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

* FaceBookPlayer – שואב את הלינקים של היוזר ומפלטר אותם לרשימת שירים ממנה הוא יכול ליצור Playlist ומציג ממשק נוח להצגתן.
* Profile picture picker – מאפשר למשתמש ליצור אלבום של 5 תמונות מכל התמונות בכל האלבומים שלו ליצירת אלבום בו יעזרו לו (לפי לייקים) לבחור תמונת פרופיל חדשה.

### תבנית מס' 1 – Template Method

* סיבת הבחירה / שימוש בתבנית:

באפליקציה שלנו, קיים הפיצר FacbookPlayer שמשמיע שירים שהעלה המשתמש.  
כאשר אנחנו מקבלים לינק שפרסם המשתמש – בהתאם לאתר ממנו הם פרסמו , בין אם YouTubeSong, FacebookSong או SoundCloudSound, עלינו "להפוך" את הלינק ל2 סוגי לינקים:

1. אודיו – לינק שיעבוד בautoplay ולא חשוב התצוגה שלו.
2. וידאו – לינק שיהיה embedded וימתח על כל תצוגת הBrowser.

עבור כל אתר מהאתרים שיציינו, קיימת לוגיקה שונה להפיכת הלינק לאודיו ווידאו, אבל מצד שני לכל אחד מהאתרים האלה נרצה עדיין להתייחס בצורה דומה עם הפעולות של קבלת הלינק לאחר הפילטור וכדומה..

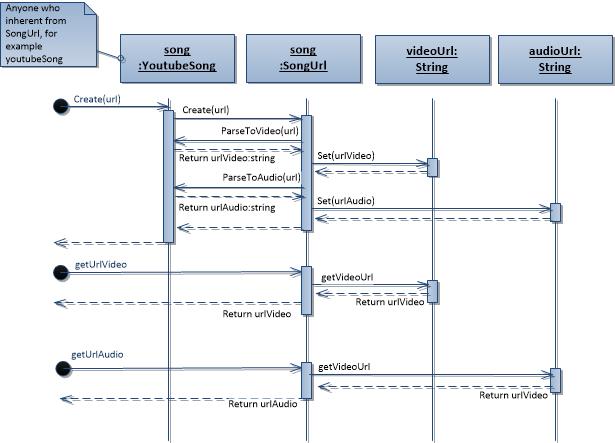
* אופן המימוש:

באפליקציה שלנו, תחת הפרוייקט FacebookPlayerLogics קיימת משפחה פולימורית עם המחלקה האבסטרקטית SongURL ממנה יורשים YouTubeSong, FacebookSong ו SoundCloudSong.

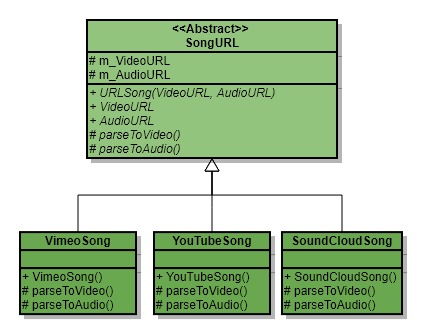
האב האבסטרקטי הוא SongUrl, המתודות Potected override (שהן בעצם הinjection point ) הן parseToAudio ו praseToVideo שיתממשו שונה בהתאם למי שירש מ.UrlSong

עוד בקונסטור של SongUrl אנחנו מזמנים את המתודות על מנת לשמור אותן ב Members m\_VideoURL ו m\_AudioURL. ומחזירים אותם באמצעות הProperties VideoURL ו AudioURL.

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – Strategy

* סיבת הבחירה / שימוש בתבנית:

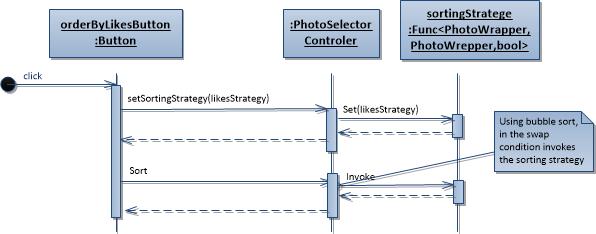
באפליקציה שלנו קיים פיצ'ר Profile Picture Picker, כחלק מאפשרויות הפיצ'ר, אנחנו מציגים את התמונות מאלבומי המשתמש. לנוחות השימוש בפיצ'ר, רצינו לאפשר לסדר את את התמונות לפי מספר שיטות: מספר לייקים, מספר תגובות, זמן וכדומה..

ראינו צורך ליצור אפשרות לinjection point עבור אסטרטגית המיון שלנו.

