SmartBiz

# מסמך תכנון ועיצוב

גרסה 1.0

**שמות הסטודנטים**:

רותם אלישדה

עומר בר-און

# תוכן עניינים

## תיאור כללי של המערכת

המערכת בנויה על סמך מודל 3-Tier, שבו המערכת מופרדת לשלוש שכבות:

- שכבת בסיס הנתונים (Database Layer)
- שכבת הלוגיקה (Business Logic Layer)
- שכבת התצוגה (Presentation Layer)

כל שכבה מתקשרת עם השכבה מעליה באמצעות ממשק קבוע מראש.
ההפרדה לשכבות מאפשרת לנו להחליף בעתיד את הממשק בין שתי שכבות מבלי להחליף את הממשק בין שתי השכבות האחרות, וכן, להחליף את המימוש של אחת השכבות מבלי להחליף את המימוש של השכבות האחרות, ובכך לחסוך בשכתוב קוד קיים, ולאפשר גמישות לשינויים.

## הנחות עבודה בכתיבת הפרויקט

1. בכל רגע מחובר רק משתמש אחד במערכת.
2. לא ניתן למחוק מהמערכת משתמשים/לקוחות בכדי למנוע מצבים של חוסר-עקביות בנתונים, לכן בכדי לממש "הסרה" שלהם, השדה הבוליאני is_active ישנה ערכו לערך האמת false.
3. שם פרטי ושם משפחה מורכבים ממילה אחת בלבד ללא רווחים.
4. שם מלא מורכב משם פרטי ושם משפחה עם רווח ביניהם.
5. שם משתמש מכיל אותיות בלבד.
6. במערכת זו, לכל מוצר ושירות קיים מחיר בתווך 0 עד 1,000,000. מחיר זה לא כולל את המע"מ ואת ההנחות התקפות. חישוב ההנחה מתבצע **לאחר** הוספת המע"מ.
7. המערכת מאפשרת יצירה של עד 1,000 סוגי מוצרים שונים במלאי ועוד 1,000 סוגי שירותים שונים. המספר הסידורי של סוג המוצר נע בתווך 1,000 – 1,999. המספר הסידורי של סוג השירות נע בתווך 2,000 – 2,999.
8. כל המודולים במערכת שעובדים עם תאריכים מצפים לערכים בפורמט: dd/MM/yyyy. לכן צריך להתאים את הגדרות התאריכים במערכת ההפעלה בהתאם (מידע נוסף על כך נמצא במדריך למשתמש).
9. מערכת זו איננה תומכת בהתקשרות מול ספקים חיצוניים.

**מוסכמות רישום**

1. כל שם מחלקה/שיטה/שדה שכוללת יותר ממילה אחת, נכתבת עם אות גדולה בתחילת כל מילה נוספת.
2. כל מחלקה מתחילה באות גדולה. לממשקים נוספת האות I לפני שם המחלקה.
3. מחלקות וממשקים שהינם מהווים משתתפים בדפוס עיצוב כלשהו, אך אינם מייצגים אובייקט קונקרטי, יקבלו את שם המשתתף בדפוס העיצוב.
4. כל שדה (Member) פרטי במחלקה מתחיל ב-'_'.
5. קבועים ו-enum-ים מכילים אותיות גדולות בלבד.
6. פרמטרים לשיטה או משתנים בתוך שיטה מתחילים באות קטנה.
7. שיטות שהן ציבוריות מתחילות באות גדולה, ושיטות פרטיות (או מוגנות) מתחילות באות קטנה.
8. כל פקד בטופס מתחיל בשם (עם אות קטנה), ואח"כ בסוג הפקד כאשר כל שתי מילים מופרדות ע"י '_'.

## שכבת בסיס הנתונים (SmartBiz.Dal namespace)

שכבה זו אחראית על התקשורת מול מסד הנתונים, וביצוע הפעולות השוטפות מולו, כגון שמירה, עדכון, הסרה, ושליפת נתונים. המערכת עושה שימוש במסד הנתונים Microsoft SQL Server 2008.

שכבה זו מכילה את בסיס הנתונים smartbiz.mdf.

שכבה זו מכילה מחלקה בשם DBManager אשר מכילה רק שיטות סטטיות. מחלקה זו אחראית ליצור את ההתקשרות מול מסד הנתונים באמצעות אובייקט מסוג SqlConnection כאשר ה-connectionString מוגדר בקובץ App.config. בנוסף, מחלקה זו אחראית לשלוף נתונים באמצעות שאילתות מסוג Select ולעדכן נתונים באמצעות שאילתות מהסוגים: Insert, Update ו-Delete.

מחלקה זו מכילה את השיטות הבאות:

InitSqlConnection – אתחול האובייקט SqlConnection ופתיחת ההתקשרות מול מסד הנתונים.

CloseSqlConn – ניתוק ההתקשרות מול מסד הנתונים.

QueryCmd – שיטה המקבלת כפרמטר מחרוזת המייצגת שאילתה מסוג Select. השיטה שולחת את השאילתה למסד הנתונים ומחזירה אובייקט מסוג DataTable המכיל את תוצאות השאילתה.

NonQueryCmd – שיטה המקבלת מחרוזת המייצגת שאילתה מסוג Insert, Update או Delete. השיטה שולחת את השאילתה למסד הנתונים ומחזירה ערך בוליאני המציין האם הפעולה בוצעה בהצלחה.

ScalarCmd - שיטה המקבלת מחרוזת המייצגת שאילתה מסוג Select הכוללת שימוש באחת מהפונקציות הבאות: Avg, Count, Sum, Min, Max. השיטה מחזירה מחרוזת המייצגת את הערך המוחזר מהשאילתה.

בכדי לעשות שימוש בשכבת בסיס הנתונים, ראשית יש לאתחל את ההתקשרות מול בסיס הנתונים. דבר זה מתבצע בבנאי של הטופס הראשון המופיע במערכת (טופס ה-Login). הבנאי של הטופס מאתחל את ההתקשרות מול בסיס הנתונים באמצעות פניה למחלקה DBController שנמצאת בשכבת ה-Bl. מחלקה זו יוצרת את ההתקשרות ע"י קריאה לשיטה הסטטית InitSqlConnection במחלקה DBManager.

השכבה ממומשת באופן מודולארי כך שניתן יהיה בעתיד להחליף את בסיס הנתונים לאחר, על ידי ביצוע שינויים מינימאליים ביותר, וכל זאת מבלי לשנות את פרטי מימוש השכבות האחרות.

עקרונות מנחים לגבי שכבת בסיס הנתונים:

- הטבלאות מעוצבות בצורת BCNF (Boyce-Codd normal form).
- הטבלאות עוצבו כך שתיווצר כפילות מינימאלית ככל האפשר של נתונים הנשמרים בהן.
- בחרנו לממש את ההתקשרות למסד הנתונים בשיטה המקושרת (Full Connected). כלומר החיבור בין היישום למסד הנתונים הינו קבוע. בחרנו לממש את ההתקשרות למסד הנתונים בשיטה זו מכיוון שעדכניות המידע חשובה למשתמשי המערכת.
- החיסרון הוא ששיטה זו יוצרת עומס על השרת שבו נמצא מסד הנתונים.
- היתרונות של שיטה זו הינם:
  - תמיד עובדים עם המידע העדכני ביותר.
  - כמעט ואין צורך לטפל בקונפליקטים המתרחשים בין לקוחות שונים המטפלים באותו המידע.
  - הפעולות מול מקור המידע מתבצעות במהירות.

### טבלת משתמשים (Users)

טבלה זו מכילה מידע אודות המשתמשים הקיימים במערכת. הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| username | מחרוזת | שם משתמש. מפתח ראשי |
| first_name | מחרוזת | שם פרטי |
| last_name | מחרוזת | שם משפחה |
| is_manager | בוליאני | הרשאת מנהל? |
| is_teamleader | בוליאני | הרשאת ראש צוות? |
| is_salesmanager | בוליאני | הרשאת מנהל מכירות? |
| is_worker | בוליאני | הרשאת עובד? |
| password | מחרוזת | סיסמא |
| is_active | בוליאני | האם פעיל במערכת? |
| is_must_change_password | בוליאני | האם המשתמש חייב לבצע שינוי סיסמא בהתחברות הבאה למערכת? |

### טבלת הגדרות כלליות (Global Settings)

טבלה זו מכילה מידע אודות הגדרות כלליות הקיימות במערכת. כל שורה בטבלה היא הגדרה המורכבת ממפתח וערך של המפתח. הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| [key] | מחרוזת | הגדרה במערכת. מפתח ראשי |
| value | מחרוזת | ערך עבור ההגדרה |

בטבלת ההגדרות הכלליות ישנן 8 הגדרות קבועות, עבורן יש לקבוע ערכים. להלן ההגדרות:

| שדה | הסבר |
|---|---|
| vat | אחוז המע"מ |
| coin_type | סוג המטבע |
| company_name | שם החברה |
| company_address | כתובת החברה |
| company_phone | טלפון החברה |
| module | שם הקובץ של המודול שנטען |
| open_at | שעת הפתיחה של העסק |
| close_at | שעת הסגירה של העסק |

### טבלת לקוחות (Customers)

טבלה זו מכילה מידע אודות לקוחות בית העסק. הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| cust_id | מחרוזת | מזהה לקוח. מפתח ראשי. |
| birth_date | תאריך | תאריך לידה |
| first_name | מחרוזת | שם פרטי |
| last_name | מחרוזת | שם משפחה |
| city | מחרוזת | עיר |
| street | מחרוזת | רחוב |
| zip_code | מספר שלם | קוד מיקוד |
| phone_home | מחרוזת | טלפון בבית |
| phone_mobile | מחרוזת | טלפון נייד |
| phone_fax | מחרוזת | פקס |
| date_created | תאריך | תאריך יצירה |
| time_created | זמן (מומר לטיפוס bigint) | זמן יצירה |
| is_active | בוליאני | האם הלקוח פעיל במערכת? |

### טבלת היסטוריית לקוחות (Customer History)

טבלה זו מכילה מידע אודות היסטוריית הלקוחות. הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| hid | מספר שלם | מזהה רשומה. מפתח ראשי. מספור אוטומטי החל מ-1000. |
| cust_id | מחרוזת | מזהה לקוח. מפתח זר. |
| type | מחרוזת | סוג הרשומה |
| date_created | תאריך | תאריך יצירה |
| time_created | זמן (מומר לטיפוס bigint) | זמן יצירה |
| description | טקסט | תיאור הרשומה |

### טבלת מוצרים (Products)

טבלה זו מכילה מידע אודות סוגי המוצרים השונים הקיימים במערכת. הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| pid | מספר שלם | מזהה מוצר. מפתח ראשי. מספור אוטומטי החל מ-1000. |
| type | מחרוזת | סוג המוצר |
| manufacturer | מחרוזת | יצרן |
| name | מחרוזת | שם המוצר |
| price | מספר ממשי | מחיר המוצר. שדה זה יכול להכיל את הערך null. |

