### 

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

**משחק אלבומים:**

* הפיצ'ר מציג למשתמש ארבע תמונות Cover מתוך אלבומיו.
* למשחק בממשק הנוכחי – שתי אופציות שונות:

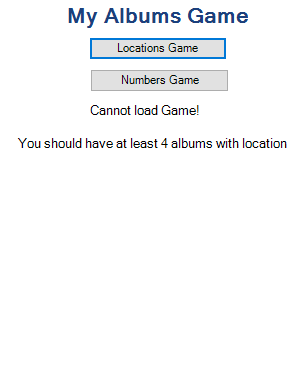
משחק מיקומים – על המשתמש לנחש את המיקום שהוגדר לאלבום (Album.Location).

משחק כמות תמונות – על המשתמש לנחש כמה תמונות באלבום (Album.Count).

* בעת לחיצה על הכפתור של סוג המשחק, יאותחל המשחק הרלוונטי.

משתמש ללא 4 אלבומים בעלי תמונה ומיקום/כמות תמונות חיובית, הפיצ'ר של המשחק הספיציפי לא יהיה זמין עבורו.

כלומר: במידה והמשתמש בחר לשחק משחק "מיקומים" ואין לו ארבעה אלבומים עם תמונה ומיקום (Location) תוצג לו הודעה מתאימה כי המשחק אינו זמין.



אך ייתכן כי אם ילחץ המשתמש על משחק "כמות תמונות" ויש לו 4 אלובמים עם תמונה וכמות תמונות חיובית (גדולה מ-0) יאותחל ויופיע המשחק.

* בעת לחיצה על אחת התמונות, יפתח טופס המשחק-
  + שם האלבום
  + שאלת המשחק
  + תמונת האלבום
  + 4 תשובות אפשריות נכונות
* בעת מענה על תשובה נכונה- מוצגת הודעה למשתמש, מספר הנקודות שצבר במשחק מתעדכן והתמונה בטופס הראשי של האפליקציה מוחלפת לתמונת אלבום אחרת.
* בעת מענה על תשובה לא נכונה- מוצגת הודעה למשתמש והתמונה על הטופס הראשי של האפליקציה אינה מתחלפת. כלומר המשתמש יכול לנסות לשחק שנית ולנחש שוב את התשובה.

**מציאת דייט:**

* הפיצ'ר מאפשר למשתמש למצוא דייט בעזרת הפייסבוק עם אדם שאינו זר לחלוטין.
* על המשתמש לבחור לפחות אחד מהנתונים הבאים: מין וטווח גילאים.

לאחר לחיצה על כפתור Find Match יוצגו למשתמש כל החברים שלו אשר:

* + המצב המשפחתי שלו אינו מוגדר או מוגדר כרווק
  + עומד בחתך הנתונים שהוגדר: מין ו/או טווח גילאים

במידה ונמצאה התאמה, המשתמש יוכל לבחור מתוך רשימת המתאימים/ות את החבר שהוא מעוניין לצפות בנתוניו.

### תבנית מס' 1 – Command

* **סיבת הבחירה / שימוש בתבנית:**

בחרנו להשתמש ב-Command Pattern כדי לממש אופרציות שונות של משחק האלבומים.

ההבדלים בין משחק 'מיקום אלבום' למשחק 'כמות תמונות', נבדלים זה מזה באופן אתחול המשחק ובתשובות האפשריות.

בעוד שבמשחק מיקומי התמונות – נציג למשתמש מיקומים שונים כתשובות אפשריות,

במשחק כמות התמונות - נציג למשתמש מספרים שונים כתשובות אפשריות.

בעתיד, במידה ונרצה לייצר עוד משחק שמתבסס על נתוני האלבומים, כל שנזדקק הוא לממש מחלקה נוספת שמממשת את Interface ICommandGame ובה נממש את אופן האתחול והנתונים הרלוונטים עבור המשחק שנרצה לבצע.

בנוסף, נצטרך לקשר בין ה'לחיצה' על הכפתור המתאים ב-UI ליצירת הCommand החדש הרלוונטי בלוגיקת המשחק (כלומר הוספת שורה נוספת במתודה SetCommand הממומשת בFacade).

