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

* [הפיצר הראשון יוצר רשימה ומציג למשתמש, את חבריו בפייסבוק, החברים החדשים שלו, וחברים שהסירו אותו מרשימת החברות.]
* [הפיצר השני יוצר שני רשימות למשתמש לפי החיפוש שבחר בעצם כדי למצוא דייט מרשימת החברים הקיימים, בין אם הכי מתאים או מתאים אבל לא לגמרי]

### תבנית מס' 1 – [סינגלטון]

* סיבת הבחירה / שימוש בתבנית:

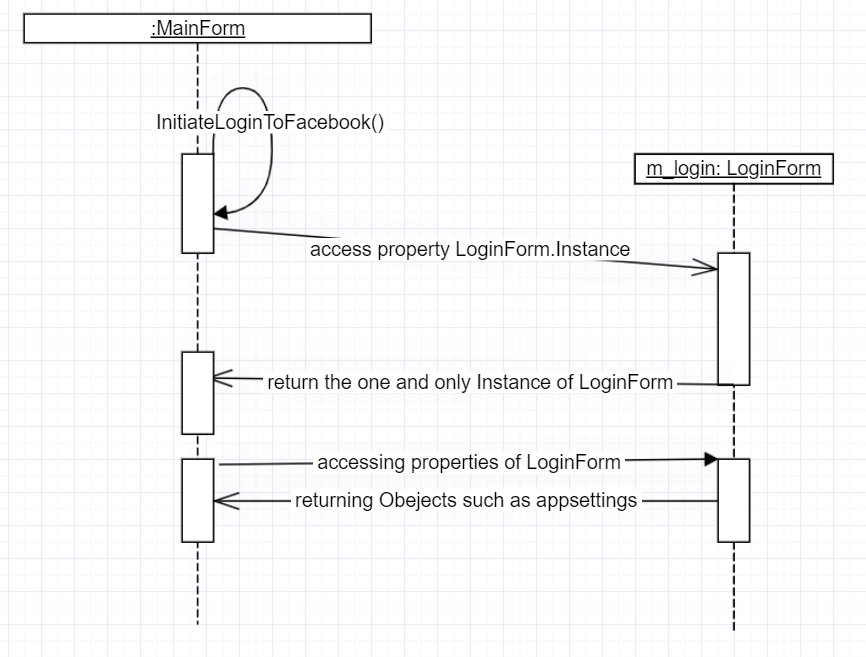
[במערכת שלנו קיימת מחלקה מיוחדת ויחודית שנקראת LoginForm, היא מחלקה שיש לה מטרה אחת, היא מציגה חלון (form) שנועד להתחברות המשתמש לשרת של פייסבוק, ומאיין הצגה ראשונית של האפליקציה (מסך כניסה) – ברור מאליו שמחלקה זו צריכה להיות אחת ויחידה בכל המערכת שלנו, יש לה מטרה אחת, ועם יהיה מספר איתחולים ממנה יווצרו המון בעיות: התחברות לפייסבוק מספר פעמים עלולה לשבש את המערכת, המחלקה בתוכה appsetting, שומרת קובץ עם השמירה של המשתמש, כמה מחלקות יכולות לדרוס את השמירה זו של זו ללא צורך. לכן, בחרנו להשתמש בתבנית סינגלטון שפותרת לנו את הבעיה ההנדסית הזו.]

* אופן המימוש:

[המימוש נמצא במחלקה LoginForm עצמה, בחרנו במימוש פשוט של semi-lazy, משום שאין צורך למימוש סינגלטוני מורכב יותר, אין עוד מטודות סטטיות במחלקה, כמו כן המילה השמורה volatile, מגנה עלינו מבעיות עם תהליכים מרובים. המטודה הסטטית (היחידה) Instance מחזירה לנו את האובייקט LoginForm.]

* 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

### 

### תבנית מס' 2 – [Factory method]

* סיבת הבחירה / שימוש בתבנית:

[הצורך ליצור אובייקטים מסוגים שונים בעלי אב פולימורפי משותף בזמן ריצה / במערכת שלנו יש פיצרים הקשורים לחברים של המשתמש, לפוסטים של המשתמש, ועוד. בכל אחד מן המקרים הללו קורה משהו משותף : יש צורך לחלון נוסף שיקפוץ בזמן לחיצה על האובייקט שבפיצר, ואז יועבר אובייקט שמכיל את את המידע עבור אותו האובייקט שבפיצר, למשל מידע כמו שם הפוסט, מסר הפוסט, התגובות של הפוסט וכולי. במערכת שלנו בחרנו לייצר אובייקט אבסטרקטי שממנו נורשים האובייקים הספציפיים יותר שבהם יש מימוש של אותו האובייקט שעוטף את האובייקט של הפייסבוק (של הפיצר) ודולה ממנו את המידע הרלוונטי - שאחר כך החלון הקופץ יוכל להשתמש במידע זה שבתוך האובייקט.

