Design patterns Ex02

מגישים : [מקס אליצוב] , [309350072] , [ניב גיטצי] , [204389209]

**: First Application Feature**

**Friend :Evolution**

זה , מאפשר למשתמש שביצע כניסה לאפליקציה , לבחור את אחד החברים מתוך רשימת החבריםFeature

שלו ובכך לראות את כל תמונות הפרופיל של אותו חבר לאורך השנים , כאשר מתחת לכל תמונת פרופיל מצורפת תמונה המציגה את השנה שבה צולמה תמונת הפרופיל.

כמו כן , משתמש אשר יעבור בעיון על התמונות שהוצגו לפניו ויזכור באיזו שנה צולמה כל תמונה, יגדיל את סיכויו לענות נכונה על שאלת גילו של החבר בתמונה שתוצג לו בפיצר הטריביה.

כלומר מימשנו את שני הפיצרים כך שיהיה קשר נחמד ביניהם ובכך נוסיף לחווית המשתמש. (הסבר על הפיצר השני יוסבר בהמשך).

**:Secnd Application Feature**

**:Friend Trivia**

זה , מאפשר למשתמש באפליקציה , לבחור את אחד מחבריו , מתוך רשימתFeature

החברים שתוצג לפניו . לאחר מכן יחל משחק טריביה בו תיבדק בקיאותו של המשתמש לגבי

חברו , ע"י 4 שאלות כאשרר בסוף המשחק , יוצג למשתמש מספר השאלות שנענו נכונה , ובנוסף משפט יוצג משפט על סמך תוצאתו בחידון.

תבניות מספר 1 ו- 2

**FAÇADE + SINGLETON**

תבנית עיצוב זו ממומשת בתוכנית כמחלקה בשם :

(מחלקה זו הינה סינגלטון) . מטרת התבנית הינה להוות חוצץ **ApplicationFacade**

בין החלק הלוגי לחלק של ממשק המשתמש. בתוכנית שלנו החלק של ה- UI אינו מכיר את

החלק הלוגי בכלל , כלומר כל פעולה שאינה באחריות ה- UI עוברת דרך ApplicationFacade וזו פונה לחלק הלוגי , ומעבירה את חזרה את הנתונים או הפעולות הלוגיות שנעשו ע"י החלקים הלוגיים חזרה ל – UI.

בחרנו לממש תבנית זו כדי ליצור מודולריות בתוכנית ובנוסף לאפשר reuse בתוכנית בעתיד . כלומר , אם בעתיד ארצה לשנות או להחליף את ממשק המשתמש , אוכל לעשות זאת מבלי לשנות שום דבר שקשור לחלק הלוגי של התוכנית.

תבנית העיצוב singleton קשורה לתבנית העיצוב הראשונה (facade) . מחלקה זו מחזיקה את ה- data של ה- logged in userאחד ויחיד.

המחלקה ApplicationFacade הינה סינגלטון במערכת , כלומר ישנו מופע אחד ויחיד של המחלקה בתוכנית , וכל יתר הצרכנים בתוכנית יכולים לעשות שימוש במחלקה זו ביוזמתם.

בחרנו לממש את המחלקה ApplicationFacade בתור סינגלטון משום שמחלקה זו עונה על התנאים למימוש סינגלטון :

