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

* **הוספת תמונות ופוסטים לרשימת מועדפים** – האפליקציה מאפשרת סימון מיוחד לתמונות ופוסטים נבחרים. בכל חלון בו מוצגות תמונות או פוסטים, ניתנת האפשרות למשתמש ללחוץ הוספה למועדפים וניתנת אפשרות למשתמש להסיר תמונה או פוסט מרשימת המועדפים בכל רגע נתון.
* **חיפוש תמונות לפי מועד** – האפליקציה מאפשרת למשתמש חיפוש אחר תמונות מתאריך המכיל יום וחודש והמערכת מציגה את אוסף התמונות מהמועד הנבחר לאורך השנים.

### תבנית מס' 1 – Observer

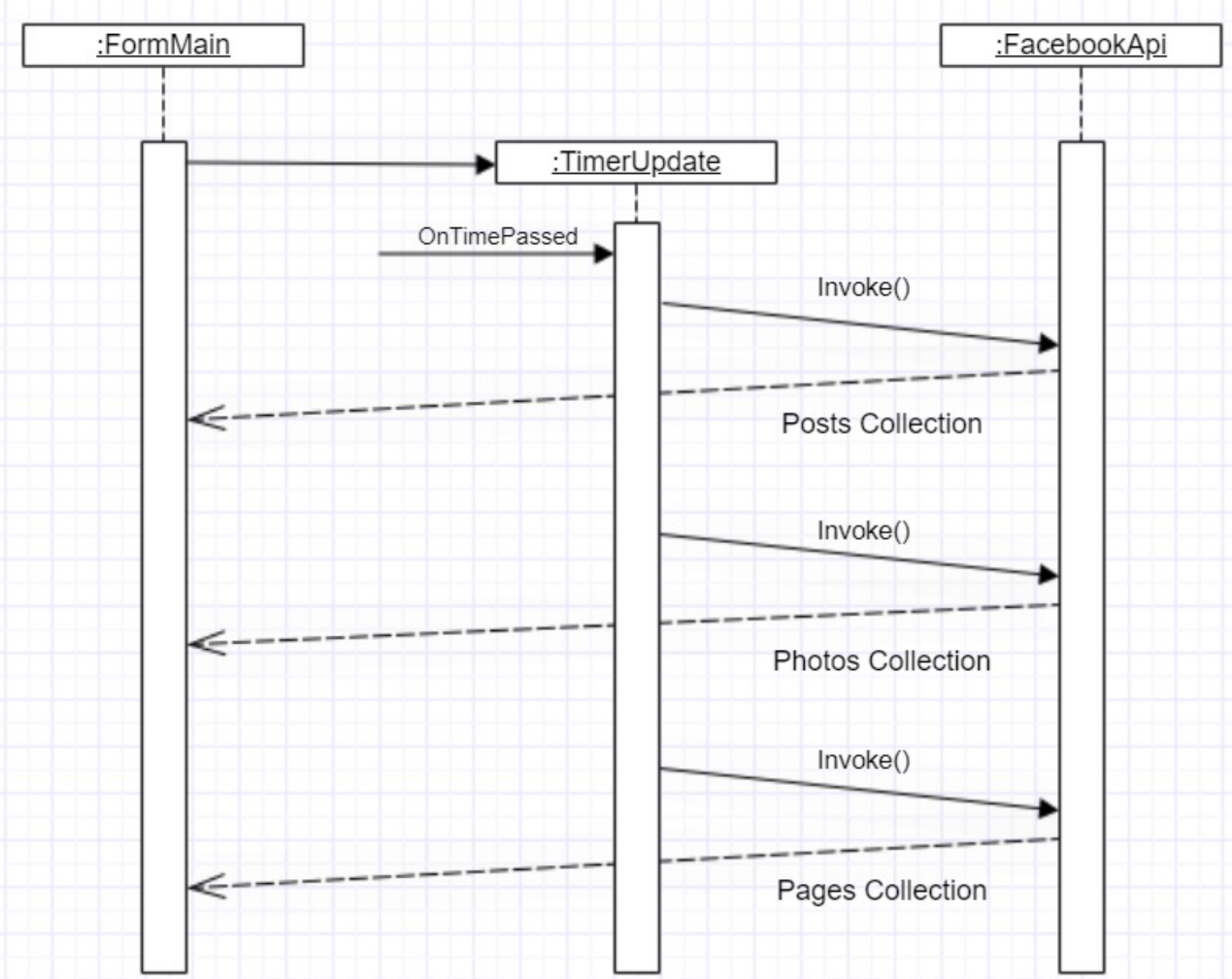
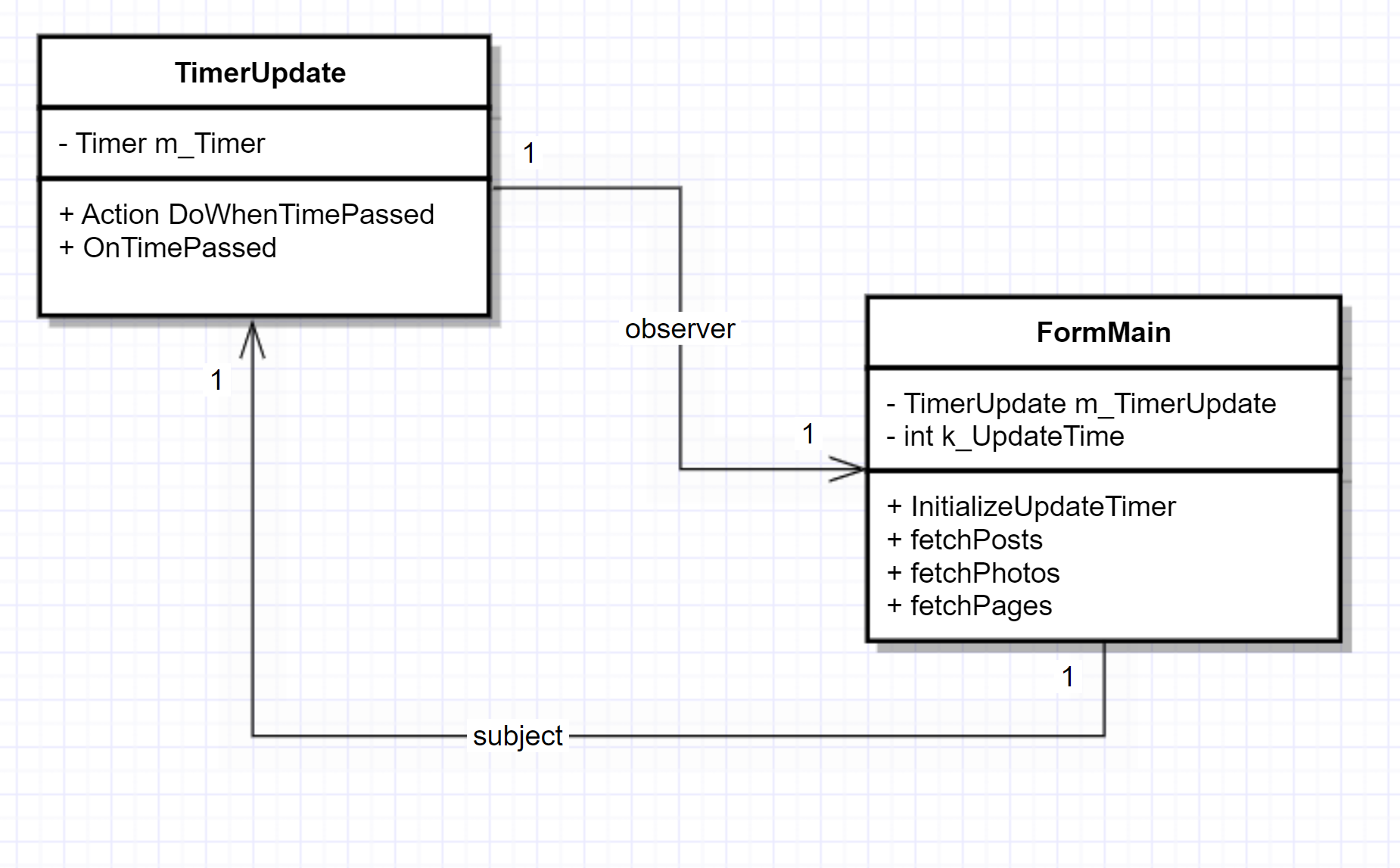
* סיבת הבחירה / שימוש בתבנית:

