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

* הצגת חידות שקשורות לנתונים של המשתמש בפייסבוק:

האפליקציה מציגה למשתמש חידות מתוך מאגר של חידות שהוספנו, חידה שקשורה לנתון כלשהו על המשתמש – בכמה קבוצות הוא חבר, כמה אלבומים יש לו, מה שם העמוד האחרון אותו הוא אהב ועוד.

על ידי לחיצת כפתור תוצג גם התשובה.

המשתמש יכול לבחור האם הוא רוצה שהחידות יוצגו לו בסדר מקל לקשה או מקשה לקל.

הפיצ'ר ממומש במחלקה RiddleFeature.

* יצירת כרטיס ביקור עבור המשתמש:

בלחיצת כפתור האפליקציה תיצור עבור המשתמש כרטיס ביקור הכולל נתונים שלו: שם, עיר, תמונה, לינק לעמוד הפייסבוק, אימייל, מגדר (הנתונים נלקחים מחשבון הפייסבוק שלו).

בנוסף יוכל המשתמש להוסיף מידע נוסף לפי רצונו שיופיע בכרטיס ביקור (טלפון, משפט ועוד).

בעזרת תבנית decorator הוספנו פונקציונליות לפאנל של רקע עם תמונה. בעזרת התבנית נוכל בעתיד להוסיף עוד סוגים של רקעים/ שידרוגים אחרים לפאנל ולשלב קומבינציות ביניהם.

הפיצ'ר ממומש במחלקה FormMain במתודה showCallingCard עם שימוש בData Binding, וכן במחלקות הקשורות לתבנית decorator – יש שידרוג לפאנל של כרטיס הביקור - CalingCardPanel, CardDecorator.

### תבנית מס' 1 – iterator

* סיבת הבחירה / שימוש בתבנית:

בחרנו להסתיר מהמשתמש את מבנה הנתונים שמחזיק את אוסף החידות, כדי שאם בעתיד נרצה לשנות את מבנה הנתונים, נוכל לעשות זאת מבלי לפגוע באופן המעבר על הקולקציה – מבלי לשנות את הקוד. כדי לאפשר מעבר על הקולקציה באופן שאינו חושף את מבנה הנתונים השתמשנו בתבנית iterator.

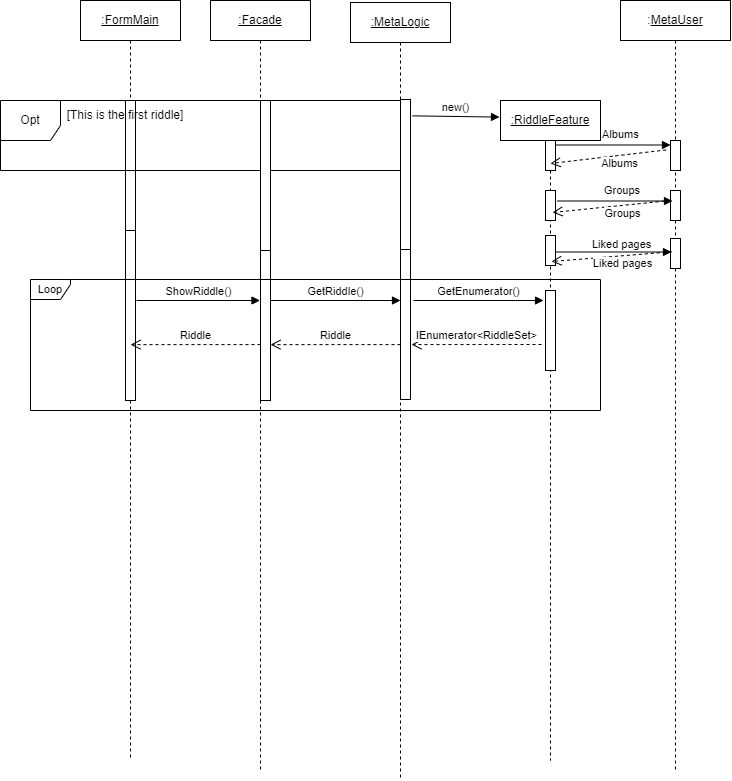
* אופן המימוש:

הקולקציה של החידות נמצאת בתוך המחלקה RiddleFeature. לכן מחלקה זו מממשת את IEnumerable<RiddleSet>, ויש לה שתי מטודות GetEnumerator בהתאם לדרישות הממשק. המטודה GetEnumerator שמחזירה IEnumerator<RiddleSet>, יוצרת את האיטרטור באמצעות שימוש ב yield return. והמטודה השנייה מפעילה את אותה אחת. כעת, במחלקה metalogic, יש רפרנס לאיטרטור של החידות, (שהוא קיבל באמצעות המטודה GetEnumerator של RiddleFeature) ובאמצעותו המחלקה של הלוגיקה יכולה לעבור על מבנה הנתונים, לבקש בכל פעם את החידה הבאה, מבלי שהיא בעצם חשופה למבנה הנתונים שמחזיק את החידות.

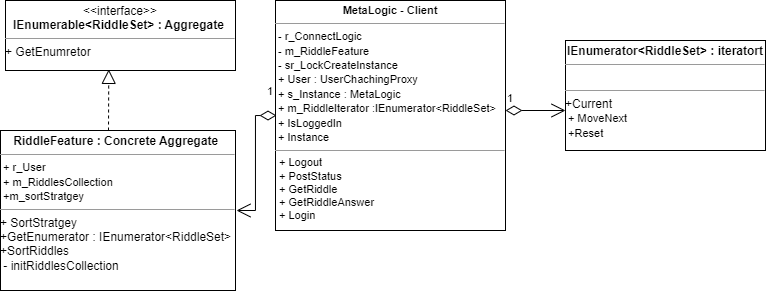
השחקנים בתבנית:

1. Client: המחלקה MetaLogic
2. Aggregate: המחלקה RiddleFeature
3. Iterator: המחלקה ממומשת אוטומטית באמצעות yield return.

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – decorator

* סיבת הבחירה / שימוש בתבנית:

רצינו להוסיף פונקציונאליות של רקע לפאנל כרטיס הביקור. בחרנו להשתמש בדקורטור ולא פשוט לשנות את הקוד המקורי מכיוון שכך האובייקט המקורי לא מודע לתוספת, ועדיין ניתן להשתמש באובייקט המקורי בתוכניות אחרות שלא מעוניינות בתוספת הרקע. בנוסף יש אפשרות להרחיב בהמשך לעיצובים שונים של צבעים נוספים, מסגרות ותוספות אחרות מבלי לגעת בקוד האובייקט המקורי. וזה גם מאפשר להוסיף בהמשך קומבינציות שונות של עיצוב – למשל פאנל עם צבע רקע סגול וגם מסגרת צהובה, ללא צורך בשכפול קוד וכתיבה של מספר רב של מחלקות חדשות.

* אופן המימוש:

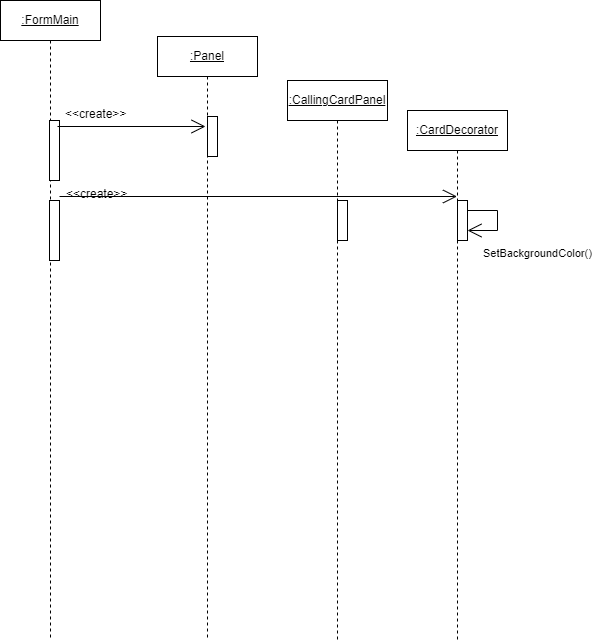
השתמשנו בinterface בשם ICardDecorator, אותו מממשים גם ה CalingCardPanel שהוא הcore decorator (שנותן רקע לבן לכרטיס) וגם ה CardDecorator שנותן רקע עם תמונה. ה CardDecorator מחזיק בתוכו ICardDecorator, מה שיאפשר לנו להגדיר מספר רב של קומבינציות ככל שנרצה להוסיף בהמשך.