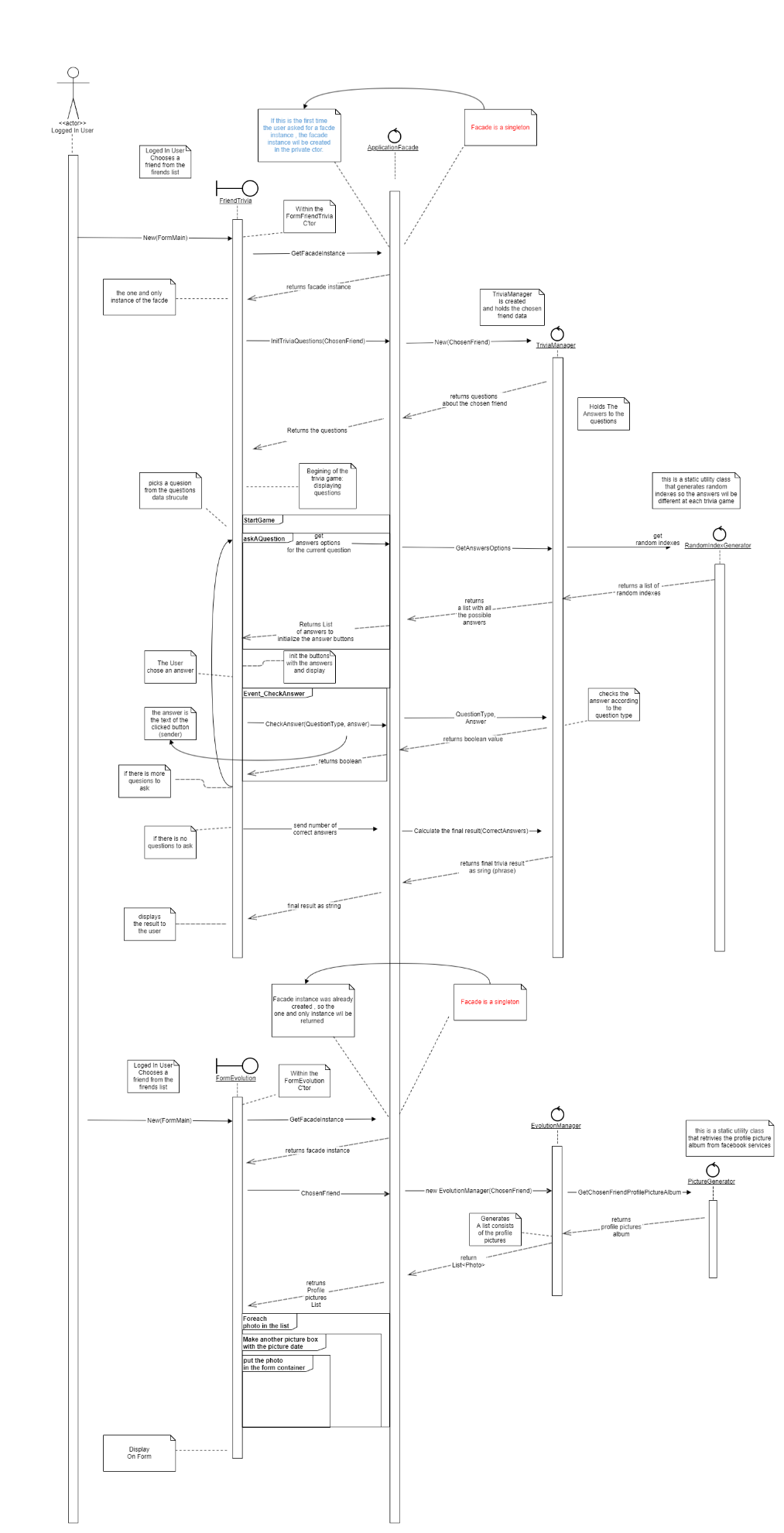
**אם יהיה יותר ממופע אחד במערכת , אזי זו תהייה טעות או בזבזנות.**

* בזבזנות משום שמחלקה זו היא בתפקיד המתווך בין ה- UI לרכיב הלוגי של המערכת ולא ייתכן מצב שיש שני מופעים שונים של מחלקה זו (state שונה).כי מחלקה זו תלויה ב – logged in user של התוכנית.

