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

* cityFriends – מביא את רשימת הערים של החברים וכמות החברים שלך שגרים בכל אחד מהם
* unfriendFriends – מביא את כל החברים שלך שאתה צריך להפסיק חברות איתם שאוהבים את "ניקלבק"

### תבנית מס' 1 – Proxy

* סיבת הבחירה / שימוש בתבנית:

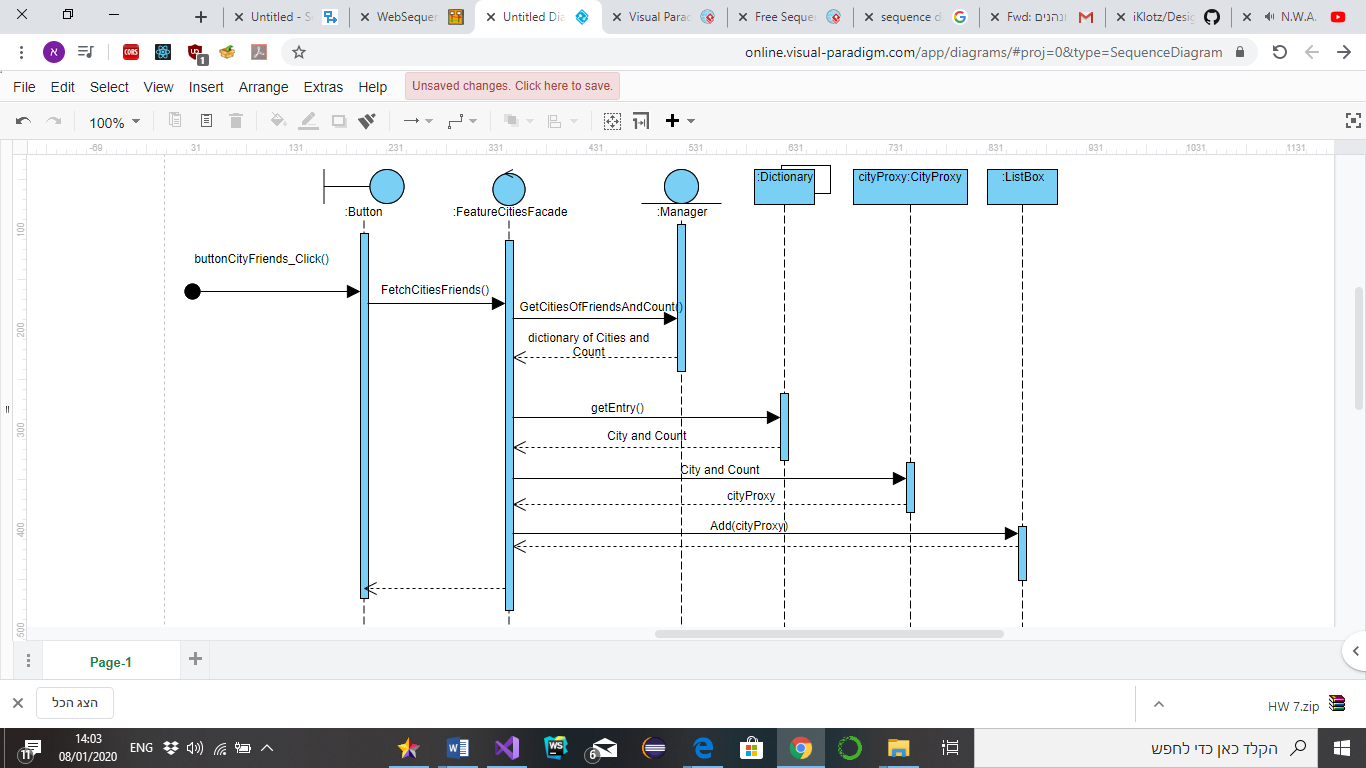
רצינו שבפיצ'ר הראשון הרשימה תהיה עדיין מסוג City, ועדיין שנוכל להוסיף טקסט של עיר וכמות החברים בתוכו, לכן השתמשנו ב CityProxy

* אופן המימוש:

