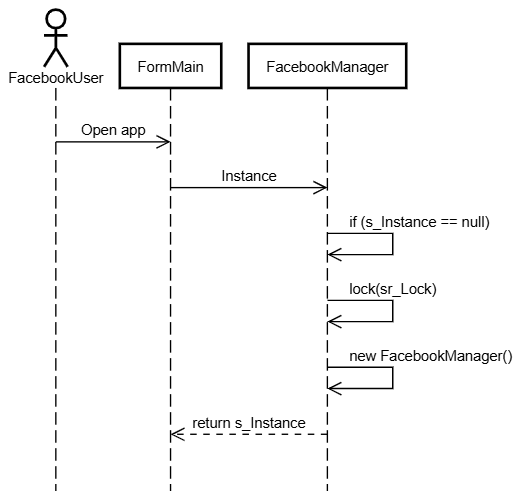
### תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל הקודם:

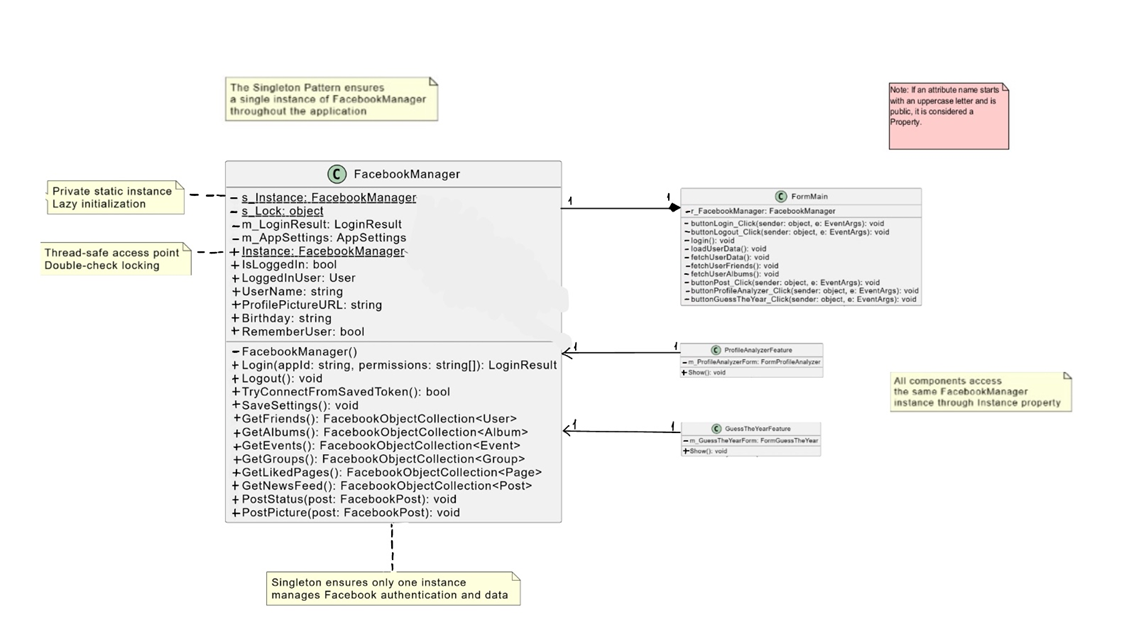
* Profile Analyzer
  + פיצ'ר שמנתח את פרופיל המשתמש ומציג מידע מקיף על הפעילות שלו בפייסבוק:
    - סטטיסטיקות כלליות (מספר לייקים, חברים, אירועים, פוסטים, סרטונים ותמונות).
    - התמונה שקיבלה הכי הרבה לייקים והתמונה שקיבלה הכי מעט.
    - רשימות חברים מסוננות (חברים שמדברים את אותה שפה, מאותה עיר, עם אותו יום הולדת).
* Guess The Year
  + משחק אינטראקטיבי שבו המשתמש צריך לנחש באיזו שנה צולמה תמונה מהאלבומים שלו:
    - המשחק מציג תמונה אקראית מהאלבומים של המשתמש.
    - מוצגות 4 אפשרויות בחירה של שנים שונות.
    - המשתמש צריך לבחור את השנה הנכונה שבה צולמה התמונה.
    - המשחק שומר מעקב אחר תשובות נכונות ושגויות.
    - בסיום מוצג סיכום של הביצועים.

### תבנית מס' 1 – Singleton

* סיבת הבחירה / שימוש בתבנית:
  + נדרש מופע יחיד שינהל את החיבור לפייסבוק לאורך כל חיי האפליקציה.
  + צריך לשמור מידע גלובלי על המשתמש המחובר (LoginResult).
  + חשוב למנוע יצירת מופעים מרובים כדי למנוע בעיות סנכרון וניהול משאבים.
  + נדרשת גישה נוחה מכל מקום באפליקציה למידע על המשתמש המחובר (לדוגמה ב ProfileAnalyzerFeature וב GuessTheYearFeature ניגשים ל LoggedInUser).
  + שמירת הגדרות המשתמש (Remember Me) צריכה להיות מרוכזת במקום אחד.
* אופן המימוש:
  + שימוש ב-sealed class למניעת ירושה.
  + קונסטרקטור פרטי למניעת יצירת מופעים מבחוץ.
  + שימוש ב-double-check locking למניעת בעיות thread-safety.
  + שמירת המופע היחיד בשדה סטטי.
  + חשיפת property סטטי Instance לגישה למופע.
  + שמירת מצב החיבור והגדרות המשתמש במופע היחיד.
  + המימוש ממוקם בתיקיית Singletons.
* Sequence Diagram





* Class Diagram

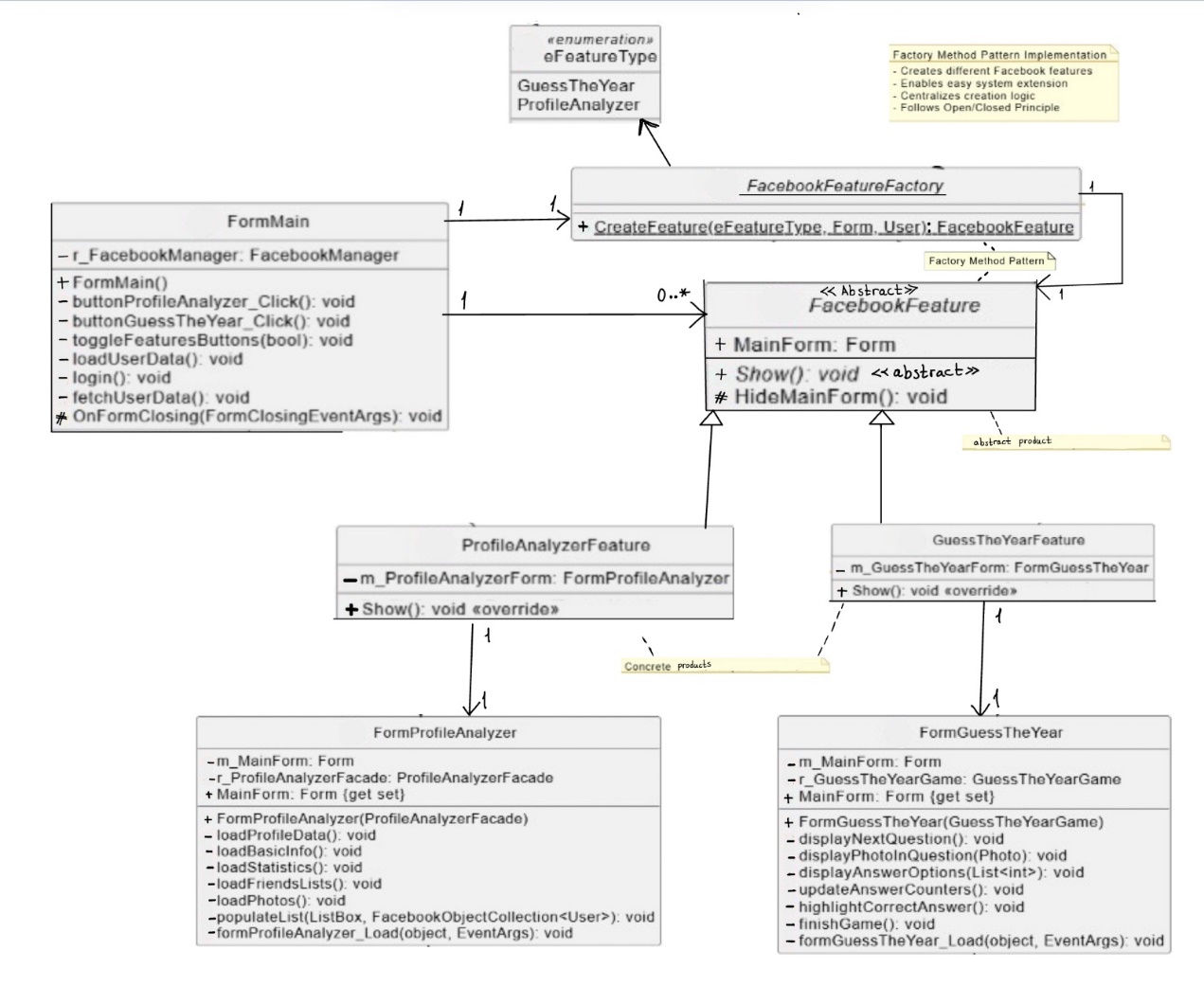
### תבנית מס' 2 – Factory Method

* סיבת הבחירה / שימוש בתבנית:
  + נדרשה דרך אחידה ליצירת פיצ'רים שונים (GuessTheYear, ProfileAnalyzer).
  + רצינו להפריד בין הלוגיקה של יצירת הפיצ'רים לבין השימוש בהם.
  + מאפשר הוספת פיצ'רים חדשים בקלות בלי לשנות את הקוד הקיים.
  + מסתיר את המורכבות של יצירת הפיצ'רים מה-FormMain.
  + מבטיח שכל הפיצ'רים יקבלו את אותם פרמטרים (MainForm) בצורה אחידה.
* אופן המימוש:
  + מחלקת בסיס אבסטרקטית FacebookFeature.
  + מחלקת FacebookFeatureFactory סטטית שאחראית על יצירת כל הפיצ'רים במערכת.
  + enum לסוגי הפיצ'רים השונים.
  + מחלקות ספציפיות לכל פיצ'ר שיורשות מ-FacebookFeature.
  + העברת הפרמטרים הנדרשים (Form) בזמן היצירה.
  + המימוש ממוקם בתיקיית Factories.
* Sequence Diagram

A diagram of a company

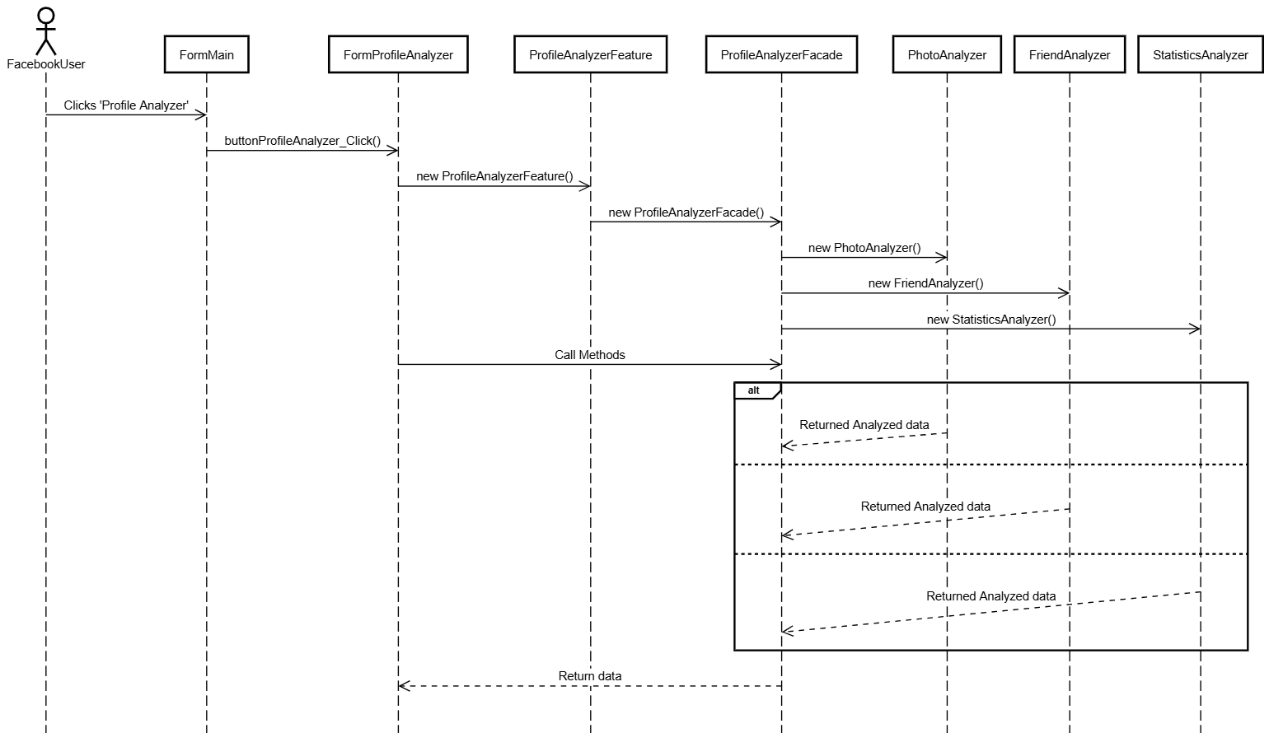
Description automatically generated



* Class Diagram

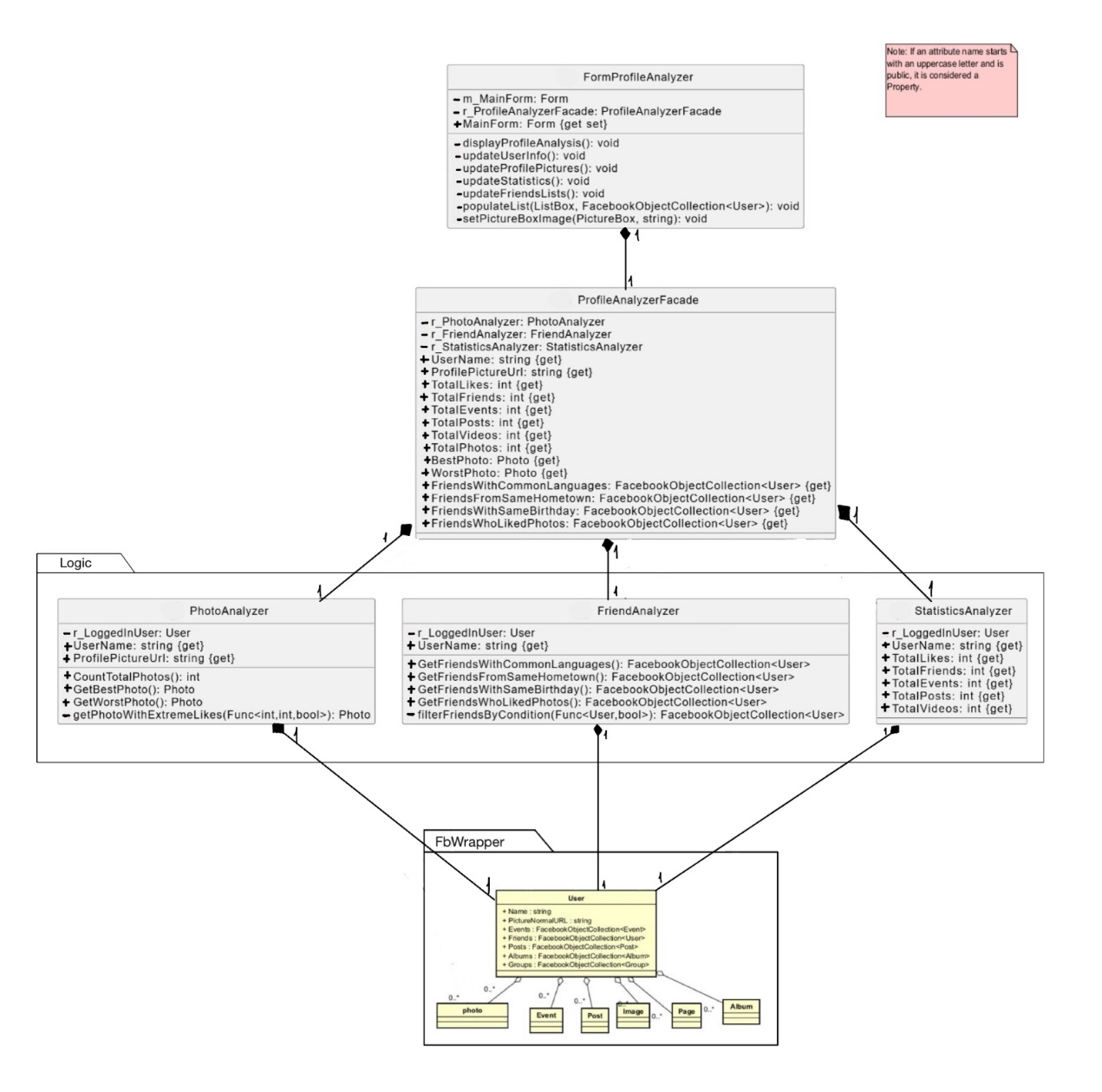
### תבנית מס' 3 – Facade

* סיבת הבחירה / שימוש בתבנית:
  + ניתוח פרופיל המשתמש דורש אינטראקציה עם מספר תת-מערכות מורכבות.
  + רצינו להפריד את הלוגיקה המורכבת של ניתוח הפרופיל מה-UI.
  + נדרש ממשק פשוט שיסתיר את המורכבות של הניתוח מ-FormProfileAnalyzer.
  + כל תת-מערכת (חברים, תמונות, סטטיסטיקות) היא מורכבת בפני עצמה.
  + מאפשר שינויים בלוגיקת הניתוח בלי להשפיע על ה-UI.
  + מקל על הטיפול בשגיאות ועל התחזוקה של הקו
* אופן המימוש:
  + ה - Facade מחזיק מופעים של כל תת-המערכות הלוגיות.
  + חשיפת ממשק פשוט דרך properties ומתודות.
  + הסתרת המורכבות של תת-המערכות.
  + העברת User בקונסטרקטור לכל תת-מערכת.
  + שימוש בread-only fields למניעת שינויים במצב ה – Facade.
  + FormProfileAnalyzer מתקשר רק עם ה – Facade.
  + המימוש ממוקם בתייקית ה Facades.
* Sequence Diagram



sd



* Class Diagram

### עבודה אסינכרונית עם ממשק משתמש

FormMain:

במסך הראשי של האפליקציה, השתמשנו בתכנות אסינכרוני בכמה מקומות מרכזיים. בתהליך ההתחברות (login), אנחנו מבצעים את הבקשה לפייסבוק ב - thread נפרד כדי לא לתקוע את הממשק בזמן ההמתנה לתשובה. כשמתקבלת תשובה חיובית, אנחנו טוענים את נתוני המשתמש (loadUserData) שגם הוא מתבצע בצורה אסינכרונית. בנוסף, כל הטעינה של רשימות (חברים, אלבומים, קבוצות ועמודים) מתבצעת ב - threads נפרדים כדי לאפשר טעינה במקביל ושמירה על UI רספונסיבי. כשמשתמש בוחר פריט מאחת הרשימות, טעינת התמונה המתאימה גם היא מתבצעת ב - thread נפרד.

FormProfileAnalyzer:

במסך ניתוח הפרופיל, כל הטעינה של המידע מתבצעת בצורה אסינכרונית. המידע מחולק לקטגוריות שונות (מידע בסיסי, סטטיסטיקות, רשימות חברים ותמונות) וכל קטגוריה נטענת ב - thread נפרד. הסיבה לכך היא שכל קטגוריה דורשת מספר בקשות לפייסבוק שיכולות לקחת זמן. הפיצול ל - threads מאפשר לנו להציג את המידע בהדרגה כשהוא מגיע, במקום להמתין שהכל יטען. בנוסף, אנחנו משתמשים ב - Invoke כדי לעדכן את ה - UI בצורה בטוחה כשמגיע מידע חדש.

FormGuessTheYear:

במשחק "ניחוש השנה", אנחנו משתמשים בתכנות אסינכרוני בעיקר בטעינת התמונות והצגת השאלות. כל שאלה חדשה דורשת טעינה של תמונה מפייסבוק, ולכן זה מתבצע ב - thread נפרד. גם כשהמשתמש עונה תשובה, הבדיקה והמעבר