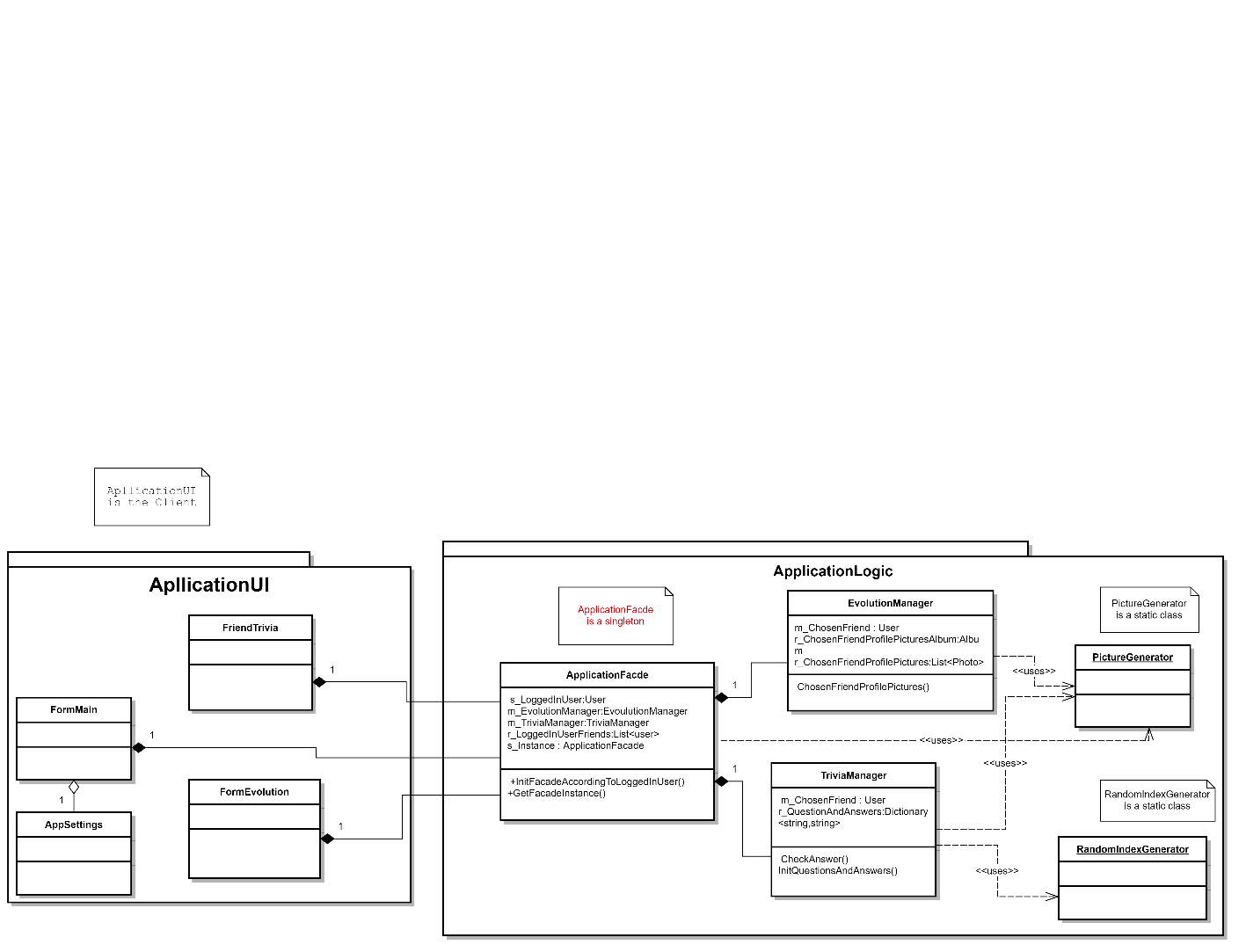
כלומר , אם יהיו שניים שונים ,בוודאות יהיה להם את אותו ה - state , ומכאן שיווצר שיכפול מיותר.

* טעות לוגית משום שאם יהיו שניים שונים זה אומר שיש שני logged in users וזה כמובן לא אפשרי בתוכנית. הסינגלטון מהווה מתווך בין ה – UI לחלק הלוגי ושניהם תלויים ב – Logged in user הנוכחי.