### טבלת מלאי (Inventory)

טבלה זו מכילה מידע אודות כמויות המוצרים השונים הקיימים במלאי העסק. הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| pid | מספר שלם | מזהה מוצר. מפתח ראשי. מפתח זר. |
| amount | מספר שלם | כמות המוצר |

### טבלת הנחות (Discounts)

טבלה זו מכילה מידע אודות ההנחות השונות בבית העסק. הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| pid | מספר שלם | מזהה מוצר. מפתח ראשי. מפתח זר. |
| one_plus_one | בוליאני | מבצע 1+1? |
| discount | מספר ממשי | אחוז הנחה על המוצר |

### טבלת חשבונית (Bills)

טבלה זו מכילה מידע אודות כל הרכישות שבוצעו בבית העסק. הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| receipt | מספר שלם | מספר חשבונית. מפתח ראשי. מספור אוטומטי החל מ-1000. |
| cust_id | מחרוזת | מזהה לקוח. מפתח זר. |
| date_created | תאריך | תאריך יצירה |
| time_created | זמן (מומר לטיפוס bigint) | זמן יצירה |
| t_price_equipment | מספר ממשי | סה"כ עלות הציוד |
| t_price_services | מספר ממשי | סה"כ עלות השירותים |

**הערה**: השדה cust_id יאפשר בגרסה עתידית של המערכת לקישור בין חשבונית ללקוח.

### טבלת פירוט רכישה (Purchase Details)

טבלה זו מכילה מידע אודות הכמויות שנרכשו מכל מוצר בכל עסקה שבוצעה.
טבלה זו מקשרת בין הטבלאות "חשבונית" ל"מוצרים" בקשר מסוג רבים-רבים.
הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| receipt | מספר שלם | מספר חשבונית. מפתח ראשי. מפתח זר |
| pid | מספר שלם | מזהה מוצר. מפתח ראשי. מפתח זר. |
| amount | מספר שלם | כמות המוצרים |

### טבלת שירותים (Services)

טבלה זו מכילה מידע אודות השירותים השונים שמציע בית העסק.
הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| pid | מספר שלם | מזהה שירות. מפתח ראשי. מספור אוטומטי החל מ-2000. |
| type | מחרוזת | סוג השירות |
| price | מספר ממשי | מחיר השירות. שדה זה יכול להכיל את הערך null. |

### טבלת הזמנות ציוד (Orders)

טבלה זו מכילה מידע אודות הזמנות הציוד של בית העסק. הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| oid | מספר שלם | מזהה הזמנה. מפתח ראשי. |
| dealer | מחרוזת | שם הספק |
| status | מחרוזת | סטאטוס ביצוע |
| cost | מספר ממשי | העלות הכוללת של ההזמנה |
| date_created | תאריך | תאריך יצירת ההזמנה |
| time_created | זמן (מומר לטיפוס bigint) | שעת יצירת ההזמנה |

### טבלת פירוט הזמנה (Order Details)

טבלה זו מכילה מידע אודות פרטי כל אחת מהזמנות הציוד.
טבלה זו מקשרת בין הטבלאות "הזמנות-ציוד" ל"מוצרים" בקשר מסוג רבים-לרבים.
הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| oid | מספר שלם | מזהה הזמנה. מפתח ראשי. מפתח זר. |
| pid | מספר שלם | מזהה מוצר. מפתח ראשי. מפתח זר. |
| amount | מספר שלם | הכמות של המוצר בהזמנה |
| cost | מספר שלם | המחיר הכולל של המוצר בהזמנה |

### טבלת חלוקת עבודה (Work Distribution)

טבלה זו מכילה מידע אודות קריאות השירות הקיימות במערכת.
הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| call_id | מספר שלם | מזהה קריאה. מפתח ראשי. מספור אוטומטי החל מ-1000. |
| assigned | מחרוזת | מזהה העובד המבצע (username). מפתח זר. |
| service | מחרוזת | תיאור השירות |
| amount | מספר שלם | כמות השירות |
| date_created | תאריך | תאריך פתיחת הקריאה |
| time_created | זמן (מומר לטיפוס bigint) | זמן פתיחת הקריאה |
| status | מחרוזת | סטאטוס הביצוע |
| description | מחרוזת | תיאור קריאת השירות |
| cust_id | מחרוזת | מזהה לקוח שעבורו נפתחה הקריאה. מפתח זר. |

### טבלת היסטוריית קריאות שירות (Service Calls History)

טבלה זו מכילה מידע אודות היסטוריית קריאות השירות. הטבלה מכילה את השדות הבאים:

| שדה | טיפוס | הסבר |
|---|---|---|
| hid | מספר שלם | מזהה רשומה. מפתח ראשי. מספור אוטומטי החל מ-1000. |
| call_id | מספר שלם | מזהה קריאה. מפתח זר. |
| comments | טקסט | הערות |
| date_created | תאריך | תאריך פתיחת הרשומה |
| time_created | זמן (מומר לטיפוס bigint) | זמן פתיחת הרשומה |
| type | מחרוזת | סוג הרשומה |

### טבלת יצרנים עבור קובץ מודול (Module Product Manufacturers)

טבלה זו מכילה את רשימת היצרנים שהתווספו למערכת לאחר טעינת קובץ המודול.

| שדה | טיפוס | הסבר |
|---|---|---|
| manufacturer | מחרוזת | שם היצרן. מפתח ראשי. |

### טבלת סוגי מוצרים עבור קובץ מודול (Module Product Types)

טבלה זו מכילה את רשימת סוגי המוצרים שהתווספו למערכת לאחר טעינת קובץ המודול.

| שדה | טיפוס | הסבר |
|---|---|---|
| type | מחרוזת | סוג המוצר. מפתח ראשי. |

### טבלת סוגי שירותים עבור קובץ מודול (Module Service Types)

טבלה זו מכילה את רשימת סוגי השירותים שהתווספו למערכת לאחר טעינת קובץ המודול.

| שדה | טיפוס | הסבר |
|---|---|---|
| type | מחרוזת | סוג השירות. מפתח ראשי. |

## שכבת הלוגיקה (SmartBiz.Bl namespace)

שכבה זו מקשרת בין שכבת בסיס הנתונים לשכבת ההצגה, והיא מהווה הפשטה של האובייקטים הקיימים במערכת. כל אובייקט מיוצג באמצעות מחלקה משלו. מחלקות אלו יתוארו בהמשך.

**עקרונות מנחים בשכבת הלוגיקה:**

- בעת פניה לשכבת בסיס הנתונים (Dal), שכבת הלוגיקה תספק את בדיקות אימות הנתונים (Validation) וכן, בדיקת חריגות (Exceptions) עבור הפלטים שמתקבלים ממנה.

- עקרון הכימוס (Encapsulation) בא לידי ביטוי בשכבת הלוגיקה, בכך שאנו מסתירים את המידע הנשמר במחלקות וחושפים אותו כלפי חוץ באמצעות תכונות (Properties) בלבד. מעבר ליתרונות הידועים של עקרון הכימוס, הקוד הופך להיות יותר גמיש, מהבחינה שבמידה וירצו להוסיף הגבלות מסוימות על ערכי המידע או בדיקות ניתן יהיה להוסיף זאת לכל מידע שנשמר וזאת מבלי צורך לשכתב קוד קיים במקומות אחרים.

- עקרון ה-SRP (Single Responsibility Principle) יבוא לידי ביטוי בשכבת הלוגיקה בכך שכל מחלקה תקבל אחריות אחת בלבד.

- עקרון ה-LSP (Liskov Substitution Principle) יבוא לידי ביטוי בשכבת הלוגיקה בכך שכאשר המימוש יאפשר זאת, אז פונקציות העושות שימוש במצביעים ל-Base Class יהיו מסוגלות לעשות שימוש ב-Derived Classes מבלי לדעת על כך. (למשל בביצוע fill של רשימת מוצרים או שירותים ל-Data Grid, הפונקציה תקבל רשימת אובייקטים מטיפוס Purchasable מבלי לדעת בפועל מהו ה-Derived Class).

- עקרון ה-Open-Closed Principle יבוא לידי ביטוי בכך שהקוד ייבנה בצורה מודולארית וגמישה תוך תמיכה בהרחבות עתידיות, כך שהמודולים יהיו סגורים לשינוי, אך ניתנים להרחבה.

- שימוש בדפוסי עיצוב (Design Patterns) רלוונטיים כדי שהעיצוב של שכבה זו יהיה גמיש לשינויים ויעמוד ביעדי הנדסת תוכנה נוספים.

נעבור עתה על המחלקות השונות.

## מחלקות כלליות

### Constants.cs

מחלקה זו מכילה אוסף של קבועים אשר רלוונטיים לכל המערכת. במחלקה זו נגדיר קבועים אשר השימוש בהם ייעשה מכמה מחלקות שונות במערכת. דוגמה לקבועים אשר צריך לעשות בהם שימוש בכלל המערכת הינם: שם המשתמש והסיסמא של מנהל המערכת.

### Convertor.cs

מחלקה זו מכילה אוסף של שיטות סטטיות הממירות בין ערכים וטיפוסים. כל שיטה במחלקה זו מקבלת פרמטר אחד ומחזירה את הערך לאחר ביצוע ההמרה. דוגמה לשיטה השייכת למחלקה זו היא שיטה אשר מקבלת תאריך מטיפוס DateTime ומחזירה מחרוזת המייצגת תאריך בפורמט: dd/MM/yyyy.

מחלקה זו תכיל את השיטה הבאה:

```
public static string ConvertToDBDate(DateTime date)
```

### CustomerValidator.cs

מחלקה זו מכילה אוסף של שיטות סטטיות המבצעות אימות עבור תכונות השייכות לישות 'לקוח'. כל שיטה במחלקה זו מקבלת מחרוזת המכילה את ערך התכונה ומחזירה ערך בוליאני הקובע האם המחרוזת מכילה ערך חוקי עבור תכונה זו.

מחלקה זו תכיל את השיטות הבאות:

```
public static bool ValidateCustomerID(string cid)

public static bool ValidateCustomerZipCode(string zipCode)
```

כמו כן, למחלקה זו ניתן להוסיף שיטות נוספות הבודקות את התכונות השונות של הישות 'לקוח'.

### DBController.cs

מחלקה זו מכילה שתי שיטות סטטיות המבצעות חיבור וניתוק התקשרות למסד הנתונים. שיטות אלה פונות אל ה-DBManager בשכבת ה-Dal כדי לאתחל ולנתק את ההתקשרות בהתאמה.
מחלקה זו תכיל את השיטות הבאות:

```
public static void InitDB()

public static void CloseDB()
```

### GlobalSettings.cs

מחלקה זו מכילה שיטות סטטיות המחזירות ערכים של הגדרות כלליות במערכת. כל שיטה במחלקה זו מחזירה ערך של הגדרה כללית כלשהי. לכל אחת מהשיטות מוגדר קבוע המכיל את ערך ברירת המחדל למקרה שהגדרה זו לא נמצאה בבסיס הנתונים.

