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

* **שליחת הודעות למספר קבוצות במקביל**

המשתמש מסמן מספר קבוצות בהן הוא חבר, כותב פוסט ומפרסם אותו במקביל בכל הקבוצות שסימן.

* **ניגון יוטיוב פלייליסט מהמוסיקה האהובה**

המשתמש בוחר להקה אחת מרשימת הלהקות האהובות עליו והאפליקציה מציגה לו את היוטיוב פלייליסט של הבחירה שלו בתוך האפליקציה.

* **הצגת פרופיל משתמש חלקי**

הצגת פרטי פרופיל המשתמש הבאים: שם, תמונה, רשימת החברים, פרטי החברים, סטאטוס אחרון שפורסם וימי ההולדת של היום הנוכחי והיום שלאחריו.

בנוסף – אפשרות לפרסם פוסט על ה Wall.

### תבנית מס' 1 – [Façade]

* סיבת הבחירה / שימוש בתבנית:

לפשט למשתמש את משיכת ימי ההולדת של היום הנוכחי וזה שלאחריו.

במקום לפנות פרטנית למתודה שמחזירה את ימי ההולדת של היום הנוכחי ולמתודה שמחזירה את ימי ההולדת של מחר, המשתמש יפנה למתודה אחת שתחזיר את כל ימי ההולדת שהמערכת תומכת בהם. זה מאפשר גם להוסיף בעתיד תמיכה לעוד ימי הולדת (למשל: ימי הולדת של עוד יומיים מהתאריך הנוכחי) בלי שהמשתמש יצטרך לשנות את אופן השימוש.

* אופן המימוש:

Client

מחלקת FacebookUserProfile – מכילה אובייקט מסוג FriendsBirthdayFacade.

Facade

מחלקת FriendsBirthdayFacade – מכילה את התכונה Birthdays שפונה למתודה

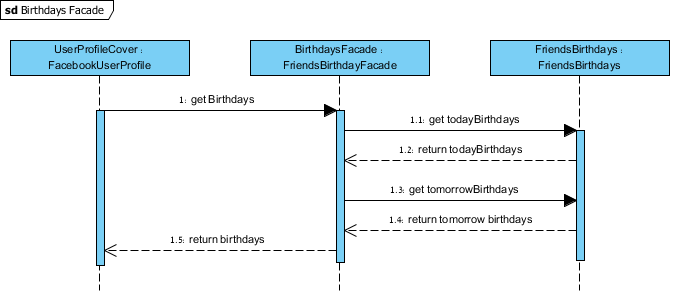
getTodayAndTomorrowBirthdays המחזירה Dictionary עם ימי ההולדת.

Library

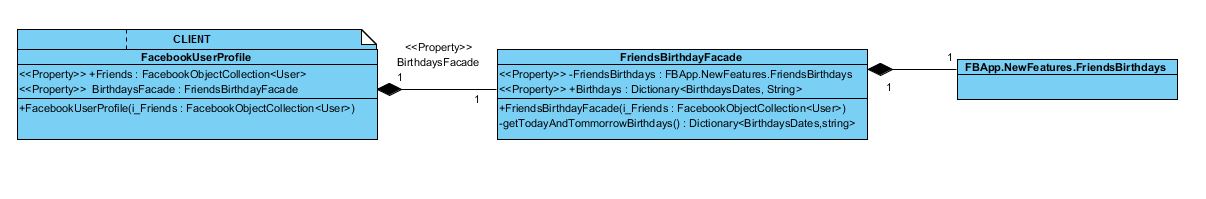
מחלקת FriendsBirthdays – מכילה את התכונות todayBirthdays ו tomorrowBirthdays בהן עושה שימוש מחלקת ה Facade.

את המימוש ניתן למצוא בקוד לפי שמות המחלקות הנ"ל.

