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

.כרטיסי ברכה לימי הולדת- הכנת כרטיסי ברכה לימי הולדת.  
המשתמש יכול להכין כרטיסי ברכה לימי הולדת בכל זמן כך שיהיו מוכנים בשעת צורך, במידה והמשתמש מתנתק הכרטיס אינו נמחק אלא יופיע שוב כשיתחבר.  
ע"י בחירה של אחד הכרטיסים שיצר המשתמש המחובר (גם אם התנתק מאז) - הוא יכול לפרסם פוסט ברכה לאחד או יותר מחבריו שחוגגים היום יום הולדת (ובו תוכן הכרטיס הנבחר).  
ניתן למצוא אותו בקובץ של features בתיקייה של הלוגיקה והמימוש שלו בא לידי ביטוי במימוש הטופס FormBirthday.

אישור Events: בפיצ'ר זה אנו טוענים למשתמש את כל האירועים שהוא לא אישר הגעה עבורם ונותנים לו את האופציה לסמן אם הוא מעוניין להגיע לאירוע או לסרב.  
ניתן למצוא את המימוש שלו בקובץ של Manager בתיקייה של logic והמימוש שלו בא לידי ביטוי במימוש הטופס FormEvents.

### תבנית מס' 1 – [Faҫade אטום]

* **סיבת הבחירה / שימוש בתבנית:**

יש לנו מורכבות לוגית שרצינו שהמשתמש לא יהיה חשוף אליה אז רצינו שיהיה מנהל חזותי שדרכו הדברים המורכבים מתנהלים ולא נחשפים באופן ישיר למשתמש.

כמו-כן, מחלקה זו ממומשת כ-Singelton שמטרתו העיקרית היא היכולת של רכיבי המערכת השונים ומערכות עתידיות לקבל גישה לרכיב אחד שמנוהל ע"י סינגלטון .

**השימוש ב- FaçadeMangerיוצר את המצבים הבאים:**

* ממשק המשתמש אינו מכיר כלל את העולם הלוגי.
* נוחות שימוש המשתמש אינו צריך להכיר את המורכבות של הפעולות הלוגיות
* בטיחות שימוש לא יוכל לגרום נזק על ידי הפעלה לא תקינה של רכיב לוגי
* נוחות תחזוקה באמצעות עדכון ה-Facade במידת הצורך.

תפקידה של מחלקה זו היא לפשט את העולם הלוגי, היא זאת שמתמודדת מול העולם הלוגי.

בין השאר היא אחראית על פרטי החיבור של המשתמש המחובר היחיד בכל הפעלה של המערכת, ולכן יש בה מן התכונות של סינגלטון.

היא יוצרת את הפישוט של העולם הלוגי בכך שהיא "מסתירה" מהלקוח את תתי המערכות המורכבות שאיתם היא עובדת. כך, במקום שהלקוח יתמודד עם מורכבות תתי המערכות, הוא מבחינתו עובד מול אובייקט יחיד שמנהל ומתווך עבורו את כל שהוא זקוק לו מבחינה לוגית.

