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

* סינון סטטוסים של החברים של המשתמש ע"י מילת (או משפט) מפתח.

כיום לא ניתן לסנן את הסטטוסים של החברים ע"י מילת חיפוש, לדוגמא, אם מעניין אותי לראות מי מדבר על "חוק הלאום" מהחברים שלי, אני יכול להכניס את זה לשורת הטקסט ולאחר לחיצה על כפתור "filter" לקבל את כל הסטטוסים של חבריי שמכילים את הביטוי "חוק הלאום".

לאחר קבלת הסטטוסים אפשר ללחוץ על הסטטוס ולראות מי רשם אותו (וכמובן את התמונה שלו).

* קבלת כל האיבנטים של החברים שלי במקום אחד.

משתמש פייסבוק לא יכול לראות במקום מרוכז את האיבנטים של כל החברים שלו. בנוסף בלחיצה על איבנט מתוך הרשימה, מתקבלת תמונות האיבנט והתיאור שלו, וכל החברים של המשתמש שבסטטוס "go" לאיבנט.

המשתמש בנוסף יכול להצטרף לאיבנט (אם הגדרות הפרטיות של האיבנט מאפשרות).

אם מעניין את המשתמש לקבל רק איבנטים שקשורים למסיבה, לדוגמא, הוא יכול להכניס "מסיבה" בשדה טקסט ולקבל את האיבנטים שבכותרת שלהם מופיעה המיל מסיבה – יכולת המאפשרת סינון האיבנטים ע"פ מילת עניין.

### תבנית מס' 1 – Proxy

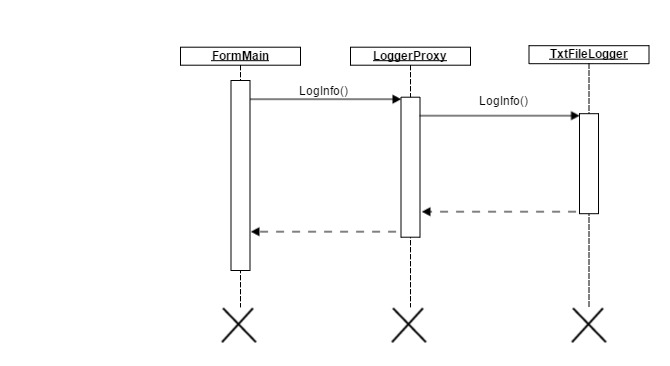
* סיבת הבחירה / שימוש בתבנית:

התבנית מאפשר בקרה על הגישה ל-ConcereteLogger, כך שה-client ניגש רק ל-proxy ולא ישירות ל-logger. בגישה ל-proxy מתבצעת ספירה של כמות הגישות ל-logger. ניתן להוסיף לוגיקה נוספת ל-proxy שאין ב- ConcereteLoggerכמו לבדוק האם יש למשתמש הרשאה לכתוב ל-log.

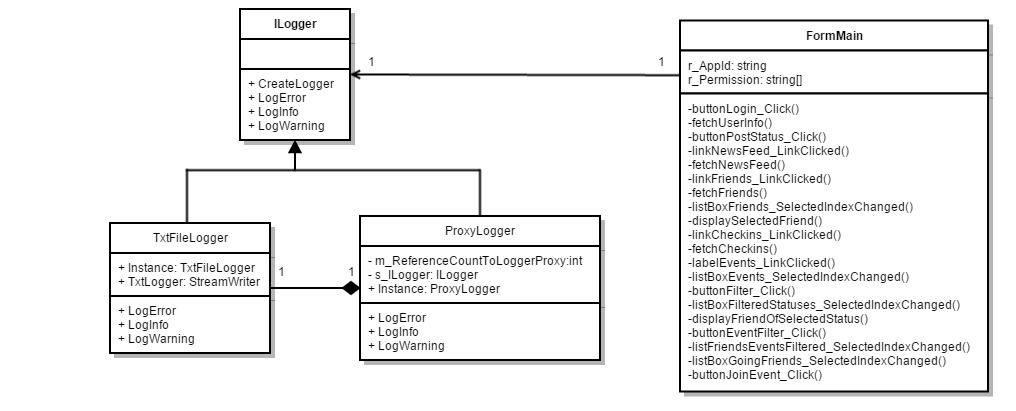
* אופן המימוש:

המחקלות TxtFileLogger (or EventLogger) ו-ProxyLogger יורשות מהמחלקה ILogger. V-ה-ProxyLogger מחזיק instance של ה-TxtFileLogger וה-client (FromMain) פונה ל-Logger רק דרך ה-ProxyLogger.

