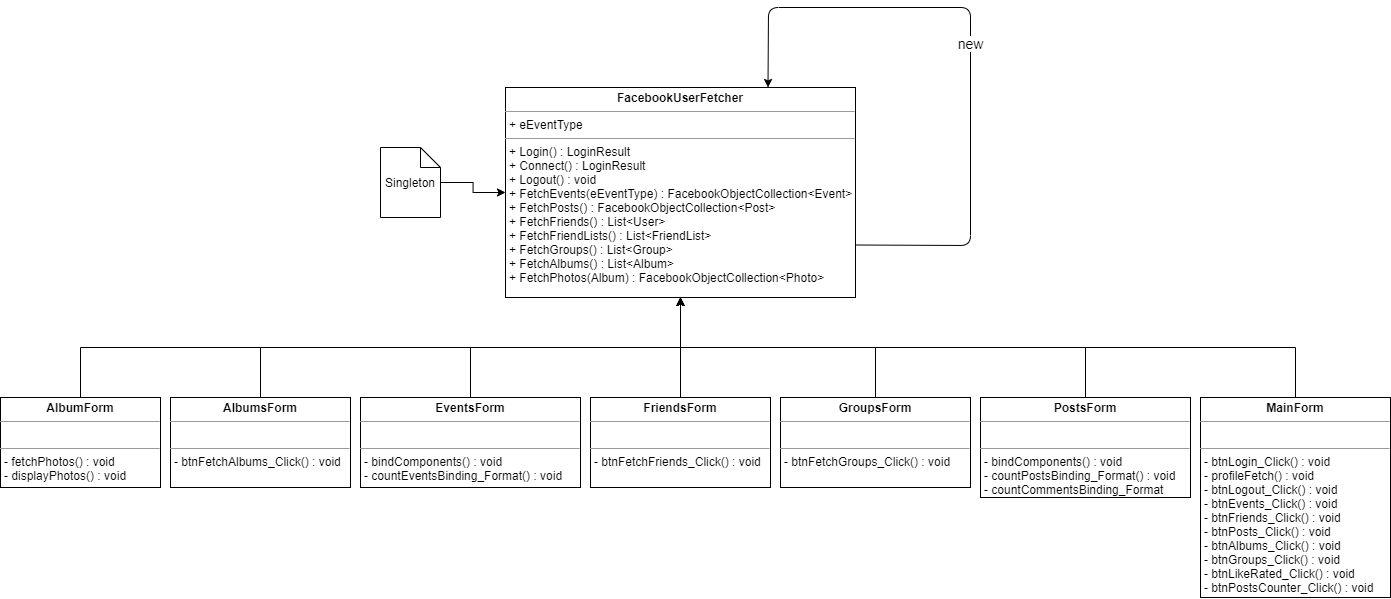
במחלקה יש משתנה סטטי m\_Instance ובאמצעות המשתנה Instance נבדוק אם כבר קיים מופע של m\_Instance, אם יש אז נחזיר אותו אחרת נייצר אחד חדש.

מיקום

\* ניתן למצוא בפרויקט Logic -> FacebookUserFetcher.

* 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 – Factory Method

* סיבת הבחירה / שימוש בתבנית:

בחרנו להשתמש בתבנית הנ"ל מכיוון שישנה בתוכנית משפחה פולימורפית של טפסים שהיצירה שלהם נעשת ב-Client.

* אופן המימוש:

יצרנו מחלקה חדשה שאחראית על יצירת הטפסים המחלקה החדשה נקראת FormFactory ובתוכה מטודה אחת סטטית שמקבלת שני פרמטים שלפיהם יוצרת את הטופס המתאים. כתוצאה מכך המחלקה לא מכירה את האוביקטים שהיא יוצרת ואת תהליך היצירה שלהם.