בכדיי לספר חווית פייסבוק בסיסית ומעודכנת, נרצה שהמידע שמוצג למשתמש יתעדכן כל פרק זמן מסוים. לכן יצרנו רכיב שעוקב אחרי הזמן שעובר, ומעדכן את נתוני המשתמש בכל פרק זמן כזה. השתמשנו בתבנית זו כיוון ששימוש בתבנית זו מאפשר לנו לבצע שימוש חוזר ברכיב זה כך שאם בעתיד נרצה ליצור רכיבים נוספים שנרצה לעדכן גם אותם בכל פרק זמן מסוים, נוכל להוסיף את אופציית הרענון מבלי לשכפל קוד.

* אופן המימוש:

יצרנו מחלקה חדשה בשם TimerUpdate אשר סופרת זמן כך שבסוף כל פרק זמן מופעלת המתודה OnTimePassed שמפעילה את הActions שנרשמו ל event DoWhenTimePassed.

בבנאי של המחלקה FormMain, נוצר מופע של TimerUpdate ובנוסף המתודה fetchUserInfo נרשמת ל event DoWhenTimePassedולכן כאשר פג הזמן מופעלת המתודה fetchUserInfo, בכך נתוני המשתמש מתעדכנים בכל פרק זמן מחדש.

* Sequence Diagram
* Class Diagram

### תבנית מס' 2 – Iterator

* סיבת הבחירה / שימוש בתבנית:

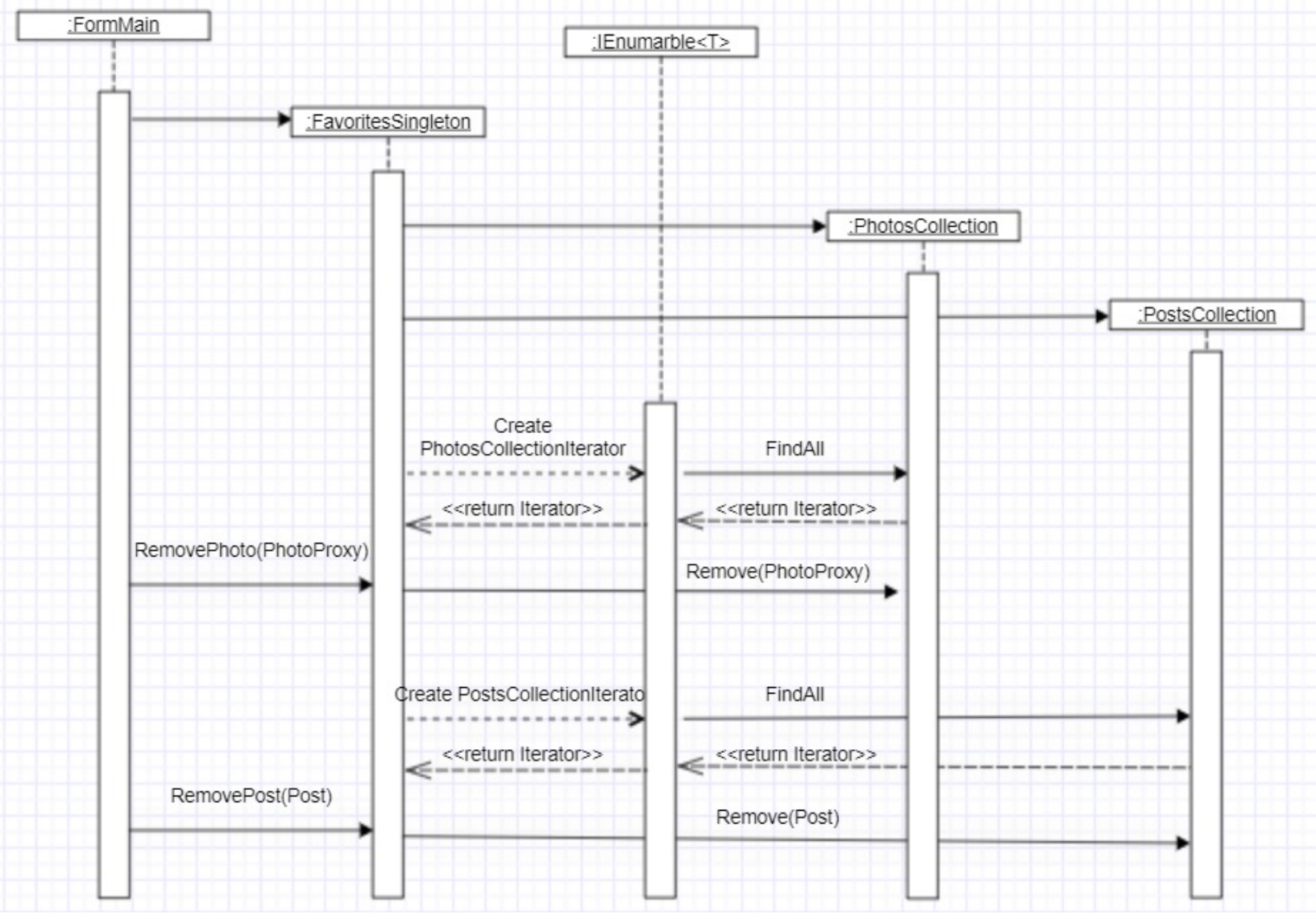
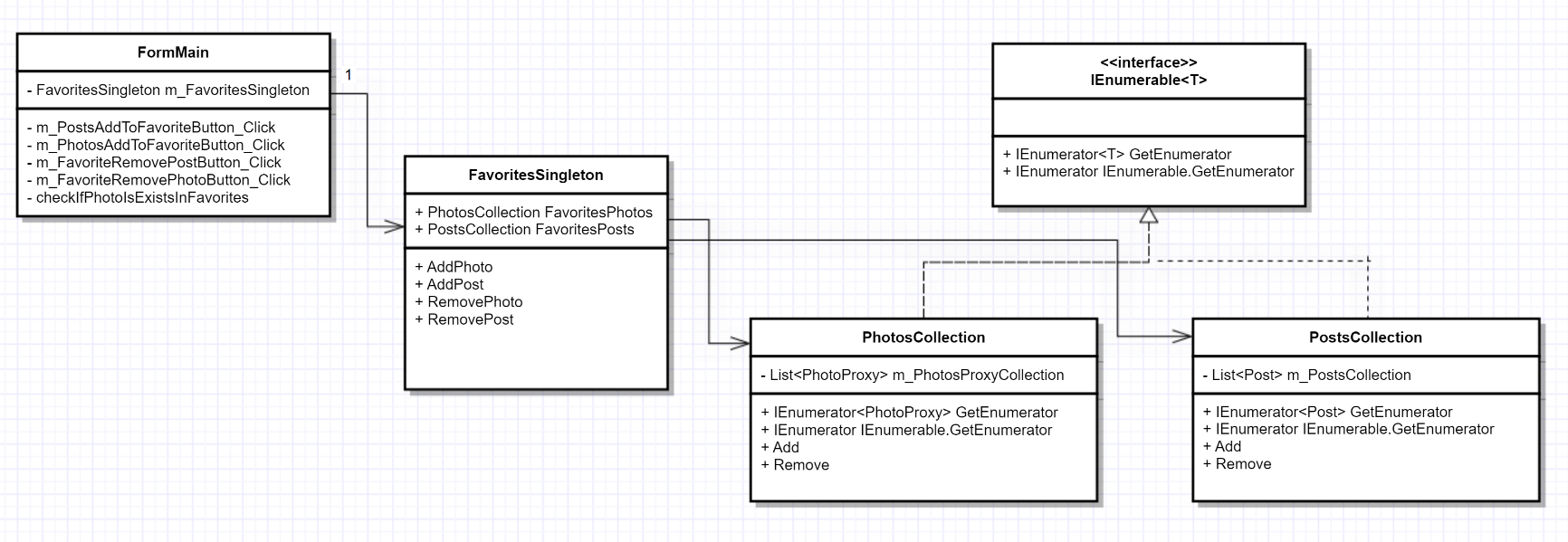
באפליקציה שלנו יש לכל משתמש רשימת מועדפים יחידה (סינגלטון מהמטלה הקודמת) של תמונות ופוסטים. ברצוננו לאפשר את האופציה שבעתיד נוכל לשנות את מבנה הנתונים של רשימת התמונות והפוסטים המועדפים כך שלא נצטרך לשנות את הקוד אלא רק במקום אחד. כשמוסיפים/מוחקים תמונה או פוסט מרשימת המועדפים, הformMain לא יודע לאיזה מבנה נתונים התבצע השינוי ולכן שינוי של מבנה הנתונים ברכיב שיצרנו, אינו משנה את הקוד בשום מקום.

