### תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל מספר 1:

* פיצ'ר ראשון: הפיצ'ר מציג למשתמש המחובר בלחיצת כפתור את תחזית המזל היומית שלו, יחד עם המצב רוח שלו, מספר המזל שלו.

הפיצ'ר מקבל את המידע מ API חיצוני באמצעות HttpRequest מהלינק:

<https://aztro.sameerkumar.website>

בתרגיל הראשון מימשנו אותו במחלקה של Horoscope וכעת העברנו אותו

לCache Proxy אשר מחזיק בתוכו RemoteAPI שבו מבצעים את הRequest.

* פיצ'ר שני: הפיצ'ר מציג למשתמש המחובר בלחיצת כפתור כמה ימים נותרו עד יום ההולדת הבא שלו ואיזה גיל הוא יהיה.

הפיצ'ר ממומש בAppEngine.

### תיאור קצר של מה שממימשנו בתרגיל מספר 2:

* תבניות Singleton, Façade, Cache proxy
* Data Binding של Name, Description עבור כל אלבום בעת כניסה אליו.
* עבודה אסינכרונית עם async await במתודה בשם SetHoroscope()

### תבנית מס' 1 – [Strategy]

* סיבת הבחירה / שימוש בתבנית:

השתמשנו בתבנית זו על מנת לאפשר למשתמש לפלטר את האלבומים ע"פ קריטריון מסוים לבחירתו. בצורה זו באופן גמיש ניתן לשנות את הלוגיקה בתוך הרכיב האחראי על הצגת האלבומים ובכך לאפשר בעתיד עוד אופציות שונות של פלטור.

השתמשנו בStrategy הנ"ל בתוך מחלקה בשם FilterIterator בתוך מתודה Filter שעוברת על כל רשימת האלבומים ולפי הקריטריון הרלוונטי מפלטרת אלו אלבומים להציג.

* אופן המימוש:

תחת Engine,

יצרנו Interface בשם IStrategyIterator שבתוכו יש מתודה המחזירה האם אותו אלבום שייך לקריטריון הפלטור הרלוונטי.

מימשנו במחלקה StrategyImplementation מימושים שונים של מחלקות המממשות את הממשק הנ"ל, כל אחת באופן אחר המספקת פלטור שונה.

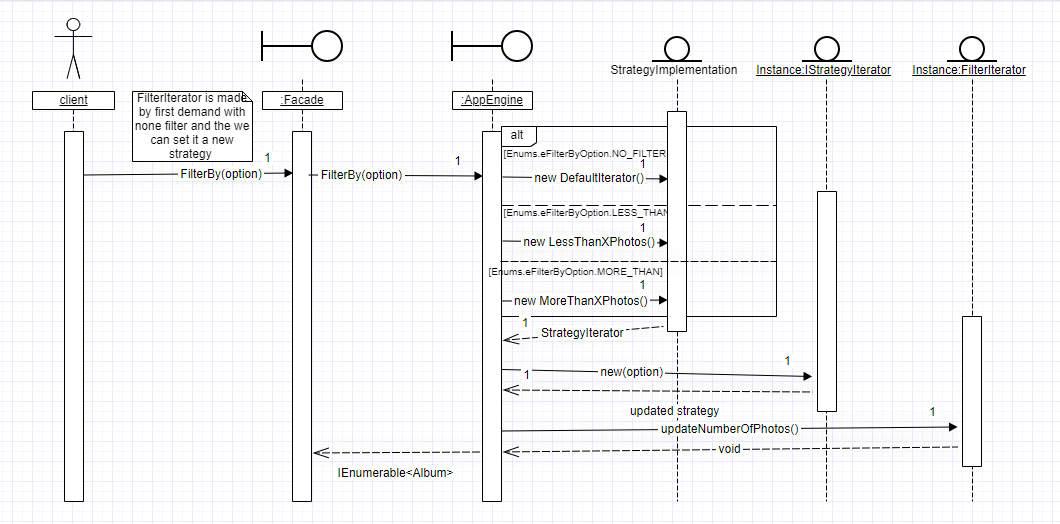
המימושנים שעשינו בתוך StrategyImplementation:

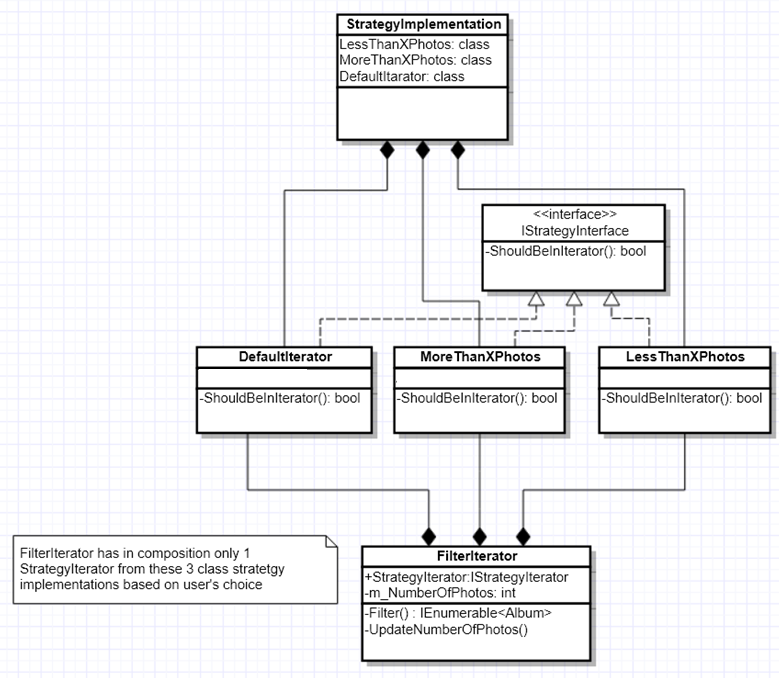
מחלקה בשם LessThanXPhotos שהיא מחזירה true אם באלבום יש לכל היותר X תמונות כאשר X הוא פרמטר (מספר) שהמשתמש בוחר להציג.

מחלקה בשם MoreThanXPhotos שהיא מחזירה true אם באלבום יש לפחות יותר מ X תמונות כאשר X הוא פרמטר (מספר) שהמשתמש בוחר להציג.

מחלקה בשם DefaultIterator שהיא מחזירה true עבור כל אלבום כי ב Default כל אלבום מוצג.

* Sequence Diagram



* Class Diagram

### תבנית מס' 2 – [Iterator]

* סיבת הבחירה / שימוש בתבנית:

השתמשנו בתבנית זו על מנת שיהיה ניתן להציג למשתמש ע"פ קריטיון הפלטור שבחר את האלבומים הרלוונטים לבחירתו בריצה על האלבומים באמצעות Iterator.