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – [Singleton]

* סיבת הבחירה / שימוש בתבנית:

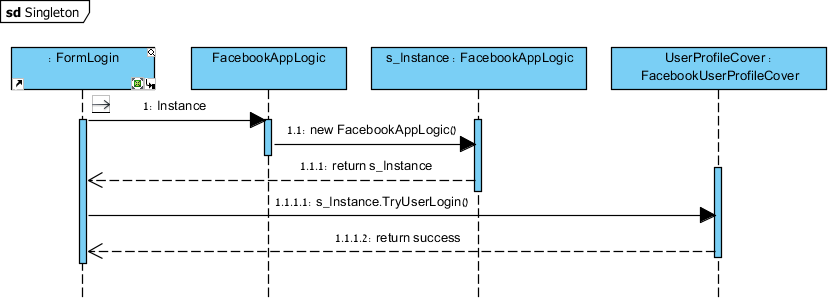
באפליקציה שלנו יש אובייקט FacebookAppLogic שמכיל את הלוגיקה של האפליקציה. החלטנו להפוך אותו לסינגלטון מכיוון שאנו לא רוצים שיהיה יותר מאחד כזה ומכיוון שיש המון גישות אליו לאורך חיי האפליקציה – עבור כל פעולה שהמשתמש באפליקציה מבצע יש קריאה לאובייקט הנ"ל.

* אופן המימוש:

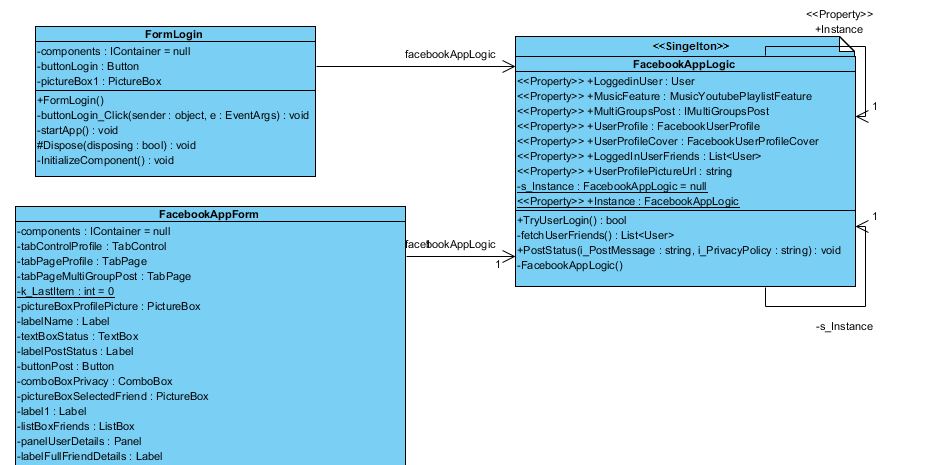
הפכנו את המחלקה FacebookAppLogic למחלקה שנעולה לירושה, הוספנו לה אובייקט סטטי מסוג FacebookAppLogic שנקרא s\_Instance, הוספנו מתודת Instance שמחזירה את s\_Instance – אם האובייקט אינו קיים, המתודה תיצור אותו, והפכנו את ה constructor ל private. כל השאר נשאר אותו דבר.

ניתן למצוא את המימוש במחלקת FacebookAppLogic.

* Sequence Diagram – מתארת את השימוש הראשון בסינגלטון בקוד.



Class Diagram



• קו החיים של המופע האחד מתחיל מהתחלה ולא רק מהרגע שהוא נוצר. (-4)

• הקשר בין הסינגלטון לעצמו לא מלא ולא מדוייק ( קומפוזיציה מלאה ) , ולמה 2 חצים מהסינגלטון לעצמו ?? (-5)

• איפה התיאור של התהליך שמבצע הסינגלטון במתאר את הסינגלטוניות שלו, כשמבקשים ממנו מופע בפעם הראשונה? הדיאגרמה אמורה לתאר את השימוש בתבנית, וזה לגמרי לא קורה בדיאגרמה הזו.(-4)

### תבנית מס' 3 – [Proxy]

* סיבת הבחירה / שימוש בתבנית:

פרוקסי מאפשר להוסיף לוגיקה לפיצ'ר שמבצע שליחת הודעות למספר קבוצות במקביל. בעזרתו אנו מבצעים שליחת הודעה לקבוצות ב Thread נפרד, שומרים את ההודעות ב cache במידה והחיבור לרשת אבד ודואגים שהן יישלחו ברגע שהחיבור חזר. בדרך זו האפליקציה עדיין שימושית בזמן שליחת ההודעות ומאפשרת לשלוח הודעות במהירות בלי להמתין עד ששליחה אחת תסתיים.