* אופן המימוש:

יצרנו את המחלקות: PhotosCollection, PostsCollection שמממשות את IEnumerable<T> בהתאמה. כל אחת מהמחלקות מחזיקה List<T> (שאת מבנה נתונים זה נוכל בעתיד לשנות לפי הצורך), בנאי שמאתחל את הרשימה, ומימושים של Add, Remove בהתאם למבנה הנתונים (במקרה זה רשימה).

בנוסף, במחלקת FavoritesSingleton, אנו מחזיקים פרופרטיז של PhotosCollection, PostsCollection במקום מבנה נתונים מסוים.

ע"י מימוש זה, בFormMain ובכל מקום אחר בתוכנית למעט המחלקות PhotosCollection, PostsCollection עצמן, אנו יכולים להוסיף ולמחוק תמונות ופוסטים מרשימת המועדפים בלי לדעת מה מבנה הנתונים שבו הנתונים מוחזקים וע"י כך יש לנו אפשרות לשנות זאת בכל שלב שנרצה ללא שינוי הקוד.

* Sequence Diagram
* Class Diagram

### תבנית מס' 3 – Strategy

* סיבת הבחירה / שימוש בתבנית:

נרצה לתת אפשרות למיון הפוסטים לפי קטגוריות שונות. ע"י שימוש בתבנית זו לא נצטרך לשכפל קוד לכל קטגוריה. בנוסף, אם בעתיד נרצה להוסיף קטגוריות נוספות ו/או נרצה למיין פריטים מTypes שונים, נוכל לעשות זאת בעזרת המימוש הנוכחי ללא שכפול קוד.