לא נצטרך לעשות שינוי אפילו של שורה אחת בלוגיקת המשחק PictureGameFeature ובכך השימוש בפטרן הזה הופך את הפיצ'ר להיות Reuseabily. ניתן לממש משחקים רבים כאוות נפשנו.

* **אופן המימוש:**

ניתן למצוא את הקוד של הInterface במחלקה ICommandGame.

ניתן למצוא את הקוד של המימושים השונים של המשחק ב- CommandGameByLocationו- CommandGameByNumbers.

בממשק המשתמש כאשר המשתמש לוחץ על אחד מהכפתורים "Location Game" או "Numbers Game" תבוצענה הפעולות הבאות:

1. SetCommand(CurrentButton.Text): מתודה הנמצאת בFacade ואחראית לייצר את הCommand הרלוונטי בלוגיקת המשחק PictureGameFeature

במידה ולחצו "Location Game" תייצר המתודה new CommandGameByLocation()

במידה ולחצו "Numbers Game" תייצר המתודה new CommandGameByNumbers()

1. CreatePictureGameFeature(Albums): מתודה הנמצאת בFacade ואחראית להפעיל את המתודה CreateGame(Albums) מהלוגיקה של המשחק.

* PictureGameFeature.CreateGame(Albums) אחראית להפעיל את הCommand.ExectueGame שמייצר את המשחק עם הנתונים להתחלת המשחק בהתאם לCommand שנוצר.

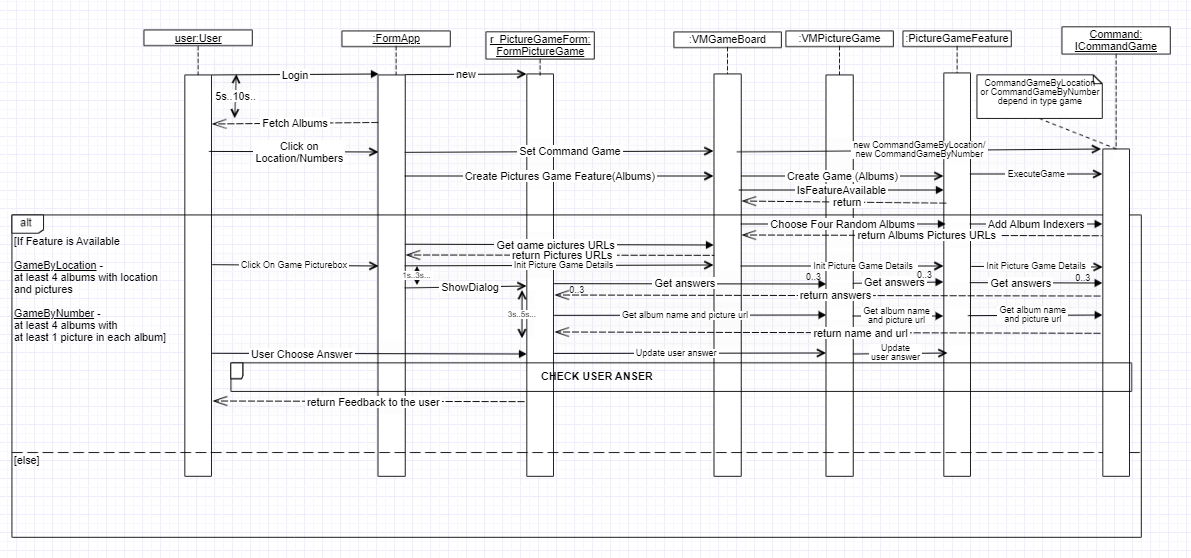
1. setPictureBoxsAndLablesForGame(): מתודה האחראית לאתחל את הנתונים על גבי הUI , בהתאם לנתונים שבלוגיקת המשחק. המתודה בודקת אם קיים הפיצ'ר (אם הוא זמין), ומאתחלת את הPictureboxs שע"ג הטופס הראשי FormApp. במידה ולא קיים הפיצ'ר יבוצע Abort ויוצג למשתמש הודעה במקום התמונות.

