תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל הקודם:

- **סינון פוסטים/צ'ק-אין ע"פ זמן**
הפיצ'ר מהווה סינון לנוחיות המשתמש של הפוסטים/צ'ק אין ע"י בחירת המשתמש (כל הזמנים, יום נוכחי, חודש נוכחי, שנה נוכחית).

תוצאת החיפוש תחזיר רשימת פוסטים/צ'ק אין בהתאם לבחירת המשתמש , ובנוסף תאריך פרסומם.

  - הסינון של הפוסטים כולל **הצגת הפוסט הפופולרי ביותר בהתאם.**
  - **בצ'ק אין ניתן גם לראות מיקום על גבי google maps** בעת בחירת הפריט.

מחלקות הקשורות לפיצ'ר:

  - ממשק משתמש: FormPosts, FormCheckins תחת FaceBookApp.UI.
  - החלק הלוגי: eNumFilter, FacebookObjectFilter<TTT> where TTT : Post תחת FaceBookApp.Logic.

- **YouTube Music:**
הפיצ'ר מציג למשתמש את דפי המוזיקה שלו ותמונותיהם ובהתאם לבחירתו יפתח דפדפן המציג לו את תוצאות חיפוש של דף המוזיקה אותו הוא בחר והוא יכול לשמוע את השירים של אותו אומן.

מחלקות הקשורות לפיצ'ר:

  - ממשק משתמש: FormMusic תחת FaceBookApp.UI.

- **Active Friend:**

בממשק המשתמש מופיע רשימת החברים, ע"י בחירת חבר מהרשימה מופיע פאנל מתחת לרשימה המציג את תמונת החבר והאם החבר פעיל, הבדיקה מתבצעת ע"י משיכת הפוסט האחרון של החבר ובדיקת תאריך פרסומו בהשוואה לחודש הנוכחי, בנוסף תוצג גם תאריך פרסום הפוסט האחרון.

מחלקות הקשורות לפיצ'ר:

  - ממשק משתמש: FormMain תחת FaceBookApp.UI.

- **חיפוש בפוסטים של החברים ע"י טקסט** (פיצ'ר שנוסף בעקבות הערה בתרגיל הקודם)
הפיצ'ר נותן אפשרות למשתמש לחפש ע"י מילת מפתח פוסטים של חבריו , תוצאת החיפוש תחזיר רשימת פוסטים בהתאם למילת המפתח.

מחלקות הקשורות לפיצ'ר:

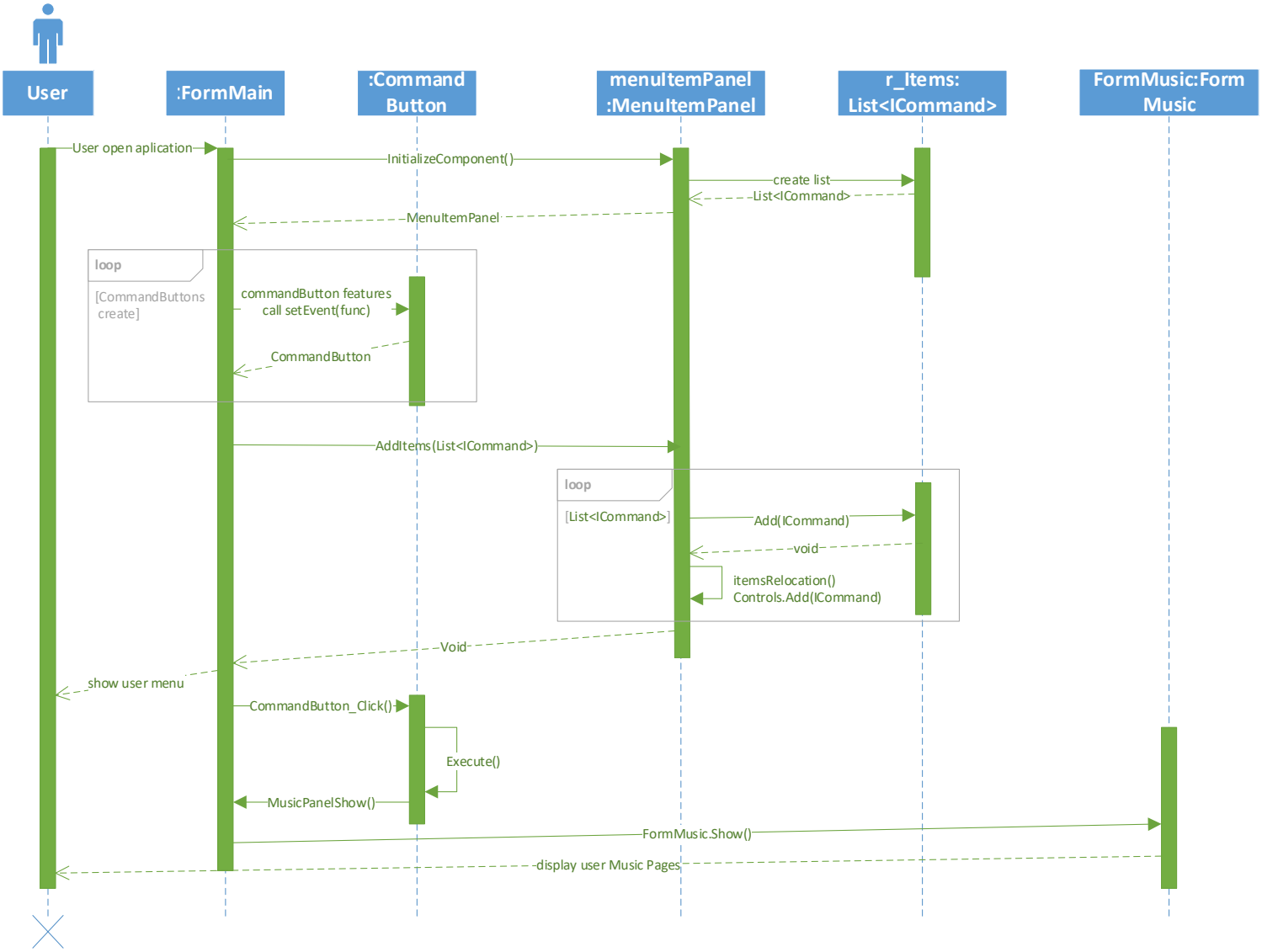
  - ממשק משתמש: FormSearchPosts תחת FaceBookApp.UI.
  - החלק הלוגי: SearchFriendsPosts.

תבנית מס' 1 – [Command Pattern]

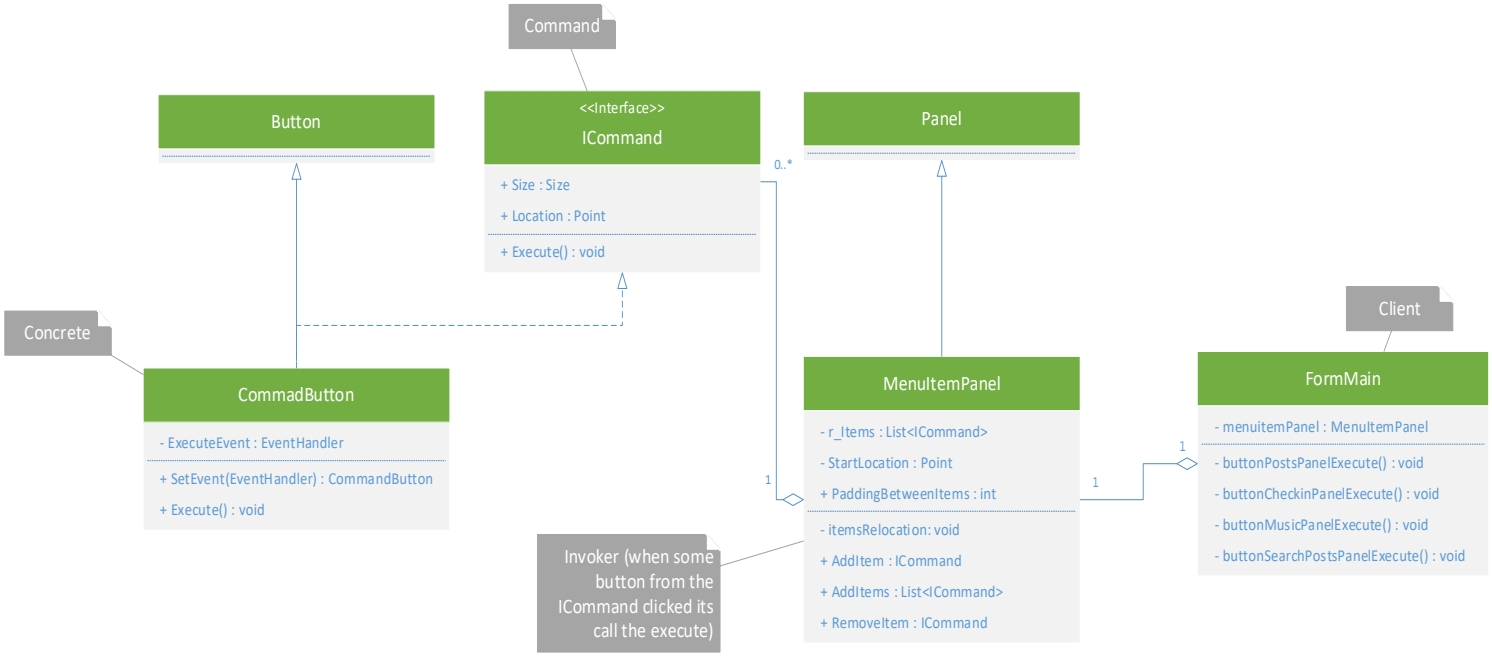
- סיבת הבחירה / שימוש בתבנית:
  באפליקצייה שלנו יש פאנל בצד שמאל שמכיל כפתורי תפריטים המועצבים באופן מותאם לשם מעבר לפיצ'ר הרלוונטי באפליקצייה, במידה ובעתיד נרצה להוסיף פיצ'רים נוספים אז נבזבז זמן יקר ביצירה, מיקום או בתיקון במידה ונשכח עיצוב מסויים או כאשר נרצה למחוק ואז למקם הכל מחדש.
  נוכל לייצר את כפתור התפריט עם כל תכונותיו בצורה אוטומטית ע"י שימוש בתבנית ה Command ללא התעסקות בעיצוב הכפתור מבחינת תמונה,מיקום, שם וכו' וכך לחסוך זמן.