לשאלה הבאה מתבצעים ב - thread נפרד כדי לאפשר אנימציות חלקות (למשל הדגשת התשובה הנכונה) ומעבר חלק בין השאלות. זה גם מאפשר לנו לשלוט בתזמון של הצגת המשוב למשתמש והמעבר לשאלה הבאה.

### עבודה עם Data Binding

**השימוש העיקרי ב-Data Binding נעשה ב-FormMain, שם יצרנו קישור בין אוספי המידע מפייסבוק לרכיבי התצוגה באמצעות Data Sources.**

FormMain:

כשמשנים את השם ב Albums Editor, נעשה Two-Way Data Binding והמידע משתנה אוטומטית ב – Data Source.

כשמשנים את השם, תאריך היום הולדת והאימייל ב Friends Editor, נעשה Two-Way Data Binding והמידע משתנה אוטומטית ב – Data Source.

מיקום בקוד:

* FacebookWinFormsApp/UI/FormMain.cs
  + בפונקציה fetchFriends - קישור רשימת החברים ל – ListBox.
  + בפונקציה fetchAlbums - קישור רשימת האלבומים ל - .ListBox

הגדרות Data Sources:

* FacebookWinFormsApp/Properties/DataSources
  + הוספנו Data Source עבור Album.
  + הוספנו Data Source עבור FriendList.

השימוש ב-Data Binding מאפשר:

* קישור אוטומטי בין אוספי הנתונים לרכיבי התצוגה.
* שימוש ב-DisplayMember להצגת השם של כל פריט.
* עדכון אוטומטי של התצוגה כשהמידע משתנה.
* תמיכה מובנית בסינון וחיפוש.