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

1. התאמה בסגנון אפליקציית ההיכרויות "טינדר". האפליקציה משתמשת ברשימת החברים של היוזר שמשתמש באפליקציה ובהתאם להעדפות של היוזר, האפליקציה מוצאת התאמות פוטנציאליות ומראה אותן למשתמש. כאשר המשתמש בוחר באחת מההתאמות, תמונת הפרופיל של ההתאמה מופיעה.

2. מציאת הפוסטים של היוזר בהתאם לתבנית שהוא מכניס. היוזר יכול לבחור האם להציג רק את הפוסט האחרון מבחינה קלנדרית שתואם לתבנית או את כלל הפוסטים שתואמים לתבנית.

3. מחולל אוטומטי של כרטיסי ברכה לאחד מחבריך בפייסבוק לפי נושא (יום הולדת, מעבר דירה וסיום לימודים) והעלתם לפייסבוק.

### תבנית מס' 1 – Facade

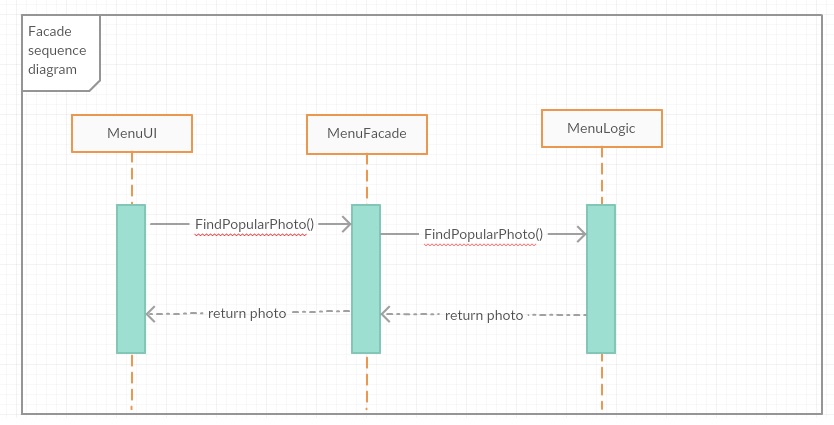
סיבת הבחירה / שימוש בתבנית:

בחרנו ב-facade כדי לממש חלוקה מוחלטת בין ה-UI ללוגיקה. יצרנו עבור כל חלון, מחלקה של facade שרק איתה ה-UI מתקשר. ה-façade יוצר קשר עם הלוגיקה ומעביר אותה ל-UI. כך אם בעתיד נרצה לשנות לוגיקה או להוסיף לוגיקה, ה-UI לא צריך להשתנות משום שהוא משתמש ב-facade.

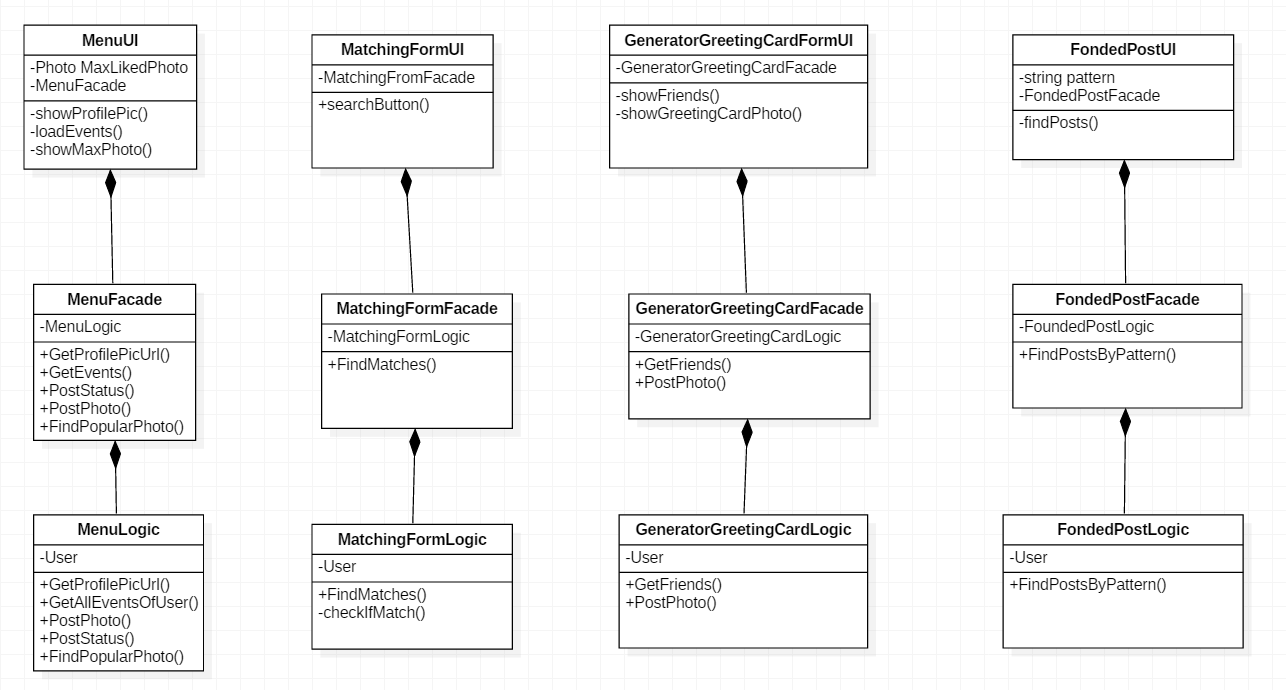
אופן המימוש:

המימוש נעשה באופן שכל חלון קיבל facade שמפריד בין ה-UI ללוגיקה.

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – Singelton

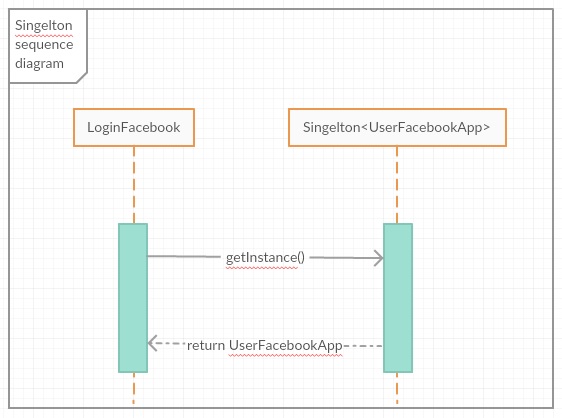
סיבת הבחירה / שימוש בתבנית:

מתחילת ריצת האפליקציה עד סיומה, קיים רק משתמש יחיד שהתחבר אליה. לכן, במקום להעביר את המשתמש הזה לכל מחלקה, יצרנו סינגלטון למשתמש זה. כל מחלקה שמעוניינת ביוזר שהתחבר לאפליקציה עוברת דרך הסינגלטון. בנוסף, אם בעתיד נוסיף מחלקה היא לא תצטרך העברה של היוזר דרך המחלקות האחרות אלא בעזרת הסינגלטון בלבד.

