**מסמך מפורט על עבודה 2**

מגישים: לירן כהן ואופל אפטר.

תעודות זהות: 201392131, 308345438.

סעיף א: תבניות העיצוב הבאות מומשו בפרויקט:

1. Abstract Factory.
2. Singleton.
3. Class Proxy.

פירוט:

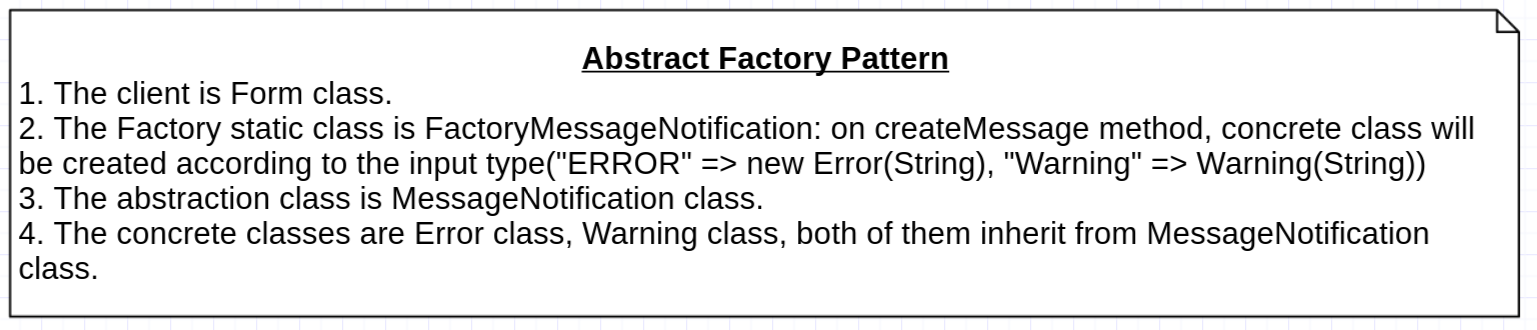
1. תבנית מספר 1 מומשה בעקבות הצורך ביצירת הודעה מתאימה לממשק המשתמש, כאשר התרחשה לוגיקה שגרמה לשגיאה או לאזהרה, לדוגמה כאשר המשתמש ינסה לפרסם סטטוס ללא תוכן באפליקציה, תקרה שגיאה באפליקציה והמשתמש צריך לדעת שמצב מערכת זה לא תקין ובכך יש להודיע לו על השימוש הלא תקין ולתת לו הודעה שמסבירה איך צריך להשתמש באפליקציה ולפרסם נכון סטטוס.

1. תבנית מספר 2 מומשה בעקבות הצורך במופע אחד ויחיד שמאפשר התחברות של משתמש, שמירת נתוני המשתמש ושימושם לכל אורך חיי האפליקציה. יש צורך במופע אחד שלא ניתן לשכפל אותו וממנו לבקש את נתוני המשתמש כגון:(חברים, פרסומים וכדומה). כלומר, כאשר משתמש מעוניין להשתמש בשירותי האפליקציה, ראשית, עליו להתחבר ולהזדהות עם שם המשתמש וסיסמא שלו מול פייסבוק. בכל שרות שנתמך באפליקציה יש צורך בנתוני המשתמש לכן נוכל לחסוך בשכפול המידע ממחלקה למחלקה על ידי שימוש בתבנית העיצוב Singleton, בעזרת תבנית זו נוכל לבקש בכל רגע נתון refernce עם נתוני המשתמש הנוכחי. בנוסף לכך, מבחינה קונספטואלית לא תקין ליצור יותר ממופע אחד שמאפשר התחברות לאפליקציה.
2. תבנית מספר 3 מומשה בעקבות הצורך ביצירת רכיב משופר לרכיב קיים, כלומר הדפדפן ברירת המחדל של WinForm לא סיפק את הצרכים של האפליקציה כגון: הגרסה שלו נמוכה מהגרסה שבה אפשר להפעיל דפים מודרניים בדפדפן, לדוגמה יוטיוב, גוגל מפות.