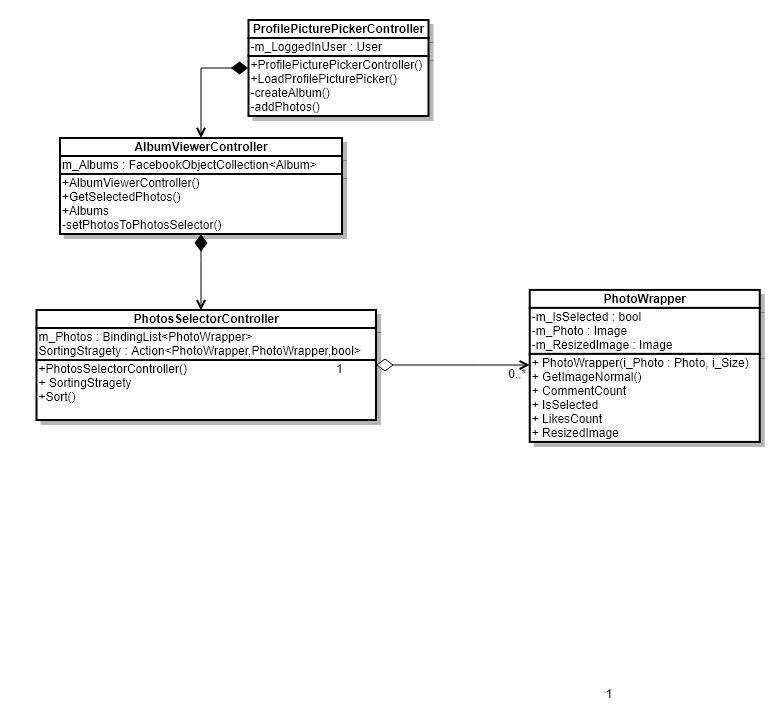
* אופן המימוש:

בקוד קיימים הControllers שיצרנו – AlbumViewrController שמחזיק בתוכו PhotoSelectorController. מכיוון ש PhotoSelectorController מחזיק בתוכו את קולקצית התמונות, שם גם נמצא הSortingStrategy שהחלטנו לממש באמצעות מצביע לפונקציה Func<PhotoWrapper,PhotoWrapper,bool> שמקבלת 2 PhotoWrapper ומחזירה bool בהתאם לאסטרטגית המיון שתמומש.

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – Observer

* סיבת הבחירה / שימוש בתבנית:

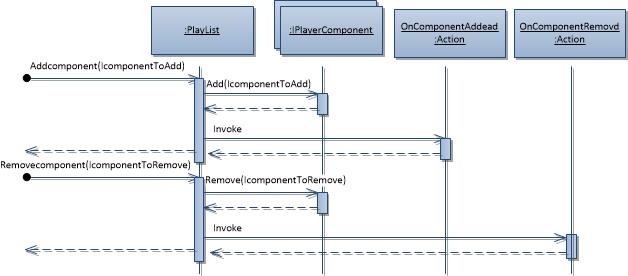
בפיצר Facebook player קיימת רשימת השמעה שהמשתמש יכול לערוך.  
רשימה זו ממומשת באמצעות תבנית העיצוב Composite, ותצוגתה בui היא בTreeViewer.

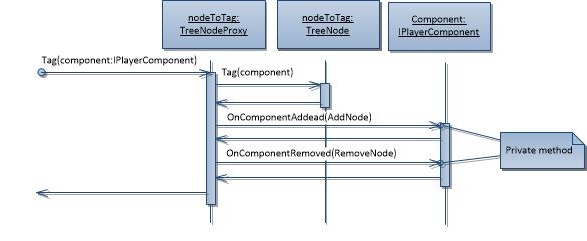
רצינו שהTreeViewer ישתנה בהתאם לשינויים שמתבצעים ברשימת ההשמעה, זאת אומרת שבעת הוספת רכיב בפלייליסט Composite, יהיה שינוי גם בTreeView  
ובמילים אחרות, שכל Node ב TreeViewer יאיזין לישנויים שקורים בחוליה אליה הוא מקושר בComposite.

* אופן המימוש:

מימשנו את תבנית העיצוב Observer בכך שתחילה יצרנו מתחזה לNode בשם NodeProxy בכך שכששמים IPlayerComponent בTag Member שלו, אנחנו דואגים גם לרשום את Node להאזנה לשינויים בעת הוספה או הסרה של רכיבים מהIPlayerComponen.  
בIPlayerComponent, שהוא הObserver, שמנו את האפשרות להירשם לOnComponentAdded וOnComponentRemoved כך שאם יוסף או ימחק אובייקט, IPlayerComponent יידע את מי שרוצה (אחד או יותר), במקרה שלנו את הNodeProxy ויפעיל אצלו פעולה למחיקה/הוספה בשביל שיתבצע שינוי בui.

* Sequence Diagram





* Class Diagram

