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

* **פיצ'ר ראשון: מיון תמונות לפי פרמטר על פי בחירת המשתמש.**

הפיצ'ר מאפשר למשתמש האפליקציה לראות את התמונות שלו, לדפדף ע"י כפתורים next ו-previous ולמיין אותם לפי סדר כמות הלייקים הגבוה ביותר על כל תמונה/ כמות התגובות הגבוהה ביותר על כל תמונה/ תמונות לפי סדר העלאה העדכני (האפשרות הזו היא ברירת המחדל). בנוסף, ע"י לחיצה על תמונה מסויימת ייפתח חלון של גוגל מפות ובו יוצג המקום שעשו לתמונה זו צ'ק אין (אם קיים).

מחלקות קשורות:

MainPageForm, PictureboxProxy, PhotosManagement.

* **פיצ'ר שני: בוחן על חבר מרשימת החברים.**

הבוחן בוחן את רמת החברות שלך ע"י הצגת מספר שאלות שעליהן המשתמש עונה ולאחר מענה על כל השאלות מתבצע סיכום של הבוחן ומוצגת התוצאה של ה"בוחן".

על המשתמש לבחור מרשימת החברים את החבר שהוא רוצה להבחן עליו וכשהוא לוחץ על הכפתור "Friendship Test" נפתח חלון בו הוא נבחן בארבע שאלות אמריקאיות. בסופו של הבוחן המשתמש מקבל תוצאה.

מחלקות קשורות:

FriendshipTestForm, AnswersObservable, ITesterObserver, FriendshipTestFacade, SetupAnswers, TestManagement, UserFriendTestData.

### תבנית מס' 1 – Strategy

* סיבת הבחירה / שימוש בתבנית:

בפיצ'ר הראשון של מיון תמונות, אנו ממיינים תמונות לפי בחירת המשתמש באמצעות מספר פרמטרים לרשותו (most recent, most comments, most likes).

עד כה, עשינו זאת ע"י מתודה ייעודית אחת למיון התמונות ע"י כך שבעזרת if-ים עברנו על כל אפשרויות המיון, וברגע שהגענו לאפשרות שהמשתמש בחר, המתודה זיהתה זאת וביצעה את הפעולה. הבנו שקוד זה אינו,Reusable Maintainable, Extendable ובנוסף לאחר שזיהינו מכנה משותף בין כל צורות המיון החלטנו להשתמש בתבנית העיצוב Startegy.