מבחינת ה"שחקנים" בתבנית זו ה-Proxy הוא ה-ProxyLoggey, ה-Subject הוא ה-ILogger וה-RealSubject הוא ה-TxtFileLogger (or EventLogger).

* Sequence Diagram

\* בדיאגרמה מוצגת ה-method LogInfo, כאשר ל-LogError & LogWarning יש אותו sequence

* Class Diagram

### תבנית מס' 2 – Singelton

* סיבת הבחירה / שימוש בתבנית:

השימוש ב-singelton מבטיח שיופעל logger יחיד ובכל קריאה ל- Log יוחזר ה-Instance של ה-Logger היחיד הזה.

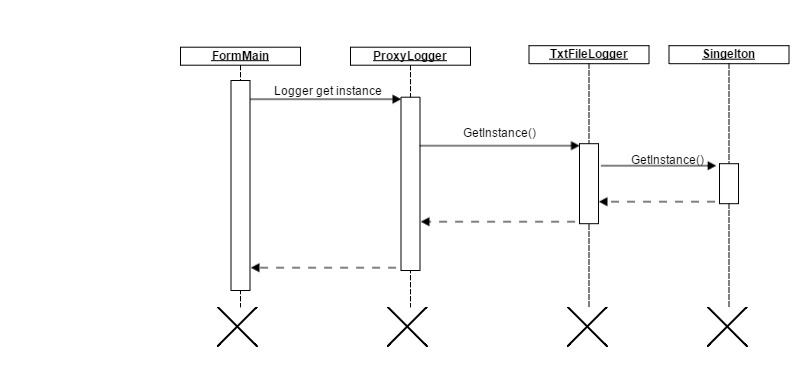
* אופן המימוש:

המחלקה Singleton מבצעת פנייה לבנאי הפרטי של האובייקט ע"י reflaction בהנחה שהוא עדיין לא נוצר, אחרת מחזירה את האובייקט הקיים.

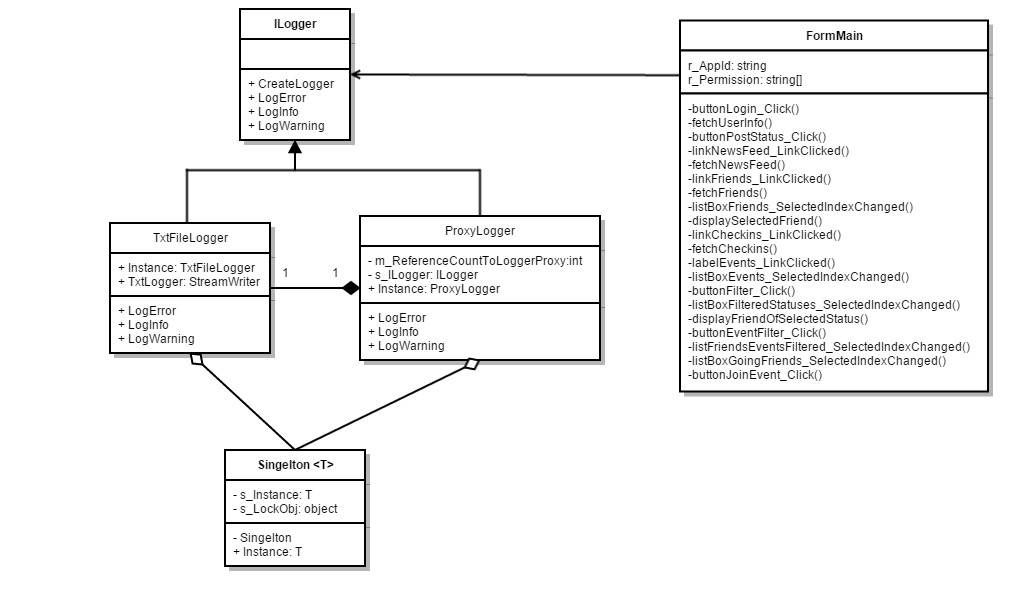
LoggerProxy ו-ConcreteLogger (TxtFileLogger או EventLogger) נוצרו בצורה זו.