- אופן המימוש:
  יצרנו ממשק בשם **ICommand** המכיל מתודה Execute (שתפקידה להיות אחראית להצגת התפריט המתאים לאותו לחצן), את הממשק מימשנו בעזרת המחלקה **CommandButton** היורשת את המחלקה Button (proxy) ומכילה משתנה מסוג EventHandler שאותו תפעיל המתודה Execute.
  על מנת שנוכל למקם את לחצני התפריטים באופן מותאם ואוטומטי יצרנו מחלקה בשם **MenuItemPanel** היורשת את Panel (proxy) ומחזיקה רשימת של ICommand המייצגים את לחצני התפריט וכך ממקם אותם בUI וזה נעשה ע"י שימוש במיקום הנוכחי של הפאנל (הנשמר במשתנה StartLocation) והרווחים ביניהם ע"י שימוש במשתנה (PaddingBetweenItems) כמו כן המחלקה מאפשרת הוספה, מחיקה מהרשימה באמצעות מתודות מתאימות.

- Sequence Diagram – תהליך של יצירת הלחצני תפריטים ודוגמא של לחיצת על תפריט של תצוגת דף FormMusic



- Class Diagram

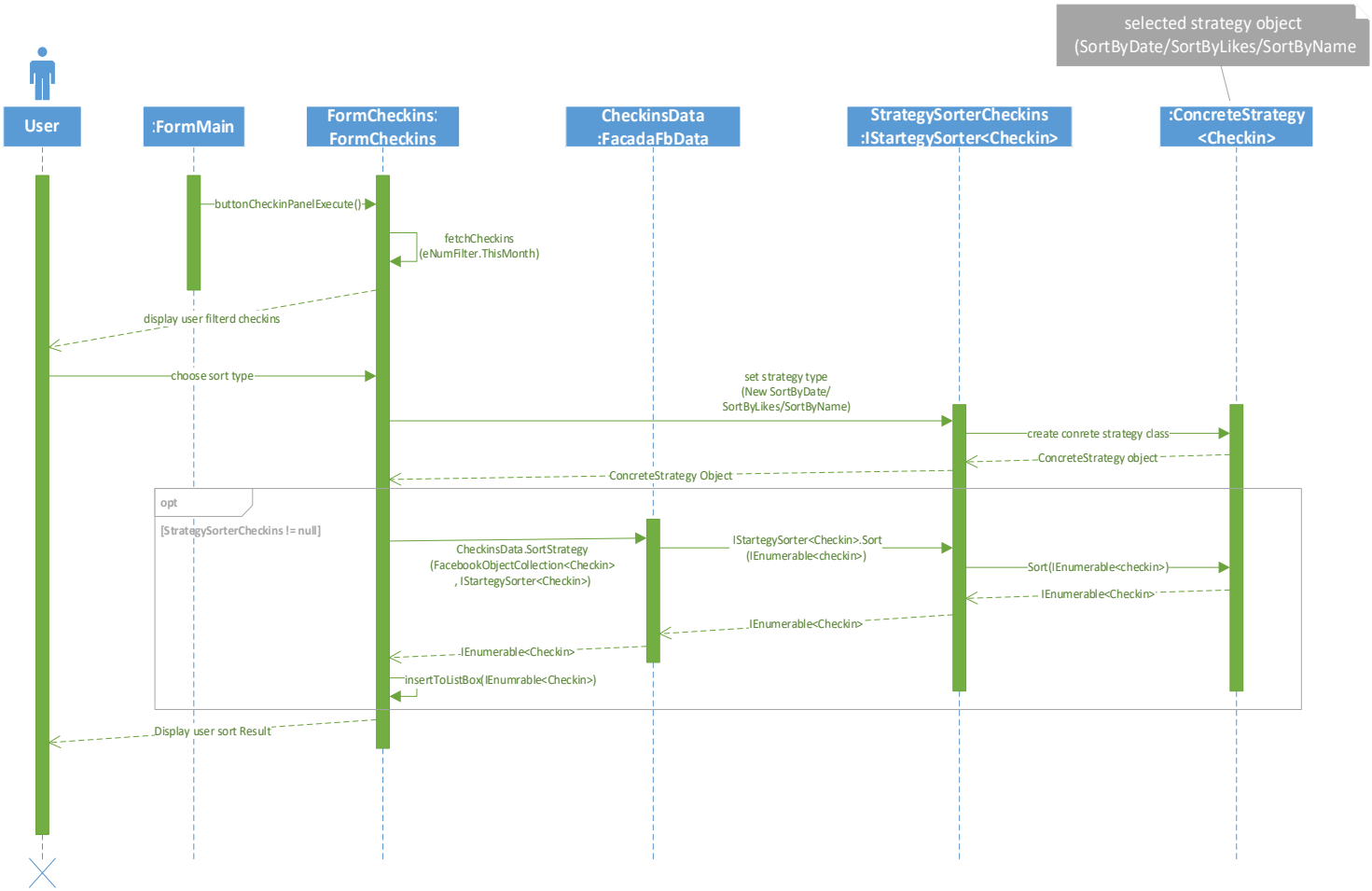


תבנית מס' 2 – [Strategy Pattern]

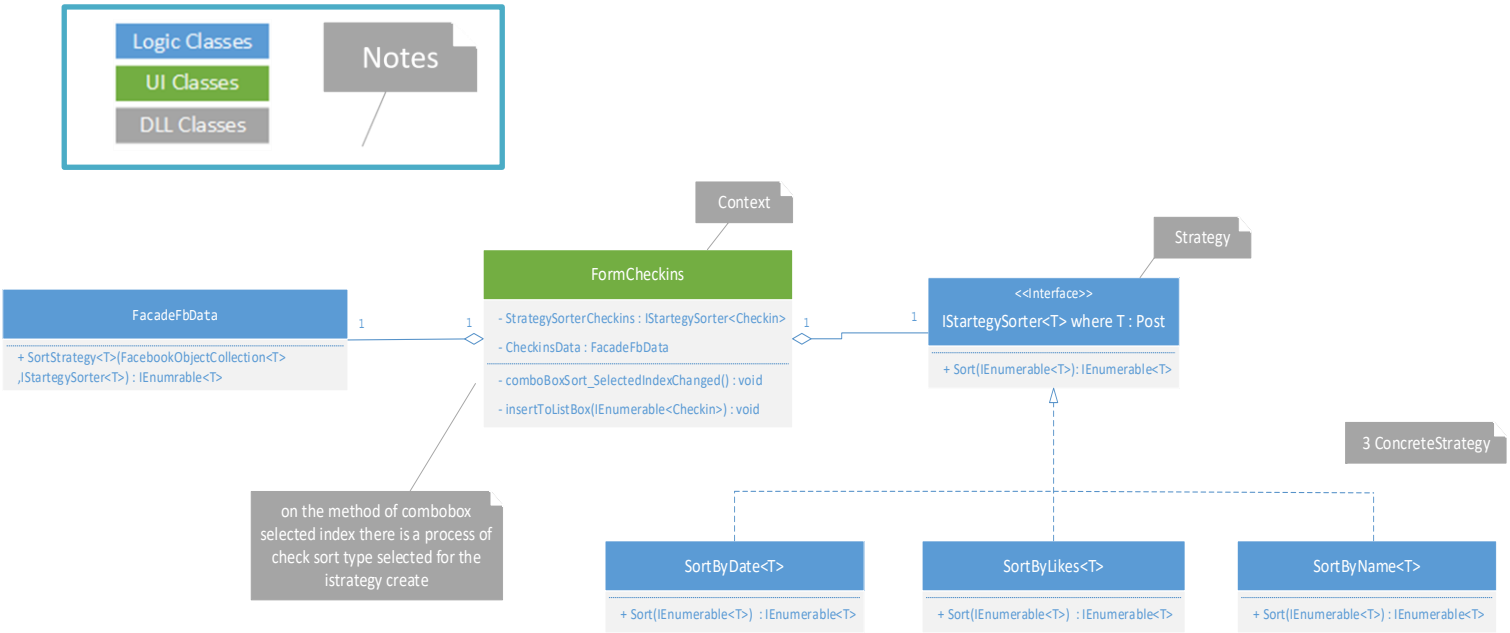
- סיבת הבחירה / שימוש בתבנית:
באפליקציה שלנו יש אפשרות למיין את תוצאות הcheckin ע"פ שם, לייקים, תאריך אשר כל מימוש שונה בהתנהגותו כלומר בדרך המיון שלהם ולכן נעזרנו בתבנית הstrategy כדי לכמס אותם תחת משפחת אלגוריתם של מיון.

