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

1. פיצ'ר המאפשר למשתמש לסנן את חבריו לפי אפשרויות מובנות מראש, הכוללות סינון לפי מין, חברים שדוברים את אותה השפה, חברים אשר נולדו באותו החודש, וחברים אשר הפרש השנים בגיל עם המשתמש הוא X (ניתן לבחירה באפליקציה). המשתמש בוחר מתוך ארבע האפשרויות את הסינון המבוקש, כאשר חלקם כוללים תתי-בחירות (כגון שפה מתוך הרשימה או בחירת שנה עבור ההפרש), ואז מוצגת למשתמש רשימה של החברים הרלוונטיים לפי הסינון שנבחר (בעת לחיצה על שם מסוים מתוך הרשימה, מופיעה התמונה של אותו חבר שנבחר).
2. פיצ'ר שמטרתו לתת למשתמש המלצות מחבריו בפייסבוק לטובת ביקור בארץ ובעולם. מכיל רשימה עם כל הערים שבהן גרים חבריו, כאשר בעת בחירה בעיר מסוימת תוצג למשתמש רשימה נוספת של כל החברים שגרים באותה העיר (לחיצה על חבר מסוים תראה את תמונת הפרופיל שלו). בעת בחירת העיר מתוך הרשימה, מקבל המשתמש מידע אודות נתוני מזג האוויר והמצב הנוכחי בעיר (באמצעות api חיצוני של חיזוי מזג אוויר), כגון: טמפרטורה, אחוזי לחות, זמני שקיעה וזריחה. בנוסף, יוצג למשתמש לינק לאתר ויקיפדיה, שם יוכל לקבל מידע נוסף אודות העיר שבה מעוניין לבקר.

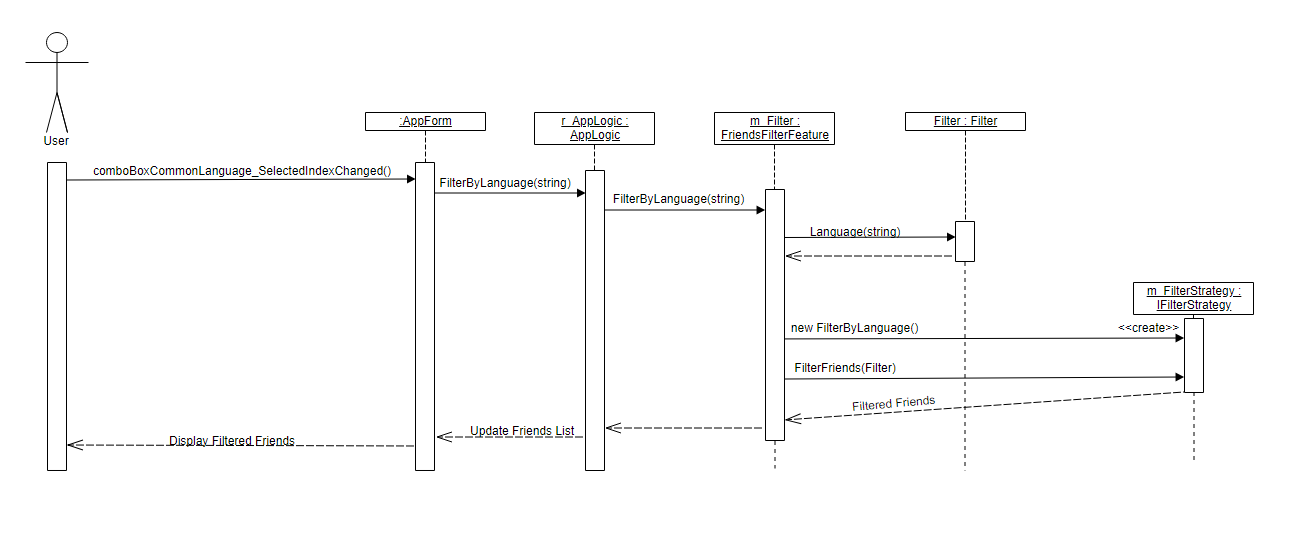
### תבנית מס' 1 – Strategy:

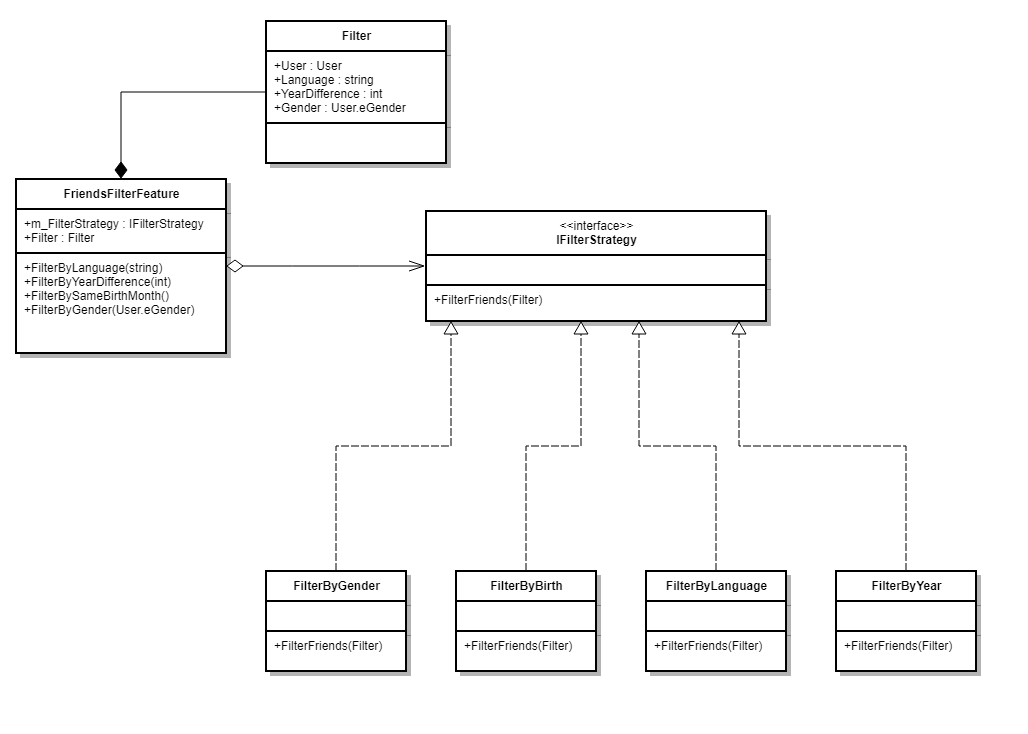
* **סיבת הבחירה / שימוש בתבנית:**

במערכת הנוכחית הרבה מאוד מתודות מהפיצ'ר הראשון (של סינון החברים) היו דומות, ונוצר מצב של שכפול קוד חלקי, כאשר רק הסינון עצמו שונה בין כל מתודה. לכן את החלק המשותף השארנו במחלקה העוטפת של הפיצ'ר השני, ואת המימוש הייחודי לכל סינון פיצלנו למחלקות נפרדות. כל מחלקה כזו מממשת בעצמה את החלק הייחודי כאשר החתימה זהה לכולם.

* **אופן המימוש:**

בכל שיטת סינון לקחנו את החלק הייחודי בכל מתודה ופיצלנו למחלקה נפרדת: FilterByGender, FilterByBirth, FilterByLanguage, FilterByYear. כל מחלקה כזו מממשת את האינטרפייס IFilterStrategy עם המתודה FilterFriends(Filter) בהתאם. המחלקה העוטפת שעד כה היתה האחראית על הפיצ'ר – FriendsFilterFeature – מחזיקה אובייקט מטיפוס IFilterStrategy, ו-Filter המחזיק פרופרטיס הנחוצים לסינון. בעת בקשה לסינון, המחלקה מעדכנת את Filter , מייצרת את האובייקט המתאים ומפעילה את המתודה FilterFriends שלו.

* **Sequence Diagram:**
* **Class Diagram:**

****

* **ההקבלה בין המחלקות בתבנית:**

Strategy = IFilterStrategy

Context = FriendsFilterFeature

ConcreteStrategy A..C = FilterByGender, FilterByBirth, FilterByLanguage, FilterByYear

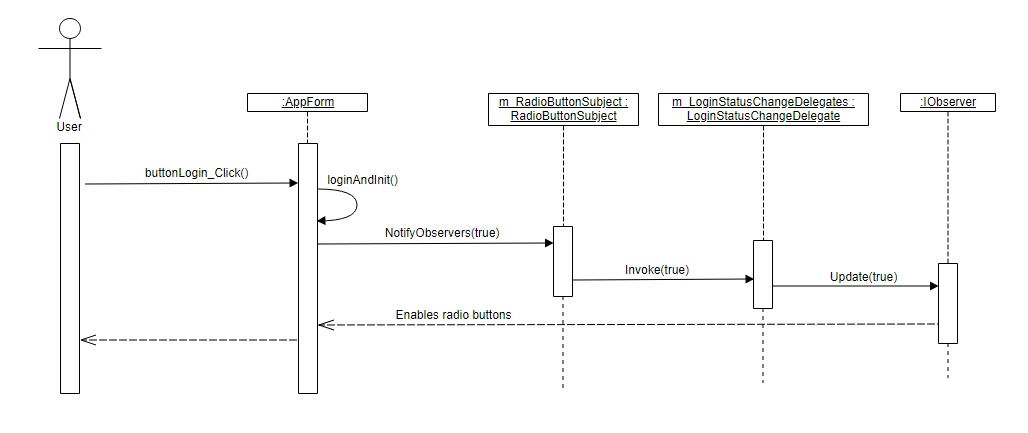
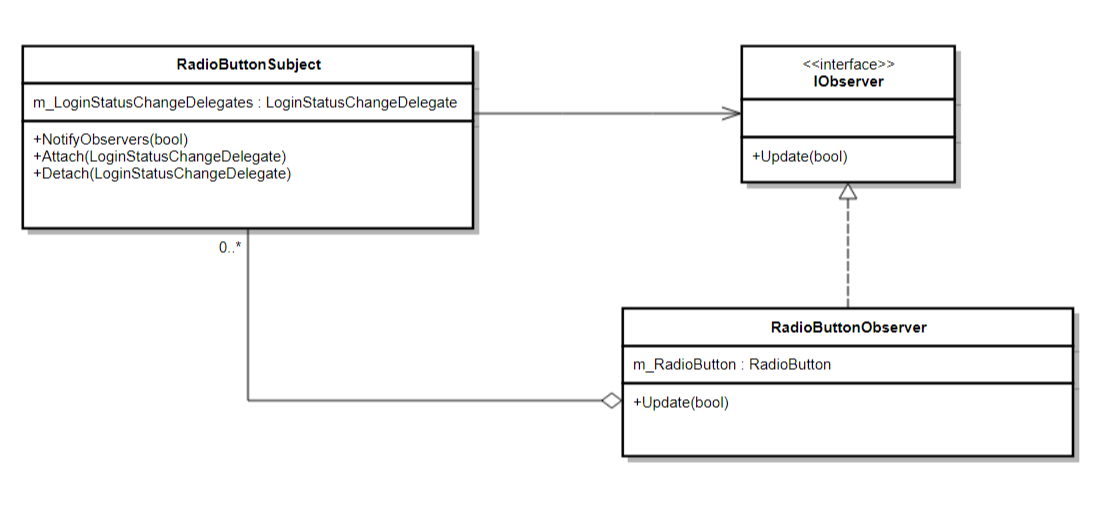
### תבנית מס' 2 – Observer:

* **סיבת הבחירה / שימוש בתבנית:**

הפיצ'ר הראשון מאפשר סינון חברים ע''פ כל מיני אפשרויות מוגדרות מראש, כאשר כל סינון מוצג כ-RadioButton שבחירה בו למעשה מפעילה אותו. על מנת למנוע מהמשתמש לסנן במקרה שיצא מהמערכת (Logout) או כדי לאפשר לו זאת כשנכנס (Login), צריך להגדיר את הפקדים כ-Enabled/Disabled בהתאמה. כעת, ברגע שאחד מכפתורי ההתחברות נלחץ, ה-RadioButtons מיודעים ומשתנים בהתאם (במקום לבדוק בכל לחיצה האם המשתמש מחובר או לא), כאשר ממשק המשתמש הוא זה שמימש אותה.

* **אופן המימוש:**

כל ה-RadioButtons מוגדרים כ-Observers ע''י מעטפת במחלקה RadioButtonObserver. מחלקה זו מחזיקה בקומפוזיציה את RadioButton ומממשת את האינטרפייס IObserver. כל ה-Observers נרשמים לשינויים בכפתורי ה-Login וה-Logout בהתאמה במחלקה RadioButtonSubject, האחראית על התווספות/מחיקת Observers ועל יידוע במקרה ונלחצו אחד מהכפתורי ההתחברות. בנוסף, כל המנגנון במחלקה זו מומש באמצעות Delegates.

