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

**הסבר על הפיצ'רים:**

**פיצ'ר 1:** מיון אלבומים לפי מספר תמונות באלבום.

הסבר: יצרנו מערך של אלבומים שגודלו כגודל מספר האלבומים שיש למשתמש.

לאחר מכן רצנו על האלבומים של המשתמש והוספנו למערך את האלבום בעל מספר התמונות הגדול ביותר. מימשנו את המיון בעזרת אלגוריתם bubble sort, בהנחה שכמות האלבומים הוא מספר שחסום על ידי 1000, ולכן זמן הריצה לא נפגע.

בעתיד השאיפה למיין גם לפי מספר Likes או מספר comments.

את הפיצ'ר אפשר למצוא לאחר שהשתמש לוחץ על כפתור Albums ולאחר מכן על Sort By, האלבומים יופיעו ממוינים לפי כמות התמונות בכל אלבום.

בקוד הפיצ'ר נמצא תחת AlbumsOps בתוך פונקציית SortLabel\_Click().

**פיצ'ר 2:** משחק הניחושים.

הסבר: משחק- נחש את הגיל של החברים שלך.

כאשר המשתמש לוחץ על כפתור Games, מופיע לו משחק הניחושים.

המערכת מגרילה חבר ראשון, ומופיעה שאלה – מה הגיל של החבר \_\_? (הערה: באפליקציה השאלה- מה הגיל של גיא רונן? התשובה 45)

על המשתמש לענות ולאחר מכן מופיע לו האם צדק או טעה, לאחר מכן מוגרל חבר נוסף, ככה המשתמש יכול לאסוף נקודות עד שנגמר המשחק.

את הפיצ'ר ניתן למצוא תחת כפתור Games. ובקוד ניתן למצוא את הפיצ'ר תחת GamesOps.

**פיצ'ר 3:** רשימת ספורט.  
זהו פיצ'ר שהוספנו למטלה מספר 2 על מנת שנוכל לממש ולהשתמש בתבניות הנדרשות.  
יצרנו פיצ'ר שמטרתו לעזור למשתמש לנהל את המטלות בספורטיוביות שהוא רוצה לבצע בעתיד הקרוב. המשתמש יכול להוסיף פעילות לרשימה, להוסיף לה תאריך יעד, לסמן שהפעילות נעשתה. המידע של המשתמש נשמר גם במעבר בין הדפים השונים. במידה והמשתמש מבצע פעילות ומסמן על כך, קופצת התראה המעודדות אותו להמשיך!

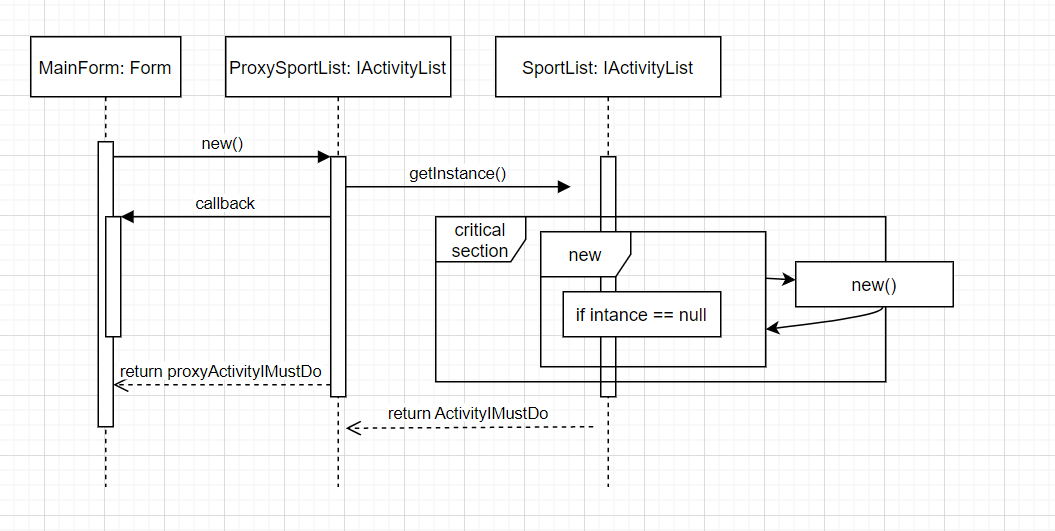
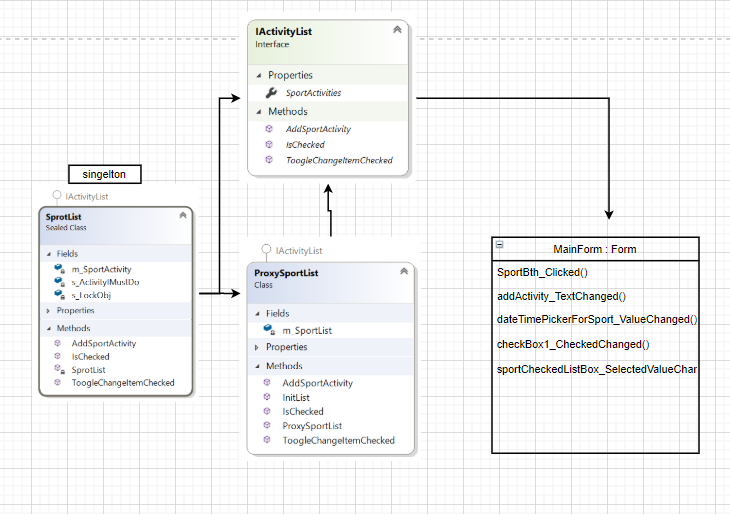
### תבנית מס' 1 – Singleton ממשפחת Creational -