- אופן המימוש:
יצרנו ממשק בשם **IStartegySorter** אשר מגדיר משפחת אלגוריתמים של מיון, כרגע אופציית המיון פועל על פיצ'ר הצ'ק-אין על מנת להתאים לשימוש עתידי גם לפוסטים יישמנו את הממשק שלנו generic כאשר האובייקט T הוא מסוג Post, בנוסף על מנת לא להגביל את מבנה הנתונים השתמשנו בIEnumrable גם בערך קבלה וגם בערך החזרה.
את הממשק מימשנו באמצעות 3 מחלקות (ניתן להוסיף עוד בהתאם לסוגי המיון) והן: **SortByDate** אשר תחזיר מיון לפי תאריך, **SortByLikes** אשר תחזיר מיון לפי כמות לייקים, **SortByName** תחזיר מיון לפי השמות.
כל אחת מהמתודות מחזירה IEnumrable (כי הפעולה שנעשת על המבנה נתונים שחוזר הוא רק foreach בשביל להציג את הנתונים) כמו כן המימוש של המיון נעשה באמצעות שימוש בLinq .

- Sequence Diagram – תהליך של יצירת הconrete המתאים והפעלתו (מוצג כתהליך כללי כי יש 3 סוגי מיון שונים שהתהליך שלהם אותו הדבר).



