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

**Friends With Common Interest:**

פיצ'ר זה מציג רשימה של החברים איתם יש למשתמש באפליקציה עמודי פייסבוק משותפים (עמודים להם המשתמש והחבר עשו "לייק"). ברשימה זו יוצגו כל החברים עם עמוד או יותר משותפים.

הפונקציה שמבצעת זאת מופיעה בקוד ב-MainLogic, וה-form הראשי של האפליקציה עושה בה שימוש בעת לחיצה על כפתור "Click here to see them!".

**Artists Recommendations:**

פיצ'ר זה עושה שימוש ב-API חיצוני של last.fm.

האפליקציה בודקת מי האמנים האהובים על המשתמש, ובעבור כל אחד מהם ממליצה לו על X אמנים דומים (הכמות לבחירתו).

עיקר ההופעה בקוד היא במחלקה LastFmApi, במחלקה קיימת מתודה שמבצעת קריאת API לשרת של lastfm באופן א-סינכרוני. הקריאה מחזירה תשובה בפורמט XML עם מידע על אומנים דומים לאמן המבוקש. מתוך המידע אנו מסננים רק את שמות האמנים ומחזירים רשימה שלהם.

**שימוש ב-Data Binding:**

עשינו שימוש בפיצ'ר זה בטבלת ה-events שלנו. כמו כן קישרנו בין שמות ה-events לחלק מהשדות שלו והם מוצגים בהתאם ל-event הנוכחי שהמשתמש בוחר לראות מידע עליו.

Graphical user interface, application

Description automatically generated

השימוש בקוד הוא במחלקת ה-MainForm, במתודה fetchEvents() שמופעלת בעקבות הלחיצה על כפתור 'Click here to get Upcoming Events'.

**שימוש בתכנות אסינכרוני:**

בקוד יש 2 שימושיים עיקריים בתכנות אסינכרוני:

1. בפיצ'ר Artist Recommendations הנ"ל, המתודה GetArtistRecommendations ב-Façade שיוזכר בהמשך (ובהתאמה המתודה fetchRecommendationsאשר מפעילה אותה) היא מתודה אסינכרונית והיא משתמשת במתודה אסינכרונית ספציפית אחרת שנמצאת במחלקה LastFmApi.

השימוש בה נעשה על מנת לאפשר למשתמש פונקציונליות של שימוש שוטף ורציף בעת טעינת ההמלצות, שלוקח זמן מכיוון והוא תלוי בתשובה של שרת חיצוני (של lastfm).

השימוש נעשה באמצעות async ו-await.

1. ב-MainForm, כלל הפונקציות שתפקידן להביא מידע ולהציג אותו למשתמש עושות שימוש בתכנות אסינכרוני, שכן המידע מיובא משרת חיצוני של פייסבוק. זאת באמצעות שימוש ב-threads – כמובן עם שימוש בפעולות מהצורה Invoke new Action כדי שלא תהיה חפיפה בין ה-threads (Cross thread operation). השימוש בתכנות אסינכרוני באופן דומה נעשה כדי לאפשר למשתמש להינות משימוש שוטף באפליקציה בעת שהנתונים נטענים.

### תבנית מס' 1 – [Template Method]

* **סיבת הבחירה / שימוש בתבנית:**

רצינו לאפשר סדר הצגת תמונות מתוך אלבום בכל מיני צורות ולכן בחרנו בפטרן Template Method שמאפשר לנו להכניס צורות וסדר הצגה שונים לאלבומים, ע"י הגדרה מחדש של מקומות באלגוריתם ללא שינוי במבנה שלו.

כרגע, אנחנו מציגים אלבום בצורה רנדומלית (כל פעם מגרילים תמונה מתוך רשימה), אם בעתיד נרצה להוסיף שיטות סידור נוספות (לדוגמה לצפות בתמונות לפי תאריך הוספתן) לא נצטרך לשנות את הקוד ולכן פטרן זה תורם לתחזוקתיות הקוד.

* **אופן המימוש:**

יצרנו מחלקה אבסטרקטית בשם .PhotoSelectionAlgorithmBase

למחלקה הזאת יש מטודה של סידור עם injection point אשר המחלקה היורשת ממנה ( PhotoRandomizer) מיישמת את שיטת סידור התמונות, כך המחלקה אחראית לסידור התמונות ומחזירה לקליינט את התוצאה הממויינת.