השתמשנו בתבנית factory method שיוצרת בעבורנו את האוביקט המתאים לפי ה type שהוא בעצם האובייקט הספציפי של פייסבוק, ואז ה"מפעל" מחזיר לנו את האובייקט הנדרש לחלון הקופץ, שיטת המפעל מסדרת לנו ועוטפת לנו (מכמסת) את עניין היצירה של האויבייקטים הללו, ונותנת לנו דרך פשוטה לעדכן ולהוסיף אובייקטים אם בעתיד יהיה צורך בשינוי כלשהו.]

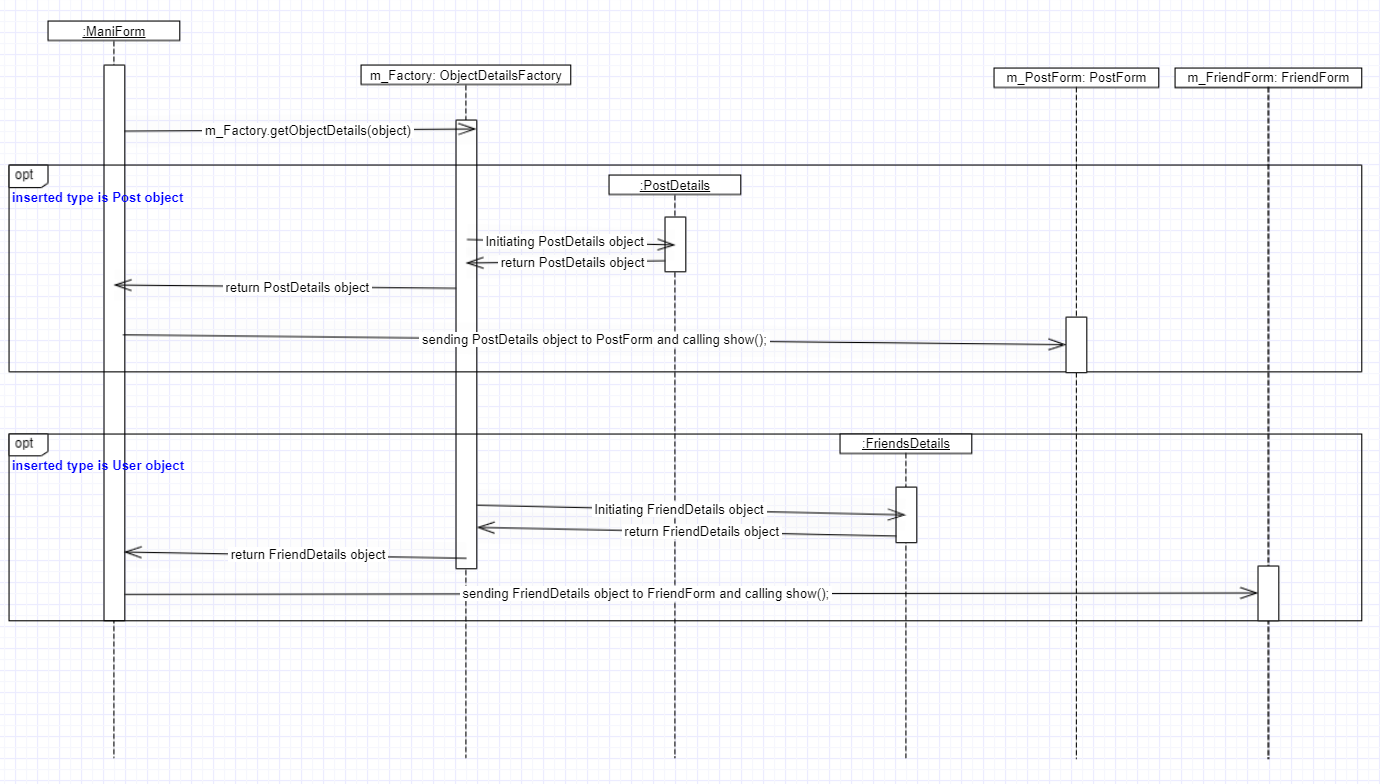
* אופן המימוש:

[בקובץ "ObjectDetails.cs" יש את רוב המימוש. האובייקט ObjectDetailsהוא האוביקט האבסטרקטי שאמור להכל את המידע הרלוונטי מתוך האובייקט הספציפי של פייסבוק. אחרי זה המחלקות הממשות אותו: PostDetailsו FriendDetails ואחר כך המפעל עצמו ObjectDetailsFactoryשמקבל type שהוא האובייקט מתוך פייסבוק, ומייצר בישבילנו את אובייקט המידע הדרוש. אותו אובייקט המידע יועבר בזמן ריצה לחלון הקופץ, שבתורו ישתמש במידע שבתוכו לצורך הצגתו.

ניתן לימצוא שימוש בזה בפועל במחלקה MainFrom במטודות InitPostFormו InitFriendFormכמו כן המפעל עצמו מוכרז כאובייקט member של MainForm.]

