**Facebook For Desktop – Design Patterns Ex03**

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

* User data analyzer – **ניתן למצוא אותו במחלקת UserDataAnalyzer**

פיצ'ר זה מלקט מגוון נתונים סטטיסטיים אודות המשתמש המחובר לאפליקציה.

באמצעותו המשתמש יכול להסיק מסקנות ולקבל אינדיקציה לגבי רמת הפופולריות / חשיפה ורמת הפעילות שלו לאורך כל השנים שבהן המשתמש רשום לפייסבוק.

פיצר זה מחולק ל-3 דיאגרמות ועם מידע זה המשתמש יוכל לשמר/לשפר מגמות שונות בהתאם לסטטיסטיקה ולהבין טוב יותר את אופן התנהגותו בפייסבוק.

* Post by category- **ניתן למצוא אותו במחלקת PostByCategory**

פיצ'ר זה מאפשר למשתמש לפרסם פוסט שמיועד רק לקבוצת חברים ספציפית.

הפיצר מקטלג את קבוצות החברים לפי הקטגוריות הבאות: חברים מאותה עיר, חברים שנולדו באותו חודש, חברים מאותו מוסד לימודי, חברים עם אותו ניסיון תעסוקתי, ויתר החברים.

המשתמש יכול לבחור קבוצת חברים ולכתוב להם סטאטוס ו/או להעלות תמונה ממחשבו האישי .

המשתמש גם יכול לראות מי הם החברים באותה קטגוריה טרם פרסום הפוסט.

### תבנית מס' 1 – Template Method

* **סיבת הבחירה / שימוש בתבנית:**

הסיבה שבחרנו להשתמש בפטרן זה היא שזיהינו שכפול קוד מסוים במימוש הפיצ'ר Post by category. היו לנו מספר מתודות שמבצעות השוואה בין שני משתמשי פייסבוק על פי קטגוריות שונות: יום הולדת, ניסיון תעסוקתי, לימודים ועיר מגורים. בכל המתודות הללו ההשוואה התבצעה באופן דומה, ולכן נוצר שכפול קוד. כדי להימנע משכפול זה השתמשנו בפטרן הנ"ל – הגדרנו מחלקת אב אבסטרקטית עם מתודה שמכילה פעם אחת את הקוד שחזר על עצמו, ובנים שיורשים מהאב ומממשים באופן ספציפי את לוגיקת ההשוואה שלהם.

בנוסף, פטרן זה העניק תחזוקתיות בפיצ'ר הזה. שהרי אם בעתיד נרצה להוסיף קטגוריה נוספת שעל פיה נשווה בין שני משתמשים, אז רק נצטרך להוסיף עבורה מחלקת בן שיורשת מה-Base Class ומממשת את ההשוואה באופן ספציפי. לא נצטרך לשנות את הקוד הקיים בכלל.

* **אופן המימוש:**

יצרנו מחלקת אב אבסטרקטית – ComparatorByCategory. מחלקה זו מכילה מתודה בשם CompareByCategory אשר כל בן מקבל בירושה והיא מכילה את קטע הקוד המשותף לכל הבנים –

JobComparator

BirthdayComparator

EducationComparator

CityComparator

בנוסף מחלקת האב מכילה מתודה protected abstract בשם NeedCompare, והיא מממומשת באופן ספציפי על ידי כל בן. אנו מבצעים קריאה לNeedCompare בתוך המתודה CompareByCategory, ולכן NeedCompare מהווה Injection Point – כלומר בזמן ריצה היא מופעלת אצל הבן המפעיל בלבד.

ניתן למצוא מימוש זה במחלקות:JobComparator, CityComparator,

EducationComparator, ComparatorByCategory BirthdayComparator,.

* **Sequence** **Diagram**

**Diagram

Description automatically generated**

* **Class** **Diagram**

**Graphical user interface, text, application

Description automatically generated**

**ההקבלה לPattern-:**

מחלקת ComparatorByCategory – המקבילה שלה בפטרן זה היא ה-,Mecanism כלומר מחלקת ה- Base Class.

BirthdayComparator, CityComparator, EducationComparator, JobComparator- מחלקות אלו מייצגות את הבנים שיורשים מה- Base Class.

### תבנית מס' 2 – Observer

* **סיבת הבחירה / שימוש בתבנית:**

מימשנו מעין ערכת צבעים עבור האפליקציה (Dark mode, Bright mode), וזיהינו צורך בשימוש בפטרן Observer כחלק מפיצ'ר זה. כאשר המשתמש משנה את ערכת הצבעים, אז הצבע של הפאנל הראשי והצבע של פאנל התפריט (וכל הכפתורים שעליו) צריכים להשתנות בהתאם. לכן התייחסנו אל הפאנלים האלו כמאזינים שרוצים לקבל דיווח על אירוע של שינוי ערכת הצבעים, ולהגיב אליו.