**sequence diagram**



**class diagram**



תבנית מספר 3

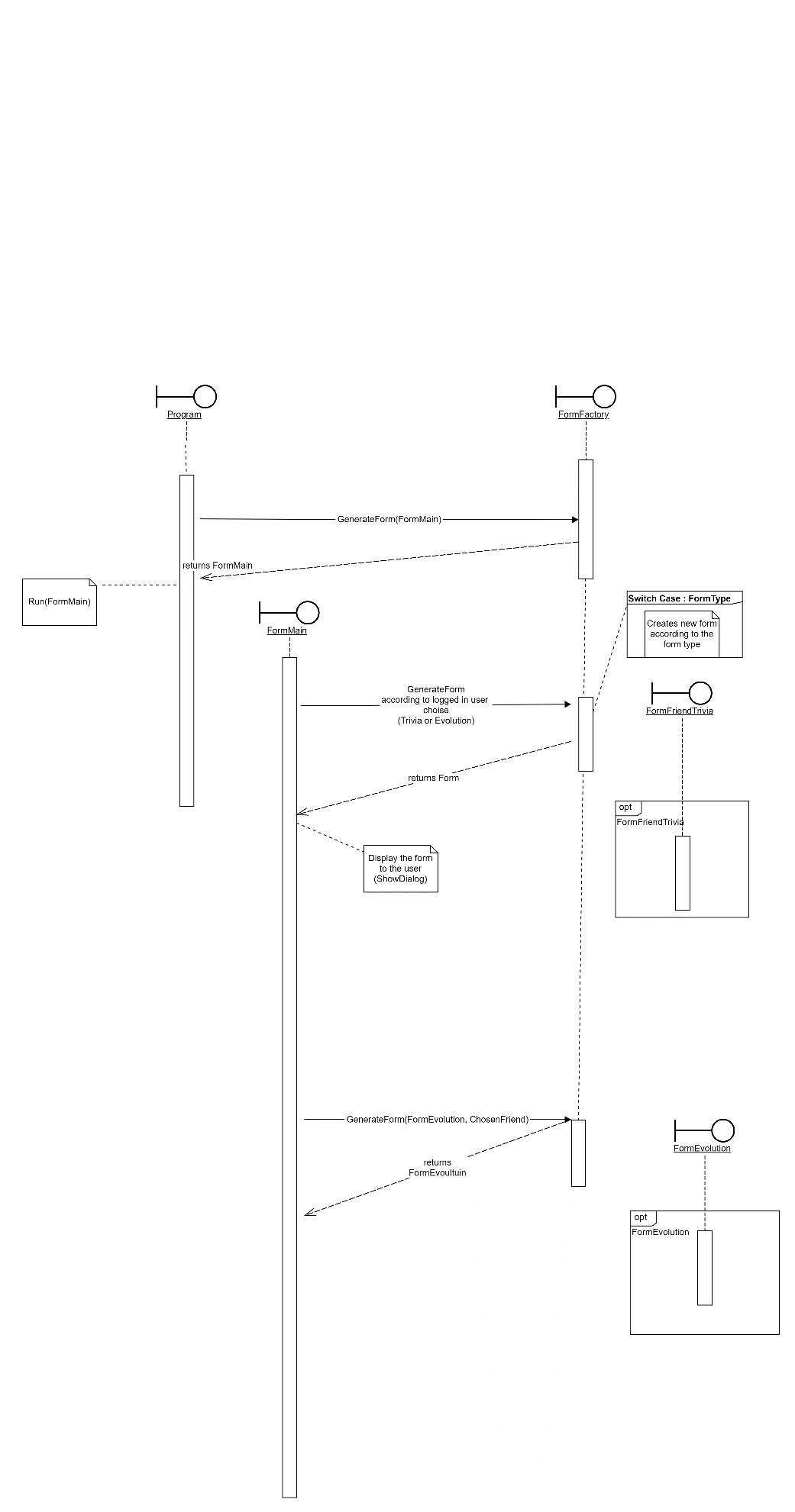
**Static Factory Class**

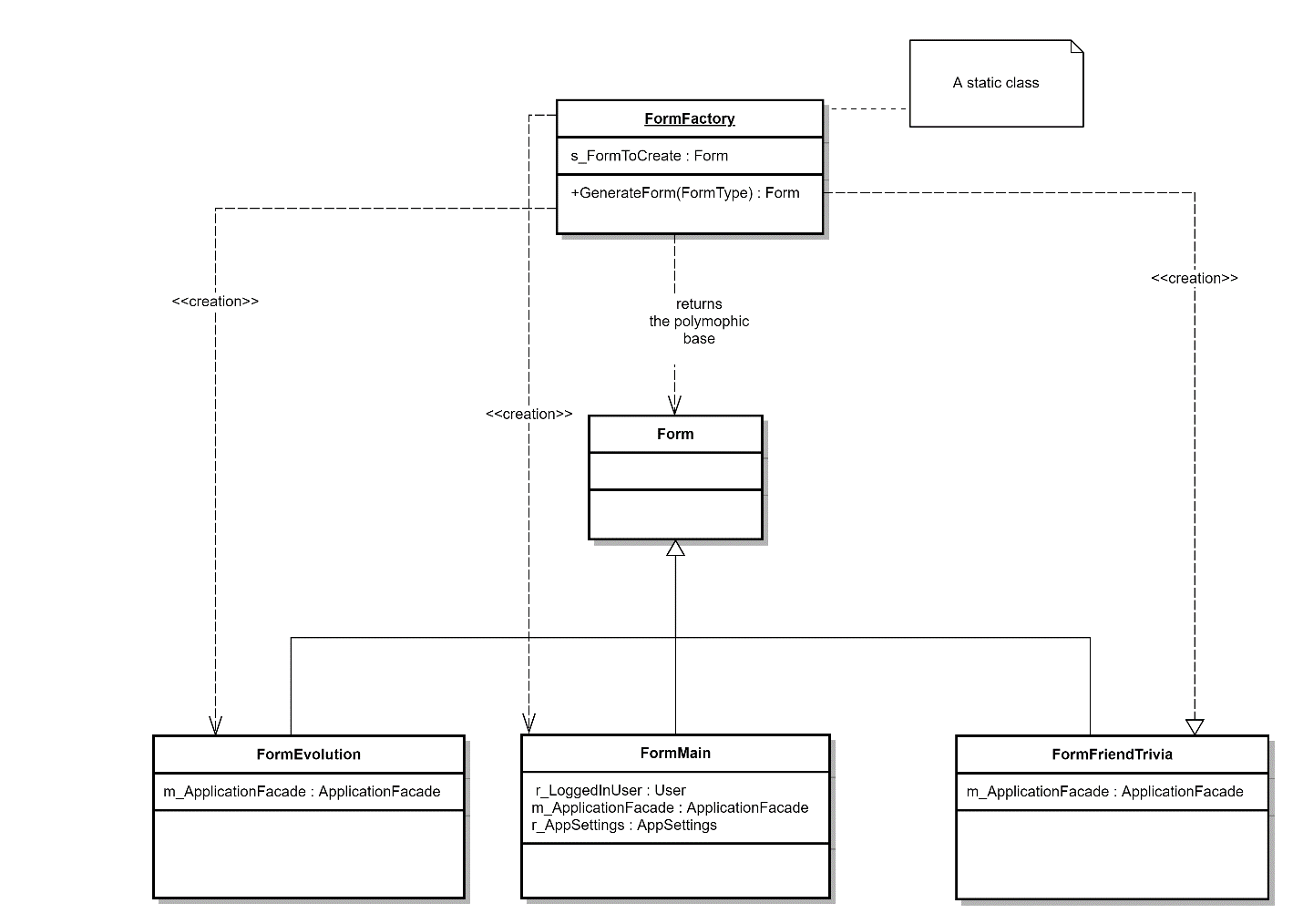
בתוכנית ישנה מחלקה בשם FormFactory אשר משמשת כ – Static Factory Class.