• מה הקשר בין ההסבר לבין התבנית? איפה בעיית A ו- B? מי כאן הרכיב המקורי שה- Client רוצה ויכול להשתמש בו? איזו לוגיקה מוסיף הפרוקסי, מדוע? מדוע לא יכולתם פשוט לממש את הרכיב המקורי (שיש לכם את הקוד שלו) בהתאם לנדרש? לא תיארתם את העובדה שהפרוקסי מתחזה לרכיב המקורי ולא הסברתם מדוע וכיצד זה תורם לכם.(-6)

* אופן המימוש:

ממשק ה ISubject

ממשק IMultiGroupsPost - מכיל את

1. המתודה SendPostToMultipleGroups שתפקידה יהיה שליחת ההודעה

לקבוצות הנבחרות.

1. תכונה LoggedInUserGroups שתפקידה יהיה להחזיר את הקבוצות של המשתמש.

מחלקת ה RealSubject

MultipleGroupPostFeature – מממשת את IMultiGroupsPost. מכילה מימוש למתודה SendPostToMultipleGroups – היא זאת שמבצעת בפועל את השליחה של ההודעה לקבוצות בפייסבוק.

מחלקת ה Proxy

MultiGroupFeatureProxy – מממשת את IMultiGroupsPost ומכילה:

1. אובייקט מסוג MultipleGroupPostFeature
2. מימוש ל SendPostToMultipleGroups – המתודה מכניסה את ההודעה והקבוצות לאובייקט שנקרא MultiPostObject ואותו היא מכניסה ל cache. לאחר מכן – היא מפעילה את המתודה sendMessagesToGroupsInTheBackround ב Thread נפרד. המתודה מבצעת את השליחה של כל ההודעות הנמצאות ב cache על ידי פנייה למתודה

SendPostToMultipleGroups מהאובייקט MultipleGroupPostFeature. במידה והשליחה נכשלת עקב בעיית תקשורת, נעשה ניסיון נוסף לשלוח את ההודעה עד להצלחה.

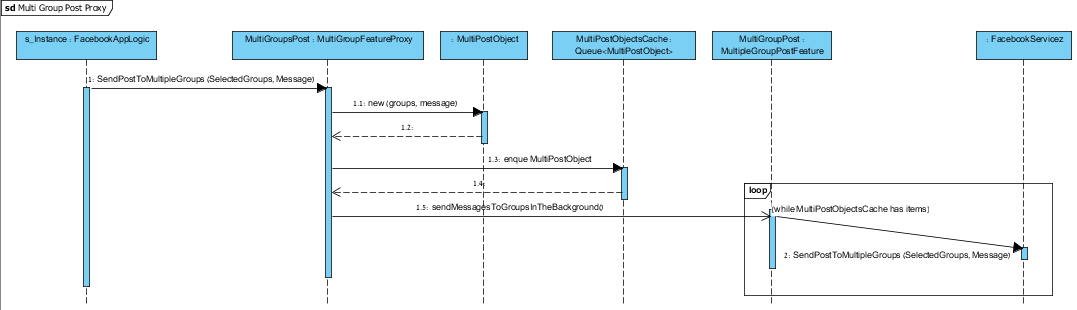
בזמן פעולת sendMessagesToGroupsInTheBackround האפליקציה זמינה לשימוש וניתן לשלוח הודעות נוספות והן נכנסות ל cahce.

מחלקת ה Client:

FacebookAppLogic – מכילה אובייקט מסוג IMultiGroupsPost שמייצג את ה Proxy.

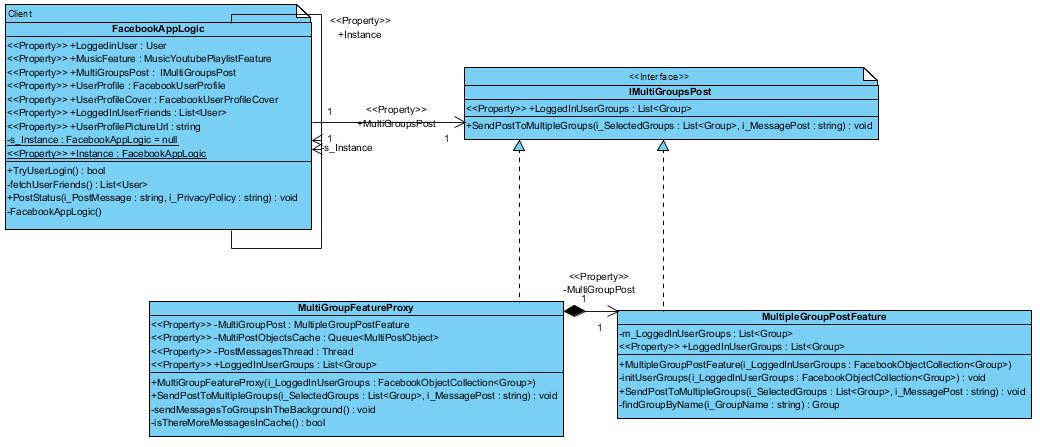
את המימוש בקוד ניתן למצוא לפי שמות המחלקות והמתודות הנ"ל.

