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

1. **Contest:**

פיצ'ר זה מאפשר למשתמש לנהל הגרלות בקלות, כאשר עליו להכניס רק את תיאור התחרות ותמונה מתאימה (אופציונלי), מהו האינדיקציה שלפיה משתמשים נוספים ייכנסו לרשימת ההגרלה (לייק לפוסט / תגובה לפוסט / גם וגם) וכמות המנצחים בסיום ההגרלה. הפיצ'ר כותב אוטומטית פוסט ומדפיס אותו לקיר של המשתמש. לאחר מכן, המשתמש יכול לעקוב אחרי רשימת הנרשמים להגרלה על ידי לחיצה על כפתור בודד, שמעדכן אוטומטית את הרשימה לפי אינדיקציות ההשתתפות. כשהמשתמש מחליט שנגמרה ההגרלה, ורוצה לפרסם מנצח/ים – עליו ללחוץ על כפתור בחירת מנצח, והפיצ'ר אוטומטית מגריל כמות מנצחים ככמות שהמשתמש בחר מראש, ומציג למשתמש את המנצחים בחלון חדש. בנוסף, מאפשר מעקב נוח אחרי כל ההגרלות שמנוהלות על ידי אותו משתמש. כמו כן, ישנה אפשרות למחיקת הגרלה.

ניתן למצוא את הפיצ'ר הנ"ל בחלון הראשי של האפליקציה תחת כפתור “Contests menu”.

1. **Memories:**

פיצ'ר זה מאפשר למשתמש להציג תחת קורת גג אחת מגוון פעילויות שביצע באתר פייסבוק, עם אפשרות סינון לפי טווח תאריכים. על המשתמש לבחור אילו נושאים ירצה שיוצגו בפניו, ועליו לבחור תאריך התחלה וסיום (כאשר יש אופציית בחירה של יום בודד). כאשר ילחץ על כפתור ה "Fetch", כל המידע המבוקש יוצג בפניו.

ניתן למצוא את הפיצ'ר הנ"ל בחלון הראשי של האפליקציה תחת כפתור “Memories”.

### תבנית מס' 1 – [Strategy]

* סיבת הבחירה / שימוש בתבנית:

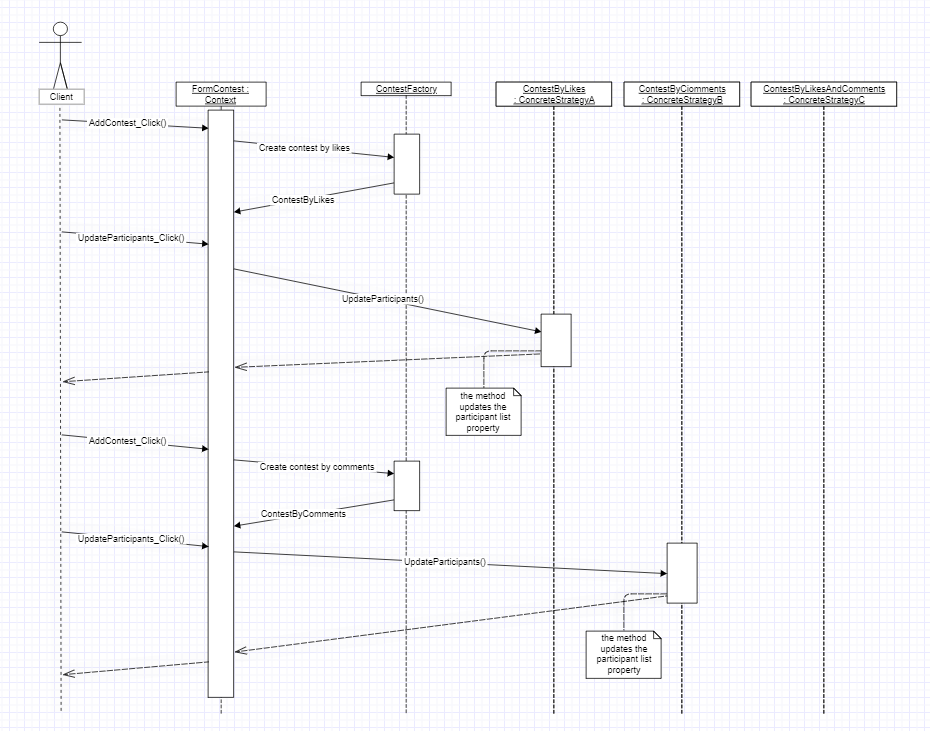
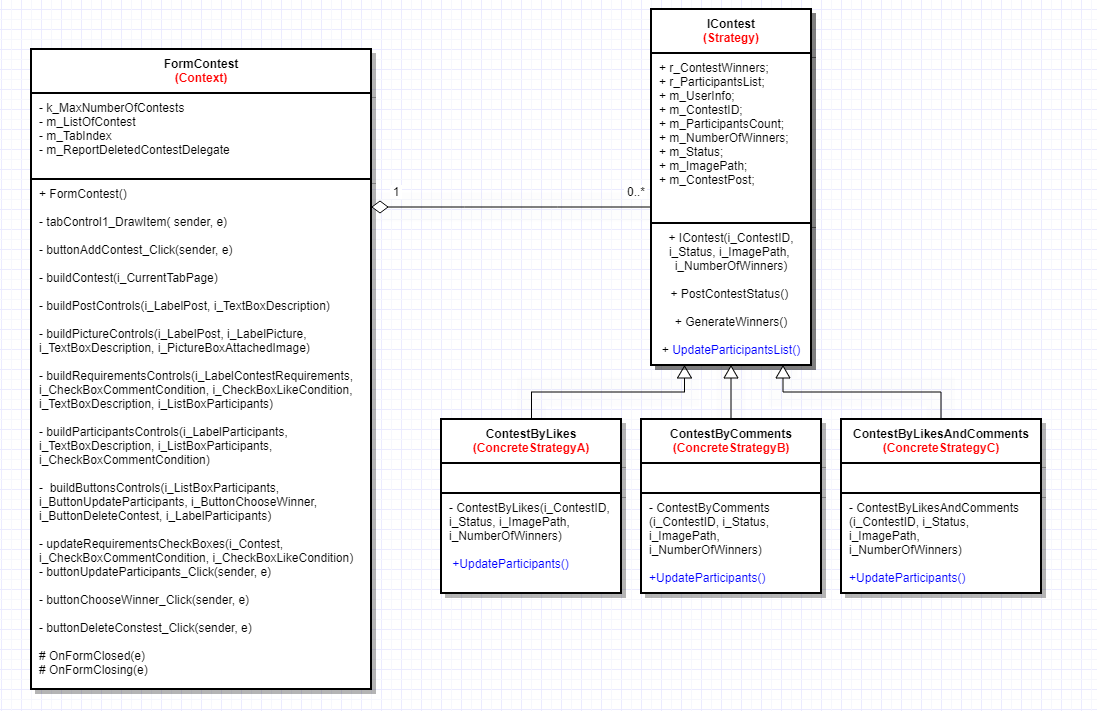
באפליקציה שלנו קיימים 3 סוגי תחורויות שונים, כאשר אלגוריתם בחירת המנצח בשלושת התחרויות מאוד דומה במימושו, אך שונה באופן בחירת המשתתפים.

לכן בחרנו להשתמש בתבנית עיצוב זו, שכן strategy מאפשר יצירה של התחרות (האסטרטגיה) המבוקשת בזמן ריצה. במימוש החדש הוצאנו את האלגוריתם מהמחלקה האבסטרקטית והעברנו אותו לתחרויות השונות, ובכך אם נרצה להוסיף סוג תחרות חדש – עלינו רק לממש את התחרות החדשה ולא צריך לשנות כלום בקוד קיים.

* אופן המימוש:

מימשנו את תבנית העיצוב הנ"ל על ידי יצירת 3 מחלקות שונות (ContestByComments, ContestByLikes, ContestByLikesAndComments) כאשר כל מחלקה מייצגת סוג תחרות אחר והן יורשות מהמחלקה האבסטרקטית IContest. במחלקה האבסטרקטית ממומש רוב הלוגיקה הנדרשת לתחרויות, וכל אחת מהמחלקות היורשות מממשת את האסטרטגיה המתאימה עבורה במתודת UpdateParticipantsList.

