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

* SortingFriends: המשתמש מקבל את רשימת חבריו ויכול למיין אותם לפי סוג המיון אותו בוחר. כאשר המשתמש בוחר חבר מסוים, מוצגות תכונות רלוונטיות עבור אותו סוג מיון
* FindBestFriend: המשתמש מחפש את החבר הכי טוב לו (ע"פ מתן משקולת לכל תכונה) שיש לו יום הולדת ב4 חודשים הקרובים ומציג את פרטיו. לאחר מציתא החבר הכי טוב, ניתן ליצור לחבר מסיבת הפתעה
* נציין כי בחלק מן התבניות לא ציינו בדיאגרמות עצמן מה כל מחלקה מייצגת לפי הדיאגרמה שנלמדה בכיתה, מכיוון שחלק מן התבניות מומשו באמצעות פיצ'רים שקיימים ב c# ולכן האדיגרמות לא זהות למה שנלמד בכיתה.

### תבנית מס' 1 – Strategy

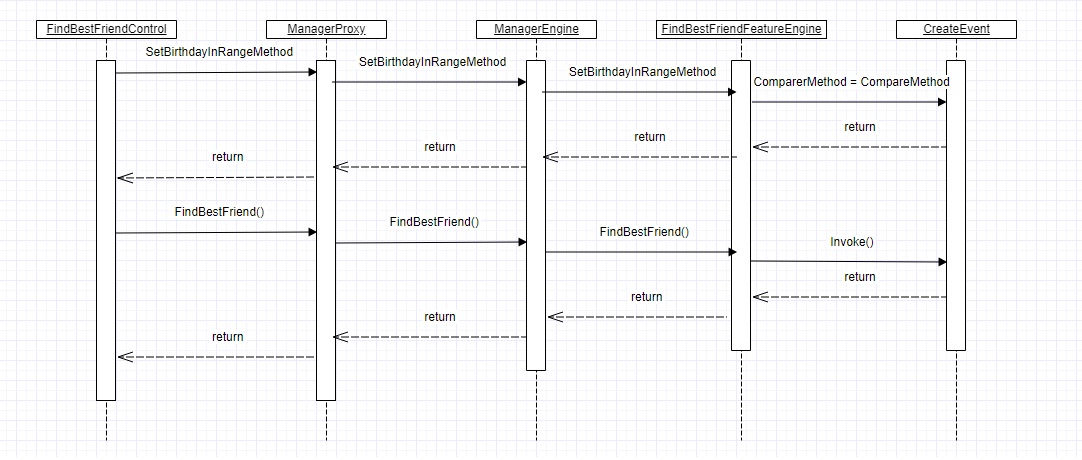
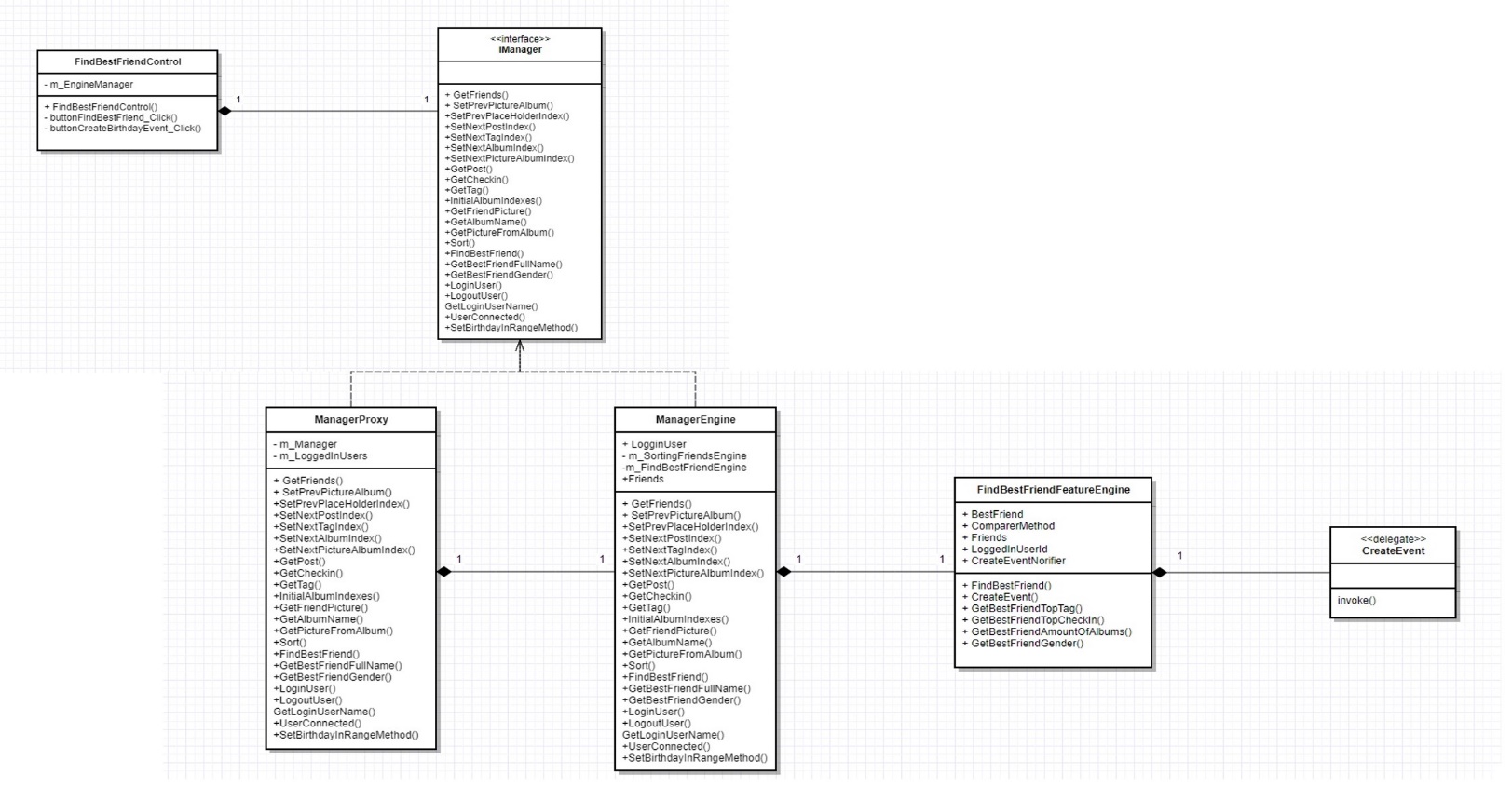
* סיבת הבחירה / שימוש בתבנית:

באפליקציה שלנו מחלקת FindBestFriendFeatureEngine מהווה את התבנית Strategy.

בחרנו בתבנית זאת על מנת לאפשר קוד גמיש אשר נוכל להשתמש בו שוב ללא צורך בשינוי קוד (reuse). הבעיה שאחנו באים לפתור היא לספק קוד גמיש וללא צורך בשינוי ובכך לספק למשתמש אשר משתמש במחלקה זאת לבחור את האלגוריתם השוואה לפיו הוא רוצה לעבוד באופן כזה שהוא רק צריך לספק את פונקציית השוואה ובכל לא צריך לשנות את הקוד הקיים. השתמשנו בתבנית זאת שהמשתמש יוכל לספק למחלקת FindBestFriendFeatureEngine את מתודת ההשוואה למציאת החברים בטווח תאריכים מסויים.

* אופן המימוש:

FindBestFriendFeatureEngine – זוהי מחלקה אשר מטרתה למצור את החבר הכי טוב של המשתמש. במחלקה זאת אנו מספקים למשתמש במחלקה זאת לבחור את מתודת ההשוואה של מציאת החברים שיש להם יום הולדת בטווח זמנים מסויים. מימשנו את התבנית במחלקה זאת ע"י Func Property אשר קיים ב c#. המשתמש במחלקה יכול לבצע השמה לפונקציית השוואה לפי בחירתו. בנוסף, על מנת להימנע מפוטנציאל לבאגים בכך שהמשתמש לא יספק מתודת השוואה, סיפקנו בעצמנו מתודת ברירת מחדל.

