### 

### 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 ו(מיקום/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.

* שירה לצרף פה צילום משחק של איך זה נראה כשאי אפשר–

אך ייתכן כי אם ילחץ המשתמש על משחק "כמות תמונות" ויש לו 4 אלובמים עם תמונה וכמות תמונות חיובית (>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 עולה והתמונה בטופס הראשי של האפליקציה מוחלפת לתמונת אלבום אחרת.
* בעת מענה על תשובה לא נכונה- מוצגת הודעה למשתמש. התמונה נשארת על הטופס הראשי של האפליקציה, 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 להתשמש ב-Command Pattern כדי לממש אופרציות שונות של משחק האלבומים.

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

בעוד שבמשחק מיקומי – נציג למשתמש מיקומים שונים כתשובות אפשריות,

במשחק כמות- נציג למשתמש מספרים שונים כתשובות אפשריות.

בעתיד, במידה ונרצה לייצר עוד משחק שמתבסס על האלבומים, כל שנזדקק, הוא לממש מחלקה נוספת שממשת 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 של אפילו שורה אחת בלוגיקת המשחק PictureGameFeature ובכך השימוש בפטרן הזה הופך את הפיצ'ר להיות Reusabliy. 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:

SetCommand(CurrentButton.Text) – מתודה הנמצאת בFacade ואחראית לייצר את הCommand הרלוונטי בלוגיקה.

במידה ולחצו "locations Game" תייצר new CommandGameByLocation()

במידה ולחצו "Numbers Game" תייצר new CommandGameByNumbers()

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.

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, Command.InitGame(Album) יאתחל את המשחק ע"פ האלבום שנבחר.

לצורך קבלת התשובה הנכונה, Command.RightAnswer יחזיר 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 וName.

הסיבה למימוש, היא כדי שנוכל לבצע שימוש חוזר במחלקה שמבצעת את המיון. במידה ונרצה להוסיף מיון לפי פרמטר נוסף, נצטרך לייצר מחלקה שממשת רק את מתודת ההחלפה "SwapMethod", תוך שהיא יורשת את הקוד המקורי SortListBox.

* **אופן המימוש:**

המחלקה SortListBox היא מחלקה אבסטרקטית בעלת Propirty 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 כדי שבאופן פולימורפי נוכל להפעיל את המיון הזה על אובייקטים שונים.

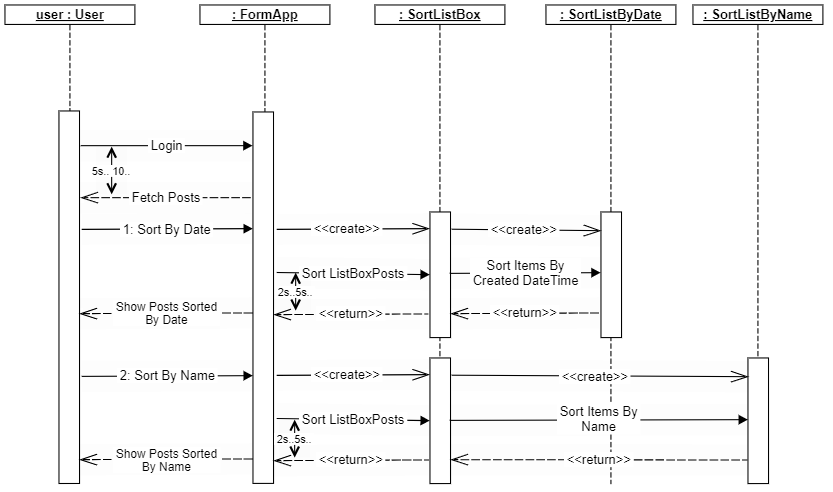
בדוגמה שלנו, מכיוון שPost הוא Legacy Component, הגדרנו כי המחלקה AdapterPost מממשת גם את ISortableItem.

ListBoxPosts מכיל אובייקטים מסוג AdapterPost שעוטפים אובייקטים מסוג Posts ולכן ניתן כנדרש לבצעת את פקודת המיון.