* **אופן המימוש:**

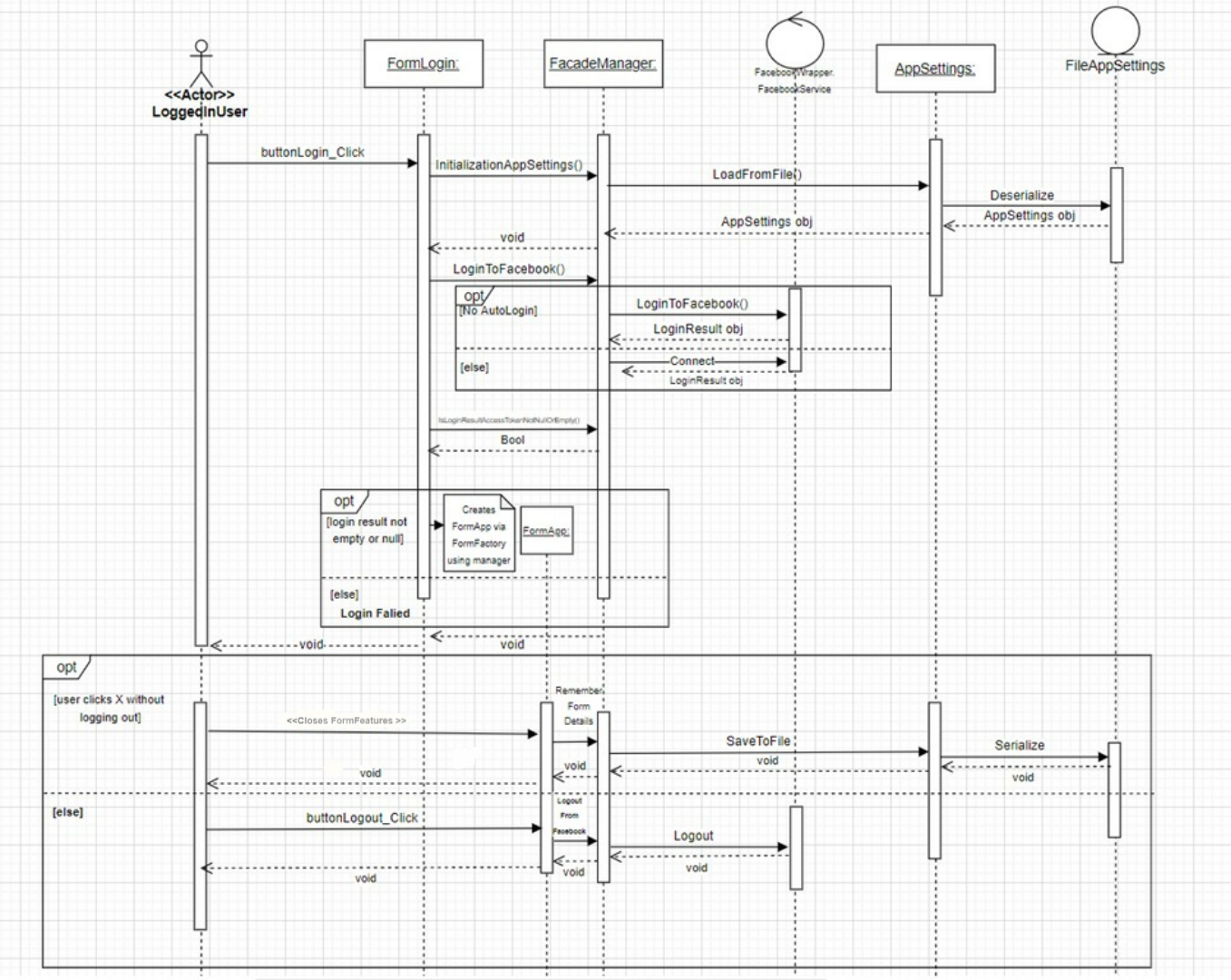
המחלקה FacadeManger היא בעצם ה-façade האטום שיצרנו.  
ה-Client הוא למעשה ספריית ה-UI, ובפרט FormLogin, FormFeatures, ושאר הטפסים תחת Forms.  
Fromlogin מייצר מופע של FacadeManger וכך נחשפים לשאר ה-Forms השירותים שמספק   
ה- Facade Manager.