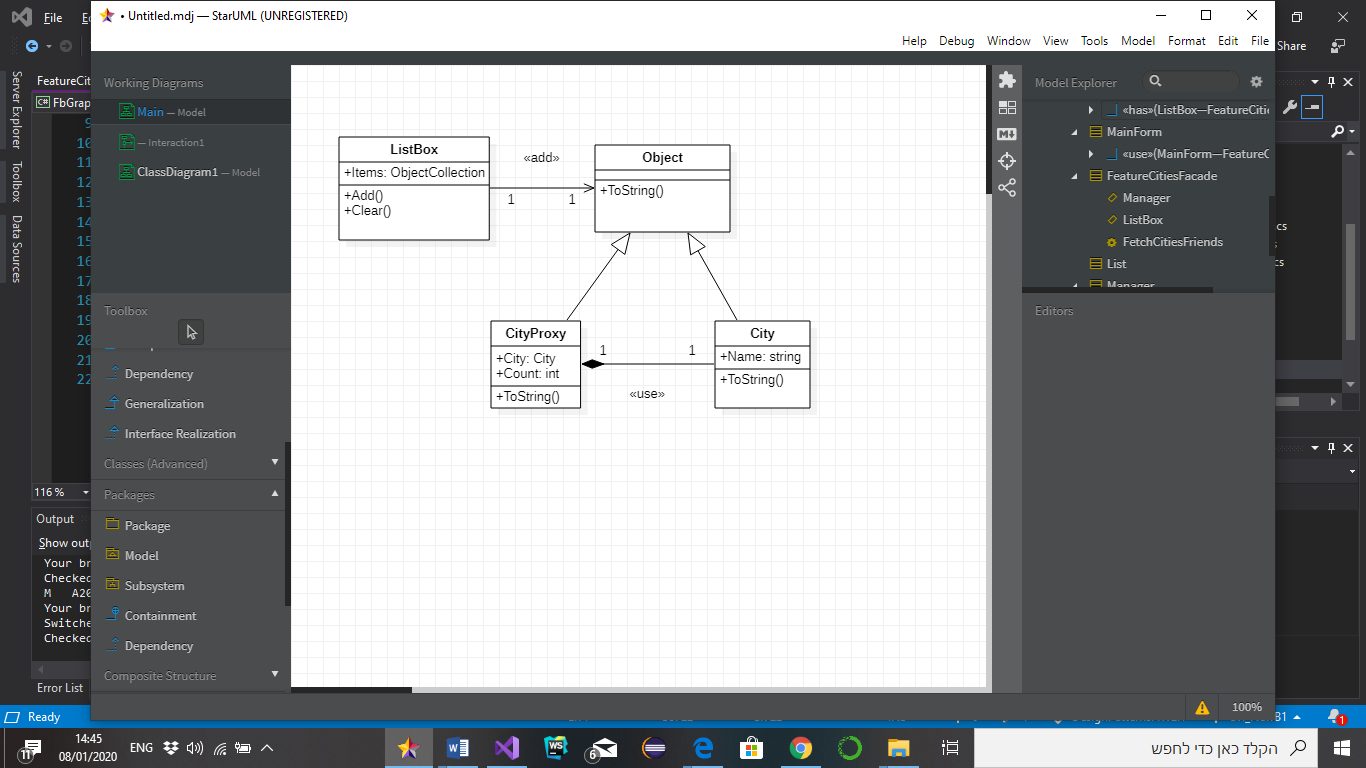
שמרנו Proparty מסוג City והפעולה שרצינו לחקות הייתה ToString() של City אך יותר משוכלל עם כמות החברים בתוכה.