כעת, הלוגיקה של המשחק מכילה אובייקט מסוג ICommandGame שהוא למעשה האובייקט CommandGameByLocation/ CommandGameByNumbers. ובאופן פולימורפי הפעולות מהלוגיקה על הCommand הן זהות.

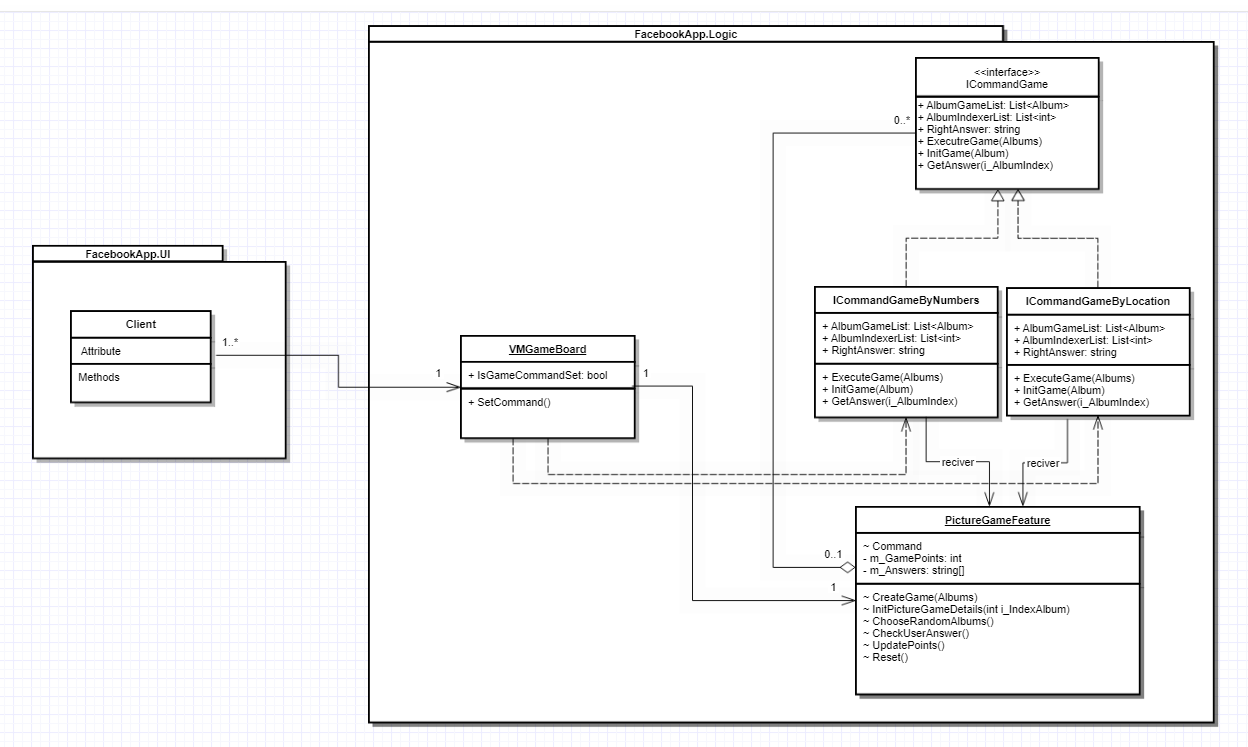
לצורך אתחול משחק ספיציפי, המתודה Command.InitGame(Album) תאתחל את המשחק ע"פ האלבום שנבחר.

לצורך קבלת התשובה הנכונה, המתודה Command.RightAnswer תחזיר את התשובה הנכונה בהתאם לאופי המשחק וכו'.

* **Sequence Diagram**



* **Class Diagram**



### תבנית מס' 2 – Template Method

* **סיבת הבחירה / שימוש בתבנית:**

בחרנו לממש את Template Method Pattern לצורך מתן אפשרות ב-UI למיין את הרשימות השונות (ListBox) לפי פרמטרים שונים- לדוגמא:Date ו-Name.

הסיבה למימוש, היא כדי שנוכל לבצע מיונים שונים ע"פ אינדיקציות שונות ללא צורך במימוש מלא וחוזר (שכפול קוד) של פונקצית המיון.