* Sequence Diagram



הערה: בתוכנה שבה השתמשנו לבניית הדיאגרמה חץ הודעה אסינכרונית מיוצג על ידי חץ שראשו אינו מלא, לעומת חץ שראשו חצוי כפי שנלמד בשיעור.

* Class Diagram



### עבודה אסינכרונית

משיכת פרטי חבר בעת בחירתו מרשימת החברים

* היכן בקוד – בעת אירוע listBoxFriends\_SelectedIndexChanged, שורה 223 במחלקת FormApp.
* כיצד – בעת בחירה של חבר מרשימת החברים, נקראת המתודה fetchUserDetails מ Thread נפרד.
* מדוע – משיכת פרטי חבר לוקח הרבה זמן. בזמן זה אי אפשר להשתמש באפליקציה. ההפרדה של הפעולה הנ"ל ל Thread נפרד מאפשרת למשוך את המידע ולהמשיך להשתמש באפליקציה.

עדכון רשימת החברים בעת עליית האפליקציה

* היכן בקוד – במתודה initLoggedInUserProfile במחלקת FormApp.
* כיצד – קריאה למתודה initListBoxFriends מ Thread נפרד.
* מדוע – כדי שהאפליקציה תהיה זמינה למשתמש בעת שרשימת החברים מתמלאת וכדי שחלון האפליקציה יעלה מהר יותר.

עדכון רשימת הקבוצות שהמשתמש חבר בהן בעת עליית האפליקצייה

* היכן בקוד – בבנאי של FacebookAppForm.
* כיצד – קריאה ל initMultiGroupsPostFeature מ Thread נפרד.
* מדוע – כדי שהאפליקציה תהיה זמינה למשתמש בעת שרשימת הקבוצות מתמלאת וכדי שחלון האפליקציה יעלה מהר יותר.

עדכון רשימת הלהקות האהובות בעת עליית האפליקצייה

* היכן בקוד - בבנאי של FacebookAppForm.
* כיצד - קריאה למתודה initListBoxFriends מ Thread נפרד.
* מדוע - כדי שהאפליקציה תהיה זמינה למשתמש בעת שרשימת הלהקות מתמלאת וכדי שחלון האפליקציה יעלה מהר יותר.

שליחת הודעות למספר קבוצות

* היכן בקוד – במחלקת MultiGroupFeatureProxy במתודה SendPostToMultipleGroups.
* כיצד – המחלקה מחזיקה אובייקט Thread שמטרתו להפעיל את המתודה sendMessagesToGroupsInTheBackround. המתודה שולחת את ההודעות שנמצאות אצלה ב cache – בתוך התור MultiPostObjectsCache. בעת שליחת הודעה למספר קבוצות המתודה בודקת האם ה Thread חי או קיים – במידה ולא אז נוצר Thread חדש עבור sendMessagesToGroupsInTheBackround.
* מדוע – פעולת שליחה יכולה לערוך זמן רב בהתאם לאיכות החיבור לרשת. בזמן זה המשתמש לא יכול להשתמש באפליקציה, ובפרט לא יכול לשלוח הודעות חדשות. השימוש ב Thread נפרד מאפשר את הנ"ל.

### עבודה עם Data Binding

1. listBoxFriends – מקושרת ל friendListBindingSource.

השימוש נעשה ב:

* 1. מתודת fetchSelectedUserDetailsובמתודת fetchSelectedUserStatus הנקראות מ fetchUserDetails במחלקת FormApp.
  2. מתודה initListBoxFriends – קישור ה DataSource לחברים של המשתמש.

1. groupsBox – מקושרת ל loggedInUserGroupsBindingSource.

השימוש נעשה במתודת initMultiGroupsPostFeature במחלקת FormApp - קישור ה DataSource לקבוצות של המשתמש.

1. listBoxMusic – מקושרת ל pageBindingSource.

השימוש נעשה במתודת initMusicFeature במחלקת FormApp – קישור בין ה DataSource ללהקות האהובות על המשתמש.