ניתן למצוא את הCityProxy בחלק הלוגי של המערכת, ומשתמשים בו בFeatureCityFacade

* Sequence Diagram



* Class Diagram



הListBox(client) מוסיף לItems שלו אובייקטים מסוג CityProxy(Proxy) במקום אובייקטים מסוג City(RealSubject) שיורשים את המתודה ToString() ממחלקת Object(ISubject)

### תבנית מס' 2 –Façade

* סיבת הבחירה / שימוש בתבנית:

[תיאור הסיבה / הצורך בשימוש בתבנית במערכת שלכם]

רצינו לאפשר ללקוח שמשתמש בFacade לא לדעת על כל החלק הלוגי שקורא מאחורי הקלעים ולאפשר לו לגשת לזה בלחיצת כפתור, ובנוסף אם בעתיד נרצה להחליף את הצורה שבה אנו מפעילים את הFeatureCitiesFacade לאחת כזו שלא נרצה להשתמש בכפתור, ניתן לעשות זאת מבלי לשנות רכיבים בלוגיקה של הFacade

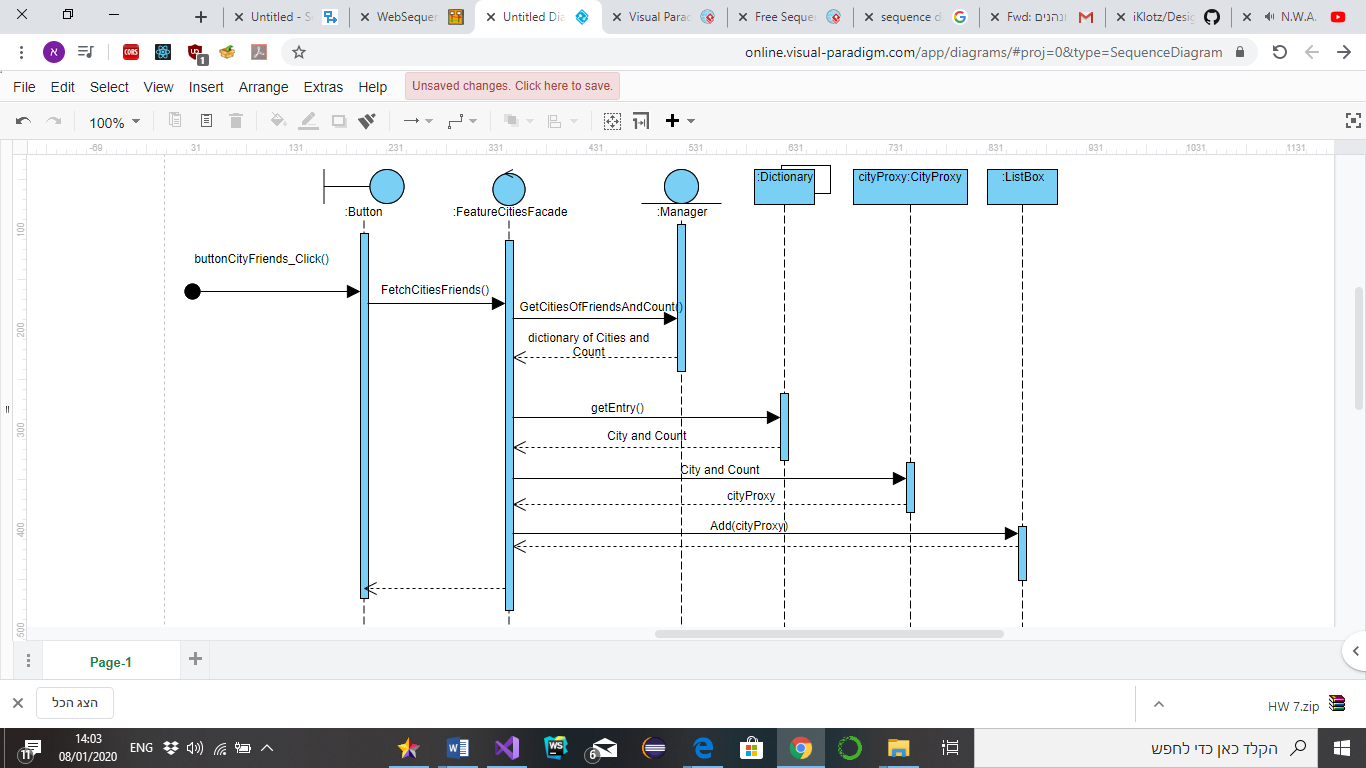
* אופן המימוש:

יצרנו Façade בשם FeatureCitiesFacade שמחזיק Property מסוג Manager ומסוג ListBox

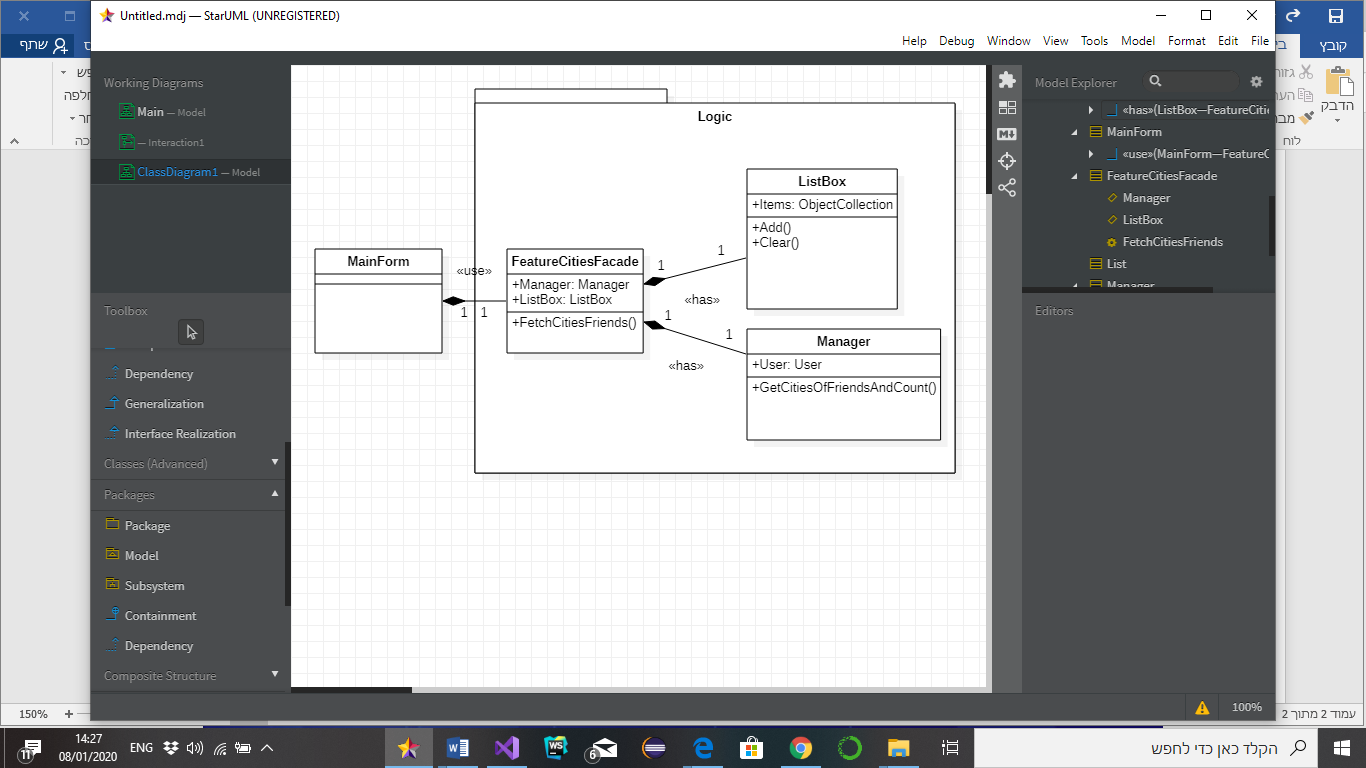
שמפשיט ללקוח, MainForm, את כל העבודה מאחורי הקלעים שהפיצ'ר עושה

ניתן למצוא את ה בFeatureCityFacade בחלק הלוגי של המערכת, ומשתמשים בו בMainForm בעת לחיצה על כפתור buttonCityFriends

* Sequence Diagram



* Class Diagram



בעת הפעלת הכפתור ב,MainForm(client) נעשה שימוש בFeatureCitiesFacade(Façade) שמחזיק אובייקט מסוג ListBox(SubSystemA) ומסוג Manager(SubSystemB) ומפעיל את המתודה FetchCitiesFriends() שעשוה שימוש בלוגיקה של התת מערכות הללו.

### תבנית מס' 3 – Builder

* סיבת הבחירה / שימוש בתבנית:

[תיאור הסיבה / הצורך בשימוש בתבנית במערכת שלכם]

בנינו רכיב שמאפשר לראות איך עמוד הפייסבוק היה נראה בצבעים שונים בילדר התאים מושלם למשימה זו.

* אופן המימוש:

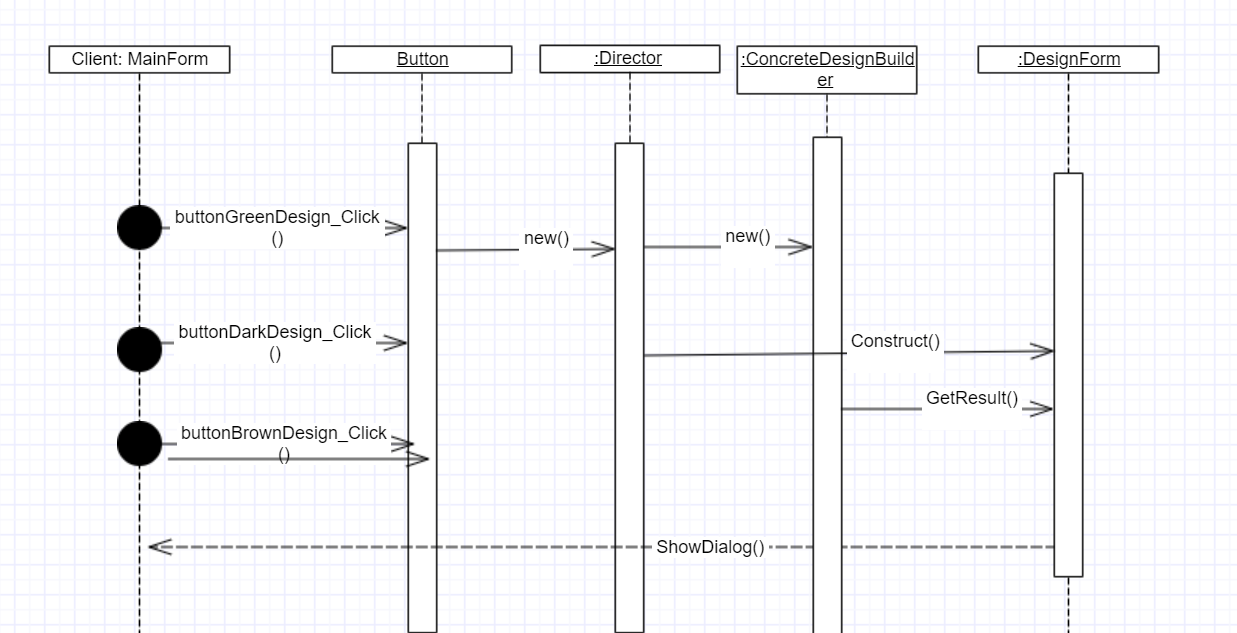
[תיאור המימוש והיכן ניתן למצוא אותו בקוד]

בעת הפעלת הכפתור See green/brown/dark version נוצר דירקטור, ובילדר מהסוג concrete builder, לאחר מכן דירקטור מפעיל מטודת Construct() היוצרת פורם עם צבעים מוגדרים מראש. (אחת משלושת האפשרויות)

מימוש BY THE BOOK, שות המחלקות מדברים מעד עצמם. בתוך התיקיה Logic בתיקייה בשם DesignBuilder

* Sequence Diagram

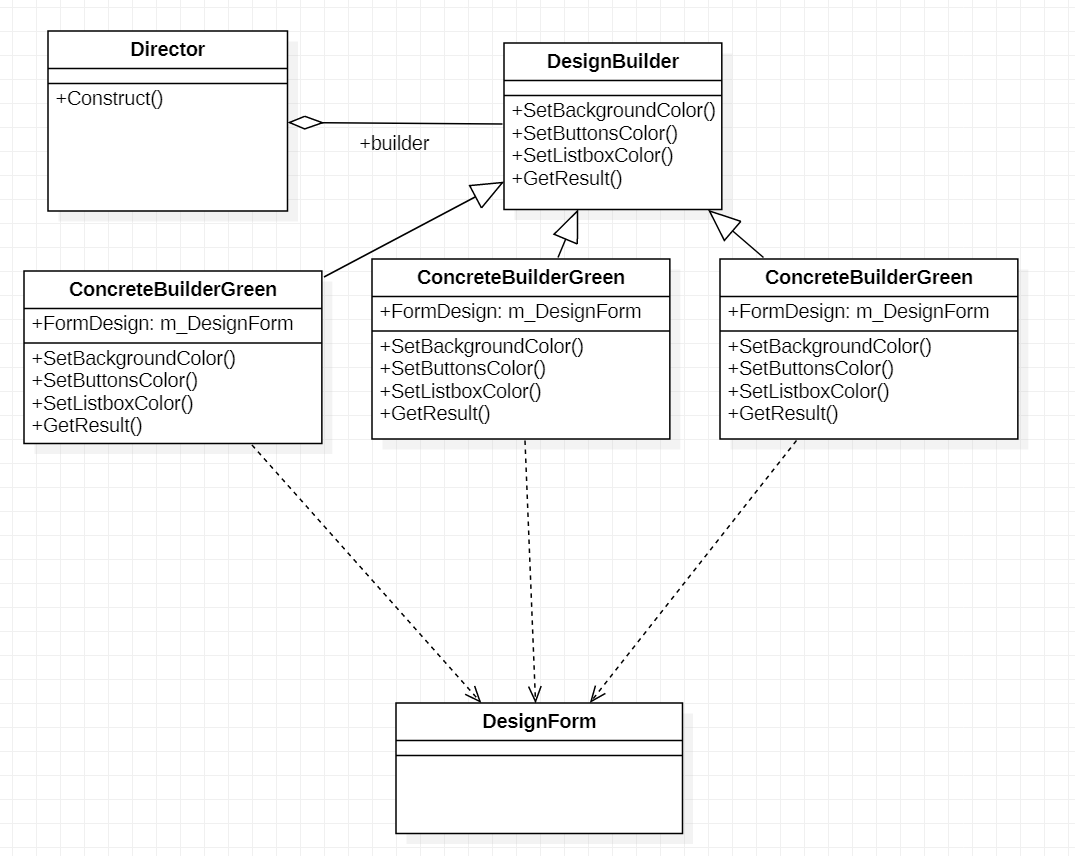
[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]



* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

שמות המקבילים בפטרן הם אותם השמות בתוספת מילהDESIGN. חוץ מProduct שהוא במקרה שלנו DesignForm.



### עבודה אסיכרונית

* היכן בקוד:

השתמשנו בעבודה אסינכרונית בMainForm בעוד שהlogin נעשה בThread אחד הבאת המידע בעזרת המתודה fetchFacebookInfo בThread אחר

* כיצד:

השתמשנו בSystem.Threading ובעזרת שליחת המתודה fetchFacebookInfo לThread חדש והפעלתו בעזרת Start() גרמנו לכל התהליך של הבאת הנתונים לרוץ בThread נוסף.

בנוסף לכל הפעולות ומתודות שעלולים לזרוק לנו שגיאה על כך שלא ניתן להם להפעיל אותם בThread שונה מהThread שהם נוצרו, דאגנו לעטוף עם Action שמפעיל אותן בעזרת lamda expression ועל כל Action שכזה הפעלנו בעזרת Invoke()

* מדוע:

על מנת להביא את המידע מהפייסבוק באופן מקביל, שנוכל לבצע בו עוד פעולות ושה MainForm לא יהיה תקוע אך ורק על הבאת הנתונים

### עבודה עם Data Bindings

* היכן בקוד:

בMainForm במתודה fetchFriends קישרנו את הUser שהגיע לנו מהData source שהוספנו FacebookWrapper.ObjectModel אל בחירת אחד החברים שלנו, והצגנו מידע עליו בעזרת

One Way Data Binding