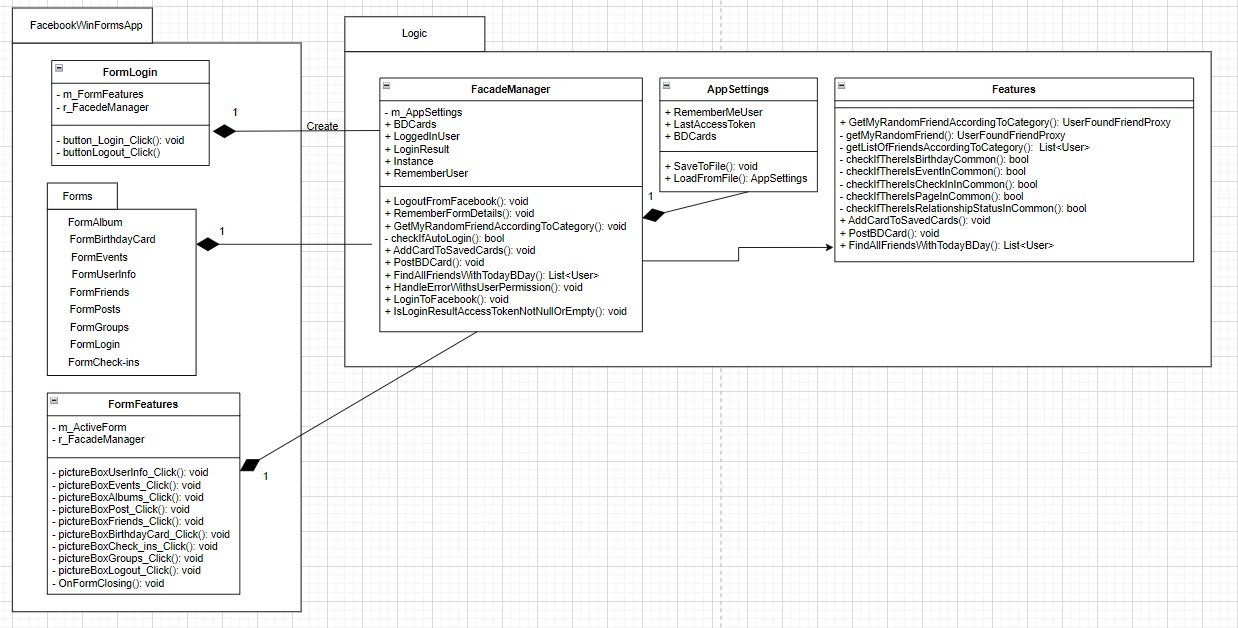
המחלקות הלוגיות הוגדרו כ-internal כאשר ה-FacadeManger נמצא איתן באותו DLL, וה-Client באחד נפרד. דבר זה מאפשר facade אטום. ה-Client לא יכול לגשת ישרות ללוגיקה (AppSettings, Features) שכן הוא לא מכיר אותם כלל.

בנוסף, ב- FacadeMangerיש מתודות שמחברות בין המשתמש לחלק הלוגי של ה-Features תוך הגנה משגיאות, ובכך מתאפשר שימוש נוח ובטוח של הלקוח בפיצ'רים של הלוגיקה.

מבחינת ה-Singelton, קיימת מתודת Instance שמייצרת מופע בפעם הראשונה שמתבקש אחד, וביתר בפעמים- מחזירה את הרפרנס לאותו מופע בדיוק.

* Sequence Diagram



* Diagram Class

### תבנית מס' 2 – Proxy

* סיבת הבחירה / שימוש בתבנית:

אנו מעוניינים להציג ב-formFriends את החבר שמצאנו.  
בנוסף לפרטים הרגילים שיש לכל User ברצוננו להציג גם את גילו המדויק של החבר וגם את מרחקו מה-LoginUser.

כלומר, אנחנו רוצים לשדרג את User כך שהוא יחזיר לנו את גילו המדויק ואת הפרש הגילאים מLoginUser בנוסף לדברים שuser מציע.

labelDetails מקבל string החלטנו לממש אובייקט נפרד שגם הוא מספק את ה ToString עם כל התוספות שרצינו.

User הינו לגסי ולכן השימוש בPROXY היה נחוץ.