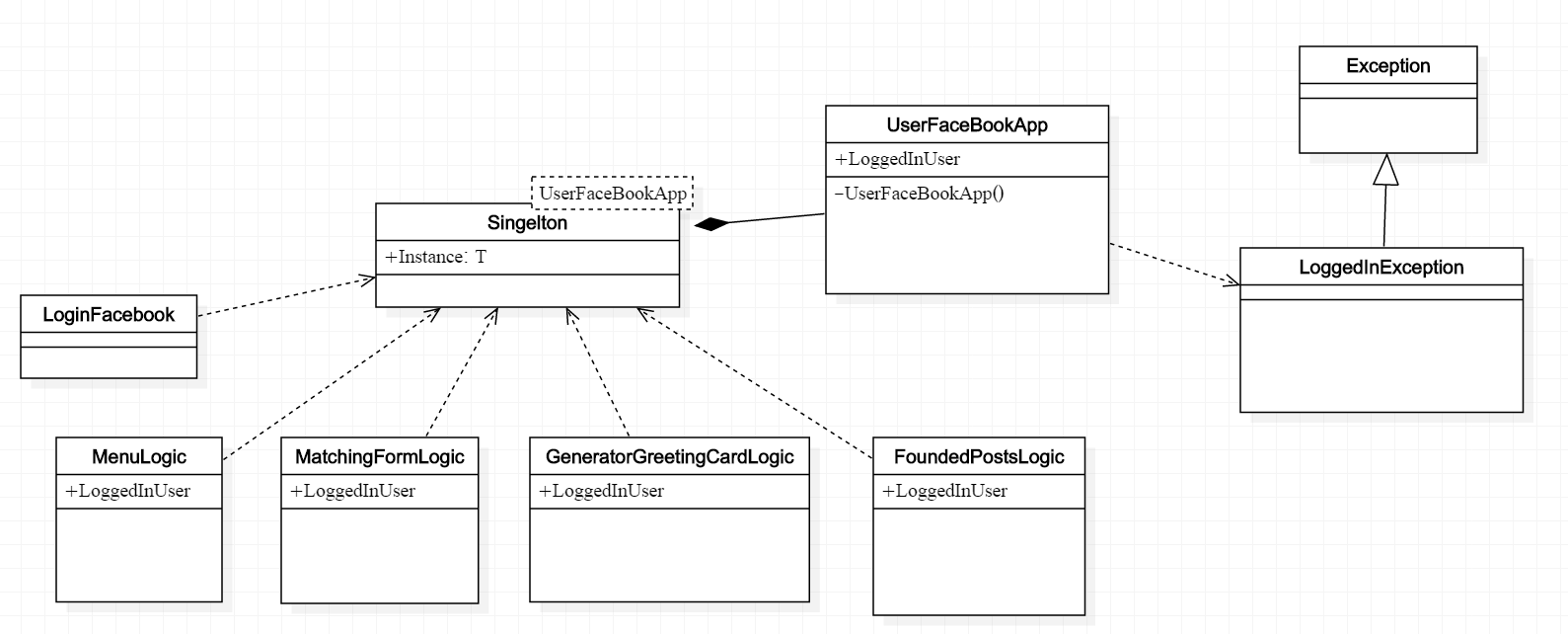
אופן המימוש:

יצרנו מחלקה (UserFacebookApp) שעוטפת את תהליך היצירה הראשוני של היוזר של פייסבוק. מחלקה זאת מכילה בנאי פרטי ואובייקט User כשדה של המחלקה. כך המחלקה הגנרית של סינגלטון יכולה ליצור מופע יחיד של מחלקה זאת.

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – Factory Method (Static factory class)

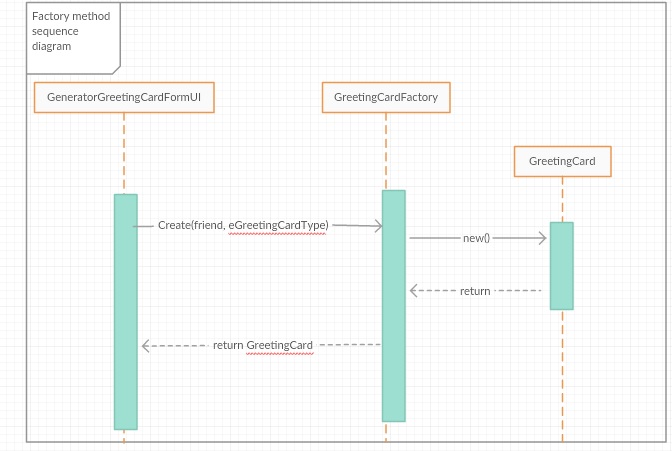
סיבת הבחירה / שימוש בתבנית:

בחרנו בתבנית זו משום שיש לנו משפחה פולימורפית של כרטיסי ברכה. הלוגיקה ליצירת כרטיס ברכה ספציפי היא זהה. לכן לא רצינו לחזור על הקוד של יצירת הכרטיס בכל פעם שניצור כרטיס חדש. כך אם בעתיד נצטרך ליצור כרטיס ברכה במקום אחר, נוכל להשתמש במחלקה הסטטית של יצירת כרטיסי הברכה ולא לחזור על הקוד היצירה.

אופן המימוש:

יצרנו מחלקה סטטית (GreetingCardFactory) שתפקידה ליצור מופע ספציפי מהמשפחה הפולימורפית של כרטיסי הברכה. כך החלון GeneratorGreetingCardFormUI משתמש במחלקה הסטטית הזאת כדי ליצור כרטיס ברכה.