* **Sequence Diagram:**
* **Class Diagram:**
* **ההקבלה בין המחלקות בתבנית:**

Subject = RadioButtonSubject

IObserver = IObserver

Observer = RadioButtonObserver

### תבנית מס' 3 – Template Method:

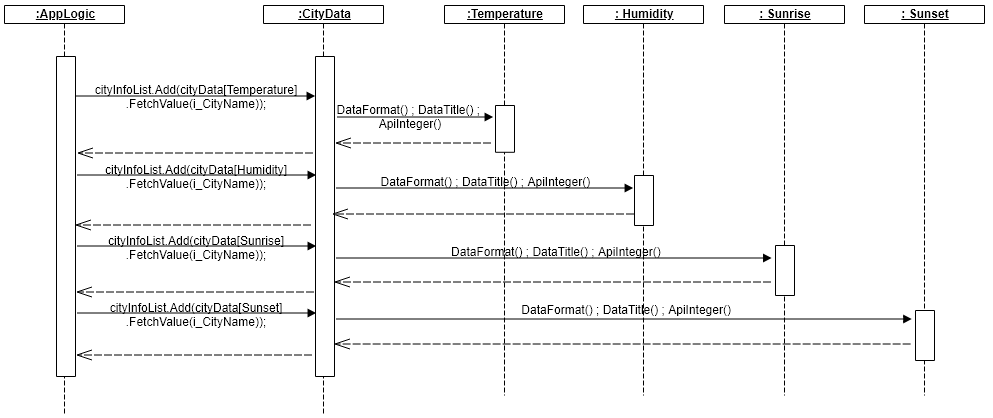
* **סיבת הבחירה / שימוש בתבנית:**

הפיצ'ר השני, שכאמור אחראי להבאת מידע על עיר נבחרת מתוך רשימת הערים שבהן גרים החברים, משתמש בין היתר ב-api חיצוני להבאת נתוני מזג האוויר ב-live. לצורך כך, הבאת המידע הרלוונטי בכל פעם עבור העיר הייתה מתבצעת עד כה באופן שהיה חוזר על עצמו עבור כל פריט מידע (בין אם זה אחוזי לחות, מזג אוויר נוכחי או שעת זריחה) – תחילה מביאים את המידע הרצוי לפי שם העיר המבוקשת, מפרסרים אותו ל-string, ולאחר מכן מוסיפים לו כותרת כלשהי בצירוף ה-data שהתקבל ב-XML (לעיתים יש גם צורך בהפעלת פונקציות על ה-data, כמו למשל המרת יחידות המידה של הטמפרטורה לצלזיוס). תהליך זה היה חוזר על עצמו עבור כל שורת מידע על העיר, וגרר לעיתים שכפול קוד וכן קושי ב-maintainability (הוספת נתון מידע חדש). לשם כך, הוספנו שימוש ב-template method באופן שמאפשר כעת גנריות ותחזוקתיות טובה יותר, וכמו כן סדר בקוד וחלוקת תפקידים לפי מחלקות.

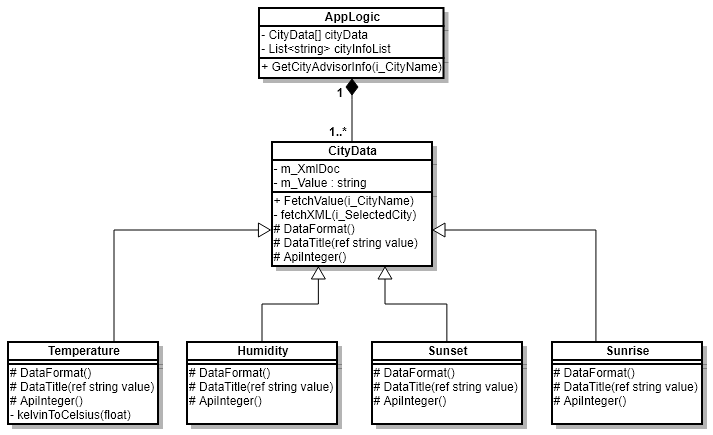
* **אופן המימוש:**

את מחלקת ה-CityAdvisorFeature המקורית, שהכילה את כל הפונקציות שטיפלו בהבאת המידע עבור כל נתון בנפרד, החלפנו במחלקת-אב אבסטרקטית בשם CityData. בנוסף, את כל הבאת המידע עבור העיר שנבחרה חילקנו למחלקות נפרדות שיורשות מ-CityData. כעת, באמצעות container החזקנו מערך שמכיל את האובייקטים של כל המחלקות היורשות הללו, שמשתמשות ב-override כדי להוסיף בכל פעם למתודת ה-Fetch את השינויים שרלוונטים לאותה שורת מידע שמגיעה מהשרת. בכך למעשה יצרנו מצב שיש מתודה גנרית שאחראית להביא שורת מידע על העיר, ומשתמשת בכל פעם במחלקה הנורשת המתאימה כדי להביא את הערך הרצוי של שורת המידע הרלוונטית.

* **Sequence Diagram:**



* **Class Diagram:**

****

* **ההקבלה בין המחלקות בתבנית:**

Abstract Class = CityData

Concrete Classes = Temperature, Humidity, Sunrise, Sunset

### Test Users:

הסיסמא לכל המשתמשים היא : Test1234

שמות המשתמשים הם:

[elliot\_pddykpa\_alderson@tfbnw.net](mailto:elliot_pddykpa_alderson@tfbnw.net)

[hannah\_tvexyzn\_baker@tfbnw.net](mailto:hannah_tvexyzn_baker@tfbnw.net)

[walter\_idmbcqb\_white@tfbnw.net](mailto:walter_idmbcqb_white@tfbnw.net)

[dolores\_vanvirn\_abernathy@tfbnw.net](mailto:dolores_vanvirn_abernathy@tfbnw.net)

[rick\_nnqhwnk\_sanchez@tfbnw.net](mailto:rick_nnqhwnk_sanchez@tfbnw.net)