עבור כל הגדרה כללית השמורה במערכת, קיימת שיטה המחזירה את הערך השמור בבסיס הנתונים של אותה הגדרה מתאימה.

מחלקה זו תכיל את השיטות הבאות:

```
public static string GetCoinType()

public static string GetCompanyName()

public static string GetCompanyAddress()

public static string GetCompanyPhone()

public static double getVatRate()

public static string GetOpeningHour()

public static string GetClosingHour()
```

### GlobalValidator.cs

מחלקה זו מכילה אוסף של שיטות סטטיות המבצעות אימות עבור תכונות כלשהן במערכת. כל שיטה במחלקה זו מקבלת מחרוזת המכילה את ערך התכונה ומחזירה ערך בוליאני הקובע האם המחרוזת מכילה ערך חוקי עבור תכונה זו. מחלקה זו תכיל שיטות לאימות תכונות שונות אשר ערכיהן נקבעים ע"י קלט מהמשתמש. דוגמאות לתכונות אשר יש לאמת לפני ששומרים את ערכיהן בבסיס הנתונים הן: כמות (מספר שלם אי-שלילי), מחיר (מספר שלם בתווך 0-1,000,000), מספר תשלומים (מספר שלם בתווך 1-24) ועוד...

מחלקה זו תכיל את השיטות הבאות:

```
public static bool ValidateVatRate(string vat)

public static bool ValidateCredit(string credit)

public static bool ValidatePayments(string payments)

public static bool ValidateDiscount(string discount)

public static bool ValidateCompanyHours(string openAt, string closeAt)

public static bool ValidateAmountAtInsert(string amount)

public static bool ValidateAmountAtCreate(string amount)

public static bool ValidateAmountAtSearch(string amount)

public static bool ValidateCost(string cost)
```

### Module.cs

מחלקה זו מכילה אוסף של שיטות סטטיות המבצעות ניתוח (Parsing) של קובץ המודול. מחלקה זו מכילה את השיטה הראשית:

```
public static bool ParseModule(string filePath)
```

המקבלת נתיב מלא הכולל את שם הקובץ, מבצעת עליו את פעולת הניתוח, מוסיפה את הנתונים המתאימים לטבלאות המודול בבסיס הנתונים ומחזירה לבסוף ערך בוליאני הקובע האם הפעולה הצליחה.

מחלקה זו כוללת את כל השיטות הפרטיות ואת הקבועים הדרושים לצורך פעולת הניתוח (Parsing).

### StringGenerator.cs

מחלקה זו מכילה אוסף של שיטות סטטיות המקבלות פרמטר אחד או יותר, כל שיטה מייצרת מחרוזת טקסט המכילה בתוכה נתונים הנגזרים מפרמטרים אלו. מחלקה זו נועדה לתמוך במנגנון יצירת האירועים האוטומטיים (עבור לקוחות וקריאות שירות), כל שיטה מחזירה את תוכן האירוע שנוצר באופן אוטומטי כאשר התוכן נבנה בהתאם לפרמטרים שהשיטה מקבלת.

מחלקה זו תכיל את השיטות הבאות:

```
public static string GenerateString_ChangeStatus(string status)

public static string GenerateString_CustomerPurchased(ShoppingCart shoppingCart)

public static string GenerateString_PriceOffer(ShoppingCart shoppingCart)

public static string GenerateString_CustomerPayment(double sum, Bill bill)

public static string GenerateString_NewServiceCall(ServiceCall newCall)

public static string GenerateString_CloseServiceCall(ServiceCall newCall)
```

### UserAuthenticator.cs

מחלקה זו מכילה שיטות סטטיות המאפשרות לאמת את פרטי ההתחברות של משתמש אל המערכת, באמצעות יצירת קשר עם מסד הנתונים והחזרת תשובה. היא מכילה את השיטה הבאה:

```
public static bool AuthenticateUser(string username, string password)
```

### UserValidator.cs

מחלקה זו מכילה שיטות סטטיות המאפשרות לוודא פרטים הנוגעים למשתמשים במערכת.
היא תאפשר לוודא שהסיסמא עומדת בתנאי חוזק מסוימים (מספר מינימאלי של תווים ומורכבות מסוימת). היא תאפשר לוודא ששם המשתמש שרוצים להגדיר עומד בקונבנציות שם מסוימות (אותיות בלבד!).
היא מכילה את השיטות הבאות:

```
public static bool ValidatePassword(string password)

public static bool ValidateUsername(string username)
```

## ממשקים ומחלקות אבסטרקטיות

### Aggregate.cs

מחלקה אבסטרקטית המכילה שיטה אבסטרקטית יחידה:

```
public abstract Iterator CreateIterator();
```

מחלקה זו הינה חלק מדפוס העיצוב Iterator אשר עליו נרחיב בהמשך.

### History.cs

מחלקה אבסטרקטית המייצגת אירוע היסטוריה במערכת. היא בעלת התכונות הבאות:

```
public int Hid { get; set; }

public DateTime DateCreated { get; set; }

public Int64 TimeCreated { get; set; }

public string Data { get; set; }

public string Type { get; set; }
```

### IDatabaseObject.cs

מחלקה זו מייצגת ממשק אחיד לכל אובייקט במערכת אשר משמש גם כאובייקט שנשמר בבסיס הנתונים. ממשק זה מאפשר את הכנסת האובייקט לבסיס הנתונים, את עדכונו או את מחיקתו. נוסף על כך, ממשק זה מגדיר שיטה נוספת המחזירה ערך בוליאני הקובע האם אובייקט זה קיים בבסיס הנתונים. יתר השיטות מחזירות ערך אמת 'true' אם הפעולה הצליחה, ו-'false' אם הפעולה נכשלה.

ממשק זה מגדיר את השיטות הבאות:

```
bool InsertToDB();

bool RemoveFromDB();

bool UpdateDB();

bool IsExistInDB();
```

### Iterator.cs

מחלקה אבסטרקטית המכילה ארבע שיטות אבסטרקטיות:

```
public abstract void First();

public abstract ShoppingCartItem Next();

public abstract bool IsDone();

public abstract ShoppingCartItem CurrentItem();
```

מחלקה זו הינה חלק מדפוס העיצוב Iterator אשר עליו נרחיב בהמשך.

### Person.cs

מחלקה אבסטרקטית המייצגת אדם. מחלקה זו מגדירה את התכונות והשיטות המשותפות למשתמשים ולקוחות. היא בעלת התכונות הבאות:

```
public string FirstName { get; set; }

public string LastName { get; set; }

public bool IsActive { get; set; }
```

### Purchasable.cs

מחלקה אבסטרקטית המייצגת פריט אשר ניתן לרכוש. מחלקה זו מגדירה את התכונות והשיטות המשותפות למוצרים ושירותים. היא בעלת התכונות הבאות:

```
public int Pid { get; set; }

public string Type { get; set; }

public double Price { get; set; }

public bool OnePlusOne { get; set; }

public double Discount { get; set; }
```

מחלקה זו מכילה את השיטות הבאות:

```
public double GetVatValue()

public double GetPriceIncVat()

public double GetPriceIncVat(double vat)

public static double GetPriceIncVat(double price, double vat)

public double GetPriceIncVatAndDiscount()

public double GetPrice(double percent)

public double GetTotalPrice(int amount)
```

### Strategy.cs

מחלקה אבסטרקטית המכילה שיטה אחת אבסטרקטית:

```
public abstract bool checkComplexity(string password);
```

שיטה זו מקבלת מחרוזת המייצגת סיסמא, השיטה מחזירה ערך בוליאני 'true' אם הסיסמא מורכבת מספיק על פי האלגוריתם הממשש את השיטה האבסטרקטית. השיטה מחזירה ערך בוליאני 'false' אם הסיסמא אינה עומדת בתנאיי המורכבות על פי אותו האלגוריתם.

מחלקה זו הינה חלק מדפוס העיצוב Strategy אשר עליו נרחיב בהמשך.

### UserDefinition.cs

מחלקה אבסטרקטית המייצגת הגדרה של משתמש. מחלקה זו נועדה כדי להגדיר את כל התכונות והשיטות אשר משותפות לאובייקט 'משתמש' ולאובייקט 'המשתמש הנוכחי'. היא בעלת התכונות הבאות:

```
public string UserName { get; set; }

public string Password { get; set; }

public bool IsManager { get; set; }

public bool IsTeamleader { get; set; }

public bool IsSalesmanager { get; set; }

public bool IsWorker { get; set; }

public bool IsMustChangePassword { get; set; }
```

בנוסף, מחלקה זו מממשת את כל השיטות של הממשק IDatabaseObject ויורשת מהמחלקה Person.

## מחלקות עבור אובייקטים במערכת

### Bill.cs

מחלקה המממשת את הממשק IDatabaseObject, מייצגת חשבונית לקוח במערכת. היא בעלת התכונות הבאות:

```
public int Receipt { get; set; }

public string CustId { get; set; }

public DateTime DateCreated { get; set; }

public Int64 TimeCreated { get; set; }

public double TotalPriceEquipment { get; set; }

public double TotalPriceServices { get; set; }
```

### ComplexityCheckerA.cs

מחלקה היורשת מ-Strategy, וממשת את השיטה האבסטרקטית:

```
checkComplexity(string password)
```

שיטה זו מממשת את אלגוריתם בדיקת המורכבות באופן הבא: השיטה בודקת האם המחרוזת מכילה לפחות 2 (מוגדר כקבוע - MINIMUM_AMOUNT) תווים אשר כל אחד מהם שייך לקבוצה אחרת, כאשר הקבוצות הן: אותיות קטנות, אותיות גדולות, ספרות וסימנים אחרים.

מחלקה זו הינה חלק מדפוס העיצוב Strategy אשר עליו נרחיב בהמשך.

### ComplexityCheckerB.cs

מחלקה היורשת מ-Strategy, וממשת את השיטה האבסטרקטית:

```
checkComplexity(string password)
```

שיטה זו מממשת את אלגוריתם בדיקת המורכבות באופן הבא: השיטה בודקת האם המחרוזת לא מכילה שני תווים זהים עוקבים. כלומר, על פי אלגוריתם זה, סיסמא מורכבת הינה סיסמא אשר לא מכילה שני תווים זהים עוקבים.

מחלקה זו הינה חלק מדפוס העיצוב Strategy אשר עליו נרחיב בהמשך.

### ComplexityCheckerC.cs

מחלקה היורשת מ-Strategy, ומממשת את השיטה האבסטרקטית:

checkComplexity(string password)

שיטה זו מממשת את אלגוריתם בדיקת המורכבות באופן הבא: השיטה בודקת האם המחרוזת לא מכילה שני תווים עוקבים השייכים לאותה קבוצה (אותיות קטנות, אותיות גדולות, ספרות וסימנים אחרים). כלומר, על פי אלגוריתם זה, סיסמא מורכבת הינה סיסמא אשר לא מכילה שני תווים עוקבים השייכים לאותה קבוצה.