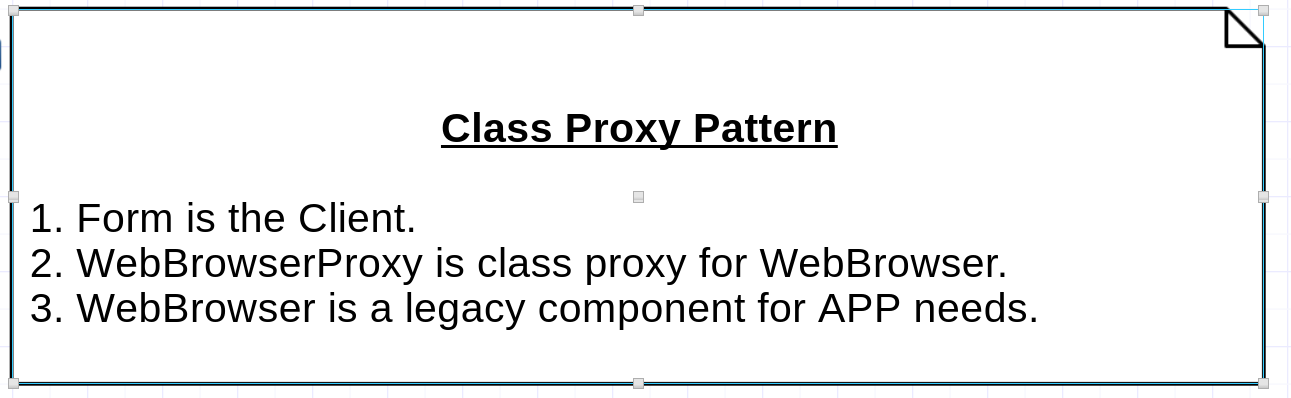
לכן, הרכיב החדש יורש מהרכיב הישן ומוסיף שיפורים וקונפיגורציה חדשה, מתחקה לרכיב הישן בכך שכל קליינט משתמש בו כמו שהשתמש ברכיב הישן.

סעיף ב:

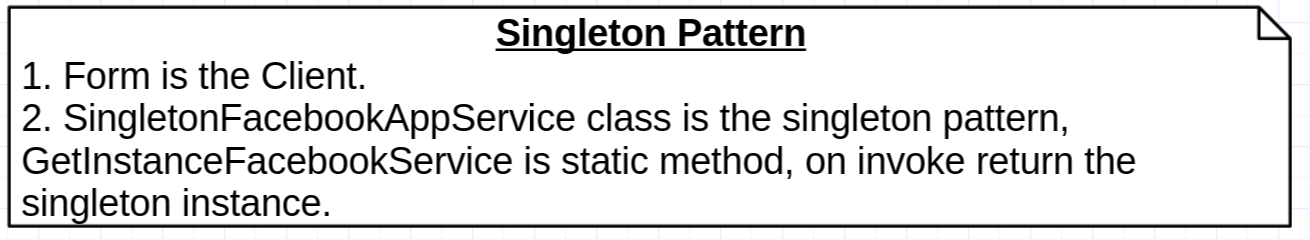
Abstract Factory details:



Class Proxy details:

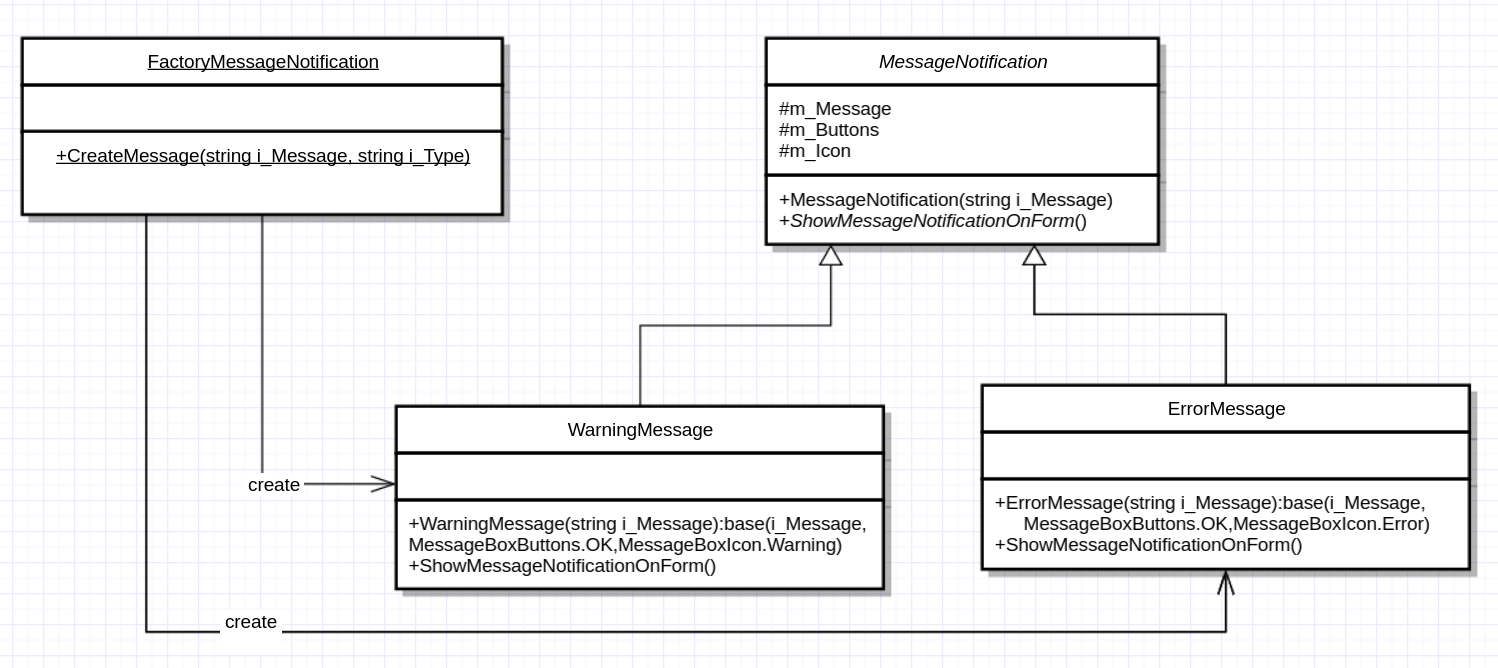


Singleton details:

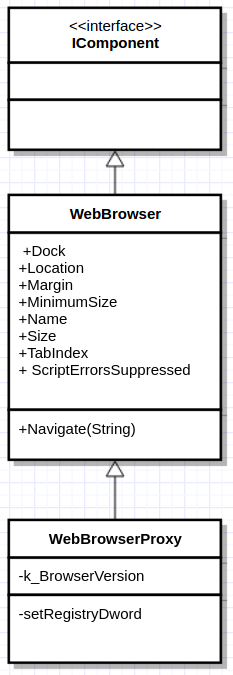


סעיף ג:

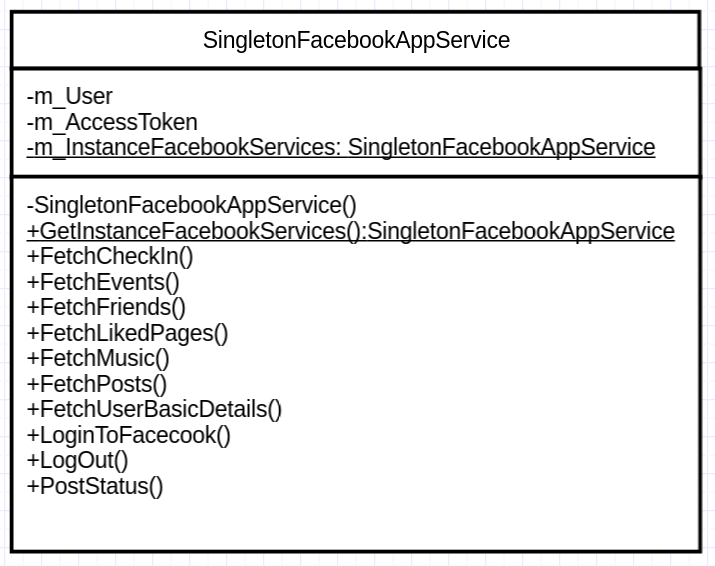
Abstract Factory Class Diagram:



Class Proxy Class Diagram:

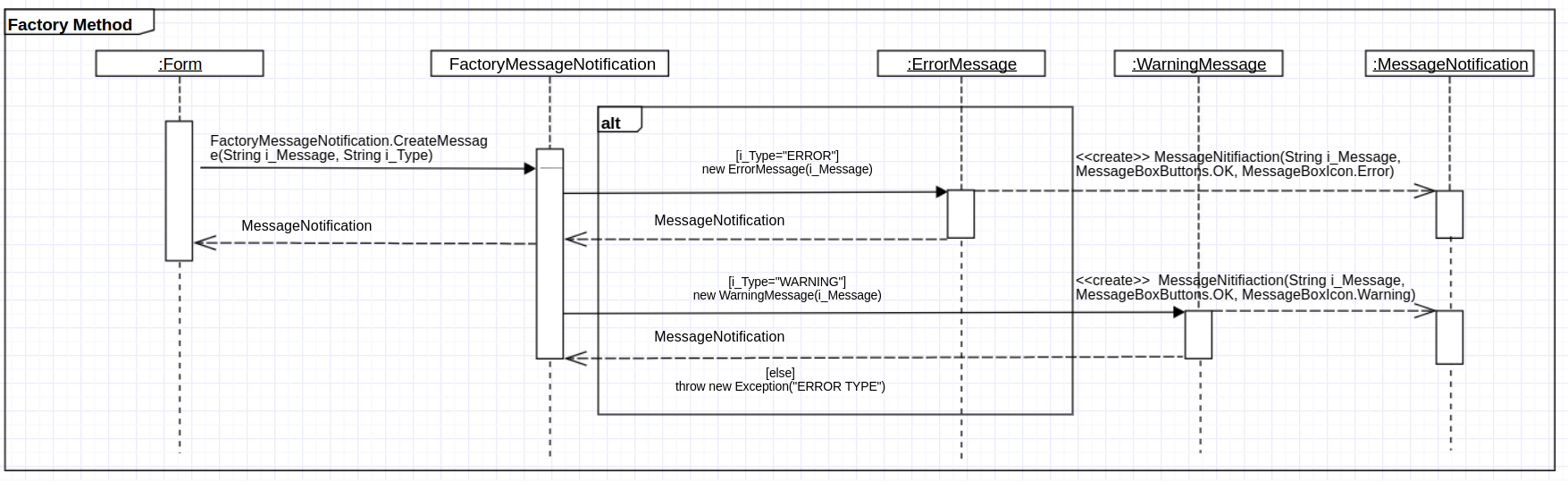


Singleton Class Diagram:

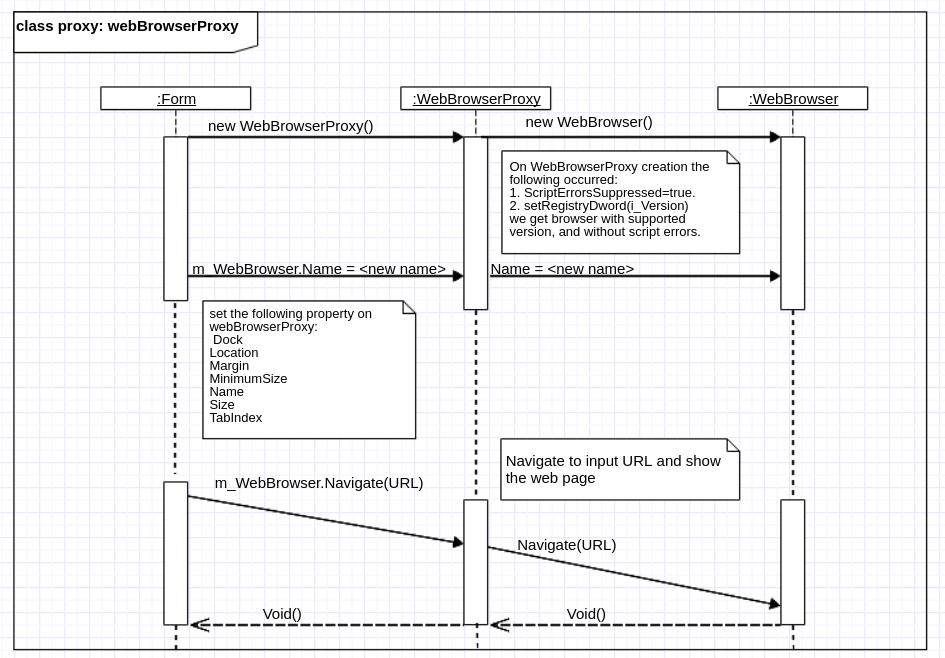


סעיף ד:

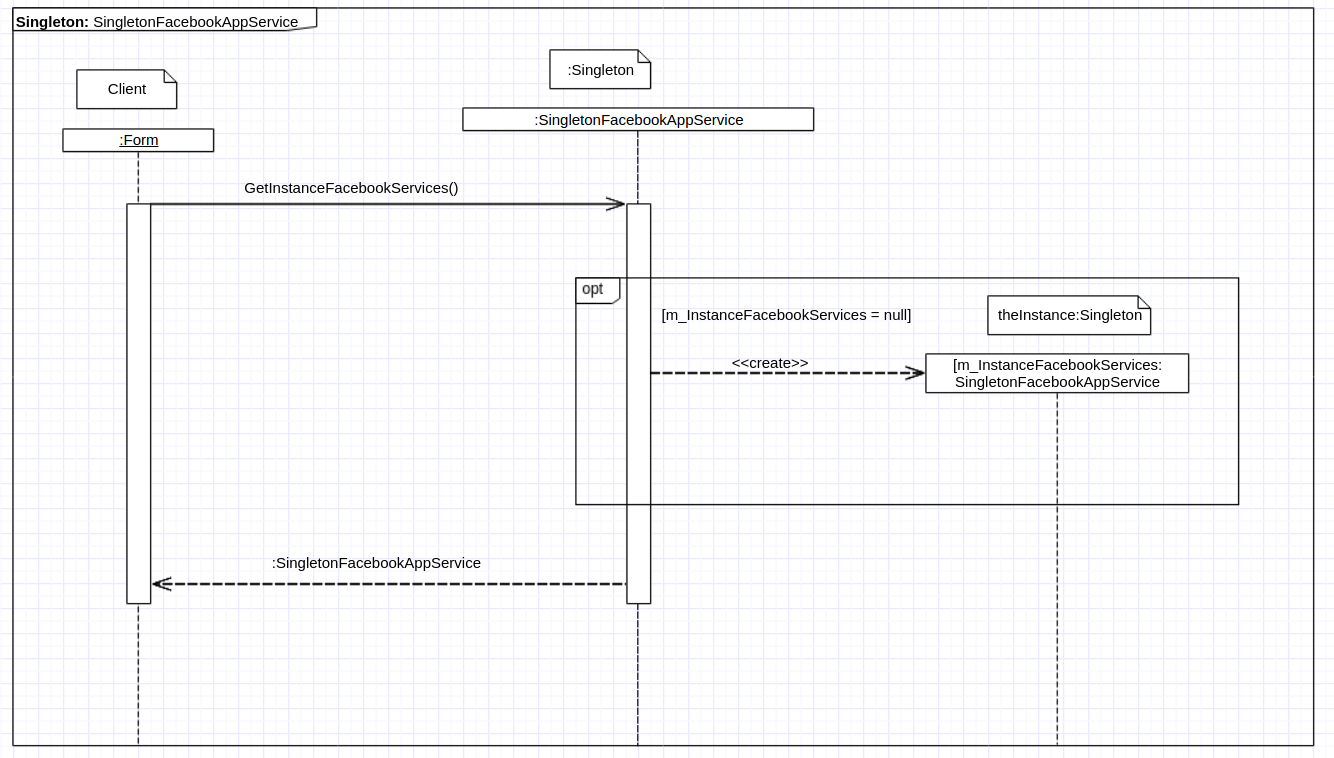
Abstract Factory Sequence Diagram:



Class Proxy Sequence Diagram:

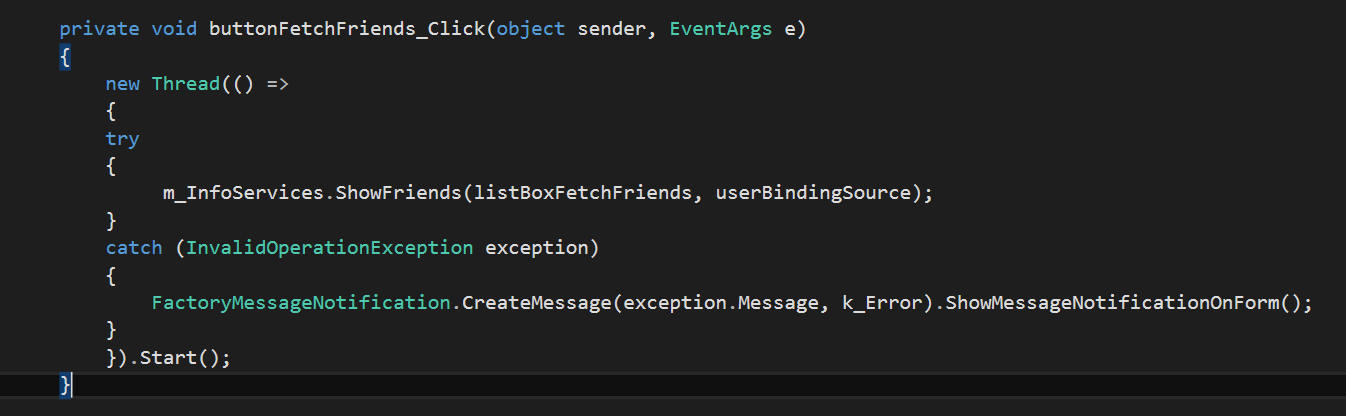


Singleton Class Diagram:



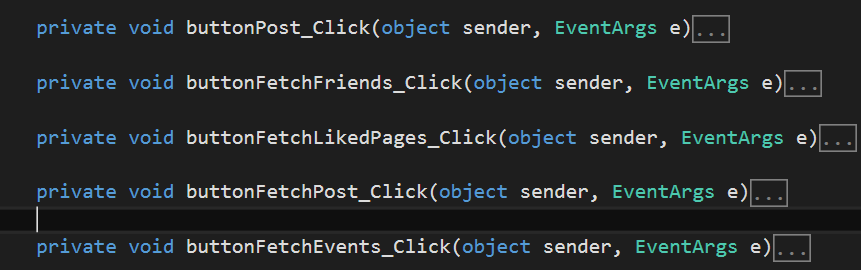
**עבודה אסינכרונית:** שימוש בעבודה אסינכרונית מבוצע במחלקות הבאות:

FormInfo, FormMusic,FormCheck-IN, בכל אירוע של לחיצת כפתור מתבצעת עבודה אסינכרונית, להלן דוגמת קוד:



דוגמה זו מהווה תבנית שבה השתמשנו בכל האירועים של לחיצת כפתור במחלקות שהוזכרו לעיל.