* Sequence Diagram
* Class Diagram

### תבנית מס' 2 – Observer

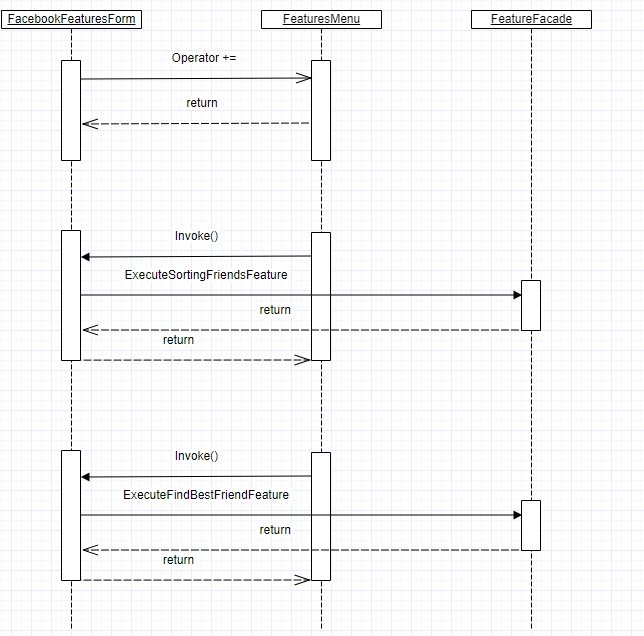
* סיבת הבחירה / שימוש בתבנית:

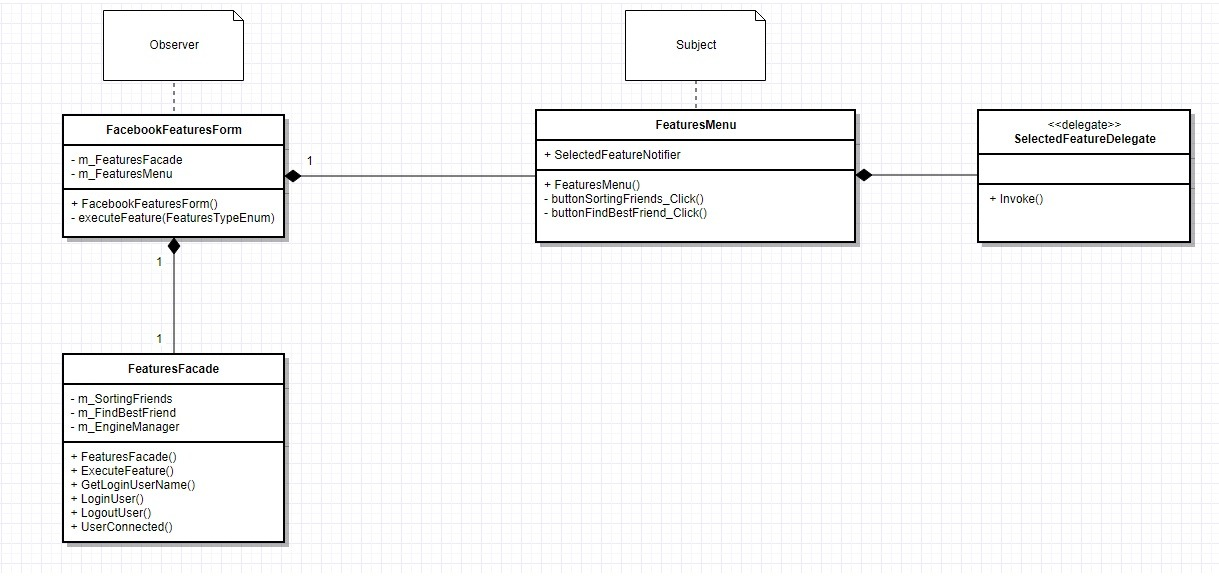
באפליקציה שלנו מחלקת FeaturesMenu מהווה בתבנית זאת את Subject ואילו מחלקת FacebookFeaturesForm את Observer. השתמשנו בתבנית זאת על מנת לאפשר שימוש של יחס מאחד לרבים באופן אסינכרוני, בכך שאם יש שינוי ב state של המחלקה הוא דואג לכך שהוא מודיע לכל המאזינים שלו. תבנית זאת מטרתה לפתור את הבעיה שאם יש שינוי ב state מסויים והוא משפיע על דברים אחרים, הוא דואג להודיע להם שהוא השתנה מבלי צורך לשנות את הקוד, כלומר כל observer רושם את עצמו אצל ה subject וברגע שהוא משתנה הוא יודיע להם (אסינכרוני).השימוש בתבנית זאת אצלנו באפליקציה מתבצעת בכך שה FacebookFeaturesForm מאזין ל FeaturesMenu. ברגע שכפתור מסויים נלחץ ב FeaturesMenu הוא מודיע ל FacebookFeaturesForm איזה כפתור נלחץ ואיזה פאנל עליו להציג כרגע.