במידה ונרצה להוסיף מיון לפי פרמטר נוסף, נצטרך לייצר מחלקה היורשת מהמחלקה האבסטרקטית SortListBox שתממש רק את המתודה האבסטרקטית לסוג המיון, "SwapMethod".

* **אופן המימוש:**

המחלקה SortListBox היא מחלקה אבסטרקטית בעלת Property בודד ושתי מתודות.

1. ListBox List - הרשימה עליה נרצה לבצע את המיון.

2. SwapMethod() - מתודה אבסטרקטית שנדרשים לממש.

3. Sort() - מתודה למיון הרשימה.

מימשנו בנוסף את המחלקות SortListByDate ו SortListByName. אלו מחלקות שיורשות מ SortListBox

וממשות את המתודה האבסטרקטית בהתאם לאופן המיון.

ב-UI, כאשר נלחץ על הRadio הרלוונטי למיון ByDate/ByName , נייצר אנונימית את המחלקה המתאימה למיון המבוקש, תוך כדי מסירת הListBox שנרצה למיין, והפעלת המתודה Sort.

לדוגמא:

new SortListByDate(listBoxPosts).Sort();

חשוב לציין: ה-ListBoxים שנשלחים חייבים להכיל אובייקטים שממשים את הInterface ISortableItem.

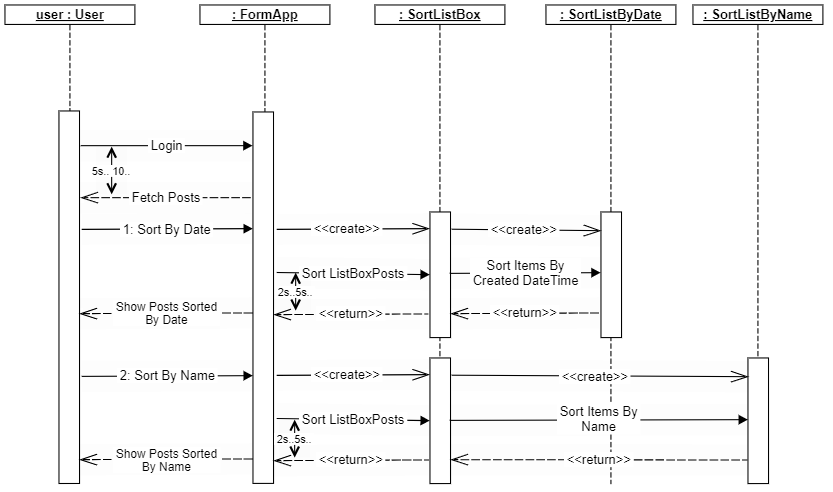
זהו Interface שמאפשר לנו באופן פולימורפי להפעיל את המיון על רשימות המכילות אובייקטים שונים.

בדוגמה שלנו, מכיוון שPost הוא Legacy Component, הגדרנו כי המחלקה AdapterPost תממש גם את ISortableItem.

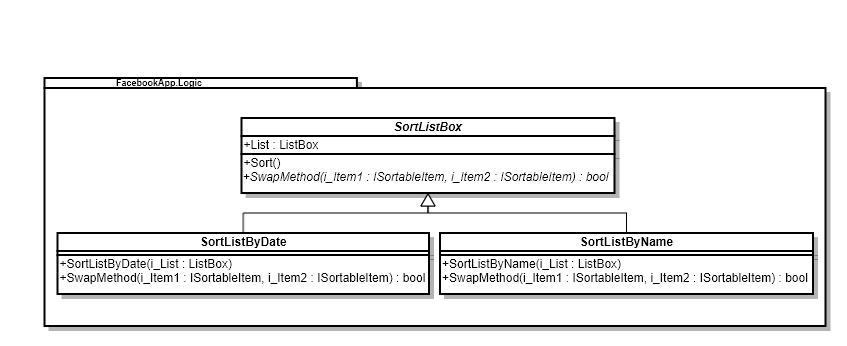
ListBoxPosts מכיל אובייקטים מסוג AdapterPost שעוטפים אובייקטים מסוג Posts ולכן ניתן כנדרש לבצע את פעולת מיון הרשימה.