הטופס FormMain מחזיק שדה מסוג ICardDecorator, אותו הוא יוצר כ- new CardDecorator . הdecorator שיצרנו מוסיף עיצוב לפאנל כרטיס הביקור בעזרת המתודה SetBackgroundColor ().

השחקנים בתבנית:

1. Client: המחלקה FormMain
2. Decorator: המחלקה CardDecorator
3. Component: המחלקה CalingCardPanel

* Sequence Diagram



* Class Diagram

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

### תבנית מס' 3 – strategy

* סיבת הבחירה / שימוש בתבנית:

למשתמש יש אפשרות לבחור האם להציג את החידות מהקל לקשה או מהקשה לקל. בהתאם לבחירתו נרצה למיין את קולקציית החידות על מנת להציג לו את זה בסדר שבו הוא בחר (מיון עולה או מיון יורד). הרכיב שאחראי על מיון יורד וזה שאחראי על מיון עולה הם זהים כמעט לחלוטין מלבד האסטרטגיה.

לכן נרצה להשתמש רק ברכיב אחד ולמעשה "להזריק" רק את המימוש (האסטרטגיה) לתוך הרכיב. לכן השתמשנו בתבנית זו על מנת לממש רכיב אחד למיון, ולשתול בו את האסטרטגיה שבחר המשתמש, באופן שלא נצטרך לשכפל קוד.

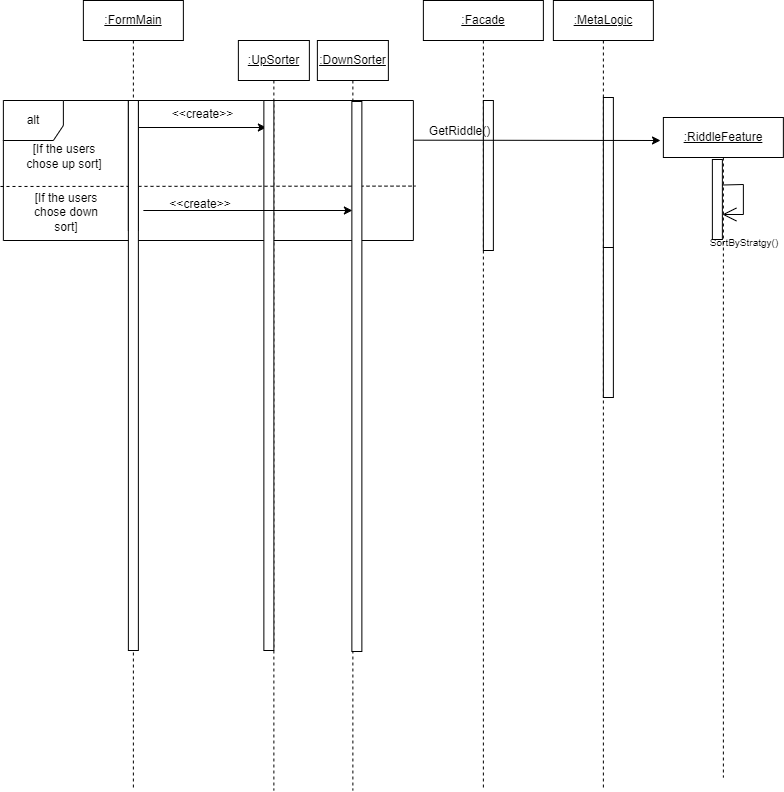
* אופן המימוש:

הגדרנו את הממשק ISortStrategy, אותו מממשות שתי האסטרטגיות – המחלקות UpSorter, DownSorter. המחלקה RiddleFeature שהיא למעשה רוצה להפעיל מטודה של מיון, יש שדה מטיפוס ISortStrategy, שהוא נוצר על ידי הFormMain בעת הבחירה של המשתמש באסטרטגיית המיון (נוצר עם אסטרטגיה ספציפית – מיון עולה או יורד). וכך למעשה במטודה של המיון, המחלקה בעצם משתילה את האסטרטגיה בעזרת השדה של האסטרטגיה.

השחקנים בתבנית:

1. Client: המחלקה FormMain
2. Context: המחלקה RiddleFeature
3. Interface: המחלקה ISortStrategy
4. Strategy1: המחלקה UpSorter
5. Strategy2: המחלקה DownSorter

* Sequence Diagram



* Class Diagram