תפקיד המחלקה הוא לייצר , לפי דרישה , את המשפחה הפולימורפית של הטפסים בתוכנית (FormMain , FormFriendTrivia ו- FormEvolution).

בחרנו לממש תבנית זו , כדי להפוך את התוכנית ליותר מודולרית בכך שהפרדנו את תפקיד יצירת הטפסים למחלקה נפרדת זו , ובנוסף שימוש בתבנית זו , שיפר את ה-flow של הקוד בכך שפעולות היצירה (new) שהיו בעבר ב – FormMain עברו ל - FormFactory , כמו כן הקוד הפך ליותר reusable משום שאם בעתיד נרצה לשנות את ממשק המשתמש , לא נצטרך לגעת בחלק הקוד שמייצר את הטפסים.

Sequence Diagram



class Diagram

שימוש בעבודה אסינכרונית :

המקומות בהם השתמשתי בתכנות אסינכרוני הינם כל החלקים בתוכנית שניגשים ל – data של facebook. השימוש ב – Thread -ים מאפשר לטפסים להיות איטרטיביים עם המשתמש במהלך הבאת הנתונים השרתי facebook.

- מתודה שמביאה משרת הפייסבוק את כלל חבריו של המשתמש.**FormMain – fetchFriends()**

מתודה שמביאה משרת הפייסבוק את כלל האירועים - **FormMain – fetchEvents()**

שלהם נחשף המשתמש.

אלו הן המתודות היחידות שנעשה להן שימוש ב – Thread אחר , משום שהתוכנית שלי ממומשת כך שכאשר בוצע loggin מוצלח כל נתוני ה – Logged in user נמצאים במחקלה ApplicationFacade. ובכך חסכתי את תהליך הבאת הנתונים הדרושים למערכת לכדי פעם אחת בלבד.

בכל פעם שנעשה שימוש ב- Thread נוסף (ששונה מה- Main Thread) נעשה שימוש פנימי במתודה Invoke()

שמפעילה delegate מסוג Action

על מנת להפעיל את אותן מתודות האחראיות על תחזוקה קונקרטית עבור ה – Main Thread

מתודה זו מכילה את קטע הקוד הדרוש על ידי שימוש ב -

" .Lambda Exression”

כל זאת כיוון שרק ה Thead - שהוסיף את האובייקטים רשאי לשנות או לעדכן אותם.

שימוש ב - DataBinding:

### **המקומות בהם השתמשתי ב – Data binding:**

בטופס FormMain ישנו listBoxEvents שמחובר ל – eventBindingSource אשר מציג פרטים על האירוע שיסומן מתוך ה- Listbox .