מחלקה זו הינה חלק מדפוס העיצוב Strategy אשר עליו נרחיב בהמשך.

### ConcreteIterator.cs

מחלקה היורשת מהמחלקה האבסטרקטית Iterator, מייצגת Iterator קונקרטי המתאים לאובייקט ShoppingCart.

מחלקה זו הינה חלק מדפוס העיצוב Iterator אשר עליו נרחיב בהמשך.

### Context.cs

מחלקה המחזיקה הפניה (Reference) לאובייקט מסוג Strategy.

מחלקה זו מכילה את השיטה:

public bool ContextInterface(string password)

מחלקה זו הינה חלק מדפוס העיצוב Strategy אשר עליו נרחיב בהמשך.

### CurrentUser.cs

מחלקה היורשת מהמחלקה האבסטרקטית UserDefinition, מייצגת את המשתמש הנוכחי שמחובר כעת למערכת.

מחלקה זו הינה חלק מדפוס העיצוב Singleton אשר עליו נרחיב בהמשך.

### Customer.cs

מחלקה היורשת מהמחלקה האבסטרקטית Person וממשת את הממשק IDatabaseObject.
מחלקה זו מייצגת לקוח רשום במערכת. היא בעלת התכונות הבאות:

```
public string Cid { get; set; }

public string City { get; set; }

public string Street { get; set; }

public int ZipCode { get; set; }

public string PhoneHome { get; set; }

public string PhoneMobile { get; set; }

public string PhoneFax { get; set; }

public DateTime DateCreated { get; set; }

public Int64 TimeCreated { get; set; }

public DateTime BirthDate { get; set; }
```

### CustomerHistory.cs

מחלקה היורשת מהמחלקה האבסטרקטית History וממשת את הממשק IDatabaseObject.
מחלקה זו מייצגת רשומת היסטוריה עבור לקוח במערכת. בנוסף על התכונות שהמחלקה יורשת מהמחלקה History, היא בעלת התכונה הבאה:

```
public string CustomerId { get; set; }
```

### Order.cs

מחלקה זו מממשת את הממשק IDatabaseObject. מחלקה זו מייצגת הזמנת ציוד מספק חיצוני.
היא בעלת התכונות הבאות:

```
public int Oid { get; set; }

public string Dealer { get; set; }

public string Status { get; set; }

public double Cost { get; set; }

public DateTime DateCreated { get; set; }

public Int64 TimeCreated { get; set; }
```

### OrderItem.cs

מחלקה זו מממשת את הממשק IDatabaseObject. מחלקה זו מייצגת מוצר כלשהו בהזמנת ציוד מספק חיצוני. היא בעלת התכונות הבאות:

```
public int Oid { get; set; }

public int Pid { get; set; }

public int Amount { get; set; }

public double Cost { get; set; }
```

### Product.cs

מחלקה היורשת מהמחלקה האבסטרקטית Purchasable ומממשת את הממשק IDatabaseObject.
מחלקה זו מייצגת מוצר במערכת. בנוסף על התכונות שהמחלקה יורשת מהמחלקה Purchasable
היא בעלת התכונות הבאות:

```
public string Manufacturer { get; set; }

public string Name { get; set; }

public int Amount { get; set; }
```

### Purchase.cs

מחלקה המממשת את הממשק IDatabaseObject. מחלקה זו מייצגת רכישה של מוצר או שירות מסוים. מחלקה זו מכילה תכונות המקשרות בין המוצר או השירות לבין החשבונית השייכת לאותה רכישה. אובייקט זה מכיל תכונה נוספת המפרטת את הכמות של המוצר או השירות הנרכש. היא בעלת התכונות הבאות:

```
public int Receipt { get; set; }

public int Pid { get; set; }

public int Amount { get; set; }
```

### Service.cs

מחלקה היורשת מהמחלקה האבסטרקטית Purchasable וממשת את הממשק IDatabaseObject. מחלקה זו מייצגת שירות במערכת. למחלקה זו אין תכונות נוספות בנוסף על התכונות שהמחלקה יורשת מהמחלקה Purchasable.

### ServiceCall.cs

מחלקה המממשת את הממשק IDatabaseObject. מחלקה זו מייצגת קריאת שירות במערכת.
קריאת שירות במערכת נפתחת באופן אוטומטי לאחר שלקוח מסוים ביצע רכישה של שירות כלשהו.
כל קריאת שירות מקושרת ללקוח אחד בלבד ובכל רגע נתון הקריאה משויכת לעובד אחד לכל היותר. היא בעלת התכונות הבאות:

```
public int Cid { get; set; }

public string Assigned { get; set; }

public string ServiceType { get; set; }

public int Amount { get; set; }

public DateTime DateCreated { get; set; }

public Int64 TimeCreated { get; set; }

public string Status { get; set; }

public string Description { get; set; }

public string CustomerId { get; set; }
```

### ServiceCallHistory.cs

מחלקה היורשת מהמחלקה האבסטרקטית History וממשת את הממשק IDatabaseObject. מחלקה זו מייצגת רשומת היסטוריה של קריאת שירות במערכת. בנוסף על התכונות שהמחלקה יורשת מהמחלקה History, היא בעלת התכונה הבאה:

```
public int CallId { get; set; }
```

### ServiceRecord.cs

מחלקה זו מייצגת רשומה של שירות מסוים. רשומה זו מכילה הפניה (Reference) לשירות (אובייקט מסוג Service) ושלושה מונים המייצגים כמה קריאות שירות עבור סוג השירות הנוכחי נמצאות בסטאטוס: לא משויך ("Unassigned"), בתהליך ("In Progress") והסתיים ("Completed"). מחלקה זו נועדה כדי ליצור אובייקטים אשר יתאימו לרשומות בדו"ח הפעילויות. היא בעלת התכונות הבאות:

```
public Service TheService { get; set; }

public int UnassignedCounter { get; set; }

public int InProgressCounter { get; set; }

public int CompletedCounter { get; set; }
```

### Setting.cs

מחלקה המממשת את הממשק IDatabaseObject. מחלקה זו מייצגת הגדרה כללית במערכת. כל הגדרה מורכבת ממפתח (string) וערך (string). מחלקה זו נועדה לייצר אובייקטים שיהוו הגדרות כלליות במערכת כגון: שם העסק, כתובת העסק, שעת פתיחה, שעת סגירה ועוד. היא בעלת התכונות הבאות:

```
public string Key { get; set; }

public string Value { get; set; }
```

### ShoppingCart.cs

מחלקה היורשת מהמחלקה האבסטרקטית Aggregate. מחלקה זו מייצגת עגלת קניות (או סל קניות) במערכת. מחלקה זו הינה מחלקה קונקרטית של המחלקה Aggregate והינה חלק מדפוס העיצוב Iterator אשר עליו נרחיב בהמשך. עגלת הקניות משמשת כדי לאסוף מוצרים או שירותים בעת ביצוע מכירה ללקוח או כדי לאסוף מוצרים לצורך ביצוע הזמנה מספק חיצוני.

מבנה הנתונים שבעזרתו נממש את עגלת הקניות יהיה רשימה של אובייקטים מסוג ShoppingCartItem. לכן היא בעלת השדה הבא בלבד:

```
private List<ShoppingCartItem> _itemsList;
```

### ShoppingCartItem.cs

מחלקה זו מייצגת פריט השייך לעגלת הקניות (ShoppingCart). מחלקה זו מכילה הפניה (Reference) לאובייקט מסוג Purchasable, כלומר כל פריט מקושר למוצר או שירות מסוים. כל מופע של מחלקה זו מכיל הפניה לפריט עצמו (המוצר או השירות), הכמות של הפריט, עלות ההזמנה, ותיאור. היא בעלת התכונות הבאות:

```
public Purchasable Item { set; get; }

public int Amount { set; get; }

public double OrderCost { set; get; }

public string Description { set; get; }
```

### User.cs

מחלקה היורשת מהמחלקה האבסטרקטית UserDefinition. מחלקה זו מייצגת משתמש קיים במערכת. להבדיל מהמחלקה CurrentUser, ממחלקה זו ניתן לייצר מופעים רבים מכיוון שהשימוש באובייקטים של מחלקה זו נועד כדי לנהל קבוצה של משתמשים במערכת. מחלקה זו אינה מוסיפה תכונות נוספות מעבר לתכונות אשר עוברות בירושה מהמחלקה UserDefinition.

### UserRecord.cs

מחלקה זו מייצגת רשומה של משתמש מסוים. רשומה זו מכילה הפניה (Reference) למשתמש (אובייקט מסוג User) ושני מונים המייצגים כמה קריאות שירות נמצאות בסטאטוס זה אשר משויכות לאותו המשתמש. הסטאטוסים של הקריאות המשויכות למשתמש הינם: בתהליך ("In Progress") והסתיים ("Completed"). מחלקה זו נועדה כדי ליצור אובייקטים אשר יתאימו לרשומות בדו"ח קריאות השירות. היא בעלת התכונות הבאות:

```
public User TheUser { get; set; }

public int InProgressCounter { get; set; }

public int CompletedCounter { get; set; }
```

## שכבת ההצגה (SmartBiz.Pl namespace)

שכבה זו מציגה ממשק משתמש (UI) שיאפשר לו ליצור אינטראקציה עם המערכת.
כמו כן, שכבה זו מתממשקת עם שכבת הלוגיקה, בכדי לקבל ממנה שירותים (שהיא עצמה מספקת ישירות או בעקיפין באמצעות שכבת בסיס-הנתונים).
בטפסים שאינם מהווים חלונות צצים (PopUps) יעמוד לרשות המשתמש סרגל כלים עליון לביצוע פעולות כלליות (כגון חזרה לתפריט הראשי, שינוי משתמש וכיוב') וסרגל כלים תחתון לקבלת חיווי על המשתמש המחובר כעת.

**עקרונות מנחים בשכבת ההצגה:**

- ממשק המשתמש יהיה נוח ופשוט לשימוש.
- כל המחלקות של הטפסים במערכת יירשו ממחלקות אבסטרקטיות אשר יגדירו הגדרות אחידות לכל אותם הטפסים היורשים. כך יהיה ניתן לחלק את הטפסים לקבוצות ובהמשך לקבוע הגדרות מסוימות לקבוצה ספציפית של טפסים. כמו כן, בראש ההיררכיה תוגדר המחלקה BasicForm כך שכל הטפסים במערכת יירשו ממנה. עיצוב זה יאפשר לנו גמישות לשינויים עתידיים.
- שימוש ב-UserControls ומחלקות כלליות עבור חלקים המשותפים לטפסים רבים כדי למנוע שכפול קוד ועל מנת לבצע שימוש חוזר.
- שימוש ב-Grid בקבצי ה-XMAL כדי לאפשר הגדלה והתאמה של החלונות לכל רזולוציה אפשרית.
- שימוש בדפוסי עיצוב (Design Patterns) רלוונטיים כדי שהעיצוב של שכבה זו יהיה גמיש לשינויים ויעמוד ביעדי הנדסת תוכנה נוספים.