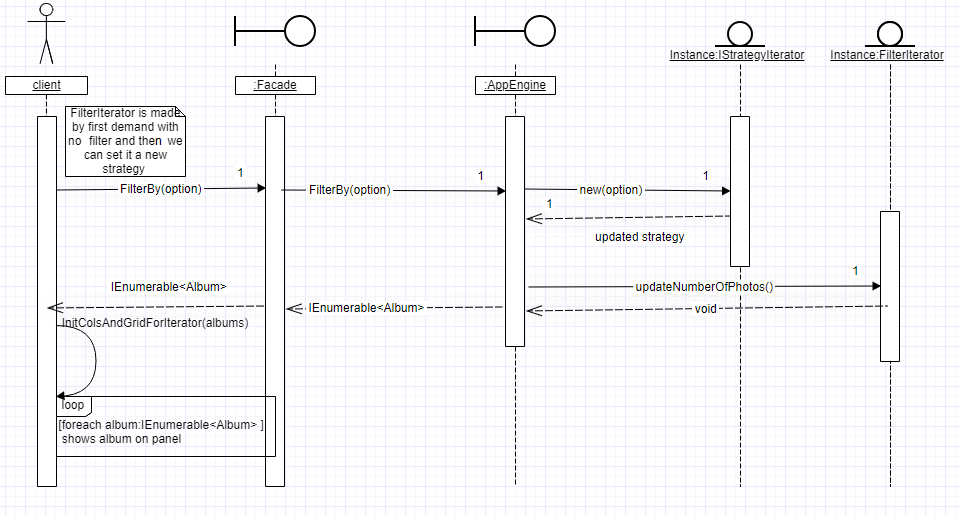
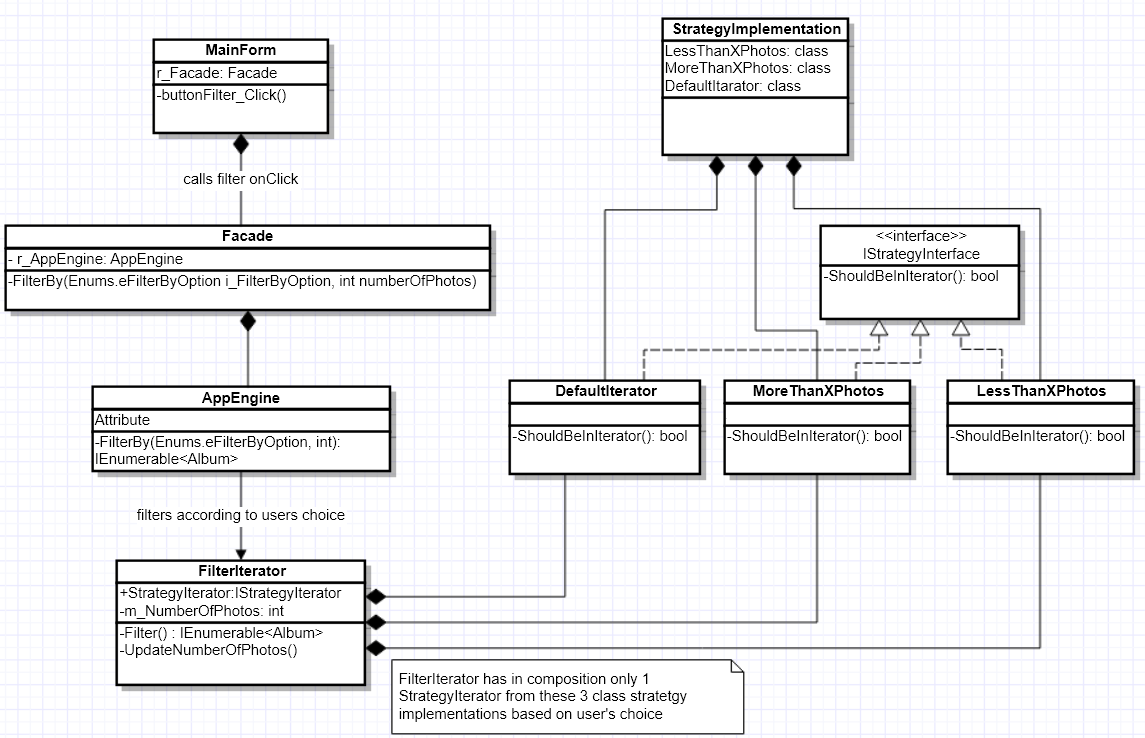
בנוסף המשתמש לא חשוף לסוג מבני נתונים אנחנו משתמשים ובכך ה Iterator נותן לנו את האפשרות לשנות בעתיד את מבני הנתונים ע"פ הצורך שלנו ולא נצטרך לשנות את הקוד בClient.

בלחיצה על כפתור Filter המוצג למשתמש בעת שהוא נמצא בחלון האלבומים, נקראת המתודה FilterBy המקבלת את מספר התמונות ואת קריטיון הפלטור, והערך המוחזר הוא ה Iterator שבריצה עליו מוצג למשתמש האלבומים הרלוונטים.