בכל קריאה ל-LoggerProxy מוחזר ה- ConcreteLoggerהיחיד שנוצר וכך מובטח שכל ה-Logs יכתבו לאותו Log.

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – FactoryMethod

* סיבת הבחירה / שימוש בתבנית:

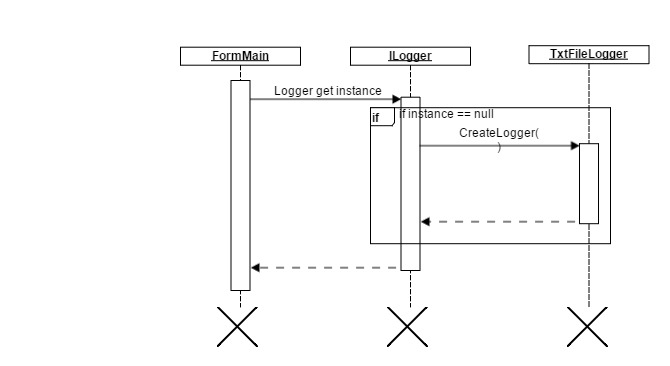
מכיוון שישנם 2 סוגי Log (וניתן לשם גם יותר מכך) אזי נדרש factory method שתאגד בתוכה את יצירת ה-Logger המתאים (ע"י גישה ל-App.config), אך בלי דעת את אופן היצירה הפרטי של ה-Logger הנבחר, ולכן בכל אפשרות מתבצעת בקשה לקבלת Instance של ה-Logger הנבחר. במימוש בקשה זו ה-Logger הנבחר יוצר (ע"י מימוש בתבנית singelton) את ה-ConcreteLogger.

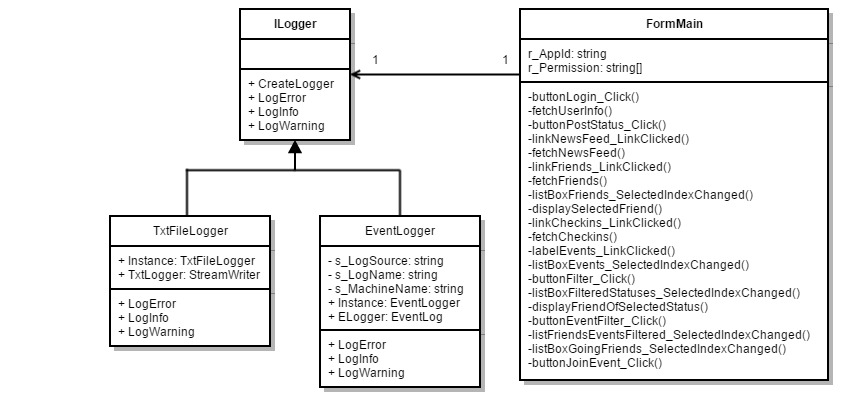
* אופן המימוש:

מימשנו את ה-FactoryMethod ע"י יצירת static method במחלקת הבסיס האבסטרקטית (ILogger).

מבחינת ה"שחקנים" בתבנית זו, ILogger היא המחלקה המייצגת את ה-Product, TxtFileLogger או EventLogger מצייגות את ה-ConcreteProduct, CreateLogger שנמצאת במחלקה ILogger מייצגת את ה-Creator וה-Instance של כל ConcreteProduct היא למעשה מי שמייצרת את ה-ConcreteProduct (בגלל ש-ConcreteProduct הוא singleton אזי הוא מיוצר רק פעם אחת).

* Sequence Diagram



* Class Diagram

### עבודה אסינכרונית

על מנת לאפשר העלאת נתונים חלקה בדף הבסיסי של האפיקציה השתמשנו ב threads.

כעת אפשר לבצע fetch והאפליקציה לא נתקעת.

מכיוון שבטאבים האחרים יש כפתור אחד אז בכל מקרה צריך להמתיין לפעולתו לאחר הלחיצה עליו ולכן לא בוצע שימוש ב-threds בטאבים אלו.

ה-methods בהן מבוצע המימוש הן:

fetchNewsFeed

fetchFriends

fetchCheckins

### עבודה עם Data Binding

פרטי ה event בדף הראשי מקושרים לרשימה של ה-events שלידם.

ה-methods בהן מבוצע המימוש הן:

labelEvents\_LinkClicked

listBoxEvents\_SelectedIndexChanged