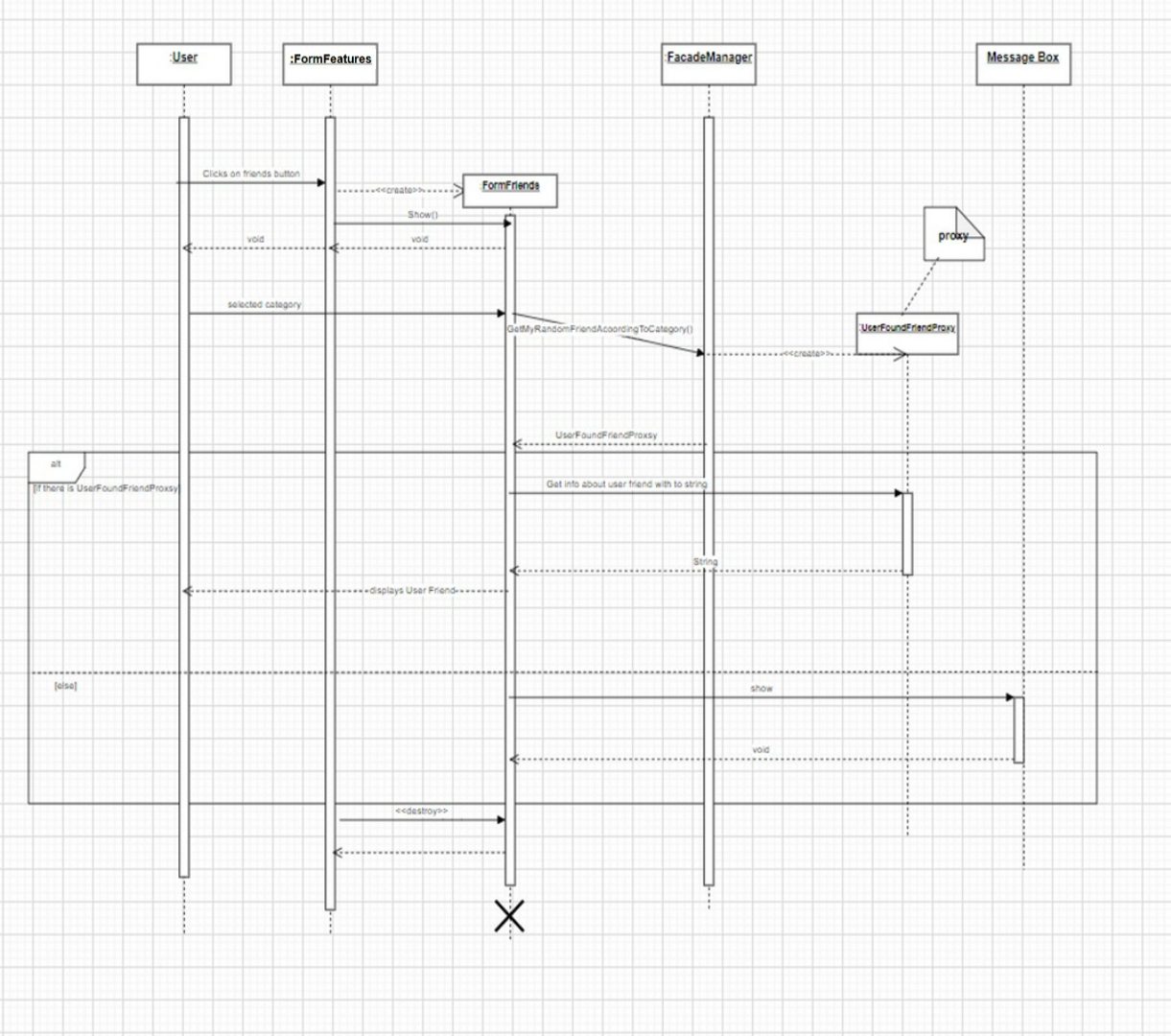
אז ניצור במקרה שלנו אובייקט שמממש את הממשק object והוא יממש את ToString המשוכלל יותר ויחזיק את User שזה ה realSubject-.

* אופן המימוש:

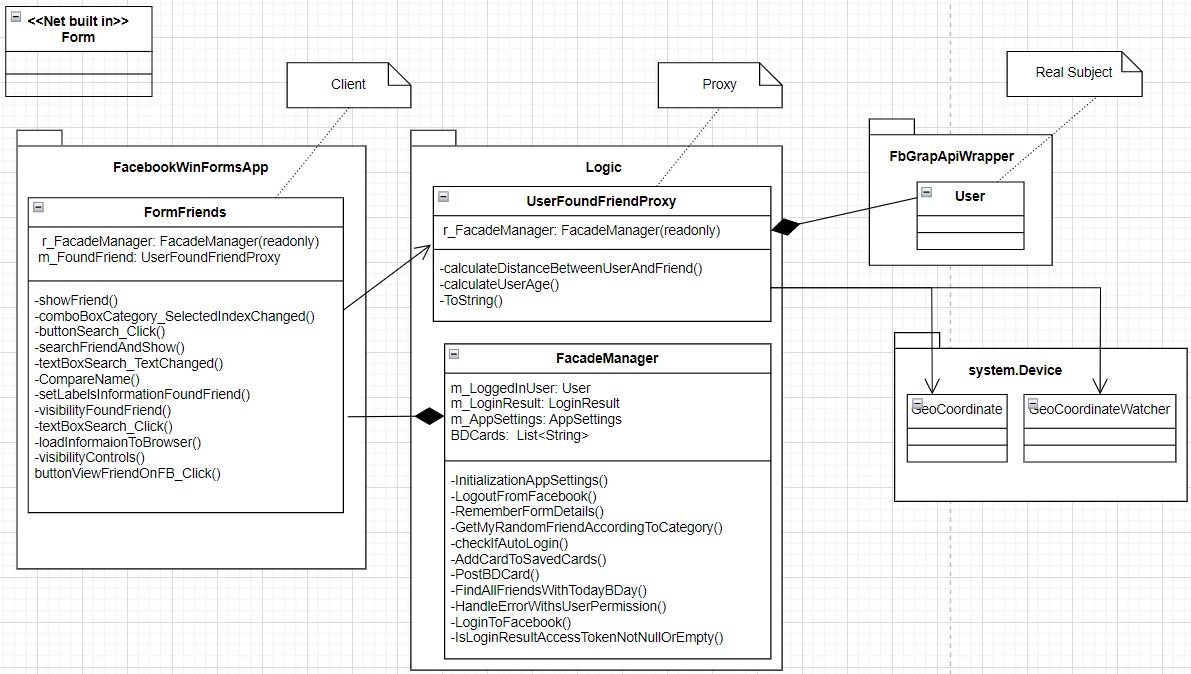
UserFoundFriendProxy – proxy  
ממשק-object  
Client- FormFriends