ניתן למצוא את המימוש של תבנית העיצוב הזאת במחלקות הבאות:

* IContest.
* ContestByComments.
* ContestByLikes.
* ContestByLikesAndComments.
* Sequence Diagram
* Class Diagram

### תבנית מס' 2 – [Observer]

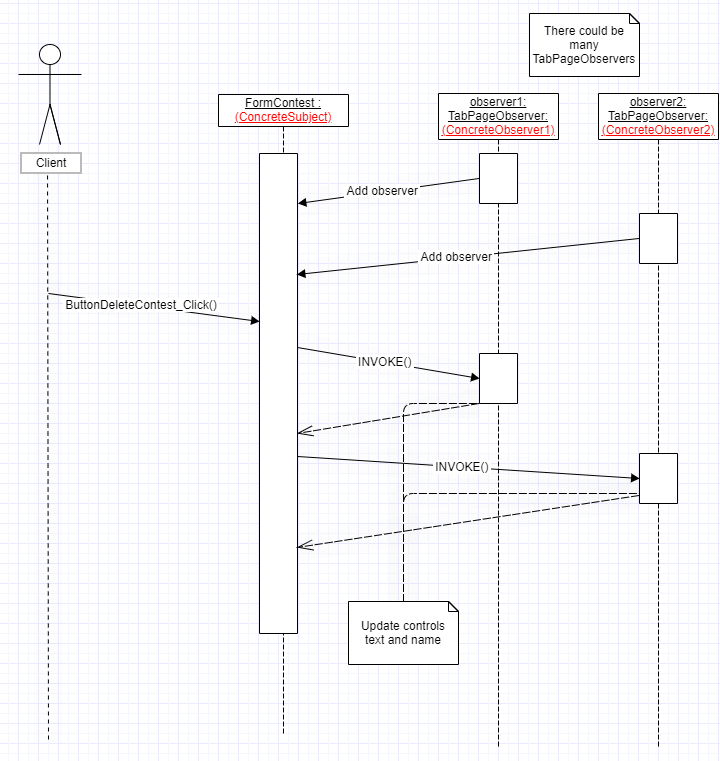
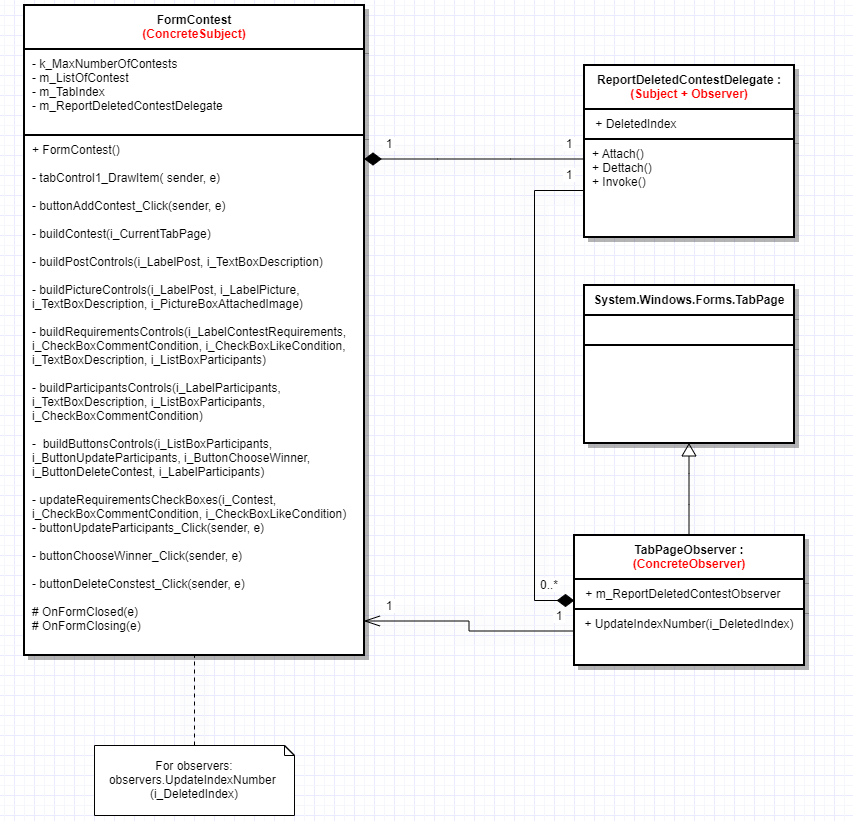
* סיבת הבחירה / שימוש בתבנית:

באפליקציה שלנו קיים פיצ'ר אשר אחראי על ניהול תחרויות עבור המשתמש.