* אופן המימוש:

כמו שרשמנו למעלה, מחלקת FeaturesMenu מהווה בתבנית זאת את Subject. במחלקה זאת השתמשנו ב event אשר קיים ב c# אשר מספק לנו מבחוץ attach ו dettach ע"י אופרטור += ו -=. כאשר נלחץ כפתור מסויים מתבצע invoke ל event ונשלח לכל המאזינים איזה כפתור נלחץ. FacebookFeaturesForm מכיל בתוכו את מחלקת FeaturesMenu ומספק מתודה לביצוע ל FeaturesMenu באמצעות אופרטור += , אשר תופעל כאשר תתבצע לחיצה על כפתור מסויים בתוך FeaturesMenu.

* Sequence Diagram



* Class Diagram

### תבנית מס' 3 – Iterator

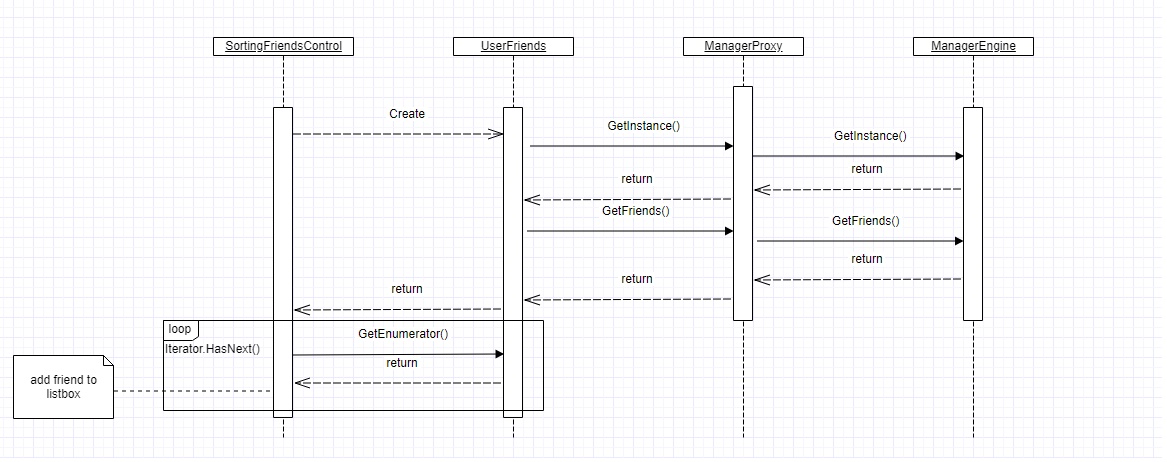
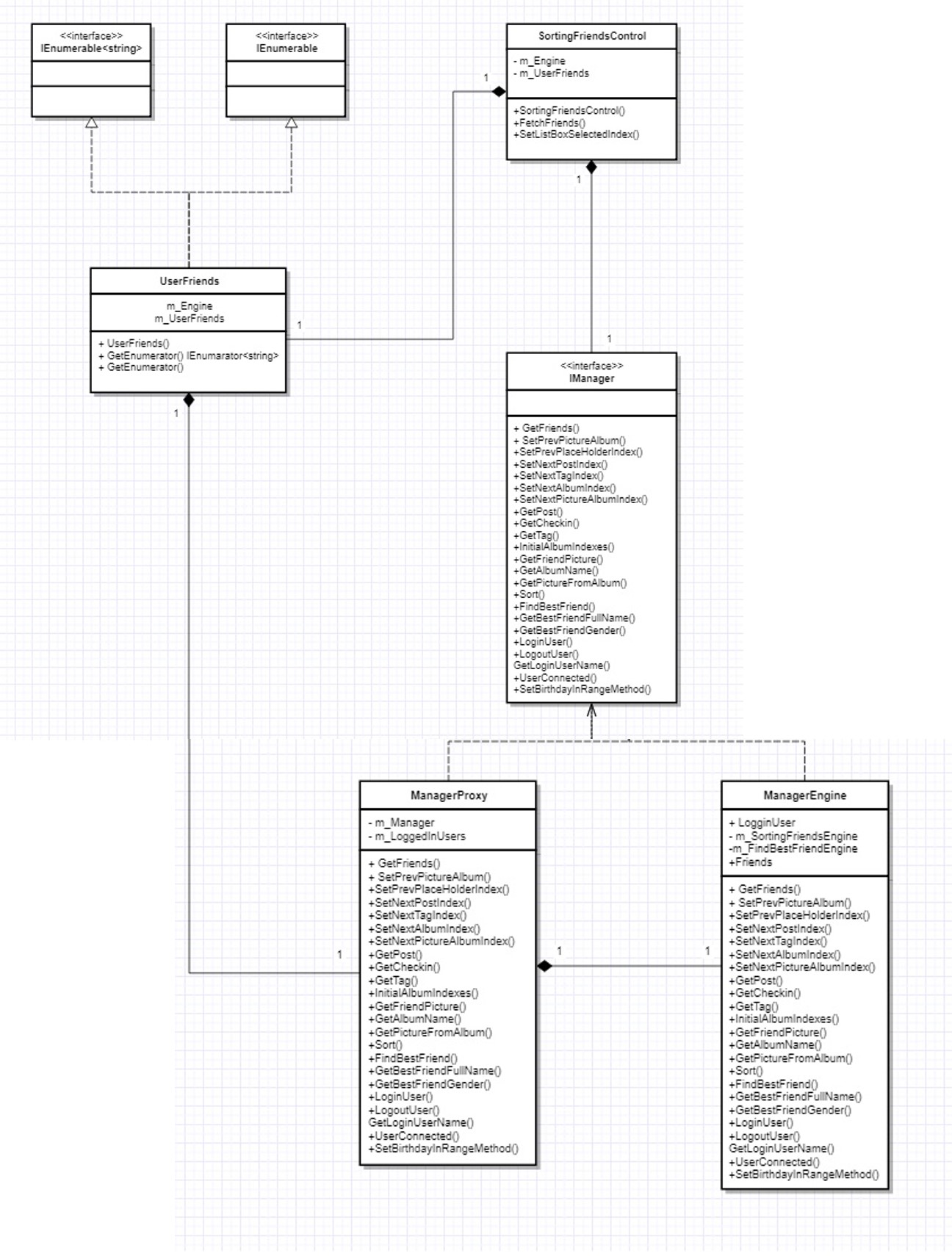
* סיבת הבחירה / שימוש בתבנית:

באפליקציה שלנו מימשנו את תבנית זאת על מחלקת UserFriends. תבנית זאת באה לפתור את הבעיה בכך שניתן לגשת ו"לרוץ" על איברי קולקציה במחלקה מסויימת מבלי להכיר את מימוש הקולקציה. בחרנו בתבנית זאת על מנת שנוכל לספק למשתמש לעבור על איברי המחלקה מבלי הצורך בלהכיר את אופן המימוש, כי ייתכן ובעתיד נרצה לשנות את מימוש הקולקציה. השתמשנו בתבנית זאת על מחלקת UserFriends אשר מספקת לי על היוזר המחובר למערכת לרוץ על שמות כל חבריו מבלי להכיר את אופן המימוש.

* אופן המימוש:

מימשנו את את תבנית זאת על מחלקת UserFriends בכך שהוא ממש את ה interfaces הבאים: IEnumerable<T>, IEnumerable. את המתודה GetEnumerator() של

IEnumerable<T> מימשנו באמצעות yield return שנלמד בכיתה. לכן באופן כזה אפשרנו למשתמש במחלקה לרוץ על איברי המחלקה ע"י לולאת forecah.

* Sequence Diagram
* Class Diagram