- Class Diagram

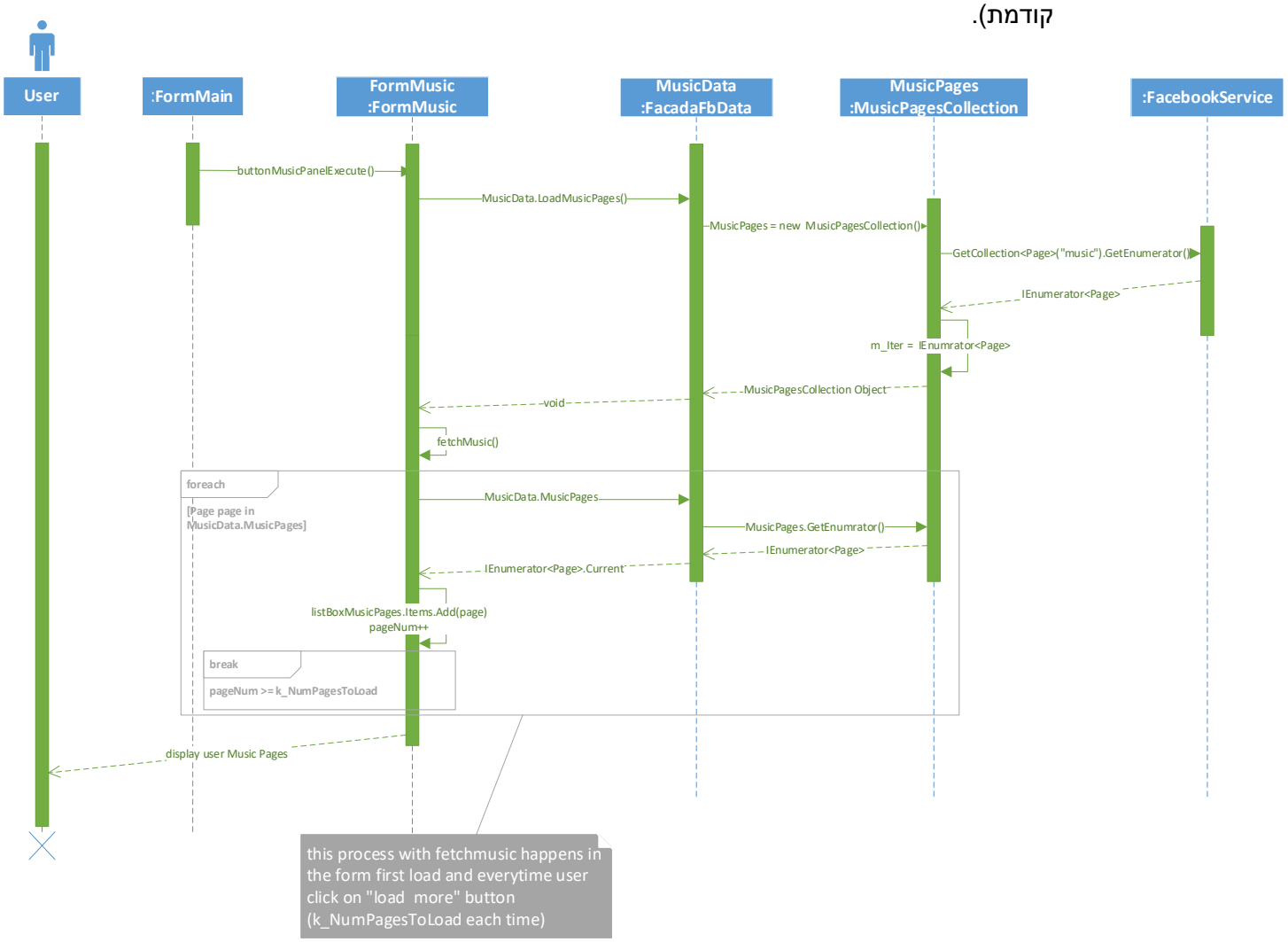


תבנית מס' 3 – [Iterator]

- סיבת הבחירה / שימוש בתבנית:
באפליקצייה שלנו אנחנו מציגים למשתמש את דפי המוזיקה שלו עם אפשרות לצפייה ביוטיוב ע"פ שם דף המוזיקה, במידה ומדובר במשתמש עם המון דפי מוזיקה הדבר יכול לגרום לטעינה ארוכה של הדפים, ולכן על מנת להקל על טעינת הנתונים ושימוש במשאבים יישמנו איטרטור אשר רץ על דפי המוזיקה כך שהוא מוגבל ל5 דפים (ניתן לשינוי ע"י משתנה k_NumPagesToLoad) ובאפשרות המשתמש לטעון עוד דפי מוזיקה. בנוסף שימוש בתבנית זו מאפשר הסתרת מבנה הנתונים.

- אופן המימוש:
יצרנו מחלקה בשם **MusicPagesCollection** אשר מממשת את הממשק **IEnumrable** (getEnumrator), מחזיקה רפרנס **לIEnumrator** (m_Iter) אשר הוא מאותחל בConstructor של המחלקה ומחזיק Enumrator של אוסף דפי המוזיקה (FacebookObjectCollection<Page>). בFacade אנחנו מחזיקים את **MusicPagesCollection** והפנייה אליו נעשית מתוך הFormMusic (במתודה fetchMusic) באמצעות לולאת ForEach שמחזירה כל פעם את הiterator הנוכחי של האוסף ומוסיפה אותו לlistbox בשביל תצוגת המשתמש, והiterator עוצר כאשר מגיעים להגבלת הדפים (k_NumPagesToLoad), כאשר המשתמש רוצה לייבא עוד דפי מוזיקה הוא יכול לעשות זאת ע"י לחיצה על כפתור "Load More Pages" אשר ממשיך מאותה נקודה האחרונה שנעצר הIterator.

- Sequence Diagram – תהליך השימוש בiterator בעת כניסה ראשונית לדף הפיצ'ר, החלק של הלולאה חוזר על עצמו כאשר משתמש בוחר לטעון עוד דפי מוזיקה בעת לחיצה על כפתור "load more" (ממשיך מאותה נקודת עצירה קודמת).



- Class Diagram