* **סיבת הבחירה / שימוש בתבנית:**

הפיצ'ר sportList מאפשר למשתמש לצפות בפעילויות הספורט שהוא רוצה לבצע בזמן מסויים. בבניית הפיצ'ר רצינו שכל המידע יוחזק במקום אחד בלבד וכי כל פעם שנוסיף פעילות לרשימה, היא תישמר בפעיליות המשתמש. כלומר, רצינו לבטל את האופציה לclient ליצירת אובייקט חדש של רשימה שכן, במידה והclient יוכל ליצור אובייקט חדש של הרשימה, הרשימה תהיה ללא כל הדברים החדשים שנוספו.   
בנוסף, אנו רוצים שהפעילויות של המשתמש יוחזקו במופע יחיד של המחלקה sportList כדי שכל המידע שלו ינוהל בצורה אחידה ובכדי שלא נצטרך לנהל סינכרון בין מופעים שונים של רשימות ספורט.

* **אופן המימוש:**

מחלקת SportList היא מחלקה שחתימתה היא public sealed, במחלקה זו כל השדות הם סטטיים. אחד מהשדות במחלקה הוא שדה מסוג SportList. המחלקה הינה חתונה בsealed כיוון ואנו רוצים לוודא שלא ירשו מחלקה זו. במידה וירשו יכול להיווצר יותר מאובייקט אחד מסוג זה וזוהי סתירה לכוונה שלנו.   
בקריאה הראשונה לשדה הInstance, השדה SportList מאותחל בפעם הראשונה ובכל אחת מהקריאות שלאחר מכן, מבוצעת בדיקה האם האובייקט אותחל, במידה וכן, נחזיר את האובייקט הקיים. בנוסף, מכיוון וקיימים מספר threads במערכת שבנינו ומכיוון ואנו רוצים להשאיר את המערכת פתוחה להתרחבות בעתיד, מימשנו את singleton sportList באמצעות thread safe, והשתמשנו בדאבל checkLock על מנת שלא יווצר יותר ממופע אחד.