המחלקה UserFoundFriendProxy היא שדרוג המתחזה ל User אשר מחזיקה את User ומממשת ToString.  
ב-formFriends כאשר אנו רוצים להציג את פרטי החבר שיצרנו נבצע פשוט labelDetails.Text=foundFriend.ToString כאשר foundFriend הוא אובייקט מסוג UserFoundFriendProxy.  
בעקבות שורה זו יוצגו פרטי החבר (שם, מין, יום הולדת, גיל והשדרוג שהוא הפרש הגילאים).

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – MethodFactory

* סיבת הבחירה / שימוש בתבנית:

ראינו שכל ה forms באפליקציה יורשים מ form ובנוסף יש להם מאפיינים משותפים ולכן הם משפחה פולימורפית כאשר אב המשפחה הוא ה – form.

כדי שלא נצטרך לממש את הלוגיקה של הבחירה והיצירה ב client בחרנו לממש את הלוגיקה והיצירה במחלקה סטטית נפרדת FormFactory שבה יש מתודה סטטית שאחראית על יצירת הform הספציפי לפי בקשת ה-client ומחזיקה אובייקט של ה- form שביקש.

בנוסף, שימוש בתבנית מספקת לצרכן קוד קונסיסטנטי, reuseable.

יתר על כן, אם בעתיד ירצו להרחיב את המשפחה הפולימורפית הוא פחות יצטרך לתחזק את הקוד.

* אופן המימוש:

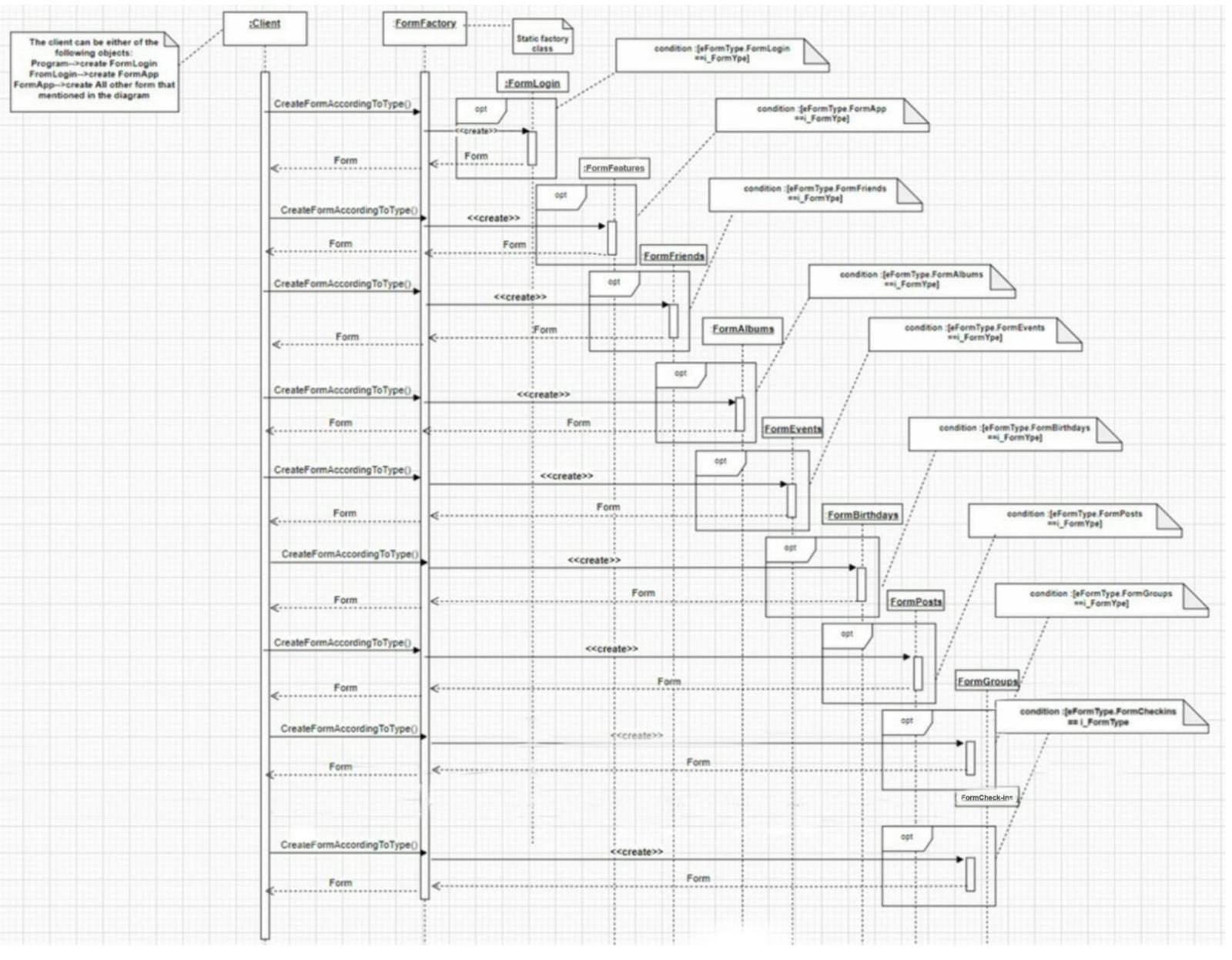
המחלקה FormFactory שאחראית על מימוש הבחירה ולוגיקת היצירה ב-Client, מספקת את הנ"ל באמצעות המתודה CreateFormAccordingToType.

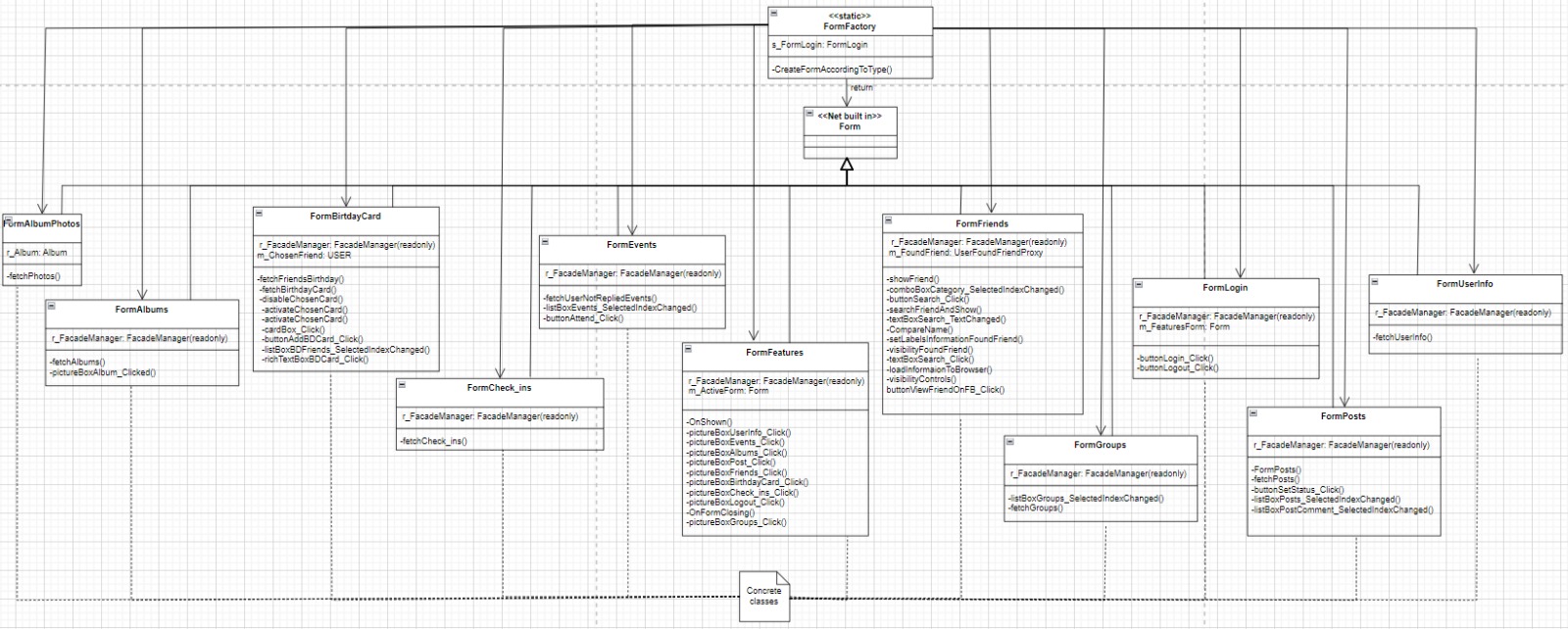
המתודה מקבלת eFormType שלפיו היא יודעת איזה form ה-client בדיוק רוצה בעזרת switch case שנמצא במתודה.