להלן פירוט המחלקות והמתודות:

**FormInfo:** 

**FormMusic:**

**Selection_175.png**

**FormCheck-in**

**Selection_176.png**

הצורך בפעולות אסינכרוניות לאירועים המוזכרים לעיל נובע מחוסר יכולת לבצע פעולות שונות במקביל וללא המתנה שפעולה תסתיים כדי שפעולה אחרת.

לדוגמה, באפליקציה, כאשר משתמש מבקש לראות את רשימת הלייקים שלו, מופעלת פעולה שרק בסיומה יוכל המשתמש לחזור ולהשתמש באפליקציה לצורך פעולות אחרות. כלומר, עד שהפעולה לא מסתיימת המערכת "קפואה".

הפעולות האסינכרוניות באפליקציה מומשו בעזרת תהליכונים - Threads, כאשר בכל אירוע של לחיצת כפתור(במחלקות שציינו לעיל) מופעלת המתודה הקונקרטית בתהליכון-Thread חדש.

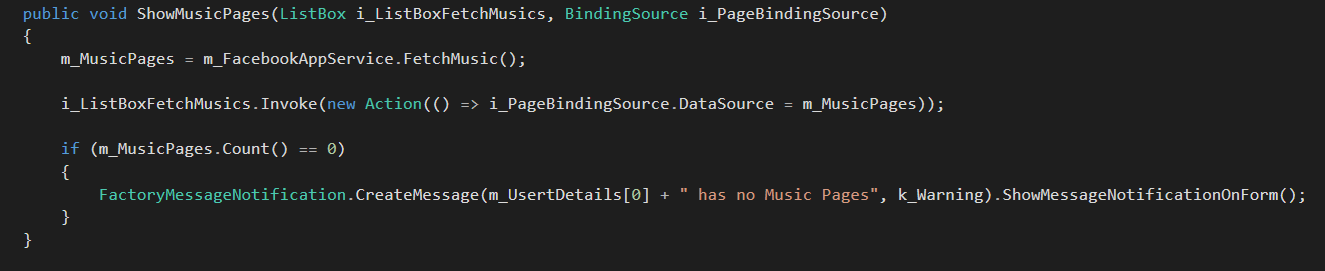
**DataBindings**

שימוש ב- DataBindings מבוצע במחלקות הבאות:

Check-InServices, InfoServices,MusicServices

כאשר נרצה לבצע חיבור של DataSource לאובייקט DATA כלשהו לדוגמה ListBox נבצע חיבור של הנתונים המבוקשים ל DataSource של BindingSource בהתאם לסוג האובייקט המבוקש , לדוגמה על מנת לטעון את דפי המוזיקה שהמשתמש אהב, נשתמש ב DataSource מסוג אובייקט Page.

להלן דוגמת קוד:



שימוש ב- DataBindings נמצא במקומות הבאים:

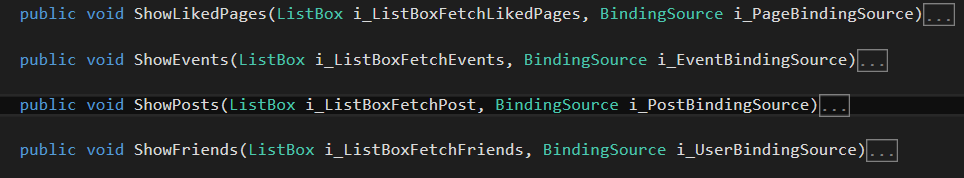
**MusicServices**

**Selection_180.png**

**Check-InServices**

**Selection_181.png**

**InfoServices**

****