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

**משחק מיקומי תמונות אלבומים:**

* הפיצ'ר מציג למשתמש ארבע תמונות Cover מתוך אלבומיו אשר קיים להם מיקום ותמונה. (משתמש ללא 4 אלבומי בעלי תמונה ומיקום, הפיצ'ר לא יהיה זמין עבורו).
* בעת לחיצה על אחת התמונות, יפתח טופס המשחק-
  + תמונת האלבום
  + 4 תשובות אפשריות נכונות
* בעת מענה על תשובה נכונה- מוצגת הודעה למשתמש, מספר הנקודות שצבר במשחק עולה והתמונה בטופס הראשי של האפליקציה מוחלפת לתמונת אלבום אחרת.
* בעת מענה על תשובה לא נכונה- מוצגת הודעה למשתמש. התמונה נשארת על הטופס הראשי של האפליקציה, כלומר המשתמש יכול לנסות לשחק שנית ולנחש שוב היכן צולמה התמונה.

**מציאת דייט:**

* הפיצ'ר מאפשר למשתמש למצוא דייט בעזרת הפייסבוק עם אדם שאינו זר לחלוטין.
* על המשתמש לבחור את חתך הנתונים: מין וטווח גילאים.

לאחר לחיצה על כפתור Find Match יוצגו למשתמש כל החברים שלו אשר:

1. במין המבוקש
2. בטווח הגילאים המבוקש
3. הגדירו את המצב הזוגי שלהם כרווקים/ לא הגדירו

במידה ונמצאה התאמה, המשתמש יוכל לבחור מתוך רשימת המתאימים/ות את החבר שהוא מעוניין ולצפות בנתוניו.

### תבנית מס' 1 – Adapter

* סיבת הבחירה / שימוש בתבנית:

במערכת השתמשנו ב-Adapter Patterns כדי להתאים לצרכינו את השימוש בחיבור לרשת הפייסבוק.

המחלקה FacebookService מאפשר חיבור לשרת פייסבוק בעזרת 2 מתודות מרכזיות בהן השתמשנו:

Login – מתודה המקבלת את AppID ו-Premission ולאחר התחברות משתמש מחזירה אובייקט LoginResult המכיל את AccessToken ו- LoggedInUser עבור המשתמש שהתחבר זה עתה.

Connect – מתודה המקבלת את ה-AccessToken של המשתמש ומבצעת חיבור לשרת.

מכיוון שבאפליקצייה שלנו קיימים תנאים שונים להתחברות –

* אם המתשמש ביקש RemmberMe יש לבצע חיבור בעזרת Connect
* אחרת, יש לבצע חיבור בעזרת Login

בחרנו לממש מחלקת AdapterLogin שתבצע את בדיקות אלו ותבצענה את הפעולות הדרושות.

* אופן המימוש:

ניתן למצוא את המתודות במחלקה AdapterLoginFacebook

בממשק המשתמש בעת יצרת האובייקט r\_LoginAdapter נמסור לו גם את AppSettings של האפליקצייה.

כעת, בעת הופעת טופס האפלקצייה הראשי נבצע חיבור בעזרת Connect.

המתודה 'המותאמת' תבצע בדיקה מתוך AppSettings (אם המשתמש ביקש לזכור אותו ביציאה הקודמת מהאפליקציה ואם קיים AccessToken), ותתחבר במידה ותנאים אלו מתקיימים.

בטופס האפלקצייה הראשית במידה ולא התבצע חיבור, כלומר חזר LoginResult=null, נציג את הטופס r\_LoginForm ונבצע קריאה למתודה r\_Login.Adapter.Login שיודעת לבצע חיבור בעזרת AppID ו- Premissions של AppSettings.

* 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 – FACADE

* סיבת הבחירה / שימוש בתבנית:

בחרנו לממש את ה-Façade Pattern בפיצ'ר של "משחק מיקומי תמונות".

הסיבה שבגינה בחרנו לממש Pattern זה, היא בגלל שהמשחק מכילה לוגיקה רבה ומורכבת ורצינו להפריד באופן כמעט מוחלט בין השכבה הלוגית לממשק המשתמש של המשחק.

בעתיד, ניתן יהיה בקלות רבה יותר להתאים ממשק משתמש חדש למשחק.

* אופן המימוש:

המחלקה הסטטית FacadePictureGame מכילה DataMembers אשר זהים לרכיבים התצוגתיים בממשק המשתמש. למשל: בטופס המשחק FormPictureGame קיימים ארבעה כפתורים עם תשובות אופציונאליות וכך גם ב FacadePictureGame קיימים ארבעה תשובות אפשריות.

הFacade שלנו מבצעת קריאות ללוגיקה של המשחק הממומשת במחלקה הסטטית PictureGameFeature ומחזיר לממשק המשתמש (הטופס המוצג FormPictureGame) את הנתונים הרלוונטים להצגה.

* 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' 3 – ABSTRACT FACTORY

* סיבת הבחירה / שימוש בתבנית:

באפליקצייה שלנו ישנם טפסים שונים המוצגים למשתמש:

טופס אלבום – המכיל כותרת, תמונה, מיקום ותאריך.

טופס חבר (משתמש) – המכיל שם, תמונה, מין, גיל ומקום מגורים.

טפסים אלו, זהים מבחינת כמות ה-Labels ומיקום התמונה ולכן, על מנת למנוע שיכפול קוד ויצירת טפסים מרובים בממשק המשתמש בחרנו לממש Abstract Factory Pattern שמייצר את הטופס המבוקש להצגה בכל פעם ומציג אותו למשתמש.

בעתיד, ואם נוסיף עוד טפסים שנרצה שיוצגו למשתמש, ניתן יהיה בקלות רבה (בהוספת שורות קוד בודדות) לבצע זאת.

* אופן המימוש:

המחלקה FormDeatils הינה מחלקה שאנו מייצרים ממנה אובייקט באפליקצייה -שמו : r\_FormDeatils.

מחלקה זו מחזיקה בIFormDetails שהוא Interface ו- Details שהוא הנתונים הרלוונטים להצגה.

כאשר משתמש בטופס הראשי לוחץ על פתיחת טופס חבר לדוגמא:

1. מתבצעת הקריאה לr\_FormDeailts.BuildForm עם המחלקה הרלוונטית שממשתמש את הInterface IFormDeatils ועם האוייבקט.

כלומר, בדוגמא זו נשלחת IFormFriend ואובייקט User friendץ

1. המתודה BuildForm תציב בDataMember IFormDeatils את המחלקה המממשת את הinterface (IFormFriend/IFormAlbum) ותקרא למתודה של הInterface הנקראת CreateForm תוך שליחת האוייבקט (User/Album) בהתאמה.
2. המתודה CreateForm תייצר אובייקט Details המתאים לUser/Album בהתאמה , Details היא מחלקה אבסטרקטית הממומשת ע"י FriendDetails/ AlbumDetails ע"פ הנתונים שנרצה להציג עבור כל אחד מאובייקטים אלו.
3. אחרי קבלת האובייקט Details עם הנתונים- בעת הצגת הטופס r\_FormDeatils.ShowDialog נלקחים הנתונים מm\_Details לתוך הLables בטופס ומוצגים.

* 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