* **Sequence Diagram**
* **Class Diagram**

### תבנית מס' 2 – Fectory Method ממשפחת Creatinal -

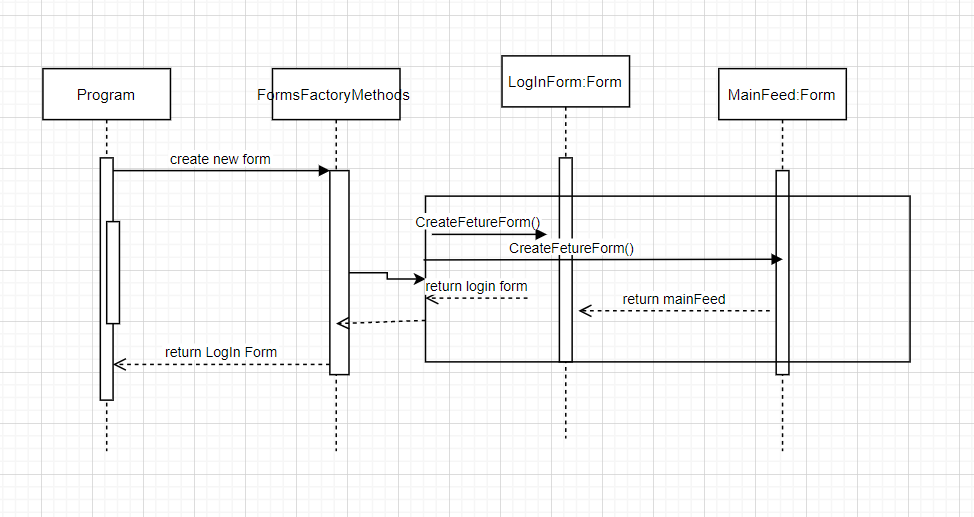
* **סיבת הבחירה / שימוש בתבנית:**

המטרה של תבנית זו, היא היכולת לממש את יצרת האובייקט במחלקה נפרדת שלא תלויה בclient. המחלקה שבנינו מחזיקה מתודה בשם CreateFetureForm. מתודה זו מחזירה Form חדש ובעצם מפרידה בין הclient לבין הלוגיקה. מחלקה זו מקלה על הclient ביצירת האובייקט ומאפשרת לקנפג (config) את הforms החדשים שיווצרו בעתיד בשלב הלוגי ובמידה ונרצה לעשות שינויים נוכל ללכת למחלקה זו ולא נצטרך לשנות את הקוד בclient.

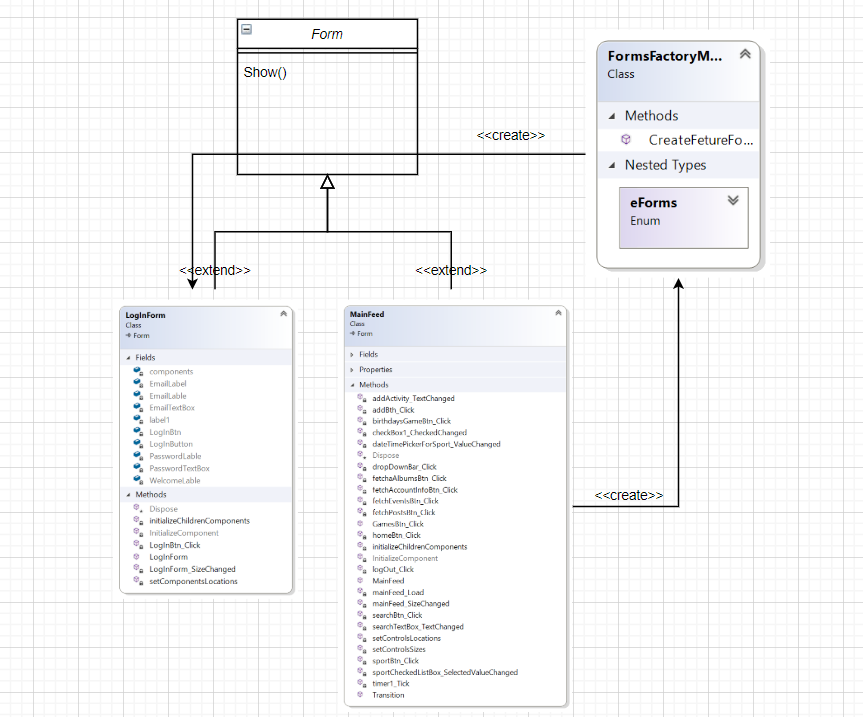
* **אופן המימוש:**

במערכת שלנו ישנה מחלקה בשם FormsFactoryMethods.בתוך המחלקה ישנה מתודה אשר eNum אשר מכילה את שני סוגי הforms שהמערכת מאפשרת לייצר. (Login and mainform). בתוך מחלקת Program במקום לעשות new login form בעצם מתצבעת קריאה למחלקת FormsFactoryMethods ושם נוצר הform הנדרש.  
בנוסף, לאחר יצירת form זה, מתבצעת קריאה נוספת על מנת ליצור את mainForm.  
נשים לב כי בעתיד נוכל להוסיף forms חדשים למערכת על ידי הוספה בeForm ושינוי בתוך המחלקה FormsFactoryMethods. כמו כן, לא נצטרך לגשת לקוד של הclient ולשנות שם דברים לוגים על מנת להוסיף טפסים למערכת שלנו.