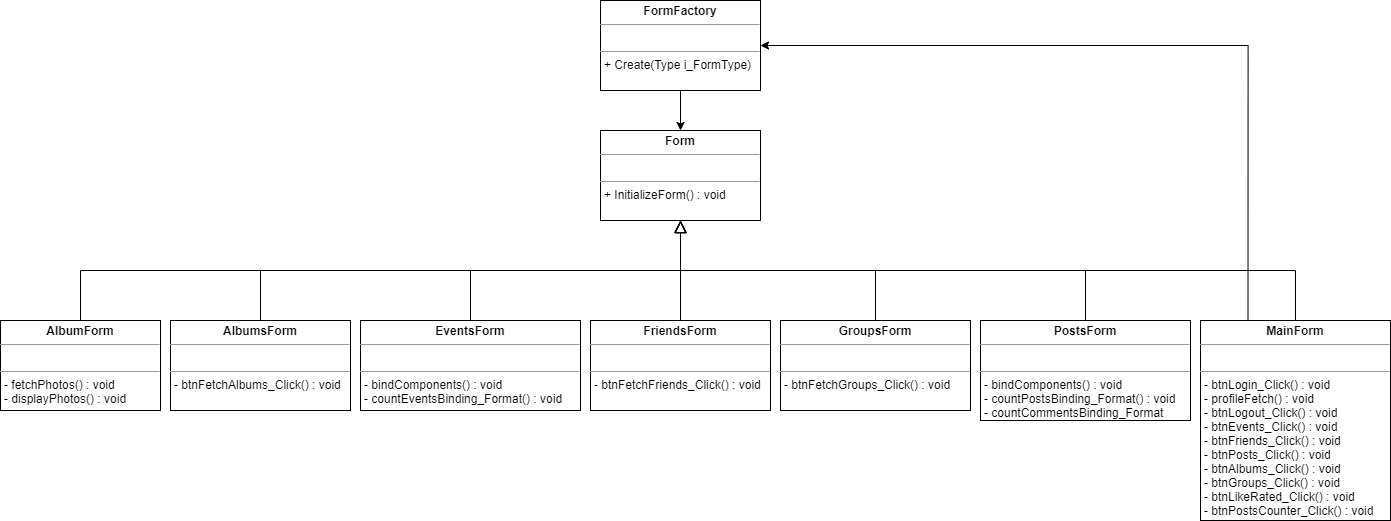
המחלקה MainForm היא המחלקה המייצגת את תפקיד ה Clientובאמצעות שימוש במטודה המתאימה נחליף בין הטפסים.

\* ניתן למצוא בפרויקט UI -> תיקייה Factory Method.

* Sequence Diagram

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

* Class Diagram



המקבילה ל-Product היא: Form

המקבילות ל-ConcreteProduct הן: AlbumForm, AlbumsForm, EventsForm, FriendsForm, GroupsForm, PostsForm, MainForm.

המקבילה ל-ConcreteCreator היא: FormFactory

### תבנית מס' 3 – [Facade]

* סיבת הבחירה / שימוש בתבנית:

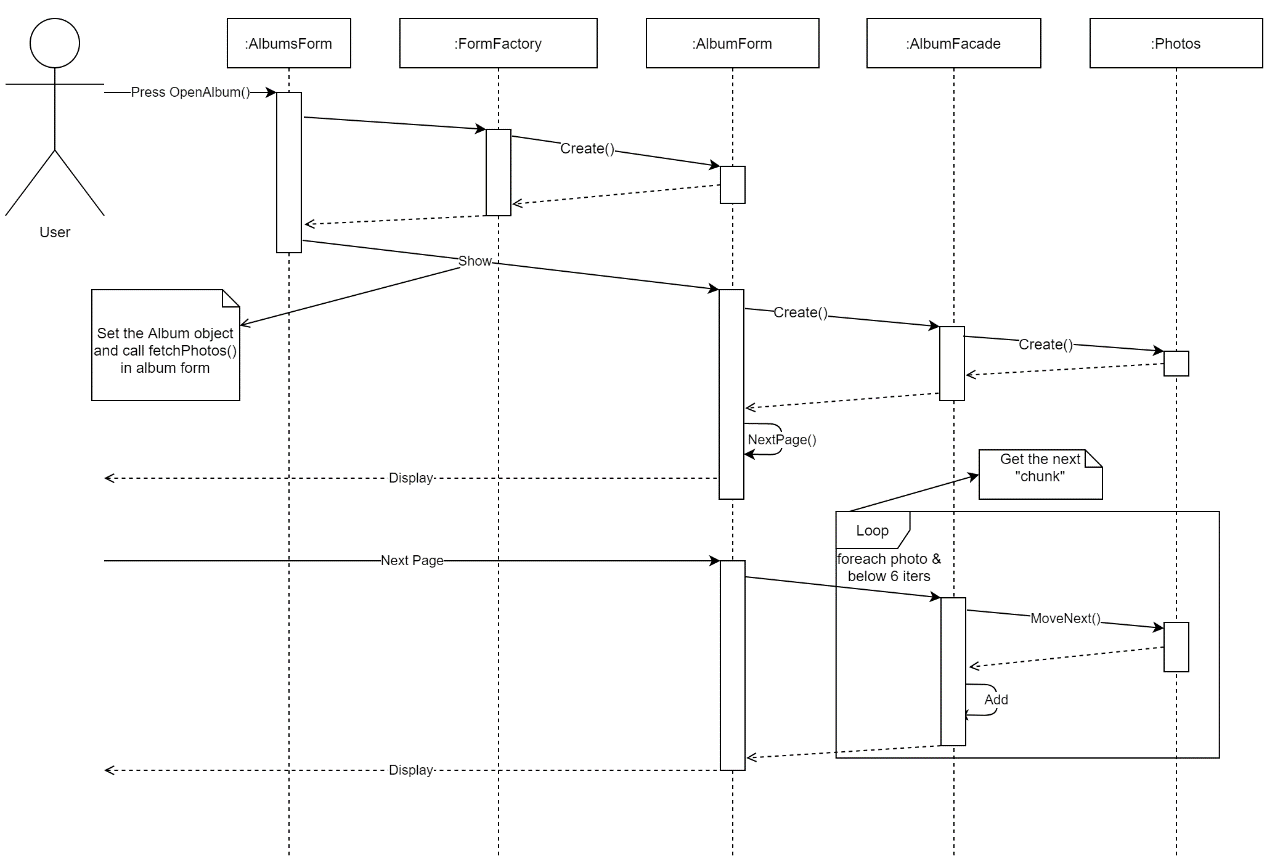
ה UI ביצע הרבה לוגיקה שנלקחה מהמנוע ועובדה כדי שתוכל להציג למשתמש את הפלט, לכן השתמשנו ב Facade בכדי שיבצע את החישוב הנ"ל לפני שהוא מגיע לUI ובכך הUI יהיה נקי יותר מפעולות לוגיות.

* אופן המימוש:

המחלקה בתכליתה צריכה להכיל אוסף תמונות ויהיה ניתן לייבא ממנה chunk-ים של תמונות. לכן נחזיק אוסף של תמונות (שנקבל מהמנוע) כך שיהיה ניתן לקבל את ה chunk הבא או הקודם מהמחלקה, ולבדוק האם בכלל קיים כזה. ומכאן שהUI לא צריך להתעסק עם חישוב ה chunk-ים והאם קיימים בכלל, אלא הוא משתמש בהם באופן ישיר מה Facade.

\* ניתן למצוא בפרויקט Logic -> AlbumFacade

* Sequence Diagram



* Class Diagram

Diagram

Description automatically generated

עבודה אסינכרונית:

ממשנו את FacebookUserFetcher כך שיתמוך ב multi threading בעזרת לוקים במקומות שנראו בעייתיים מבחינת מולטי-טראדד.

* במחלקה AlbumsForm ממשנו במטודה onload את הפעלת המטודה fetchAlbumsOnLoad בטראד נפרד.

המטודה פונה אל FacebookUserFetcher במטרה לחלץ את כל האלבומים בצורה אסינכרונית, ומוסיפה אותם לתוך הרשימה המוצגת ב UI, כדי לא לפגוע בחווית המשתמש כיוון שפעולה זו לוקחת זמן.

* במחלקה EventsForm ממשנו במטודה onload את הפעלת fetchEventsOnLoad.

שם ממשנו את לקיחת המידע אודות סוג הevent הרצוי, ופננינו אל FacebookUserFetcher בכדי לקבל את כל ה events הללו, ושינינו את ה DataSource כדי לבצע binding חדש בהתאם ל events החדשים.

כנ"ל ממשנו בצורה אסינכרונית כיוון שפנייה לשרתים ואסיפת המידעים מהם לוקחת זמן יקר בו המשתמש לא יכול לבצע שום דבר.

עבודה עם Data Binding: