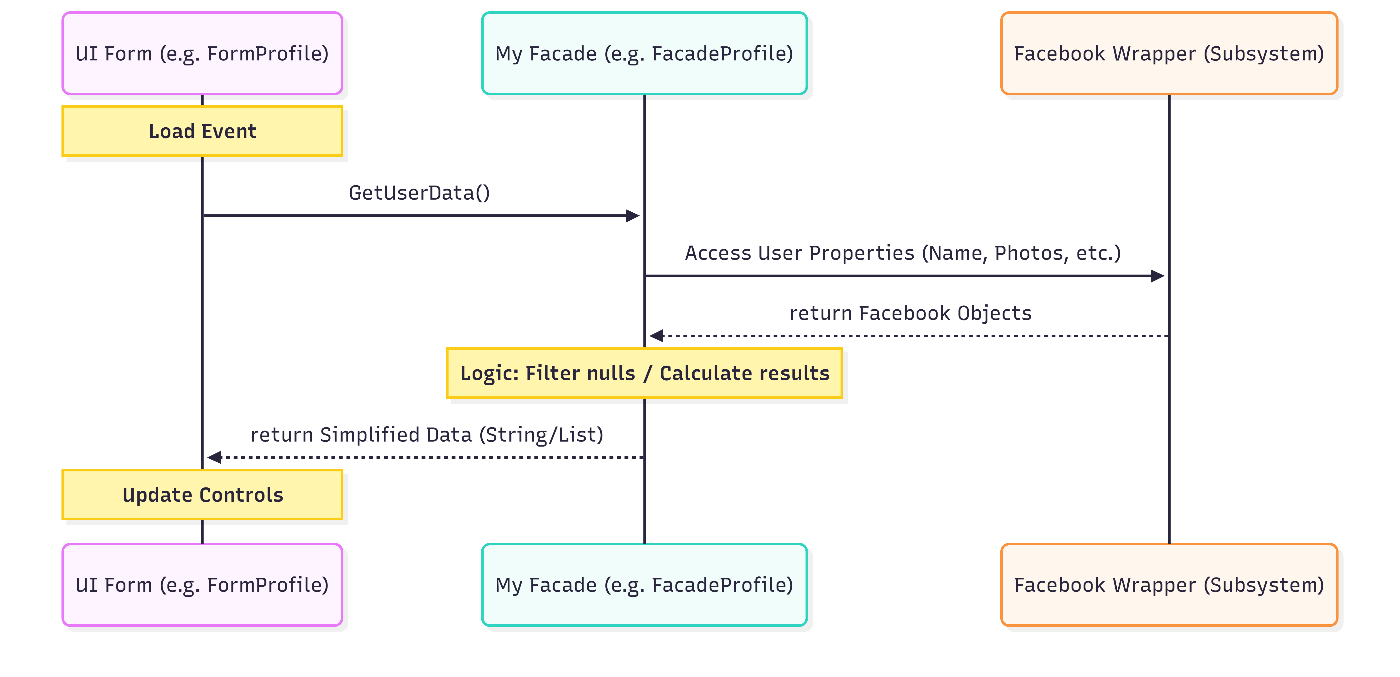
### תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל הראשון:

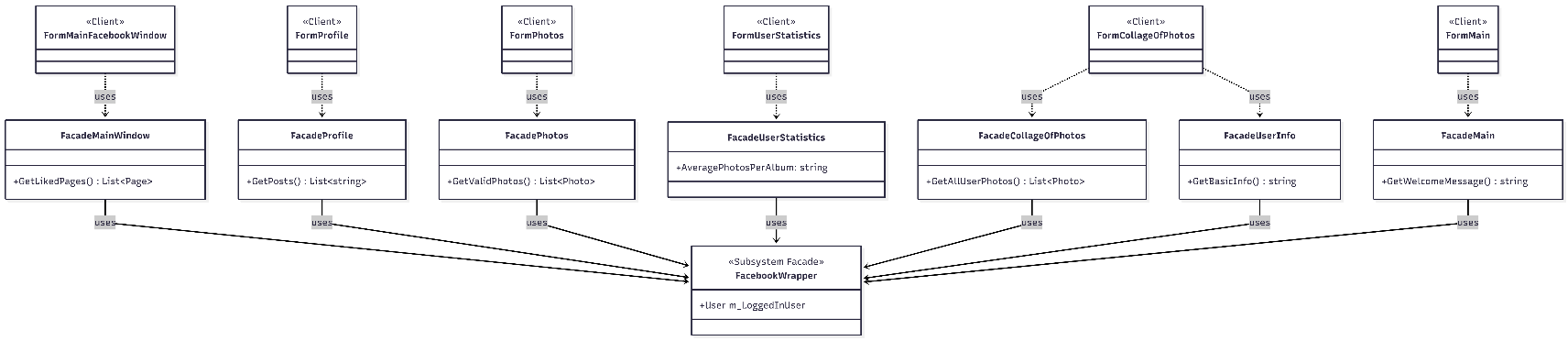
* Collage of moments: קולאז' דינמי של 4 תמונות המתחלפות אוטומטית בכל 2 שניות. עם טעינת הטופס, המערכת שולפת את כל התמונות מכל האלבומים של המשתמש ויוצרת רשימה מרוכזת. טיימר פנימי עובר על הרשימה ומעדכן את תיבות התמונה (PictureBoxes). המשתמש יכול לחזור לתפריט הראשי בכל עת באמצעות כפתור.
* User Statistics: הצגת נתוני סטטיסטיקה על פעילות המשתמש בפייסבוק. נתונים על פוסטים, אלבומים, תמונות ולייקים. סטטיסטיקות מוצגות: כמות חברים, כמות פוסטים, כמות תמונות ממוצעת לאלבום והתמונה שקיבלה הכי הרבה לייקים ועוד. הפיצ'ר מאפשר ניווט חזרה לדף הבית או המשך לדפי הפרופיל והתמונות.

### תבנית מס' 1 – Facade

* סיבת הבחירה / שימוש בתבנית: הצורך נבע מהרצון להוציא את הלוגיקה מה- Forms למקום מרכזי. ה Facadeמבודד את הטפסים מהפורמט של פייסבוק, כך שאם בעתיד פייסבוק ישנו את מבנה הנתונים שלהם, נצטרך לעדכן רק את מחלקות הfacade בזמן שממשק המשתמש ישאר בלי שינויים.
* אופן המימוש: תבנית ממומשת באמצעות מחלקות FacadeMainWindow FacadeProfile, FacadePhotos, FacadeUserStatistics, FacadeCollageOfPhotos, FacadeUserInfo, FacadeMain כל מחלקת Facade מחזיקה בתוכה את המופע של המשתמש m\_LoggedInUser ומנהלת מולו את האינטראקציה. היא מספקת לטופס מתודות פשוטות, למשל GetLikedPages, GetUserName, GetAveragePhotosPerAlbum שמחזירות מידע בלי שהטופס יצטרך להכיר את הלוגיקה. ה- Facade מרכז את כל הלוגיקה בתוכו. הטפסים נשארים קצרים ואחראים רק על הצגת הנתונים למשתמש וניהול אירועים בUI.
* Sequence Diagram

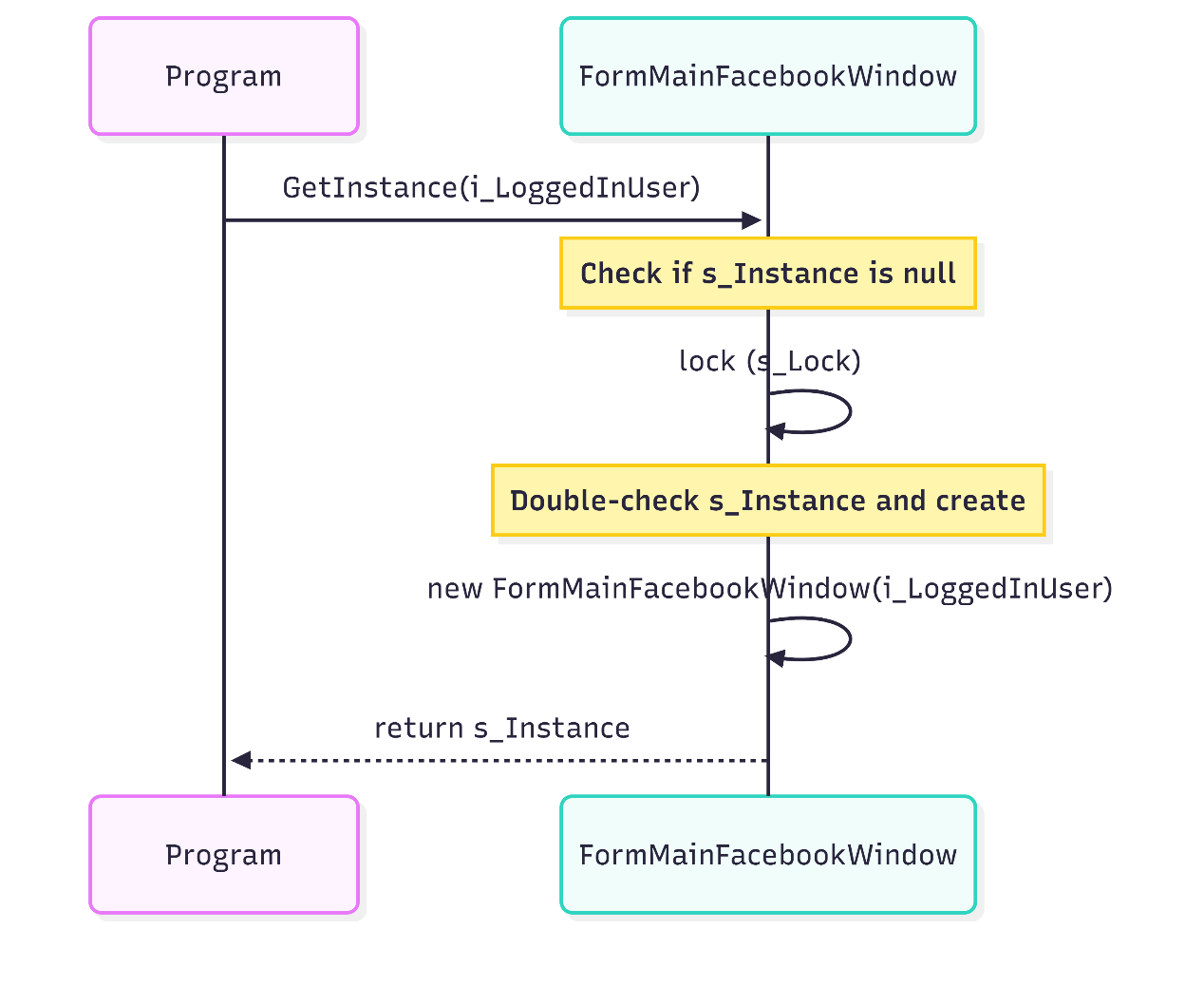


* Class Diagram

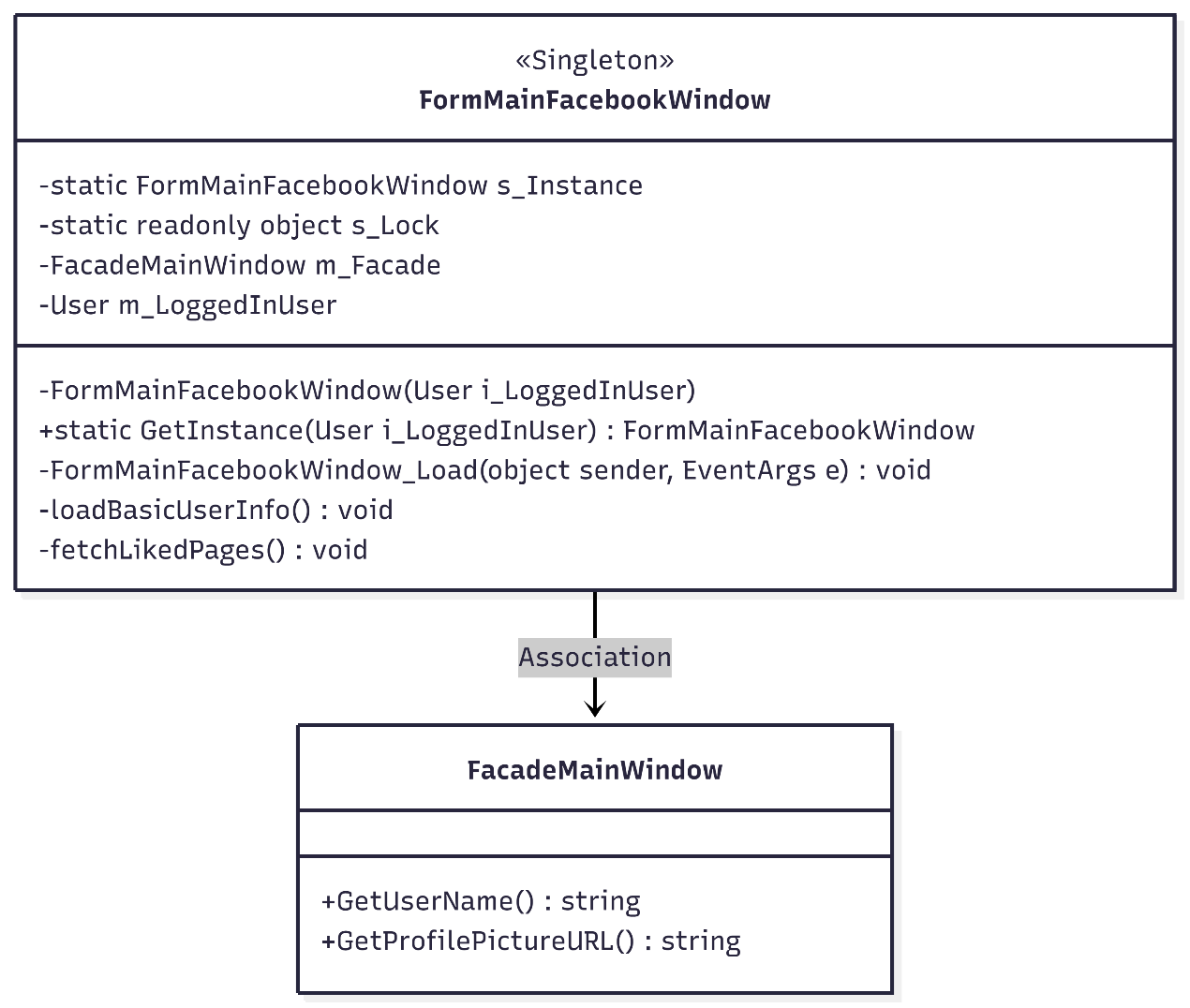


### תבנית מס' 2 – Singleton

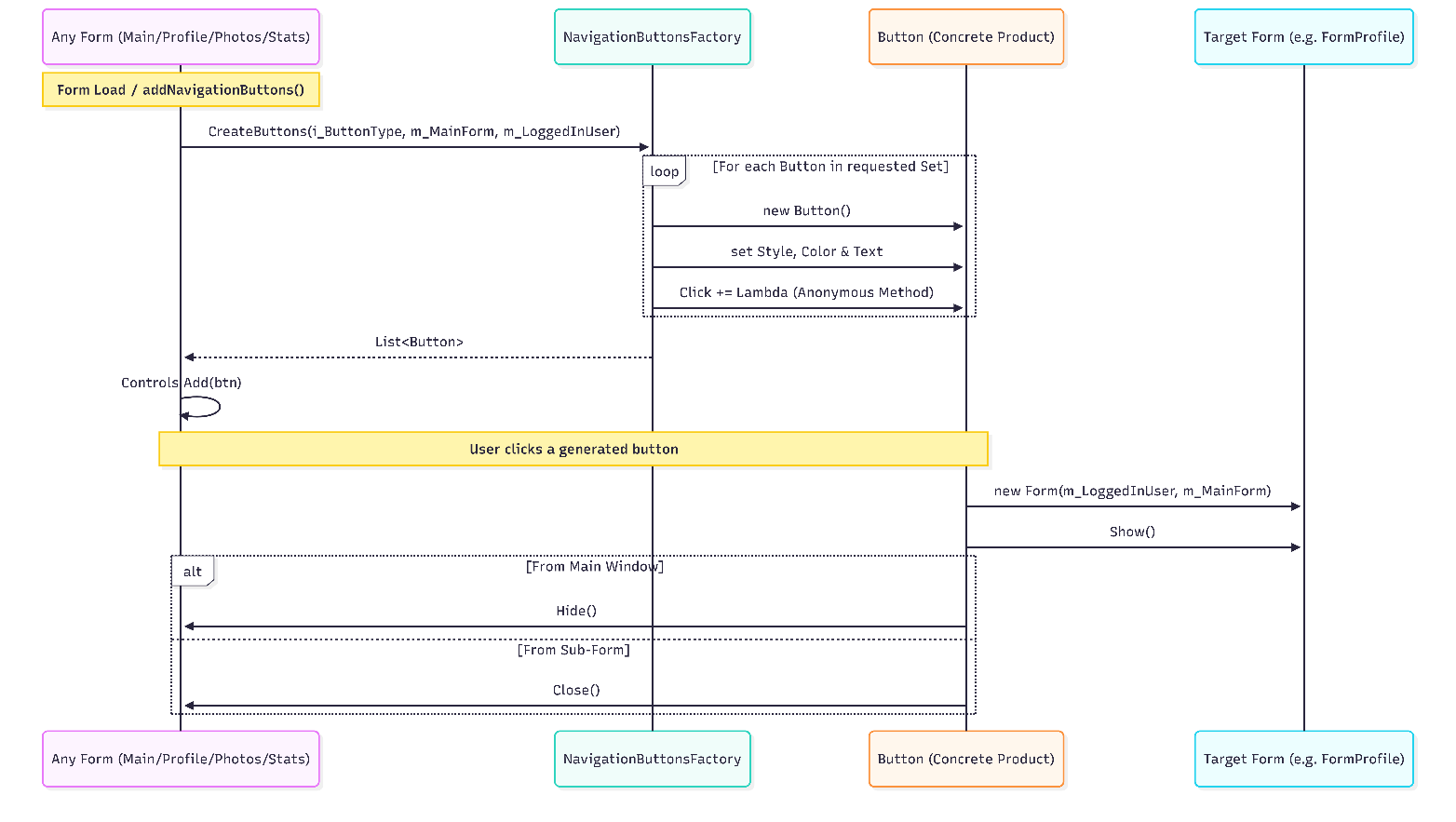
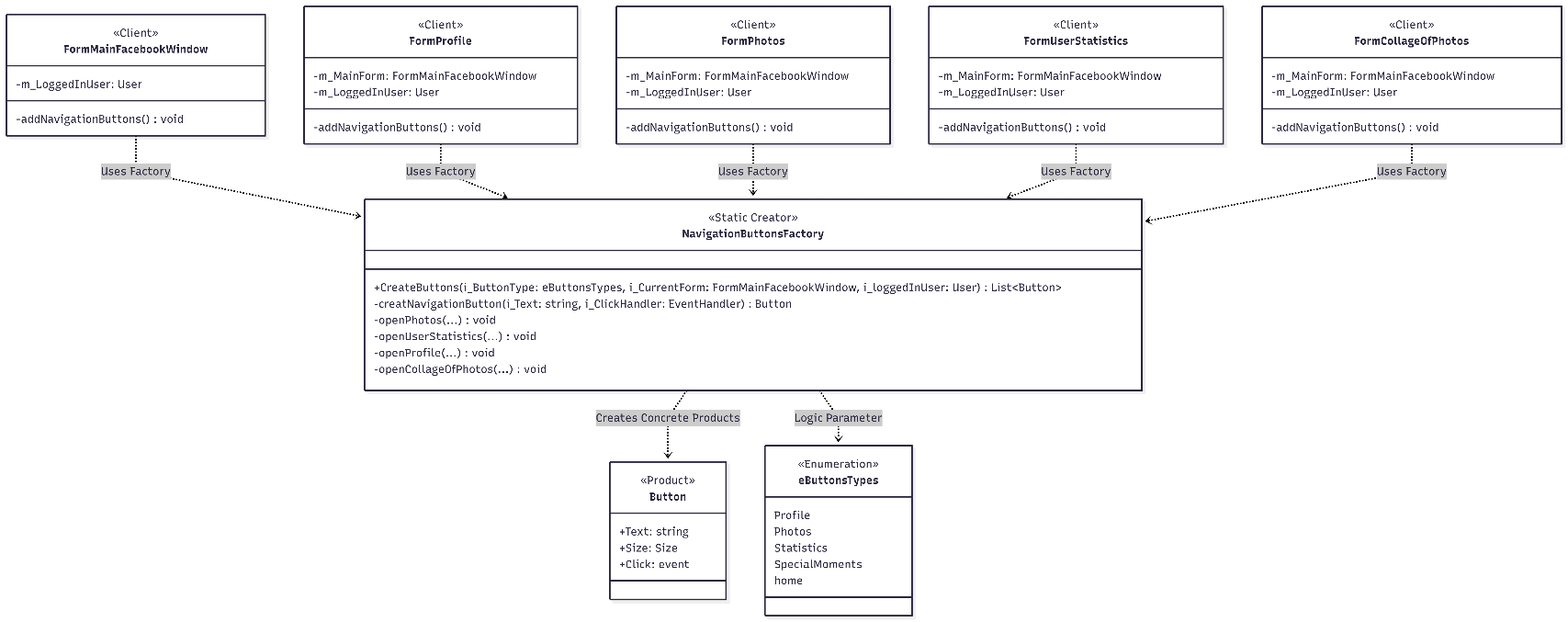
* סיבת הבחירה: השתמשנו בתבנית במחלקת "FormMainFacebookWindow". הצורך נבע מהרצון להבטיח שקיים רק מופע אחד (Instance) של החלון הראשי לאורך כל חיי האפליקציה. מכיוון שחלון זה מחזיק את נתוני המשתמש המחובר (m\_LoggedInUser) ואת ה-Facade הראשי, יצירה של מספר חלונות זהים במקביל עלולה לגרום לחוסר עקביות בנתונים ובעיות בניווט בין הדפים השונים באפליקציה.
* אופן המימוש: ה-Constructor הוגדר כ- private כדי למנוע יצירת אובייקטים מבחוץ באמצעות האופרטור new. הוגדר שדה סטטי פרטי בשםs\_Instance המחזיק את המופע היחיד של המחלקה. הגדרנו מתודה סטטית פומבית GetInstance עם אובייקט נעילה בשם s\_Lock המבטיח שהאובייקט ייווצר פעם אחת בלבד (Thread Safety). בנוסף המחלקה מוגדרת sealed כדי למנוע הורשה מהמחלקה.
* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – Factory Method