* Sequence Diagram



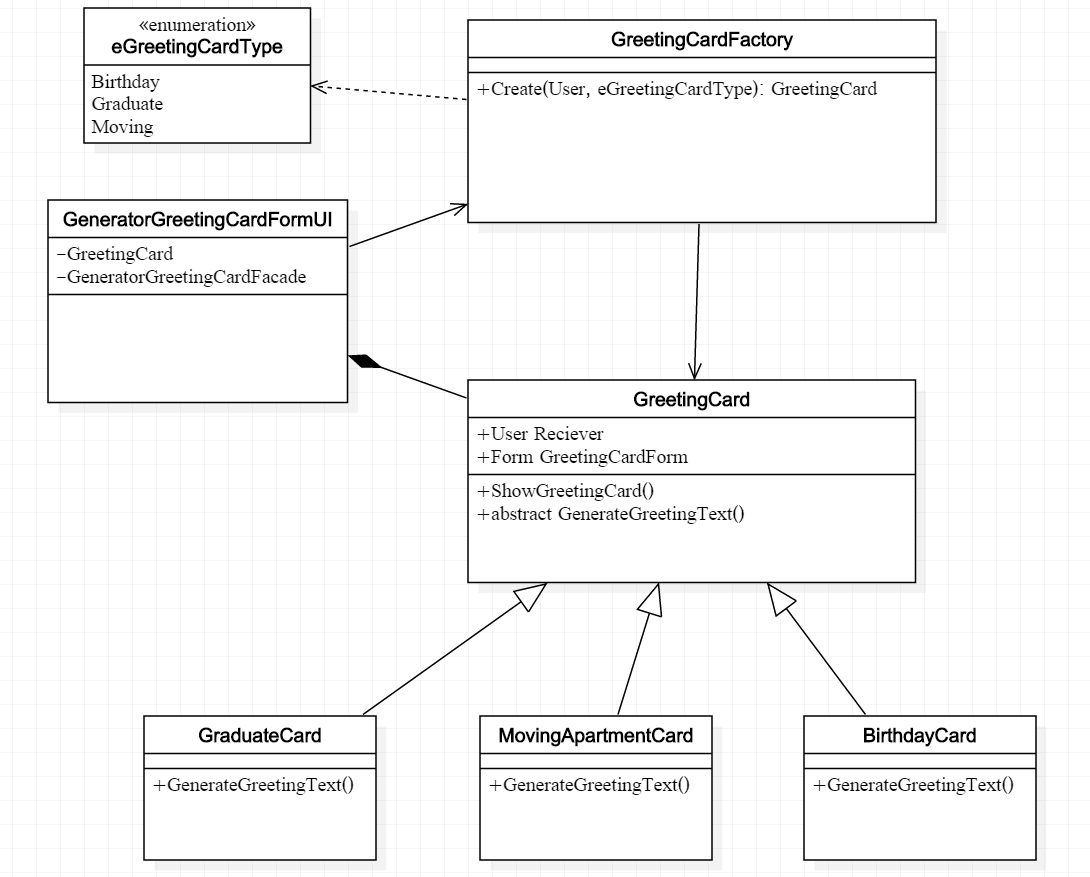
* Class Diagram

בדיאגרמה זאת, מחלקת ה-GreetingCardFactory היא בתפקיד StaticFactoryClass.

מחלקת ה-GreetingCard היא בתפקיד AbstractProduct.

מחלקת ה-GeneratorGreetingCardFormUI היא בתפקיד ה-Client.

המחלקות GraduateCard, MovingApartmentCard, BirthdayCard הן בתפקיד ConcreteProduct.



עבודה אסינכרונית:

במחלקה MenuUI השתמשנו בעבודה אסינכרונית כאשר הקצנו Thread נפרד למציאת התמונה הפופולרית ביותר. בנוסף הקצינו Thread עבור טעינת כל ה-Events של היוזר.

במחלקה FindPostByPatternUI הקצנו Thread נוסף עבור מציאת הפוסטים.

במחלקה GeneratorGreetingCardFormUI הקצנו Thread נוסף עבור טעינת כל החברים של היוזר.

עשינו שימוש בזה משום שרצינו לתת תחושה למי שמשתמש באפליקציה שהאפליקציה לא נתקעה לו וכן שהוא יכול לבצע פעולות נוסף בה בזמן שהוא מחכה לתוצאות.

עבודה עם Data Binding:

במחלקה MenuUI מימשנו data binding בין רשימת האירועים של היוזר לתמונה של האירועים. כאשר היוזר יבחר באירוע מבין הרשימה, תוצג לו התמונה של האירוע.