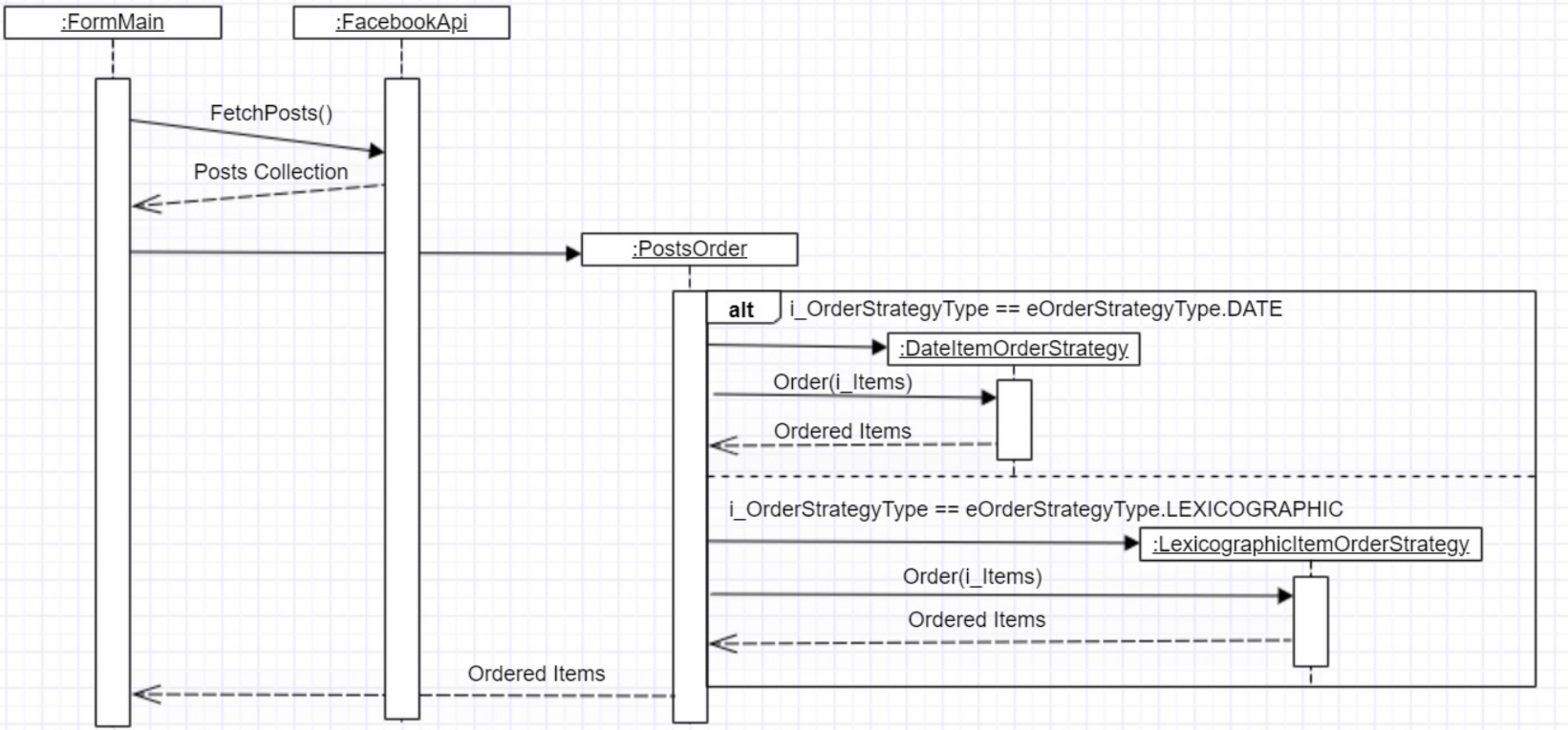
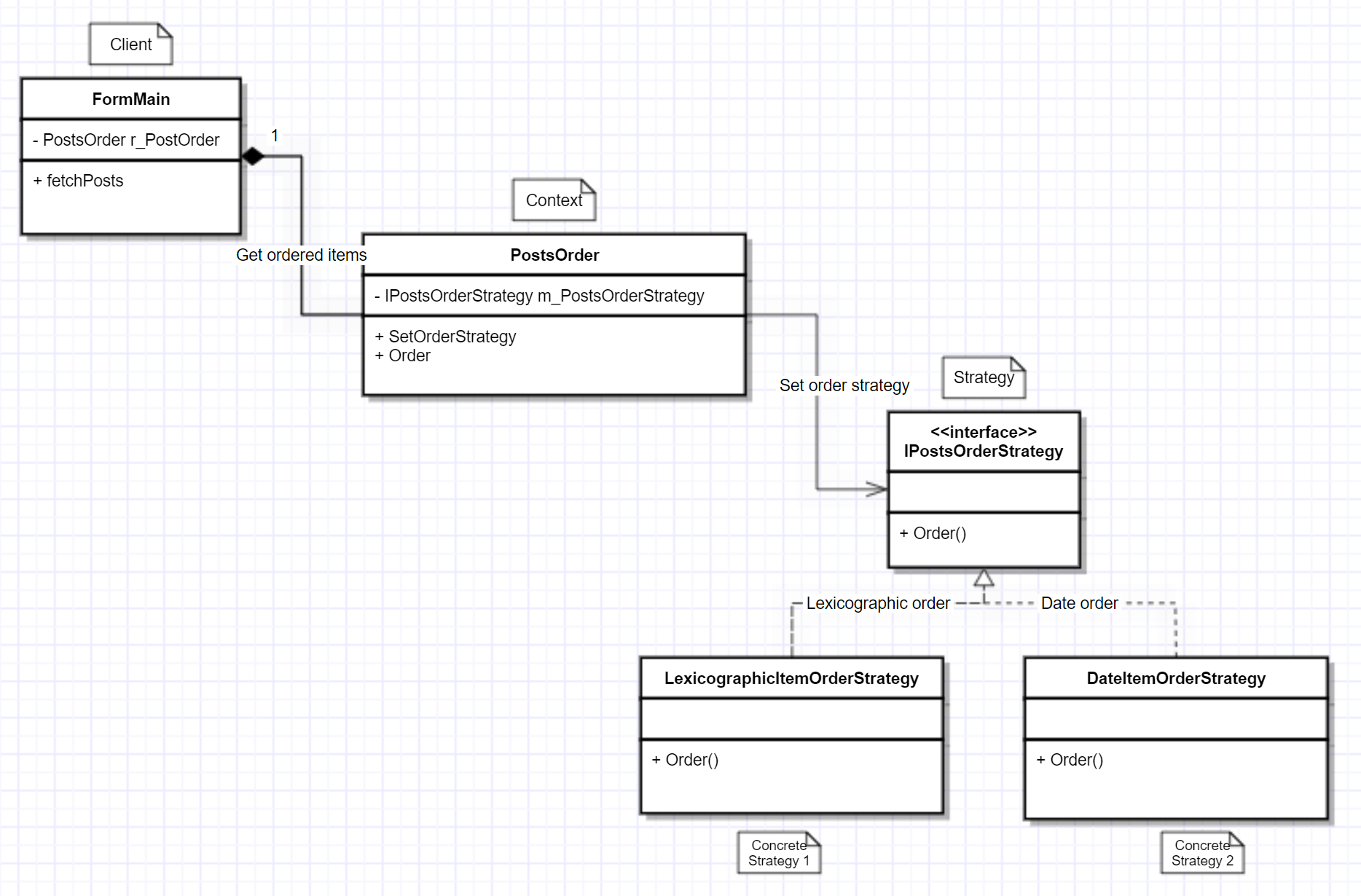
* אופן המימוש:

מימשנו אינטרפייס IPostsOrderStrategy שמכיל את מתודת Order שמקבלת IEnumerable<Post>.

בנוסף, מימשנו 2 מחלקות DateItemOrderStrategy, LexicographicItemOrderStrategy שמהוות Concrete Orderers שמממשות את האינטרפייס ע"י מימוש המתודה Order. כל מחלקה מממשת מתודה זו ע"י מיון הרכיבים לפי הStrategy שלה (בסדר כרונולוגי / בסדר לקסיקוגרפי).

כמו כן, מימשנו את מחלקת PostsOrder שאותה הFormMain מכירה, ותפקידה להחליט על הStrategy הנבחר וכן ממיינת IEnumerable<Post בהתאם.

המשתמש בוחר ע"י ComboBox את אופן המיון הרצוי והאפליקציה מציגה את הפוסטים בהתאם.

* Sequence Diagram
* Class Diagram