המתודה SwapMethod מבצעת את בדיקת ההחלפה על אוביקטים המממשים את ISortableItem.

* **Sequence Diagram**



* **Class Diagram**



### תבנית מס' 3 – Strategy

### 

* **סיבת הבחירה / שימוש בתבנית:**

בחרנו לממש Strategy Pattern בפיצ'ר של מציאת דייט כדי לאפשר בחירה מודולרית של חתך הנתונים המבוקש על ידי המשתמש.

כעת, המשתמש יכול לבחור אחד או יותר מהנתונים הבאים: גיל ו/או מגדר מבוקש.

בנוסף בחרנו לממש Pattern זה, כדי שבאופן Reuseabily נוכל להוסיף פרמטרים נוספים לסינון, ללא שינוי הקוד שמבצע את חיפוש החבר המתאים IsFriendMatch .

בעתיד, במידה ונרצה להגדירFillter נוסף כל שנצטרך הוא לממש את מתודת 'אסטרטגיית' הסינון הנוספת ולהכניס אותה לרשימת האסטרטגיות.

* **אופן המימוש:**

את קוד המימוש ניתן למצוא במחלקת הלוגיקה DatingFeature.

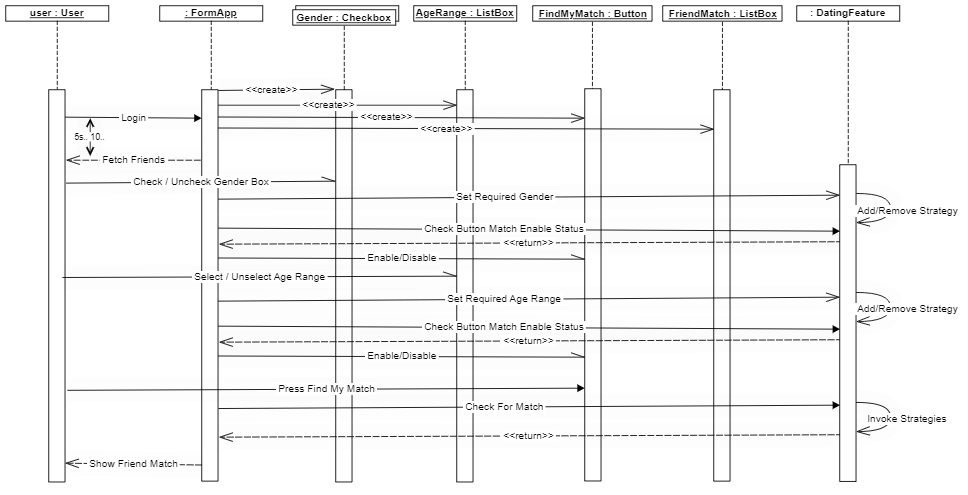
המחלקה מכילה כ-DataMember רשימה של מצביעים למתודות אשר מקבלות פרמטר מטיפוס User ומחזירות פרמטר מטיפוס Bool. מתודות אלו הם האסטרטגיות השונות לסינון רשימת החברים.

private static readonly List<Func<User, bool>> r\_MatchStrategies

כאשר המשתמש בוחר פרמטר לסינון (מגדר/גיל) מתווספת המתודה המתאימה לאסטרטגיה (matchByGender/matchByAge) בהתאמה.

במידה והמשתמש בוחר 'לבטל' את אחד הפרמטרים (מתחרט על בחירת הגיל לדוגמא), נמחקת המתודה המתאימה לאסטרטגיה מהרשימה.

בעת לחיצה על הכפתור Find Match ע"ג הטופס הראשי, מתבצעת ריצה על כל רשימת החברים תוך הפעלת המתודה IsFriendMatchToUserRequests. מתודה זו, הממומשת בלוגיקת הפיצ'ר מפעילה את כל האסטרטגיות מהרשימה שלה ומחזירה ערך אמת (החבר מתאים) אם ורק אם כל אסטרטגיות הסינון החזירו ערך אמת.

* **Sequence Diagram**
* **Class Diagram**

### 