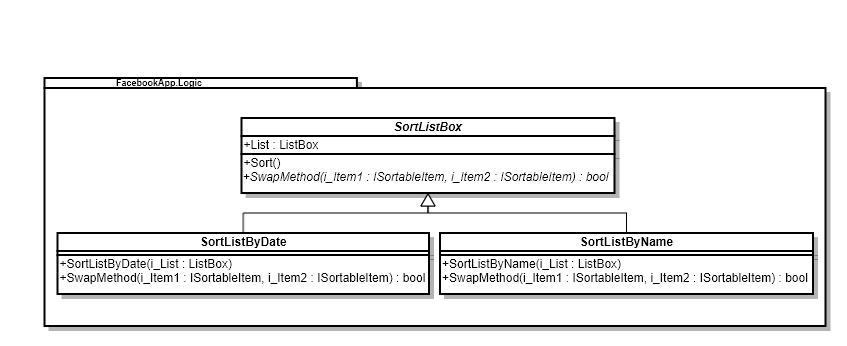
המתודות SwapMethod מבצעות את בדיקת ההחלפה על אוביקטים שמממשים את ISortableItem.

שירה – אולי ניתן ,לנעול, את המחלקה SortListBox לחייב אותה לקבל ליסטים מהסוג המבוקש

* **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 מין מבוקש.

בנוסף בחרנו לממש Pattern זה, כדי שבאופן Reusabily נוכל להוסיף חתכי מיון ללא שינוי הקוד שמבצע את חיפוש החבר המתאים IsFriendMatch .

בעתיד, במידה ונרצה לבצע Fillter נוסף כל שנצטרך הוא לממש את מתודת 'אסטרטגיית' החיפוש הנוספת ולהכניס אותה לרשימת האסטרטגיות.

* **אופן המימוש:**

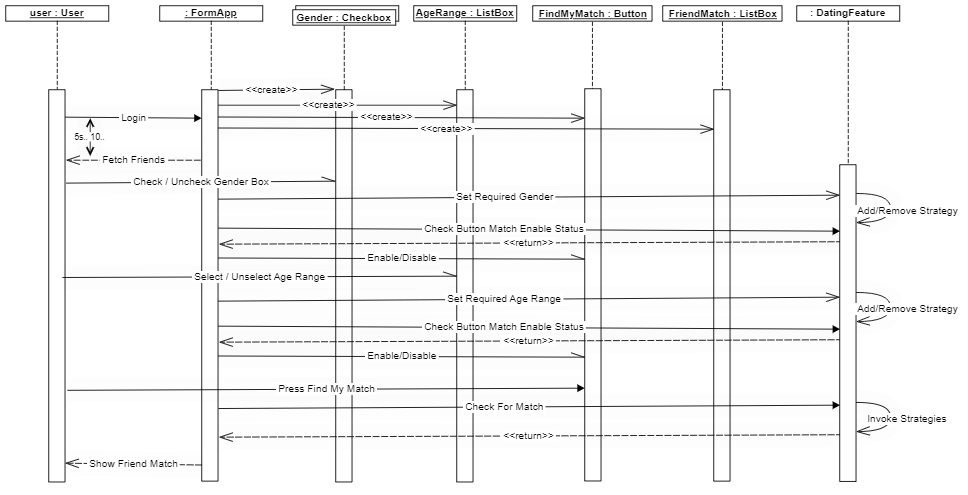
את קוד המימוש ניתן למצוא ב- DatingFeature.

המחלקה מכילה כ-DataMember רשימה של מצביעים למתודות אשר מקבלות User ומחזירות Bool מתודות אלו הם האסטרטגיות השונות שנגדיר.

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

כאשר המשתמש בוחר למשל את המין המבוקש (RequiredGender), אנו מעדכנים את הProperty המתאים בDatingFeature. ובנוסף, מתעדכנת רשימת האסטרטגיות עם matchByGender.

אם הגדרנו מין/גיל, מתווספות המתודות המתאימות matchByGender/matchByAge לרשימת האסטרטגיות בהתאמה, אם ביטלנו את ה-מין/גיל נמחקות המתודות מרשימת האסטרטגיות.

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

### 