להלן רשימת המחלקות בשכבה זו:

## UserControls

### UpperToolbar.cs

UserControl אשר משמש כסרגל כלים עילי המופיע במרבית הטפסים. הוא מאפשר לבצע פעולות כלליות כגון: יציאה מהמערכת, התנתקות של המשתמש הנוכחי, חזרה לאחור, חזרה לדף הבית,

ניקוי הטופס ובחירה מרובה. יש לציין כי לכל טופס זמינים אך ורק הרכיבים הרלוונטיים עבורו, והשאר מבוטלים.

### LowerToolbar.cs

UserControl אשר משמש כסרגל כלים תחתון. הוא מציג חיווי על המשתמש המחובר כרגע למערכת.

## ממשקים ומחלקות אבסטרקטיות

### BasicForm.cs

מחלקה אבסטרקטית היורשת מהמחלקה Window. מחלקה אבסטרקטית זו משמשת כאב-טיפוס אחד לכל סוגי הטפסים במערכת. כדי ליצור הגדרות אחידות לכל הטפסים במערכת, אנו נגדיר הגדרות אלה תחת הבנאי של מחלקה זו. במחלקה זו נגדיר שתכונת ה-Visibility תקבל את הערך Visible כדי שברירת המחדל של כל אובייקט מסוג טופס במערכת יהיה גלוי למשתמש מיד לאחר יצירת האובייקט. בנוסף, נגדיר את מיקום החלון למרכז המסך כדי שכל הטפסים במערכת יופיעו במרכז.

מחלקה זו תאפשר תמיכה בביצוע שינויים עתידיים שתקפים לכלל הטפסים במערכת, כגון: צבע הרקע של הטפסים או כל פעולה או תכונה אחרת אשר משותפת לכלל הטפסים במערכת.

### NavigationalForm.cs

מחלקה אבסטרקטית היורשת מהמחלקה BasicForm. מחלקה זו משמשת כאב-טיפוס לכל הטפסים במערכת אשר מאפשרים ניווט בין מסכים. במחלקה זו נגדיר תכונות ושיטות אשר יהיו משותפות לכל טפסי הניווט במערכת. כגון, שני סרגלי הכלים אשר יהיו זמינים בין היתר למטרות ניווט.

מחלקה זו תשמש גם לביצוע שינויים עתידיים שתקפים לכלל טפסי הניווט במערכת.

### PopupForm.cs

מחלקה אבסטרקטית היורשת מהמחלקה BasicForm וממשת את הממשק Observer. מחלקה זו משמשת כאב-טיפוס לכל הטפסים במערכת אשר מהווים מסכי Popup. במחלקה זו נגדיר תכונות ושיטות אשר יהיו משותפות לכל טפסי ה-Popup במערכת.

עבור חלק מטפסי ה-Popup ניתן ליצור מופעים ללא הגבלה כדי שהמשתמש יוכל לפתוח כמה טפסים במקביל, אך כשסוגרים את טופס הניווט הנוכחי צריך לסגור גם את כל מסכי ה-Popup הפתוחים כעת. לשם כך, מחלקה זו מממשת את הממשק Observer והינה חלק מדפוס העיצוב Observer אשר עליו נרחיב בהמשך.

### MenuForm.cs

מחלקה אבסטרקטית היורשת מהמחלקה NavigationalForm. מחלקה אבסטרקטית זו משמשת כאב-טיפוס לכל הטפסים במערכת אשר מהווים תפריט. טפסים אלו מכילים קבוצה של כפתורים המנווטים אל טפסים שונים במערכת. מחלקה זו מכילה את המימוש של אלגוריתם סידור הכפתורים. השיטה אשר מממשת אלגוריתם זה מקבלת קבוצה של כפתורים, ובהתאם למספר קבועים אשר מוגדרים במחלקה זו, השיטה מסדרת את הכפתורים לפי שורות ועמודות.

מחלקה זו תשמש גם לביצוע שינויים עתידיים שתקפים לכלל טפסי התפריטים במערכת.

### Observer.cs

ממשק המגדיר שיטה אחת בשם Update. ממשק זה הינו חלק מדפוס העיצוב Observer אשר עליו נרחיב בהמשך.

### Subject.cs

מחלקה אבסטרקטית אשר מכילה רשימה של אובייקטים מסוג Observer וממשת את הפעולות Attach, Detach ו-Notify. מחלקה זו הינה חלק מדפוס העיצוב Observer אשר עליו נרחיב בהמשך.

## מחלקות כלליות

### FilterOrders.cs

מחלקה זו מכילה אוסף של שיטות סטטיות המבצעות פעולות הקשורות לסינון ההזמנות במערכת. מכיוון שמנגנון הסינון (Filter) עבור הזמנות ממומש בכמה טפסים שונים במערכת, השיטות במחלקה זו נועדו כדי לבצע שימוש חוזר. כלומר מנגנוני הסינון של ההזמנות במערכת משתמשים בשיטות של מחלקה זו כדי למנוע כפילויות בקוד.

מחלקה זו מכילה את השיטות הבאות:

```
public static void FillDealerCmb(ComboBox dealerCmb)

public static void FillStatusCmb(ComboBox statusCmb)

public static string SearchByOrderDetails(ComboBox statusCmb, ComboBox dealerCmb,

    DatePicker fromDate, DatePicker toDate, CheckBox statusChb, CheckBox dealerChb,
    CheckBox datesChb)
```

### FilterProducts.cs

מחלקה זו מכילה אוסף של שיטות סטטיות המבצעות פעולות הקשורות לסינון מוצרים במערכת. מכיוון שמנגנון הסינון (Filter) עבור מוצרים ממומש בכמה טפסים שונים במערכת, השיטות במחלקה זו נועדו כדי לבצע שימוש חוזר. כלומר מנגנוני הסינון של המוצרים במערכת משתמשים בשיטות של מחלקה זו כדי למנוע כפילויות בקוד.

מחלקה זו מכילה את השיטות הבאות:

```
public static void FillTypesIntoComboBox<T>(ComboBox comboBox, List<T> purchasables)
where T : Purchasable

public static void FillManufacturersIntoComboBox(ComboBox comboBox, List<Product>
products)

public static void FillNamesIntoComboBox(ComboBox comboBox, List<Product> products)

public static void ManufacturerCmbSelectionChanged(ComboBox manufacturerCmb,

    ComboBox typeCmb, ComboBox nameCmb)

public static void TypeCmbSelectionChanged(ComboBox manufacturerCmb, ComboBox
typeCmb, ComboBox nameCmb)
```

### PageCache.cs

מחלקה היורשת מהמחלקה האבסטרקטית Subject. מחלקה זו מייצגת את זיכרון המטמון של הדפים במערכת. מחלקה זו מחזיקה הפניות (References) לטופס הנוכחי (Current Page), לטופס הקודם (Last Page) ולכל טפסי ה-Popup אשר פתוחים כרגע במערכת (נמצאים ברשימה של האובייקטים מסוג Observer אשר מוגדרת במחלקה Subject). מחלקה זו נועדה לתמוך בטפסי הניווט ובסרגלי הכלים כאשר יש לבצע פעולות כגון: סגירת חלון או חזרה אחורה.

מכיוון שיש צורך במופע יחיד של זיכרון המטמון במערכת, מחלקה זו הינה חלק מדפוס העיצוב Singleton אשר עליו נרחיב בהמשך.

בנוסף, מחלקה זו הינה חלק מדפוס העיצוב Observer אשר גם עליו נרחיב בהמשך.

מחלקה זו מכילה את השיטות הבאות:

```
public static PageCache GetInstance()

public void CloseAllPopupForms()
```

## טפסים קונקרטיים במערכת

### AddUpdateInventoryItemForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית Popup. מחלקה זו מייצגת את הטופס: "הוספת פריט חדש למלאי או עדכון פריט קיים". באמצעות טופס זה ניתן להוסיף מוצר או שירות חדש למלאי, בנוסף ניתן לערוך מוצר או שירות קיים.

### AddUpdateOrderForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "הוספת הזמנה חדשה / עדכון או צפייה בהזמנה קיימת". באמצעות טופס זה ניתן להוסיף הזמנה חדשה מול ספק חיצוני, בנוסף ניתן לערוך או לצפות בהזמנות קיימות. סטאטוס ההזמנה ייקבע את הרשאות הצפייה או העריכה.

### AddUpdateUserForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית PopupForm. מחלקה זו מייצגת את הטופס: "הוספת משתמש חדש או עדכון משתמש קיים". באמצעות טופס זה ניתן להוסיף/לערוך/לצפות משתמשים. בטופס זה ניתן להגדיר את פרטי המשתמש, הרשאותיו ואת סיסמתו.

### AdminMenuForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "מסך ראשי השייך למשתמש Admin". באמצעות טופס זה ניתן לנווט לטפסים אשר מנהל המערכת רשאי לפתוח. באמצעות טופס זה ניתן לפתוח טופס לניהול משתמשים או קביעת הגדרות כלליות.

### BillForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית PopupForm. מחלקה זו מייצגת את הטופס: "הפקת חשבונית". טופס זה מהווה חשבונית ללקוח ומציג את פירוט הקנייה שבוצעה ופרטים רלוונטיים נוספים.

### ChangePasswordForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית BasicForm. מחלקה זו מייצגת את הטופס: "שינוי סיסמא". טופס זה נפתח לאחר שמשתמש מסוים הזדהה במסך ה-Login ונקבעה עבורו ההגדרה שמחייבת לשנות את הסיסמא בהתחברות הבאה למערכת. באמצעות טופס זה יכול המשתמש לשנות את סיסמתו ולהמשיך לאחר מכן את הפעילות השוטפת במערכת, או לבצע יציאה ולשנות את הסיסמא מאוחר יותר. בכל מקרה לא תתאפשר כניסה אל המערכת עבור אותו המשתמש עד אשר יבצע שינוי סיסמא.

### CustomerDetailsForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "פרטי לקוח". באמצעות טופס זה ניתן להוסיף לקוח חדש או לערוך או לצפות בלקוח קיים. בטופס זה מצויים כל הפרטים האישיים של הלקוח כולל היסטוריית לקוח. ההרשאות בטופס זה נקבעות על סמך הרשאותיו של המשתמש הנוכחי.

### CustomerHistoryEventForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית PopupForm. מחלקה זו מייצגת את הטופס: "היסטוריית לקוח". באמצעות טופס זה ניתן לערוך או לצפות באירוע היסטוריית לקוח. ההרשאות בטופס זה נקבעות על סמך הרשאותיו של המשתמש הנוכחי.

### CustomersForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "ניהול לקוחות". באמצעות טופס זה ניתן לחפש לקוח על פי מספר תכונות. טופס זה מאפשר לצפות

בקבוצה של לקוחות ובנוסף להסיר או להוסיף לקוחות מהמערכת. ההרשאות בטופס זה נקבעות על סמך הרשאותיו של המשתמש הנוכחי.

### CustomersReportForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "דו"ח לקוחות". באמצעות טופס זה ניתן להפיק דו"ח המציג את כל הלקוחות שהתווספו למערכת בטווח של תאריכים ושעות.

### DiscountsForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "ניהול מבצעים". באמצעות טופס זה ניתן לקבוע מבצעים על המוצרים או השירותים הקיימים בעסק. ניתן לקבוע שני סוגים של מבצעים: 1. הנחה לפי אחוזים. 2. מבצע 1+1. הטופס מאפשר גמישות רבה בכך שניתן להשתמש במנגנון הסינון ולבחור קטגוריה של מוצר מסוים על פי יצרן או על פי יצרן וסוג המוצר, ולקבוע את ההנחה רק על קטגוריה ספציפית.

### ExpensesIncomeReportForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "דו"ח הכנסות-הוצאות". באמצעות טופס זה ניתן להפיק דו"ח המציג את הכנסות העסק מול הוצאות העסק. חשוב להעיר כי לא ניתן להפיק באמצעות דו"ח זה מידע על הרווחים של העסק. ניתן להפיק את הדו"ח על פי טווח של תאריכים ושעות.

### GlobalSettingsForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית PopupForm. מחלקה זו מייצגת את הטופס: "הגדרות כלליות". באמצעות טופס זה ניתן לערוך את ההגדרות הכלליות של המערכת. טופס זה נגיש אך ורק למנהל המערכת (Admin) והוא רשאי לקבוע הגדרות שונות אשר ישפיעו על כלל המערכת.
באמצעות טופס זה ניתן לטעון קובץ מודול על מנת להתאים את המערכת לסוג עסק ספציפי.

### InventoryForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "ניהול מלאי". באמצעות טופס זה ניתן לחפש מוצרים או שירותים הקיימים במלאי העסק. בנוסף,

טופס זה מאפשר להוסיף מוצרים/שירותים חדשים או להסיר קיימים. ההרשאות בטופס זה נקבעות על סמך הרשאותיו של המשתמש הנוכחי.

### LoginForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית BasicForm. מחלקה זו מייצגת את הטופס: "התחברות למערכת". באמצעות טופס זה ניתן להתחבר אל המערכת באמצעות שם משתמש וסיסמא. למעשה זהו הטופס הראשי שנפתח עם עליית המערכת. טופס זה מגביל את הגישה למערכת כך שרק משתמשים קיימים ופעילים יוכלו להתחבר אל המערכת.

### MainMenuForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית MenuForm. מחלקה זו מייצגת את הטופס: "מסך ראשי". באמצעות טופס זה ניתן לנווט בין טפסי המערכת שאליהם קיימת גישה. ההרשאות בטופס זה נקבעות על סמך הרשאותיו של המשתמש הנוכחי, ובהתאם להרשאות אלו, יופיעו הכפתורים המתאימים שדרכם ניתן לנווט לטפסים הרלוונטיים.

טופס זה מאפשר גמישות רבה, כאשר משתמש פעיל בעל הרשאות מתחבר אל המערכת, טופס זה מציג בפניו את כל הכפתורים הרלוונטיים גם אם למשתמש קיימות כמה הרשאות בו-זמנית.

### OperationsReportForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "דו"ח פעילות". באמצעות טופס זה ניתן להפיק דו"ח אודות סוגי השירותים השונים הקיימים בעסק. דו"ח זה מציג כמה קריאות שירות קיימות עבור כל סוג שירות בכל אחד מהסטאטוסים הבאים: "Unassigned", "In Progress" ו-"Completed". טופס זה מאפשר להפיק את הדו"ח על פי טווח של תאריכים ושעות ובנוסף על פי שיוך לעובד מסוים.

### OrdersForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "ניהול הזמנות ציוד". באמצעות טופס זה ניתן לייצר הזמנות ציוד מול ספק חיצוני. טופס זה מאפשר גם לצפות או לערוך הזמנות ציוד קיימות. הרשאת העריכה נקבעת על פי סטאטוס ההזמנה. באמצעות טופס זה ניתן לשנות את סטאטוס ההזמנה ובכך להגדיל כמויות של מוצרים קיימים במלאי העסק.

### OrdersReportForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "דו"ח הזמנות". באמצעות טופס זה ניתן להפיק דו"ח הזמנות. טופס זה מאפשר להפיק את הדו"ח על פי טווח של תאריכים או/ו על פי ספק או/ו על פי סטאטוס הזמנה. בדו"ח זה ניתן לראות רשימה של כל ההזמנות המתאימות ובנוסף את העלות הכוללת שלהן.

### PaymentForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית PopupForm. מחלקה זו מייצגת את הטופס: "קבלת התקבולים". באמצעות טופס זה ניתן לקלוט את תשלום הלקוח בעקבות רכישה שביצע. בטופס זה ניתן לבחור בין שתי אפשרויות תשלום: מזומן או אשראי. קליטת אמצעי התשלום ואישורו ייבצעו מספר פעולות כגון: הפקת חשבונית, פתיחת קריאות שירות עבור השירותים הנרכשים, גריעה מהמלאי ויצירת תיעוד בהיסטוריית לקוח.

### PricingForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "תמחור מוצרים". באמצעות טופס זה ניתן לשנות את ערך המע"מ ובנוסף להוריד/להעלות את התעריפים של המוצרים כולם באחוז כלשהו. טופס זה מאפשר להציג את השינויים לפני שמירת הנתונים.

### ReportsMainForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית MenuForm. מחלקה זו מייצגת את הטופס: "מסך דו"חות". באמצעות טופס זה ניתן לנווט בין טפסי הדו"חות שאליהם קיימת גישה. ההרשאות בטופס זה נקבעות על סמך הרשאותיו של המשתמש הנוכחי, ובהתאם להרשאות אלו, יופיעו הכפתורים המתאימים שדרכם ניתן לנווט לטפסיי הדו"חות הרלוונטיים.

### SalesForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "ניהול מכירות". באמצעות טופס זה ניתן לבצע מכירה ללקוח או להציע הצעת מחיר. כדי לבצע מכירה ללקוח יש לקשר לקוח ספציפי ולאחר מכן לבחור את המוצרים או/ו השירותים שאותם הלקוח מעוניין לרכוש. ניתן לראות בכל שלב את העלות הכוללת של ההצעה. עלות זאת כוללת את המע"מ ואת ההנחות השונות במידה וקיימות.

### SalesReportForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "דו"ח מכירות". באמצעות טופס זה ניתן להפיק דו"ח המאגד את סך הכמויות שנמכרו מכל אחד מסוגי המוצרים / שירותים. דו"ח זה מאפשר להציג מכירות שבוצעו בטווח תאריכים נתון ו/או בטווח שעות נתון.

### ServiceCallForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "קריאת שירות". טופס זה מאפשר מעקב אחר קריאת שירות. באמצעות טופס זה ניתן להקצות את קריאת השירות לאחד מעובדי הצוות, לעדכן סטאטוס ולנהל את היסטוריית קריאת השירות.
הרשאת ההקצאה לאחד מעובדי הצוות בטופס זה נקבעת על סמך הרשאותיו של המשתמש הנוכחי.

### ServiceCallHistoryEventForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית PopupForm. מחלקה זו מייצגת את הטופס: "אירוע היסטוריית קריאת שירות". טופס זה מאפשר לצפות / לערוך את ההערות של רשומת ההיסטוריה הנוכחית.

### ServiceCallReportForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "דו"ח קריאת שירות". באמצעות טופס זה ניתן להפיק דו"ח אודות כמות קריאות השירות בהם טיפל או מטפל כל אחד מעובדי הצוות בטווח תאריכים נתון ו/או בטווח שעות נתון. טופס זה מציג עבור כל אחד מעובדי הצוות מהו מספר קריאות השירות שנפתחו באותה תקופת זמן, והביאו אותן לסגירה (Completed). בנוסף, טופס זה מציג עבור כל אחד מעובדי הצוות מהו מספר קריאות השירות שנפתחו באותה תקופת זמן, והן עדיין בטיפול (In Progress).

### UserManagementForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "ניהול משתמשים". באמצעות טופס זה ניתן להוסיף או להסיר משתמשים מהמערכת. טופס זה מציג את כל המשתמשים הקיימים במערכת אשר נמצאים במצב פעיל. לחיצה כפולה על אחת הרשומות תפתח את הטופס שמאפשר לצפות / לערוך את הפרטים המלאים של המשתמש.

### WorkDistributionForm.xaml.cs

מחלקה היורשת מהמחלקה האבסטרקטית NavigationalForm. מחלקה זו מייצגת את הטופס: "חלוקת עבודה". באמצעות טופס זה ניתן לחפש אחר קריאות שירות הקיימות במערכת. ניתן לבצע סינון בעת החיפוש על פי תאריכים, סוגי שירותים, סטאטוס או/ו שיוך. לחיצה כפולה על אחת הרשומות תפתח את טופס קריאת השירות עם הפרטים המלאים של אותה הקריאה.

ראש הצוות רשאי לראות את כל קריאות השירות הקיימות במערכת. עובד רשאי לראות רק את קריאות השירות המשתייכות אליו. הרשאה זו נקבעת על סמך הרשאותיו של המשתמש הנוכחי.

## דפוסי עיצוב (Design Patterns)

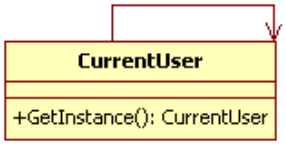
דפוסי העיצוב בהם ייעשה שימוש בפרויקט הם:

### Singleton

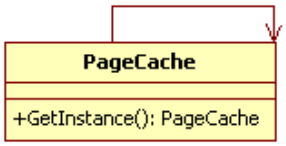
תבנית שתפקידה לוודא שקיים למחלקה מופע אחד בדיוק, בנוסף התבנית מספקת מצביע גלובלי שיאפשר לגשת למופע זה. שימוש בתבנית זו ימנע מהיווצרותן של שגיאות לוגיות הנגרמות כתוצאה מיצירה של כמה מופעים עבור מחלקות שנדרש עבורן מופע אחד לכל היותר.

בתבנית זו נעשה שימוש במחלקות הבאות:

1. CurrentUser – מחלקה זו מייצגת את המשתמש הנוכחי שמחובר למערכת. מכיוון שבכל רגע נתון יכול אך ורק משתמש אחד להיות מחובר אל המערכת, נעשה שימוש בתבנית Singleton במחלקה זו. שימוש בתבנית העיצוב Singleton יעזור לנו לוודא שלא קיים יותר ממופע אחד של המחלקה CurrentUser. למופע זה אנו ניגש מחלקים שונים בתכנית כדי לבדוק את הרשאותיו של המשתמש הנוכחי. בנוסף ייעשה שימוש בתכונות נוספות של האובייקט, לדוגמה: הצגת שמו המלא של המשתמש הנוכחי בסרגל הכלים התחתון.