וחוזר טיפוס מסוג form.

כמו כן המתודה מבצעת בדיקה שהטפסים נוצרים רק לאחר שנוצר טופס ההתחברות. במידע וכן היא מקריסה את המערכת בכדי לשמור על המנגנון של האפליקציה.

* Sequence Diagram



* ****Class Diagram

**עבודה אסינכרונית:  
formAlbum**  
**fetchAlbums()**מדובר במתודה שמביאה את כל האלבומים של המשתמש משרתי פייסבוק ומכוון שזו פעולה שלקחת זמן רב החלטנו להפעיל אותה בThread נפרד.  
**showFormAlbumsPhotos()**  
מדובר במתודה שמופעלת כאשר המשתמש לוחץ על אלבום בכדי לראות את התמונות בו.   
כאשר נטענות תמונות של אלבום ספציפי בform נפרד (פעולה גם שלוקחת זמן) לא ניתן לפתוח אלבום חדש.  
כדי לתת תחושה של שימוש רציף באפליקציה ומתן אפשרות של פתיחת כמות של אלבומי תמונות במכה בחרנו להפעיל את המתודה ב Thread נפרד.  
**formAlbumsPhotos**  
**fetchPhotos()**  
מדובר במתודה שמביאה את כל התמונות של אלבום של המשתמש משרתי פייסבוק ומכוון שזו פעולה שלקחת זמן רב החלטנו להפעיל אותה בThread נפרד.

**formBirthdays  
fetchFriednsBirthday()**  
מדובר במתודה שעוברת על כל החברים של המשתמש ומחזירה רשימה של כל החברים שיש למשתמש ולהם יום הולדת משותף, מתודה זו לוקחת את המידע משרתי פייסבוק ומכוון שזו פעולה שלוקחת זמן רב החלטנו להפעיל אותה ב Thread נפרד.  
**fromPosts**  
**fetchPosts()**  
מדובר במתודה שמביאה את כל הפוסטים והתגובות לפוסטים של המשתמש משרתי פייסבוק ומכוון שזו פעולה שלקחת זמן רב החלטנו להפעיל אותה בThread נפרד.

**fromGroups**  
**fetchGroups()**  
מדובר במתודה שמביאה את כל הקבוצות של המשתמש משרתי פייסבוק ומכוון שזו פעולה שלקחת זמן רב החלטנו להפעיל אותה בThread נפרד.

**עבודה עם DataBinding:**

השתמשנו ב DataBinding ב-FormCheck-ins .  
 listBoxCheckins של המשתמש המחובר מקושר ל- checkinBindingSource ומציג את רשימת האירועים ב- panelCheckins.