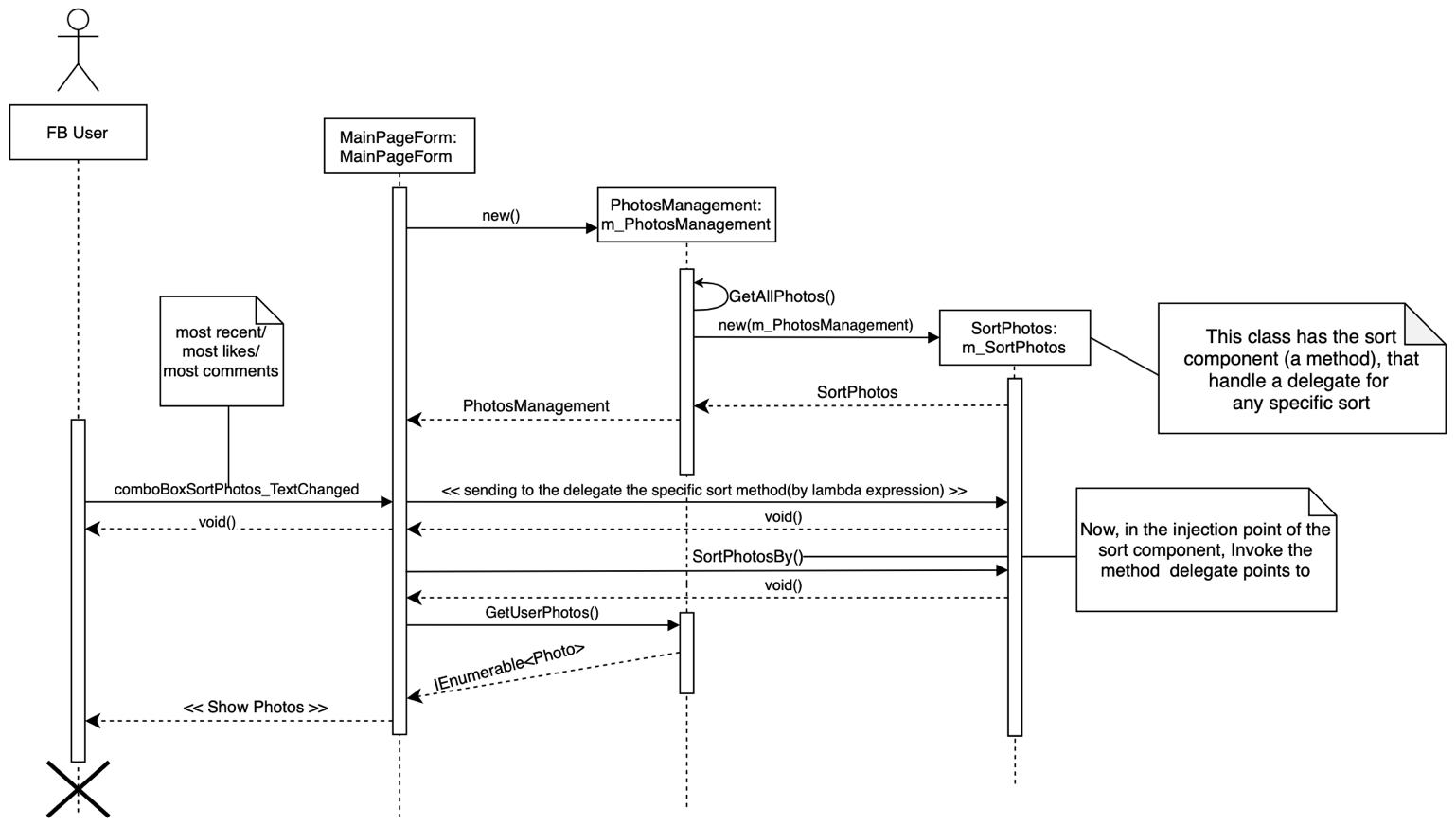
כך למעשה אנחנו מאפשרים במחשבה לעתיד, הוספת סוגים אחרים של מיונים מבלי לשנות את הקוד הנוכחי(Reuse) וכמובן פחות פוטנציאל לבאגים(Maintainable).

* אופן המימוש:

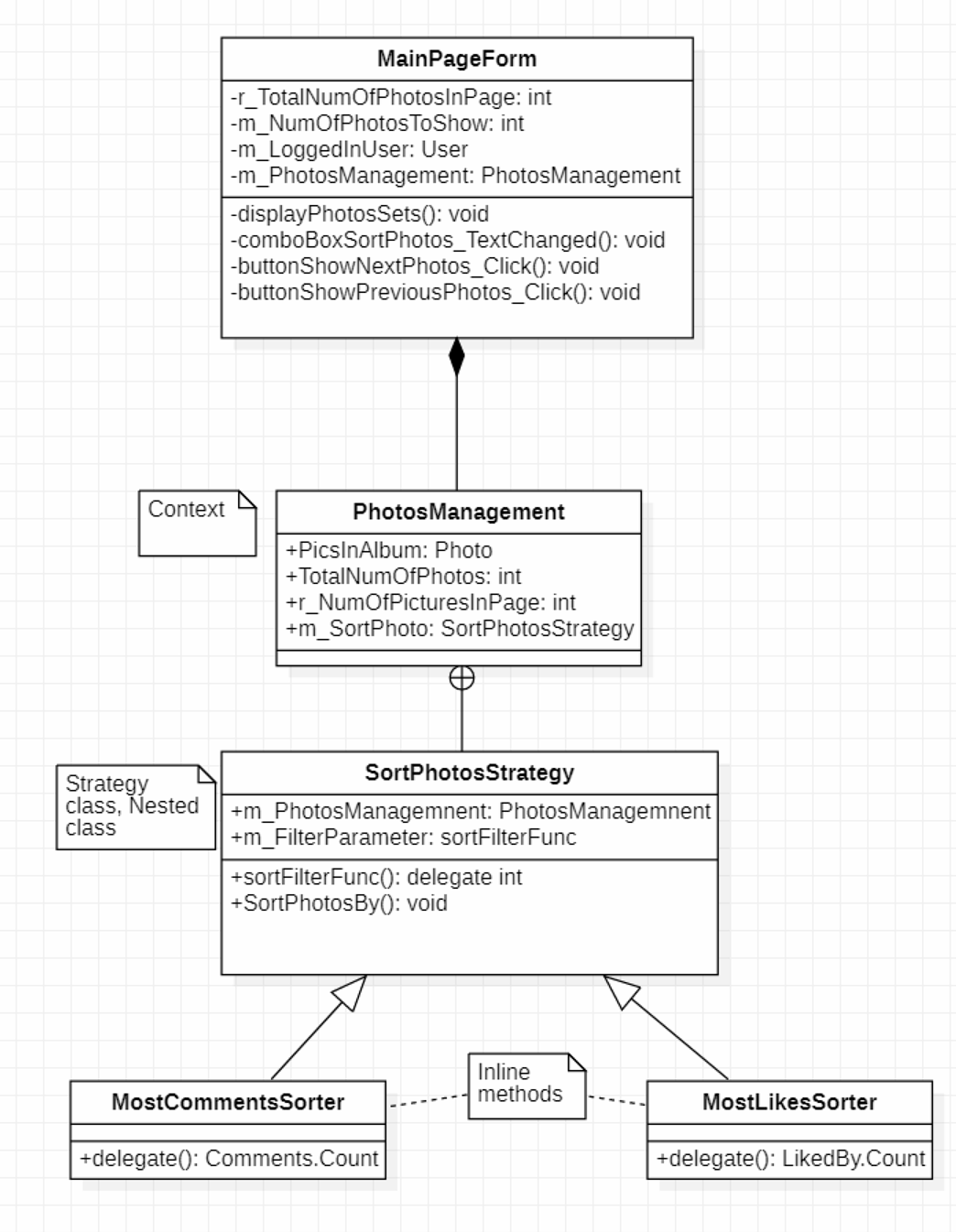
את הרכיב המדובר הוצאנו אל מחלקה בשם SortPhotos. מחלקה זו מקוננת בתוך המחלקה PhotosManagement כי בסופו של דבר הרכיב עושה פעולה שקשורה למחלקה המנהלת את התמונות. בנוסף יצרנו במחלקה SortPhotos, delegate בשם sortFilterFunc ובעזרתו כל פעם שהמשתמש בוחר אחת מן אפשרויות המיון, אנו עושים לו השמה של פרמטר המיון באמצעות lambda expression (נמצא ב- MainPageForm באירוע comboBoxSortPhotos\_TextChanged), וברכיב עצמו ובעזרת Invoke אנו מבצעים Injection לתוך רכיב המיון.

**ב- Class Diagram ציינו איזו מחלקה בקוד מייצגת איזה "שחקן".**

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – Iterator

* סיבת הבחירה / שימוש בתבנית:

בפיצ'ר הראשון של מיון תמונות, אנו מציגים למשתמש מספק קבוע של תמונות כל פעם, ע"י "דפדוף" באמצעות הכפתורים next ו- previous. את התמונות(Photo) עצמן אנו מחזיקים כיום באמצעות מערך.