כלומר מימשנו את הפאנלים כObservers- שמאזינים למחלקה שתפקידה להתריע (Notify) על האירוע. כך המחלקה שמדווחת על האירוע (מחלקת ה-Subject) לא מכירה באופן ספציפי את המאזינים שלה, אלא רק בתור כאלו שמממשים ממשק מסוים. דבר זה מעניק גם תחזוקתיות – אם בעתיד נרצה שרכיבים נוספים ישנו את צבעם כחלק מפיצ'ר זה אז נוכל לעשות זאת בקלות – הרכיב החדש רק יצטרך לממש את הממשק ולהירשם כמאזין לאירוע. כלומר לא נצטרך לשנות את הקוד הקיים, אלא רק להוסיף רישום חדש.

* **אופן המימוש:**

מימשנו מחלקת ThemeSubject שתפקידה לדווח לכל המאזינים שלה על אירוע של שינוי צבע. היא מכילה רשימת מאזינים, וכל רכיב שמממש ממשק מסוים (IThemeSubject) ורוצה להירשם כמאזין צריך לקרוא למתודה AttachObserver,

ויכול להסיר עצמו מההאזנה בעזרת DetachObserver.

יצרנו ממשק IThemeSubject עם מתודה אחת שה-Observers מממשים.

ה – Observers במימוש שלנו הם המחלקות:

MainPanelObserver – הפאנל הראשי.

MenuPanelObserver – פאנל התפריט.

כשהאירוע של שינוי צבע מתרחש, אז ה-ThemeSubject מדווח זאת לכל המאזינים בעזרת המתודה notifyThemeObservers. במהלכה הוא מפעיל את המתודה של הממשק (UpdateThemeColor) אצל כל המאזינים שלו והם מגיבים בהתאם לאירוע – משנים את הגוון שלהם במידה מסוימת.

ניתן למצוא מימוש זה בכל המחלקות שציינו:

ThemeSubject, MainPanelObserver, MenuPanelObserver, IThemeSubject.

* Diagram

  Description automatically generated**Sequence** **Diagram**
* A picture containing graphical user interface

  Description automatically generated**Class Diagram**

**ההקבלה ל-Pattern:**

מחלקת ThemeSubject – מחלקה זו מייצגת את ה- Subjectבפטרן זה (ה-Notifier) שמכיל רשימת listeners ומדווח להם על אירוע מסוים שמתרחש.

ממשק IThemeObserver – מייצג את הממשק IObserver בפטרן זה. מחלקות ה-Observers מממשות אותו ומחלקת ה-Subject מכילה רשימת מאזינים מטיפוס ממשק זה.

מחלקות MenuPanelObserver, MainPanelObserver – מחלקות ה- Observers (ה-listeners) שמממשות את הממשק הנ"ל ומאזינות לאירוע שקורה ב-Subject.

### תבנית מס' 3 – Strategy

* **סיבת הבחירה / שימוש בתבנית:**

בחרנו לממש פטרן זה כי זהינו שיש שכפול קוד במחלקת UserDataAnalyzer .

היו לנו מטודות שמבצעות איסוף סטטיסטיקה לגבי תגובות ולייקים של המשתמש וראינו שאופן הפעולה שלהם היה דומה למעט נקודת שוני אחת בינהם.  
לכן בכדי להימנע משכפול קוד בין מטודות אלו החלטנו להיעזר בפטרן Strategy .   
את קטע הקוד המשותף שמנו במטודה אחת במחלקה שמייצגת את ה-Mechanism ואת השוני בין המימושים הוצאנו למחלקות נפרדות .

מחלקות אלו מימשו את ממשק ה-IStrategy .

כך חסכנו את שכפול הקוד והקטנו את הפוטנציאל לבאגים בפיצ'ר של איסוף הסטטיסטיקה.

כמו כן, שימוש בפטרן זה העניק תחזוקתיות מכיוון שאם בעתיד נממש איסוף סטטיסטיקה על פריט נוסף אז רק נוסיף מחלקת אסטרטגיה עבורו וקוד המקור במחלקת ה- Mechanism לא ישתנה.

* **אופן המימוש:**

מימשנו את פטרן Strategy בהתאם לתפקידים המוגדרים בו :

ה-Mechanism מיוצג ע"י מחלקת UserDataAnalyzer .  
מחלקה זו מכילה מופע של הממשק IStrategyAnalyzer .

יש בה מטודה GetUserStatsAtSelectedYearשמשתמשים בה מספר רכיבים (כמו ה-CommentsAnalyzer,LikesAnalyzer) באותו האופן ובתוכה יש Injection Point שכל רכיב ממש באופן ספציפי בהתאם למידע שאותו הוא רוצה לנתח.

בכדי להחליף בין האסטרטגיות השונות אפשרנו לבצע פעולת Set strategy ע"י המטודה SetStrategyAnalyzerאשרממומשת במחלקת UserDataAnalyzer.