למשתמש יכולים להיות מספר רב של תחרויות בו זמנית, אשר מוצגים בעזרת TabPages שמוחזקים ב TabControl. בנוסף, התחרויות ממוספרות לפי מספר התחרות (בשם של ה TabPage, בטקסט שלו, ובשמם של הכפתורים השייכים לו). כאשר נמחקת תחרות מהרשימה, אירוע זה מעדכן את כל המאזינים לו (במקרה הנ"ל, מעדכן את ה TabPages לשנות את שמם בהתאם למיספור העדכני).

* אופן המימוש:

מימשנו את תבנית העיצוב הנ"ל על ידי יצירת Delegate אשר מקבל את מספר התחרות שנמחקה. ל Delegate זה מקשיבים כל התחרויות שנוצרו על ידי המשתמש. כאשר נמחקת תחרות, ה Delegate מבצע פעולת INVOKE כאשר מועבר האינדקס של התחרות שנמחקה, וכל המאזינים מעדכנים את פרטיהם בהתאם.

* Sequence Diagram
* Class Diagram

המחלקה הבאה ריקה מכיוון שהיא מחלקה של C#, ולא מחלקה שאנחנו יצרנו

### תבנית מס' 3 – [Iterator]

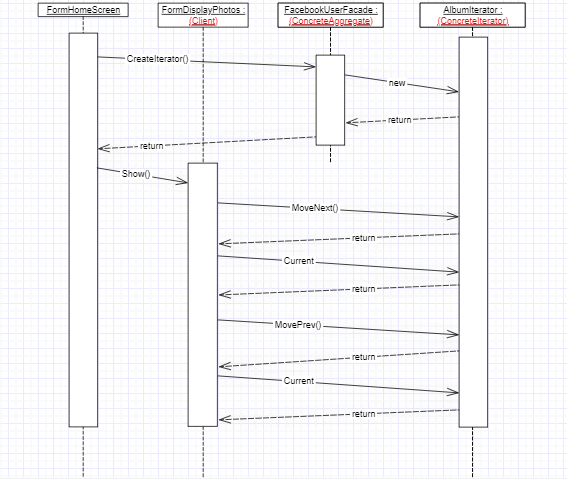
* סיבת הבחירה / שימוש בתבנית:

השתמשנו בתבנית עיצוב זו מכיוון שרצינו לאפשר למשתמש לעבור על רשימת תמונות מאלבום קיים בפייסבוק הפרטי שלו, מבלי להכיר את מבנה הנתונים של האוסף. דבר זה מאפשר לנו לשנות בעתיד את מבנה הנתונים של האלבום, מבלי שינוי כלשהו בקוד המשתמש. במידה ונרצה בעתיד לסנן תמונות לפי פילטר מסויים (לדוגמא – תיוג של אנשים מסויימים), נוכל לעשות זאת עם שינוי קוד מינימלי, רק במחלקות הקשורות ל iterator.

* אופן המימוש:

מימשנו את תבנית העיצוב הנ"ל על ידי יצירת 2 ממשקים (IAlbumIterator, IAlbumAggregate). יצרנו ConcreteIterator בשם AlbumIterator אשר מממש את הממשק IAlbumIterator. בנוסף, מחלקת הלוגיקה שלנו (FacebookUserFacade) מממשת את הממשק IAlbumAggregate ע"י הוספת המתודה CreateIterator.

כעת, במקום שמחלקת הלוגיקה שלנו תחזיר את האלבום עצמו (מחלקה מסוג אלבום שיש בה פרטים אודות האלבום, שלפעמים צריכים להיות נסתרים מהמשתמש), המחלקה מחזירה איטרטור מסוג IAlbumIterator שבו חשוף למשתמש רק התמונה הבאה / הקודמת באלבום.

* Sequence Diagram
* Class Diagram