2. PageCache – מחלקה זו מייצגת את זיכרון המטמון של הדפים במערכת. מכיוון שבכל רגע נתון יכול להיות רק זיכרון מטמון אחד במערכת, נעשה שימוש בתבנית Singleton במחלקה זו. שימוש בתבנית העיצוב Singleton יעזור לנו לוודא שלא קיים יותר ממופע אחד של המחלקה PageCache. למופע זה אנו ניגש מחלקים שונים בתכנית כדי לגשת לדף הנוכחי, לדף הקודם ולכל חלונות ה-Popup אשר פתוחים כרגע במערכת.

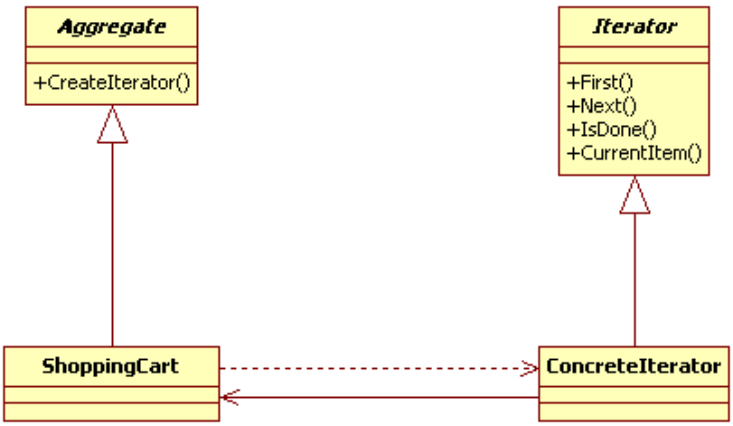


## Iterator

תבנית המספקת דרך לגשת באופן סדרתי לאיברים של אובייקט מורכב מבלי לחשוף את הייצוג הפנימי שלו. התבנית Iterator מפרידה בין הממשק של קריאת הנתונים למבנה הנתונים עצמו, ובכך מאפשרת לשנות את מבנה הנתונים ללא שינוי הקוד של המשתמש.

בתבנית זו נעשה שימוש במחלקות הבאות:

1. ShoppingCart – מחלקה זו מייצגת את עגלת הקניות (סל קניות) של המוצרים והשירותים במערכת. המשתמשים של מחלקה זו צריכים לבצע פעולות כגון: סריקה של כל הפריטים בעגלה לצורך חישוב עלות כוללת, סריקה של כל המוצרים בעגלה לצורך גריעה מהמלאי ברגע שהלקוח שילם על הרכישה וכו'. כל המשתמשים במחלקה זו אינם צריכים לדעת מהו מבנה הנתונים שבו מאוחסנים הפריטים. משתמשים אלו יכולים להסתפק ב-Iterator שבאמצעותו יוכלו לעבור על כל הפריטים בעגלה ללא חשיפה של הייצוג הפנימי שלה.
המחלקה ShoppingCart תשתתף בתבנית Iterator בתפקיד – ConcreteAggregate אשר יורש מהמחלקה האבסטרקטית Aggregate.



## Strategy

תבנית המגדירה משפחה של אלגוריתמים, מכמסת (Encapsulate) כל אחד מהם, ומאפשרת להחליף את השימוש בהם. כלומר, התבנית מאפשרת לשנות את האלגוריתמים באופן עצמאי ללא תלות במשתמשים של המחלקה.

בתבנית זו נעשה שימוש במחלקות הבאות:

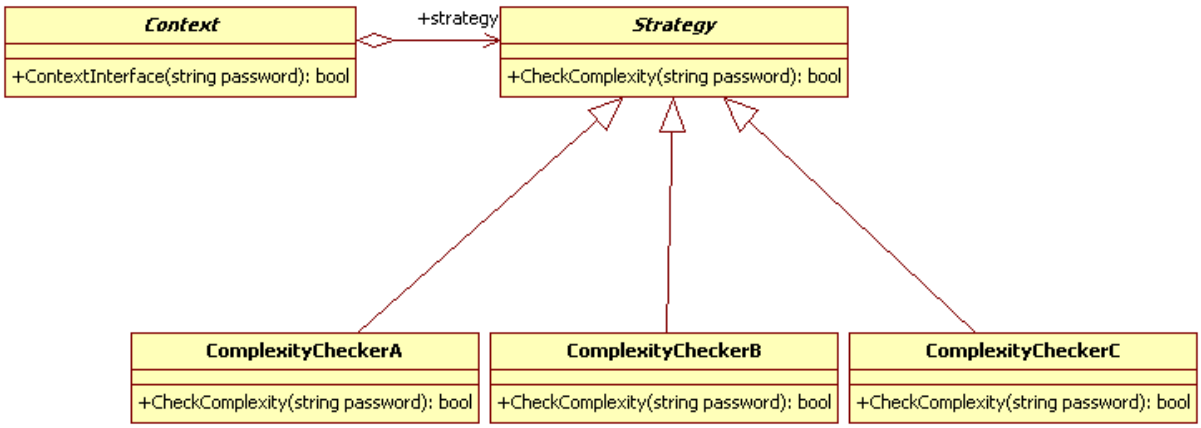
1. ComplexityCheckerA – מחלקה זו מייצגת את אלגוריתם בדיקת המורכבות הראשון. אלגוריתם זה מקבל כקלט מחרוזת ובודק האם המחרוזת מכילה לפחות 2 (מוגדר כקבוע - MINIMUM_AMOUNT) תווים אשר כל אחד מהם שייך לקבוצה אחרת, כאשר הקבוצות הן: אותיות קטנות, אותיות גדולות, ספרות וסימנים אחרים.

2. ComplexityCheckerB – מחלקה זו מייצגת את אלגוריתם בדיקת המורכבות השני. אלגוריתם זה מקבל כקלט מחרוזת ובודק האם המחרוזת לא מכילה שני תווים זהים עוקבים. כלומר, על פי אלגוריתם זה, סיסמא מורכבת הינה סיסמא אשר לא מכילה שני תווים זהים עוקבים.

3. ComplexityCheckerC – מחלקה זו מייצגת את אלגוריתם בדיקת המורכבות השלישי. אלגוריתם זה מקבל כקלט מחרוזת ובודק האם המחרוזת לא מכילה שני תווים עוקבים השייכים לאותה קבוצה (אותיות קטנות, אותיות גדולות, ספרות וסימנים אחרים). כלומר, על פי אלגוריתם זה, סיסמא מורכבת הינה סיסמא אשר לא מכילה שני תווים עוקבים השייכים לאותה קבוצה.

כל אחד משלושת האלגוריתמים הינו משתתף בתפקיד ConcreteStrategy של התבנית Strategy.

כימוס האלגוריתמים לאובייקטים שונים יאפשר לנו להחליף את האלגוריתמים באופן עצמאי ללא תלות במשתמשים של המחלקה. כך נוכל להוסיף בדיקות מורכבות סיסמא נוספות, לשלב כמה בדיקות שיתבצעו ברצף או להחליף ביניהן ללא תלות במשתמשים. שימוש בתבנית יאפשר לנו גמישות לשינויים עתידיים במידה ונרצה להוסיף למשל אלגוריתם מורכבות נוסף.




## Observer

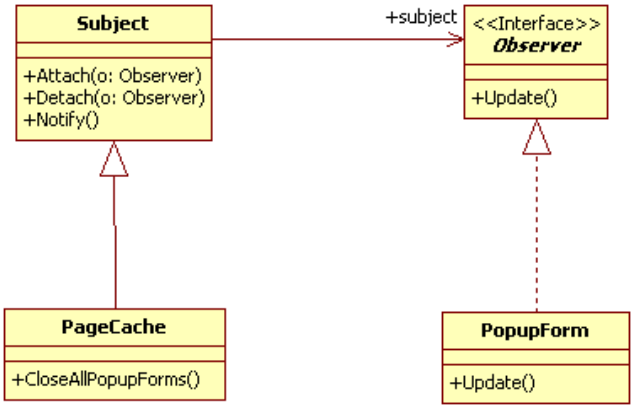
תבנית המגדירה תלות אחד-לרבים בין האובייקטים, כאשר מצב של אובייקט אחד משתנה, כל שאר האובייקטים שתלויים בו מקבלים התראה על השינוי ומתעדכנים באופן אוטומטי.

בתבנית זו נעשה שימוש במחלקות הבאות:

1. PageCache – מכיוון שמחלקה זו מייצגת את זיכרון המטמון של הדפים במערכת, מחלקה זו צריכה להכיל רשימה של הפניות (References) לכל חלונות ה-Popup שפתוחים כרגע במערכת. כאשר אנו מבצעים סגירה של חלון ניווט אשר דרכו פתחנו מספר חלונות מסוג Popup, יש לבצע סגירה מסודרת של כל חלונות ה-Popup שפתוחים כרגע, מכיוון שאינם רלוונטיים עוד. לפיכך, מחלקה זו תשתתף בתפקיד ConcreteSubject של התבנית Observer. מחלקה זו תירש מהמחלקה האבסטרקטית Subject וכך תקבל ממנה בירושה את הפעולות הבאות: Attach – להוספת אובייקטים מסוג Observer, Detach – להסרת אובייקטים מסוג Observer ו-Notify – לקריאה לכל האובייקטים הרשומים להתעדכן לפי המצב העדכני. כאשר נסגור את החלון הנוכחי, ה-PageCache יישנה את מצבו מ-REGULAR_STATE ל-EXIT_STATE, ולאחר מכן ייקרא ל-Notify כדי לעדכן את כל חלונות ה-Popup שנרשמו ל-PageCache כדי שייבצעו סגירה.

2. PopupForm – מחלקה זו מייצגת את כל טפסי ה-Popup במערכת. מחלקה זו תממש את הממשק Observer המכיל שיטה אחת בשם Update. שיטה זו תעדכן את מצב הטופס למצבו הנוכחי של ה-PageCache. כאשר האובייקט מסוג PopupForm מעדכן את מצבו, הוא בודק האם מצב זה הינו EXIT_STATE, אם כן, אזי החלון יבצע סגירה, אחרת ימשיך לפעול כרגיל. מחלקה זו מממשת את הממשק Observer ולכן היא משתתפת בתפקיד ConcreteObserver של התבנית Observer.

באמצעות שימוש בתבנית זו אנו מונעים צמידות חזקה בין המחלקות PageCache ו-PopupForm. כך בעצם נוכל בעתיד להגדיר סוגי טפסים נוספים אשר יממשו את הממשק Observer ויירשמו באמצעות Attach לאובייקט PageCache כדי להתעדכן מולו לגבי מצבו הנוכחי ולפעול בהתאם. כלומר, תבנית זו מאפשרת לנו גמישות לשינויים עתידיים.