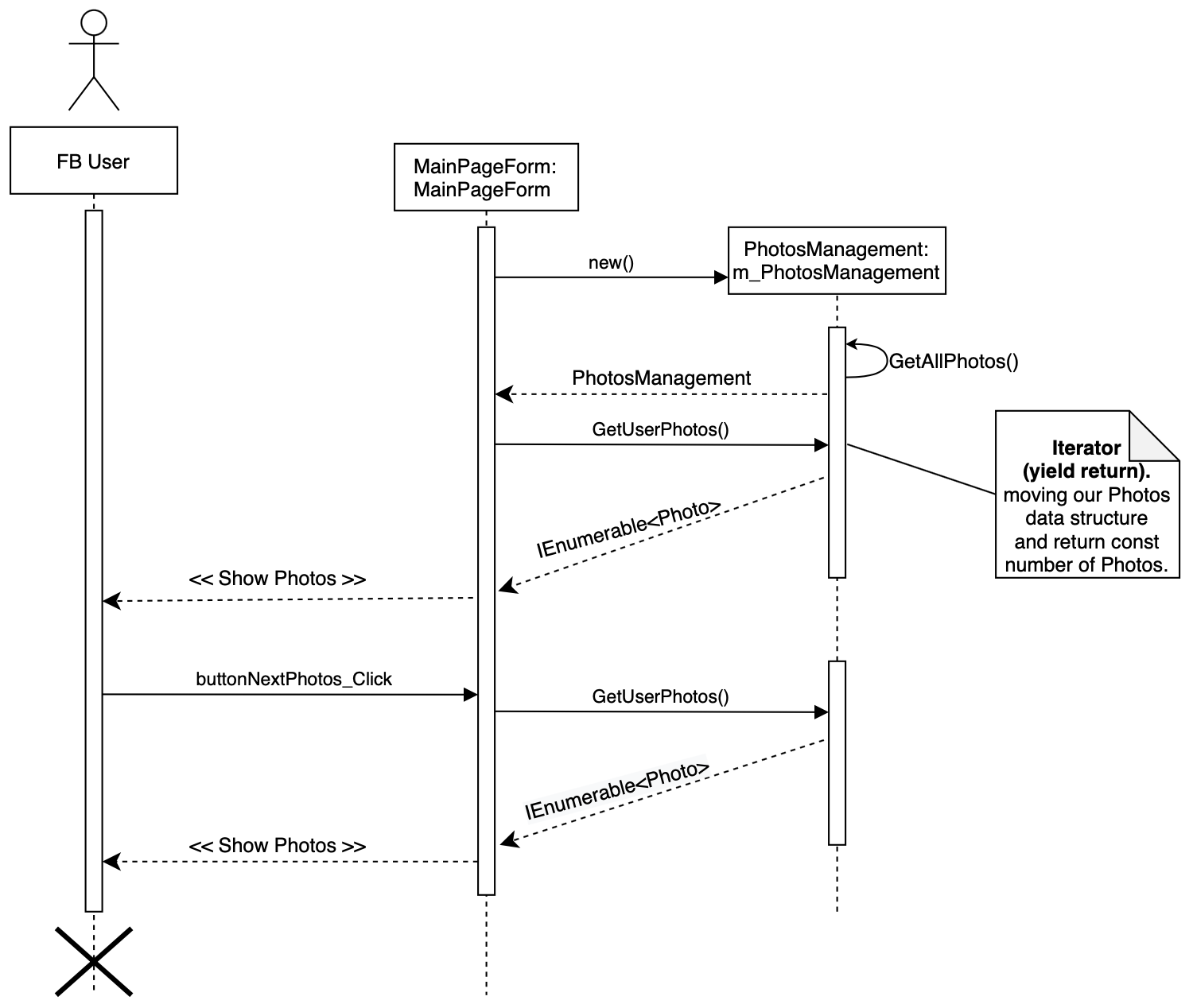
כך בעתיד אנו נוכל לשנות את מבנה הנתונים שלנו על פי הצורך ללא שינוי של קוד ה"client" ולכן השתמשנו בתבנית עיצוב Iterator המאפשרת לנו מעבר על מבנה הנתונים ולעשות בו כל שימוש ללא אבחנה בסוג מבנה הנתונים.

* אופן המימוש:

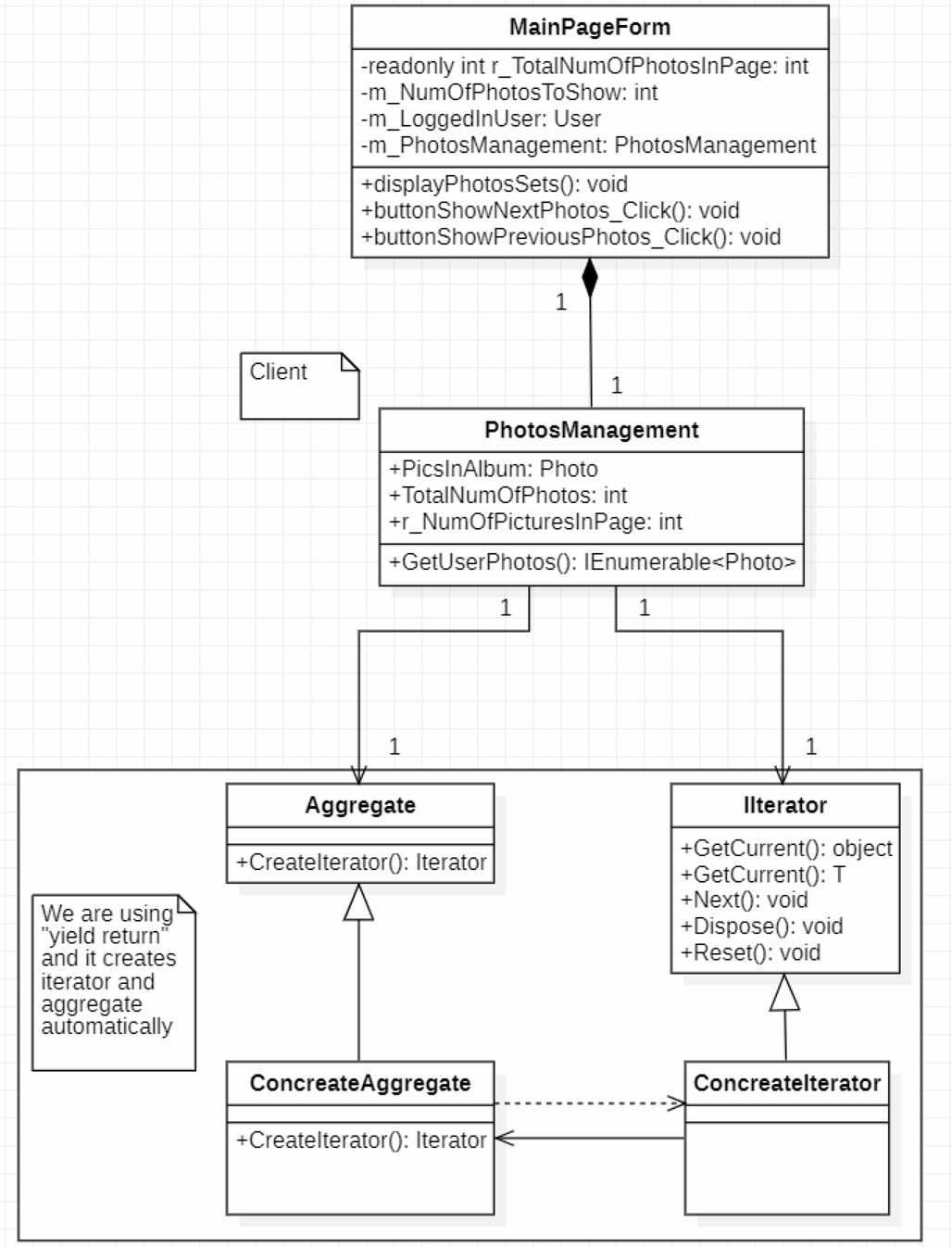
ב- UI במחלקה MainPageForm המתודה displayPhotosSets מבצעת את השמת התמונת ב- PictureboxProxy- ים שמופיעים במסך הראשי. תהליך זה קורה לא מעט פעמים בגלל דפדוף התמונות ובגלל אפשרויות המיון. במתודה זו אנו קוראים למתודה GetAllUserPhotos שנמצאת במחלקה PhotosManagement. המתודה GetAllUserPhotos אחראית לתהליך הבאת התמונות ממבנה הנתונים שלנו ע"י צמד המילים yield return מבלי להתייחס לסוג מבנה הנתונים המחזיק את התמונות. אנו מקבלים את התמונות הרלוונטיות אותן ביקשנו, ובאמצעות המילה foreach אנו טוענים את התמונות ל- Picturebox- ים.

**ב- Class Diagram ציינו איזו מחלקה בקוד מייצגת איזה "שחקן".**

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – Observer

* סיבת הבחירה / שימוש בתבנית:

בפיצ'ר השני של בוחן על חבר, בכל פעם שהמשתמש בוחר תשובה על שאלה מסוימת, מתבצעת הפעלת אירוע ובו שומרים את התשובה שהמשתמש בחר במערך, ולאחר מכן מתבצעות פעולות לוגיות נוספות, זאת על מנת לבדוק האם התשובה נכונה ובמידה שכן להוסיף ניקוד לתוצאה הסופית. בידיעה שקוד זה אינו תחזוקתי, ובמחשבה לעתיד שאם וכאשר תשובות המשתמש ישמשו לפעולות נוספות, החלטנו להשתמש בתבנית עיצוב Observer וזאת במחשבה שבחירת תשובה מסויימת תוכל להודיע למתעניינים נוספים בעתיד וזאת בלי לשנות את הקוד.

* אופן המימוש:

את המימוש ביצענו לפי שיטת Gang of four.