שחקנים**:**

Client - FacebookUserProxy

Abstract Class - PhotoSelectionAlgorithmBase

Subclass - PhotoRandomizer

* Sequence Diagram

Diagram, schematic

Description automatically generated

* Diagram

  Description automatically generatedClass Diagram

### תבנית מס' 2 – [Strategy]

* **סיבת הבחירה / שימוש בתבנית:**

אנו רוצים בזמן ריצה אפשרות להחליף את שיטת המיון של רשימת השמות של האומנים הדומים ביותר לאומנים האהובים על המשתמש. התבנית מאפשרת למשתמש לבחור בזמן ריצה את הקריטריון לפי 3 סוגי מיון: סדר אלפביתי מהקטן לגדול/מהגדול לקטן/ לפי אחוזי התאמה בין האומנים.

לוגיקת ההשוואה ממומשת במחלקה נפרדת שניתן להחליף באופן דינאמי תוך כדי ריצת התכנית.

כמובן שאם בעתיד נרצה למיין רשימת אומנים באופנים נוספים, נצטרך רק להוסיף מחלקה חדשה שמממשת סוג מיון חדש לתבנית ה-.strategy

* **אופן המימוש:**

יצרנו אינטרפייס בשם IArtistSortStrategy, אותו תהיה חייבת לממש כל מחלקה שמוסיפה שיטת מיון למערכת. במקרה שלנו, מממשות אותה RatingArtistSortStrategy, AscendingArtistSortStrategy, DescendingArtistSortStrategy.

ב RecommendationsFacade יש מופע של IArtistSortStrategy והמימוש שלו נבחר בזמן ריצה (כלומר הnew על המחלקה שמממשת ).

המשתמש בוחר את הקריטריון ב )formע"י בחירה מתוך ComboBox) ,הבחירה מועברת לפונקציה GetArtistRecommendationsמה-UI ולפיה משתמשים באחד מה-concrete strategies בתור שיטת המיון ש-sort מפעיל (המתודה setSortStrategy שמופעלת תחילה יוצרת את מחלקת המיון בהתאם לשיטת המיון שבנבחרה).

שחקנים:

Context-RecommendationsFacde

Strategy-IArtistSortStrategy

ConcreteStrategies:

AscendingArtistSortStrategy

DescendingArtistSortStrategy

RatingArtistSortStrategy

Sequence Diagram:

Diagram, schematic

Description automatically generated

Diagram

Description automatically generatedClass Diagram:

### תבנית מס' 3 – [Observer]

* סיבת הבחירה / שימוש בתבנית:

רצינו להגדיר תלות של אחד לרבים בין MainForm (אחד) למחלקות שמחזיקות מופע של MainForm ( רבים – אם בעתיד עוד מחלקות יחזיקו מופע של טופס זה) ורוצות להאזין ולבצע מתודה כלשהי כאשר MainForm נסגר, כרגע למשל לFormFlowManager יש מופע של MainForm .

כרגע FormFlowManager מעוניין בתור Observer לבצע מתודה בזמן בו MainForm נסגר , ולכן הוא רושם מתודה זאת אל MainForm.

כאשר MainForm נסגר – הוא מעדכן את המחלקות המאזינות והן מפעילות מתודה בהתאם לממשק הIExitObserver.

ביצענו שימוש בתבנית לסייע בתחזוקתיות, אם בעתיד תיהיה מחלקה נוספת שנרצה שתעודכן כאשר MainForm נסגר – כלומר ניהיה מעוניינים שהיא תפעיל מתודה כלשהי , אז נוסיף אותה לרשימת המאזינים לMainForm באמצעות Attach , והיא רק תצטרך לממש את המתודה המתאימה שמוגדרת בממשק IExitObserver – השינוי בקוד יהיה רק במחלקה הנוספת שמעוניינת להיות Observer לMainForm.

* אופן המימוש:

יצרנו ממשק בשם IExitObserver, אשר המחלקות שמאזינות לMainForm מממשות (כרגע FormFlowManager) – כלומר יש להן מתודה UpdateExited.

הוספנו מתודות Attach,Detach,notify לMainForm.

בנוסף ל MainForm הוספנו ממבר של Action אליו מוסיפים Observers באמצעות Attach וניתן לבטל Observers באמצעות Detach. המתודה notify

מבצעת Invoke ותפעיל את המתודות של הObservers כשהטופס נסגר.

שחקנים:

Subject – MainForm

IObserver(the observer interface) - IExitObserver

Observer - FormFlowManager

* Sequence Diagram

Diagram, schematic

Description automatically generated

* Class Diagram

Diagram

Description automatically generated