* **Sequence Diagram**



* **Class Diagram**



### תבנית מס' 3 – Proxy ממשפחת Structural -

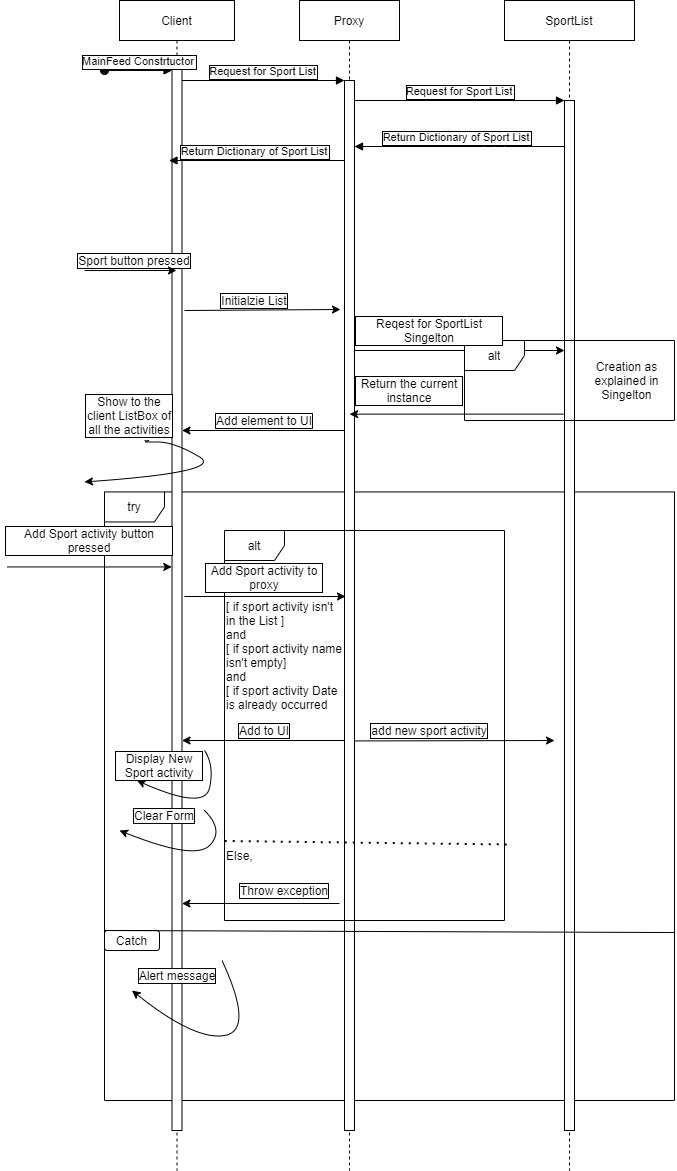
* **סיבת הבחירה / שימוש בתבנית:**

תבנית זו מאפשרת לעטוף את המחלקה שנמצאת מאחורי הקלעים ובעצם משמשת כמתווכת בין הclient לבין מחלקה זו. בנוסף, במחלקת proxy ניתן להוסיף לוגיקה שתוודא את הקלט ותמנע מעבר של קלטים שגויים למחלקה המוגנת.   
למעשה כשהclient יקרא למחלקת ProxySportList הוא אינו ירגיש הוא מדבר עם מתווך אלא ירגיש שהוא מדבר עם מחלקה מסוג IAtiviryList ובמקרה שלנו sportList.  
במידה ובעתיד נרצה לבצע שינויים בפיצ'ר ובבדיקות על הקלטים, נוכל לבצע זאת ישירות דרך מחלקה זו ומבלי לגעת ולשנות קוד בclient.