* אופן המימוש:

תחת Engine,

יצרנו מחלקה בשם FilterIterator שמחזיקה בקומפוזיציה אובייקט בשם StrategyIterator מסוג הממשק IStrategyIterator, למחלקה זו יש מתודה בשם Filter המחזירה IEnumerable<Album>, המתודה זו ממומשת באמצעות yield עבור כל אלבום שעבר את תנאי הקריטריון של ה StrategyIterator , באופן זה נוצר לנו iterator המכיל רק את האלבומים הרלוונטים לקריטריון הפלטור.

* Sequence Diagram
* Class Diagram

### תבנית מס' 3 – [Observer]

* סיבת הבחירה / שימוש בתבנית:

השתמשנו בתבנית זו על מנת להאזין לכל לחיצה על הכפתור ChangeColor וכל מי שמאזין אליו מפעיל מתודה כלשהי שעושה משהו עם הצבע השמור בתוך ColorChangerNotifier.

השתמשנו בתבנית זו בכך שפקדים מסוימים נרשמו באופן שונה לקליק הזה, וברגע שיש קליק עליו, עבור חלק מהפקדים השתנה צבע הרקע שלהם, עבור חלק אחר מהפקדים שנרשמו השתנה צבע הרקע לצבע אחר, עבור חלק אחר מהפקדים השתנה צבע הForeColor שלהם.

אם בעתיד נרצה בעת לחיצה על הכפתור ChangeColor להוסיף פונקציונליות המאפשרת עוד מניפולציות עם הצבע על הפקדים השונים אז נוכל להוסיף זאת בקלות, כלומר מאפשרים פה אקסטנסיביליות.

* אופן המימוש:

תחת UI,

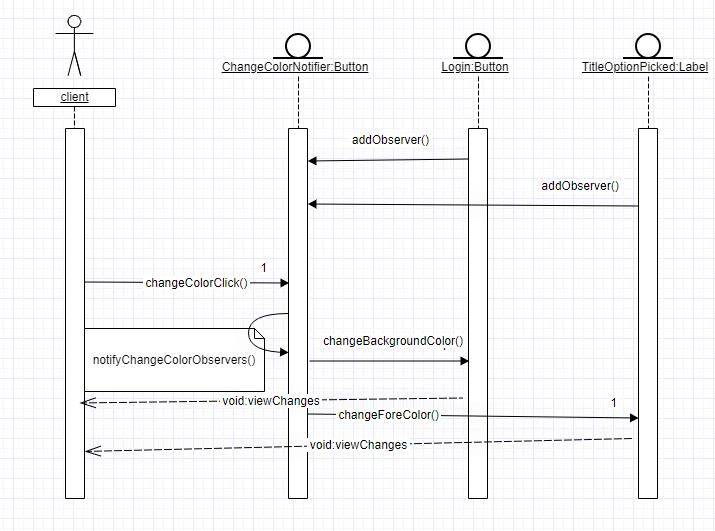
יצרנו מחלקה בשם ColorChangerNotifier היורשת מ Button כי היא בעצמה כפתור, הוספנו את פקד זה ל FormMainשלנו בצד ימין למעלה.

במחלקה זו שמור לנו שדה m\_CurrentColorשהוא int וערכו 1 עד 4 לכל היותר.

יש במחלקה זו שדה public event Action<int> m\_ReportColorChangeDelegates שאליו נרשמים כל הפקדים שרוצים להפעיל מתודה כלשהי.

דרסנו את OnClick באמצעות protected override ובתוכה כל פעם קידמנו את ערך m\_CurrentColor ב1 כשערכו לכל היותר הוא 4, ולאחר מכן קראנו לnotifyChangeColorObservers כלומר שבעת לחיצה על הכפתור הנ"ל תפעיל פונקציונליות כלשהי על אותו פקד רלוונטי באמצעות הצבע השמור m\_CurrentColor שמועבר בתור פרמטר.

ל m\_ReportColorChangeDelegates נרשמים הפקדים הרלוונטים באמצעות המתודה applyChangeColorDelegates הנקראת לאחר FormMain\_Load, המממשים מתודה שונה לדוגמה changeForeColor או changeBackColor.

* Sequence Diagram
* Class Diagram