* סיבת הבחירה / שימוש בתבנית: השתמשנו ב Factory Method כדי לנהל את יצירת כפתורי הניווט בדפים השונים. הצורך נבע מהרצון לשמור על עיצוב זהה ולוגיקת ניווט (פתיחה וסגירה של חלונות קודמים) בכל האפליקציה במקום אחד מרכזי, דבר שמקל על תחזוקתיות הקוד.
* אופן המימוש: המימוש נמצא במחלקה הסטטית "NavigationButtonsFactory" המחלקה מכילה את המתודהCreateButtons שמקבלת את סוג סט הכפתורים לפי סוג הדף (אחד מתוך eButtonsTypes) ומחזיקה רשימה של אובייקטי Button. הוא יוצר את הכפתורים עם עיצוב אחיד ומצמיד לכל כפתורEventHandler הכולל את הלוגיקה המדויקת למעבר בין הטפסים הרלוונטיים, וסגירת הטופס הנוכחי
* Sequence Diagram
* Class Diagram

### עבודה אסינכרונית עם ממשק משתמש

השתמשנו בתכנות אסינכרוני במחלקה FormProfile.

באירוע הטעינה של הטופס (FormProfile\_Load), במקום להריץ את הטעינה של הנתונים על ה-Thread הראשי, יצרנו Thread נפרד המריץ את המתודה loadUserData.

במתודות loadUserData ו- displayPosts, השתמשנו ב invoke כאשר רצינו לעדכן פקדים ב-UI (כמו TextBox או FlowLayoutPanel), השתמשנו במתודה יחד עם Action ולמדא, מכיוון שרק ל- thread שיצר את הפקדים יש את האפשרות לשנות אותם.

עשינו זאת כדי למנוע קיפאון, בגלל ששליפת נתונים ממקור חיצוני כמו פייסבוק היא פעולה שלוקחת זמן, אם לא היינו משתמשות ב- thread נפרד ממשק המשתמש היה קופא ולא מגיב ללחיצות או להזזה של החלון עד לסיום הטעינה.

### עבודה עם Data Binding

השתמשנו ב-Data Binding במחלקה FormUserStatistics.

בטופס הגדרנו רכיב מסוג BindingSource בשם facadeUserInfoBindingSource, ובזמן טעינת הטופס, קישרנו אליו את אובייקט ה-Facade של הסטטיסטיקות למקור הנתונים.

ב-Designer, קישרנו את הפקדים שמציגים את כמות הפוסטים או האלבומים ישירות ל- Properties המתאימים בתוך FacadeUserStatistics, ואז כשנטען המסך, הערכים מוצגים אוטומטית בלי שנצטרך לכתוב ידנית קוד לכל Label.