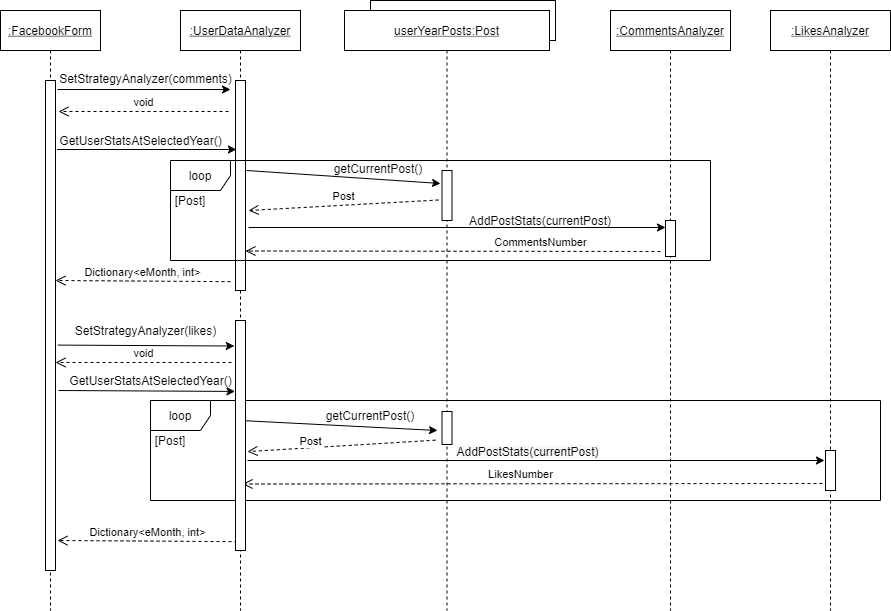
כמו כן, מימשנו ממשק IStrategyAnalyzer שממומש ע"י מחלקת ה-Concrete Strategy

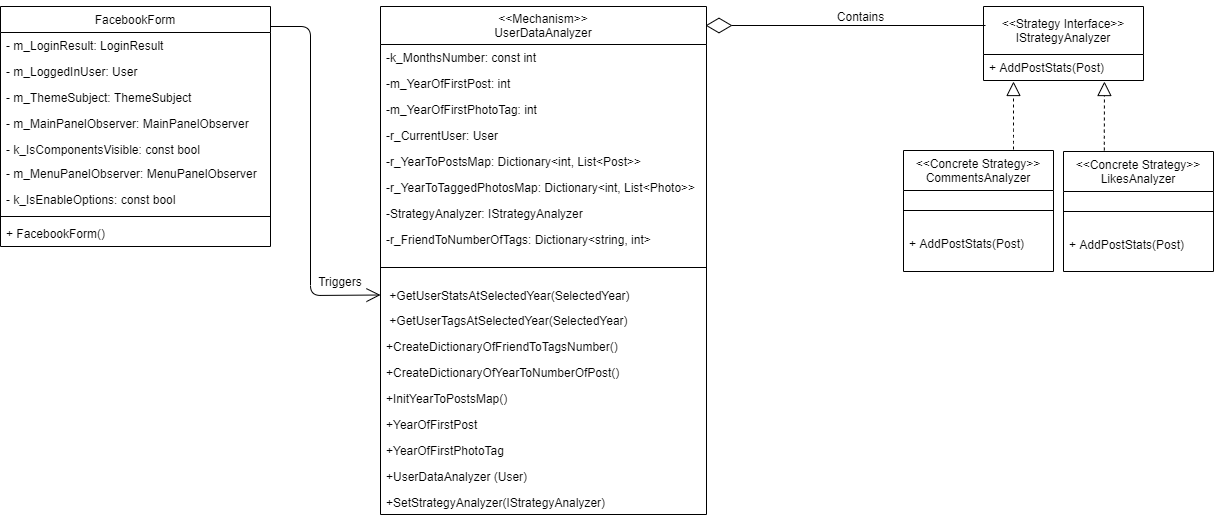
(CommentsAnalyzer,LikesAnalyzer).  
כל אחת מהן מימשה את האסטרטגיה לפי הלוגיקה שלה.

ניתן למצוא מימוש זה בכל המחלקות שציינו:

LikesAnalyzer, CommentsAnalyzer, UserDataAnalyzer, IStrategyAnalyzer.

* **Sequence Diagram**



* **Class Diagram**

**ההקבלה ל-Pattern:**

מחלקת UserDataAnalyzer– מחלקה זו מייצגת את ה-Mechanism (Context) והיא מחזיקה רפרנס למופע של הממשק IStrategy .

ממשק IStrategyAnalyzer– מייצג את הממשק IStrategy בפטרן זה.

מחלקות LikesAnalyzer, CommentsAnalyzer – מחלקות ה- Concrete Strategy שמממשות את הממשק הנ"ל .