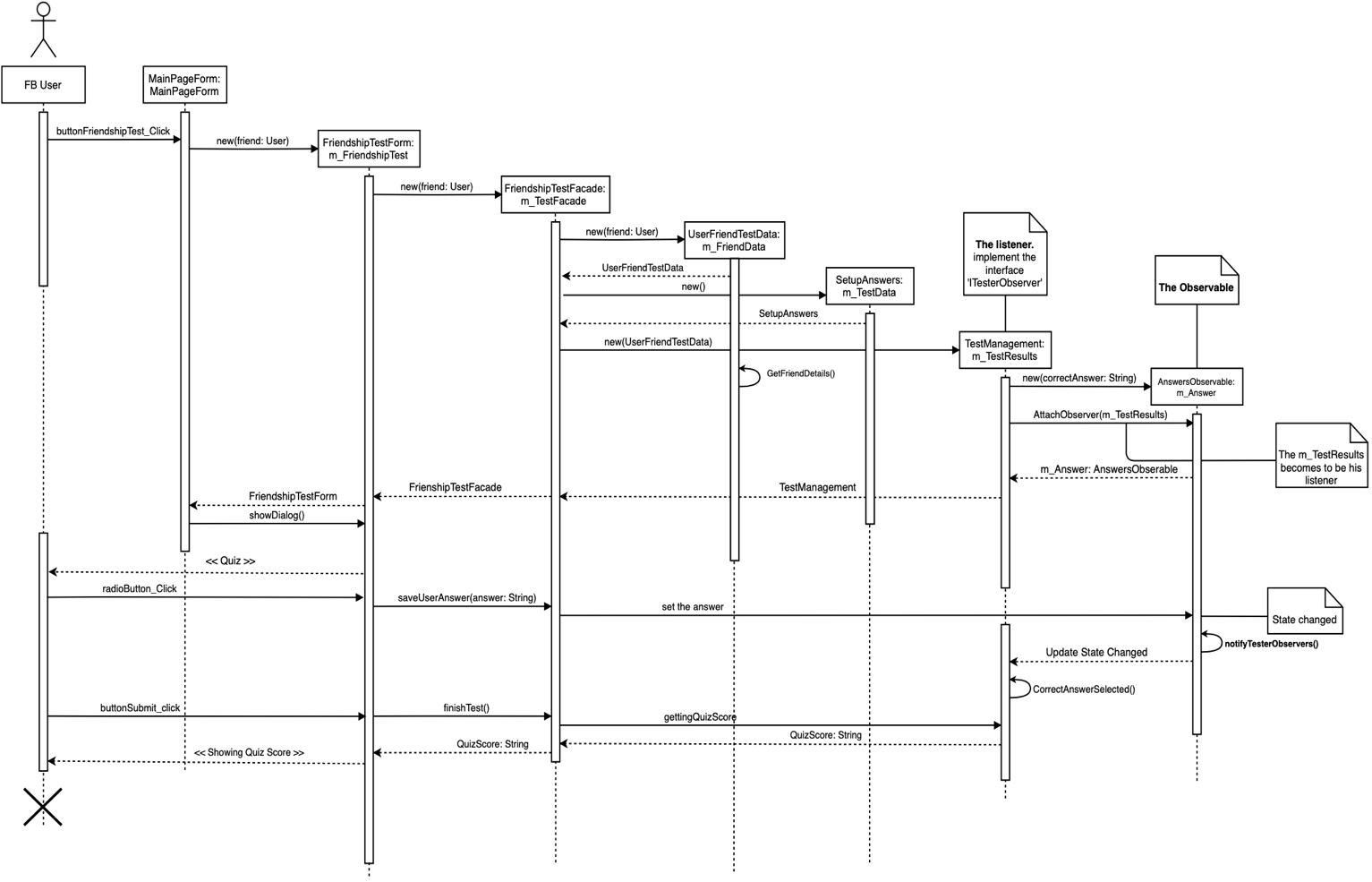
יצרנו Interface בשם ITesterObserver עם מתודה אחת בשם CorrectAnswerSelected.

המחלקה TestManagement מממשת את ITesterObserver

בנוסף יצרנו מחלקה בשם AnswersObservable שהיא מחזיקה בתוכה את התשובה הנכונה של השאלה ותשובה שהמשתמש ייבחר. מחלקה זו מחזיקה את רשימת המאזינים שלה(מסוג ITesterObserver) וברגע שמשתמש בוחר תשובה מסויימת, מתבצעת השמה באמצעות set כלומר ה- state שלה משתנה ואם התשובה זהה לתשובה הנכונה היא מודיעה לכל המאזינים שלה ובכך המתודה CorrectAnswerSelected מופעלת ומתווסף ניקוד.

**ב- Class Diagram ציינו איזו מחלקה בקוד מייצגת איזה "שחקן".**

* Sequence Diagram



* Class Diagram