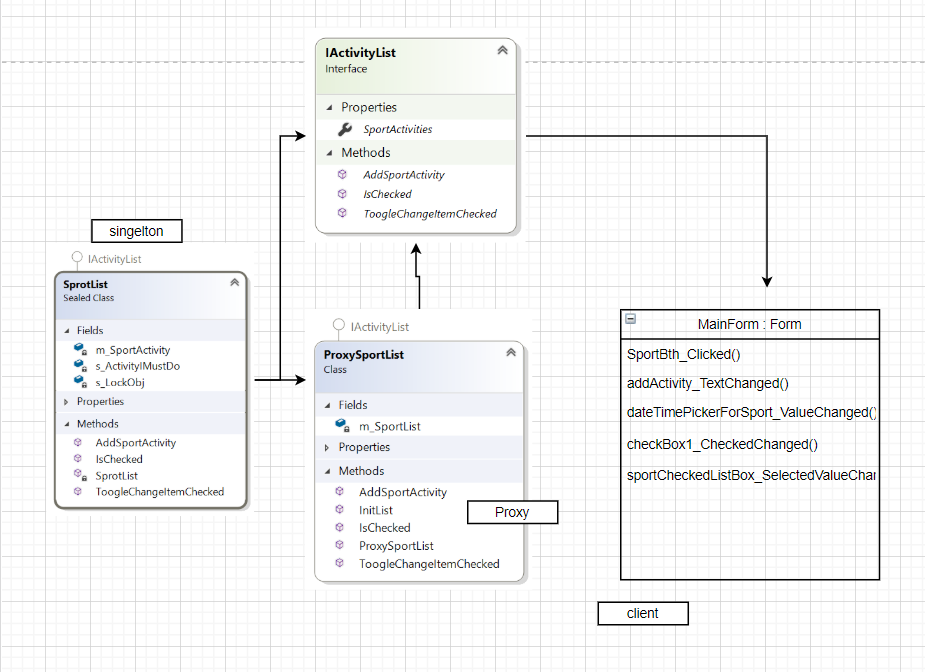
* **אופן המימוש:**

[במערכת שלנו בנינו מחלקה שנקראת porxySportList, מחלקה זו מממשת את הInterface שנקרא IActivtyList. הinterface מחזיק מתווה לממש שעליו להחזיק מילון אשר המפתח שלו הוא string והערך הוא מסוג sportActivity אשר יצרנו, בנוסף 3 מתודות אשר מאפשרות יצירה והוספה של פעילות ספורט לרשימה, בדיקה האם פעילות ספורט בוצעה ושינוי האם פעילות ספורט הקיימת ברשימה בוצעה או לא.   
מחלקת proxtSportList מממשת את הinterface ומוסיפה שכבת הגנה לוגית של בדיקות על כל המתודות שבמימוש. כלומר, הclient יוכל לשלוח את הנתונים למחלקת proxySportList ולקבל התראה במידה ואחד הנתונים שגוי/קיים/לא מתאים.

* Sequence Diagram

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

* **Class Diagram**



**Use of Multy Threading-**  
  
אנו משתמשים בmulty threading ישר לאחר הLogIn של המשתמש.   
בתוך הקונסטרקטור של מחלקת mainForm ישנה קריאה של new thread להבאת האלבומים של המשתמש. כלומר, כאשר המשתמש מתחבר למערכת שלנו, ולמעשה אנו יודעים מי המשתמש שלנו, נרצה לשפר את חווית המשתמש ולחסוך לו זמני המתנה לא רצויים בטעינת עמודים כבדים בעתיד כמו עמוד האלבומים. כל בקשה של המשתמש יוצרת פנייה למידע שנמצא אצל פייסבוק, זוהי פניה יקרה שיכולה לקחת זמן רב ולכן אנחנו מבקשים את הנתונים בעת כניסת המשתמש פעם אחת בלבד על ידי תהליכים שונים. נתונים אלו נשמרים במשתנה סטטי במחלקת albumOps.  
נשים לב כי המשתמש עדיין לא ביקש לצפות באלבומים שלנו, אבל במידה ויבקש הנתונים כבר נמצאים אצלנו במערכת. הנתונים נאספו בתהליכים נפרדים ולכן כאשר המשתמש יבקש לצפות באלבומיו, הנתונים יוצגו בצורה מהירה מספיק.

**Use of Data Binding –**

מימשנו two way data binding באמצעות vs data source כפי שנלמד בכיתה.   
לאחר שלוחצים על כפתור sport, מוצגות בפנינו רשימת הפעילות הספורטיביות שעלינו לבצע. בעת לחיצה על פעילות כלשהיא מהרשימה, יופעל האירוע שנקרא sportCheckedListBox\_SelectedValueChanged ולמעשה קושר את האובייקט שנבחר לבין רכיב הUI שמציג את השדות של האובייקט שזה עתה נבחר מתוך הרשימה.   
חלון הUI שמציג את הנתונים קשור לאובייקט באופן שאם נשנה את הנתונים בחלון זה, השדות של האובייקט ישתנו ולהיפך.