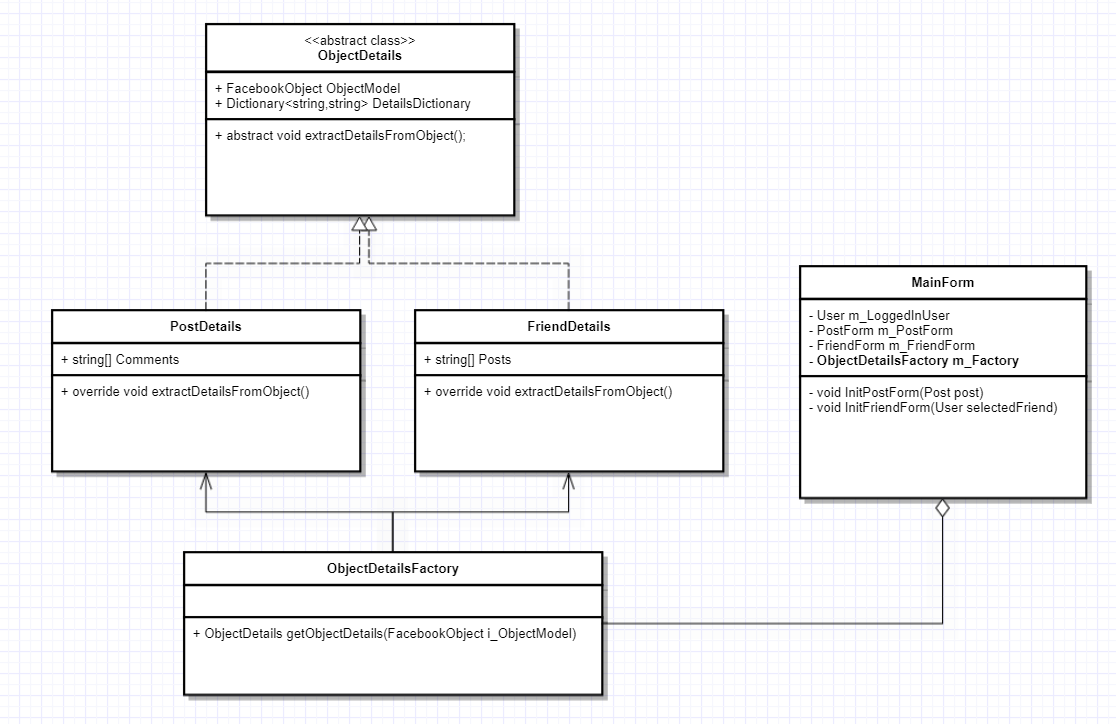
* Sequence Diagram

[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]



במחלקות שלנו, ObjectsDetails מקבילה לpizza,PostDetails ו FriendDetails ל סוגי הפיצות הממומשות, למשל, NewYorkStylePizza..

ו ObjectDetailsFactory ל מלחקת המפעל של הפיצריה עצמה ששם יש את מטודת היצירה - NewYorkPizzeria

### תבנית מס' 3 – [PROXY]

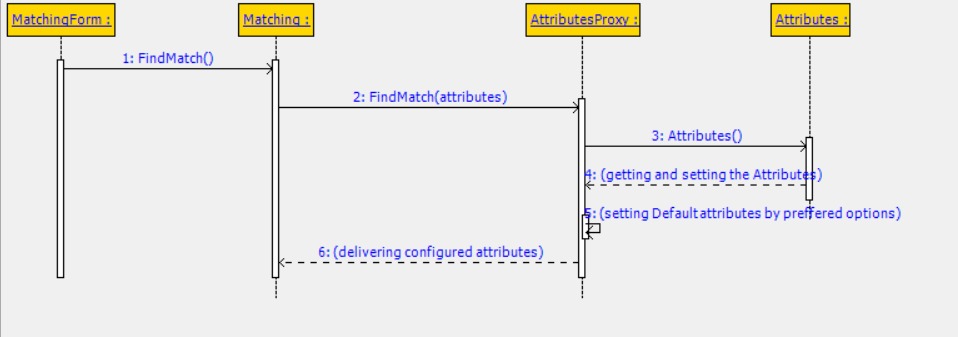
findmatch הוא LEGACY COMPONENT הסיבה לשימוש בPROXY הוא כדי לשדרג בעצם את התכונות של בן אדם מבלי לבצע יותר מדי שינויים במקור ורק ליצור בעצם מחלקה פולימורפית שתממש את ממשק בדיוק כמו המחלקה המקורית ATTRIBUTES ובמחלקה הPROXY בעצם נבצע הגדרות של תכונות שהם ברירת מחדל משהו שלא היה לנו במקור ולכן היה שימוש בPROXY

* אופן המימוש:

את המימוש ניתן לראות לאחר שהמחלקה MATCHING בעצם מבצעת את FINDMATCH() שזה הוא LEGACY COMPONENT שלנו ואחר כך יש את הממשק שלנו IATTRIBUTES והמחלקה ATTRIBUTES יורשת ממנה והיא בעצם ה REAL SUBJECT שלנו ויש את ה ATTRIBUTESPROXY ושם בעצם נמצא המימוש העיקרי והסיבה בעצם למה יצרנו PROXY

* Sequence Diagram

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



* Class Diagram ATTRIBUTESPROXY הוא בעצם ATTRIBUTES והיחס פולימורפי ATTRIBUTES הוא REAL SUBJECT ו ATTRIBUTESPROXY הוא בעצם הPROXY שלנו 