## שינויים עתידיים אפשריים

מערכת זו תוכננה ועוצבה כך ששינויים עתידיים יהיו קלים ליישום. נסכם כעת כיצד הגמישות לשינויים באה לידי ביטוי ואילו שינויים ניתן יהיה לבצע בעתיד.

1. מערכת זו בנויה על סמך מודל 3-Tier, שבו המערכת מופרדת לשלוש שכבות. הפרדה זו תאפשר לנו בעתיד להחליף את הממשק בין שתי שכבות מבלי להחליף את הממשק בין שתי השכבות האחרות, וכן, להחליף את המימוש של אחת השכבות מבלי להחליף את המימוש של השכבות האחרות. כך למשל, אם נחליט שאנו מעוניינים לעבוד מול מסד נתונים השונה מ-SQL Server נוכל להחליף בקלות רק את המימוש של שכבת ה-Dal מבלי לפגוע במימוש של השכבות האחרות. דוגמה נוספת לשינוי היא החלפת ממשק המשתמש מ-ממשק מבוסס חלונות לממשק אינטרנטי (Web) המכיל דפי HTML. גם כאן נוכל להחליף את המימוש של שכבת התצוגה מבלי לפגוע בשכבות האחרות.

2. היררכיית הירושה הקיימת בין הטפסים השונים במערכת תאפשר לנו להחיל שינויים על כל הטפסים במערכת (באמצעות המחלקה BasicForm) או רק על חלקם (באמצעות המחלקות: NavigationalForm, MenuForm או PopupForm). כך לדוגמה, אם נחליט שאנו רוצים לקבוע גודל אחיד לכל טפסי ה-Popup במערכת, נוכל לבצע שינוי זה במחלקה PopupForm ללא הצורך לשנות טופס אחר טופס.

3. לכל אורך התוכנית הוגדרו קבועים עבור הערכים אשר בהם אנו עושים שימוש. קבועים אלו יאפשרו לנו לשנות את הערכים במקום אחד בלבד ללא הצורך לבצע חיפוש בתכנית כולה, כדי למצוא היכן משתמשים בערכים אלו. כך למשל, אם נרצה בעתיד לשנות את סיסמתו של מנהל המערכת, נוכל לשנות את ערך הקבוע: PASSWORD_ADMIN.

4. מימוש המחלקה ShoppingCart באמצעות התבנית Iterator מאפשרת לנו לשנות את מבנה הנתונים של הפריטים בעגלת הקניות ללא הצורך לשנות את המימוש של המשתמשים במחלקה זו. מבנה הנתונים הנבחר לצורך מימוש עגלת הקניות הוא רשימה, מכיוון שמבנה נתונים זה הינו פשוט ומתאים לצרכים של עסקים קטנים (בדרך כלל עגלת הקניות תכיל עד 50 פריטים). אם בעתיד נגלה צורך אצל משתמשי המערכת בעגלת קניות שתכיל אלפים או יותר פריטים לעגלה, נשקול להחליף את מבנה הנתונים במבנה יעיל יותר אשר יאפשר לנו לבצע חישובים כגון עלות כוללת בזמן ריצה יעיל יותר.

5. מימוש אלגוריתמי בדיקת מורכבות הסיסמא באמצעות התבנית Strategy מאפשרת לנו להחליף את האלגוריתמים או להוסיף חדשים ללא הצורך לשנות את המימוש במשתמשים של המחלקה. כך בעצם נוכל להתאים בקלות גרסאות שונות לעסקים שונים אשר לכל אחד מהם קיימת דרישה שונה לבדיקת המורכבות. לדוגמה: עסק כגון מספרה אשר מכיל 2 עובדים בלבד (מנהל המספרה שהוא גם מנהל המכירות וראש הצוות, וספר נוסף) אינו צריך מורכבות גבוהה של סיסמאות ולכן יסתפק רק באלגוריתם בדיקת המורכבות הראשון. לעומת זאת, עסק כגון מעבדת מחשבים אשר מכיל את כל סוגי המשתמשים כאשר על ראש הצוות חל איסור לצפות בפרטי הלקוחות, ידרוש מורכבות גבוהה של סיסמאות ולכן נשייך לשיטה checkComplexity במחלקה UserValidator את שלושת האלגוריתמים שיבדקו ברצף את מורכבות הסיסמא.

6. מימוש המחלקות PageCache ו-PopupFrom באמצעות תבנית העיצוב Observer מאפשרת לנו גמישות רבה לשינויים. נוכל להחליט בעתיד כי קיים צורך לסוגים שונים של טפסים (שונים מ-PopupFrom) להתעדכן מול ה-PageCache לגבי מצבו הנוכחי ולפעול בהתאם. נוכל לבצע שינוי זה בקלות ע"י כך שנממש את המחלקה המבוקשת באמצעות הממשק Observer כדי שה-PageCache יוכל לבצע רישום של דפים ממחלקה זו ולעדכן אותם כאשר מצבו משתנה. כך למשל, נוכל להוסיף מצב חדש שנקרא DISABLE_STATE אשר יקרא לקבוצה מסוימת של טפסים לעבור למצב לא פעיל כאשר מתרחש אירוע חריג כלשהו בזיכרון המטמון.

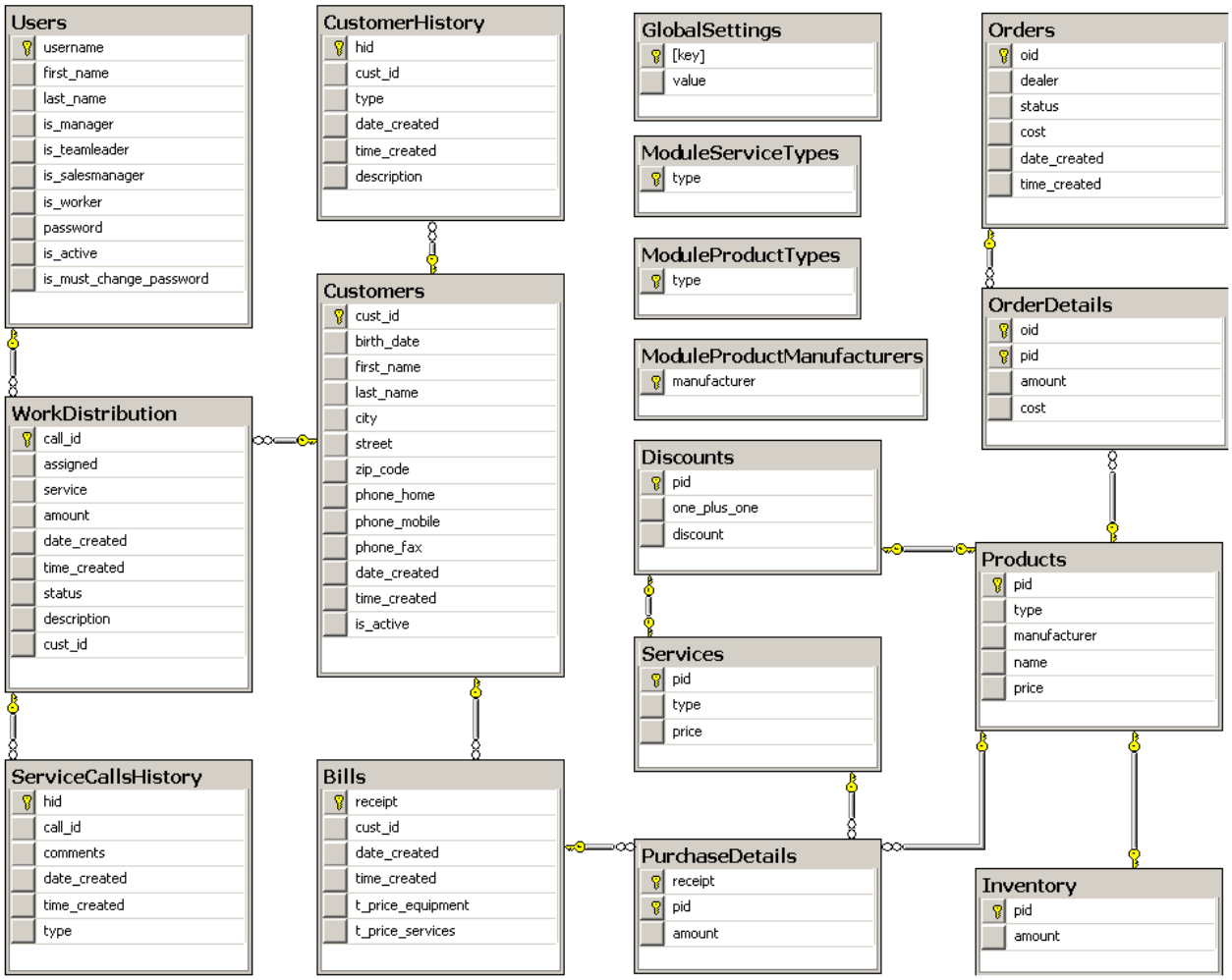
7. במערכת זו קיימים שני סוגים של פריטים אשר ניתן לרכוש: מוצרים ושירותים. הגדרת שתי המחלקות עבור פריטים אלו תחת המחלקה האבסטרקטית Purchasable בהיררכיית הירושה של המערכת, מאפשרת לנו לרכז את כל השיטות והתכונות המשותפות לפריטים אלו במחלקה זו. עיצוב זה גמיש לשינויים מכיוון שאנו יכולים לשנות את מימוש השיטות במחלקה זו, וכך השינוי יבוא לידי ביטוי גם עבור מוצרים וגם עבור שירותים במערכת. לדוגמה, השיטה GetPriceIncVatAndDiscount אשר מחזירה את מחיר המוצר לאחר

הוספת המע"מ וחישוב ההנחה, מוסיפה קודם כל את המע"מ ורק לאחר מכן מחשבת את ההנחה. אם נרצה בעתיד לשנות את המימוש של השיטה כך שקודם תחושב ההנחה ורק לאחר מכן יתווסף המע"מ, אז שינוי זה יהיה תקף גם למוצרים וגם לשירותים. נוסף על כך, עיצוב זה מאפשר לנו להוסיף בעתיד סוג נוסף של פריט אשר ניתן לרכוש שאינו מוצר ואינו שירות. מכיוון שעגלת הקניות מכילה רשימה של אובייקטים מטיפוס Purchasable, אז לא נצטרך לשנות את המימוש של עגלת הקניות ויהיה ניתן להוסיף לעגלת הקניות פריטים מהסוג החדש.

8. הגדרת המערכת כולה כמערכת גנרית המתאימה לסוגים רבים של עסקים קטנים ומימוש הפונקציונאליות של טעינת קובץ המודול מאפשרים גמישות רבה לשינויים עתידיים. ניתן להוסיף על הפונקציונאליות של טעינת קובץ המודול, שבנוסף לטעינת רשימות היצרנים, סוגי השירותים וסוגי המוצרים, שטעינת הקובץ תבצע התאמות נוספות לעסק המתאים. לדוגמה, ניתן לבנות טפסים נוספים אשר רלוונטיים רק לסוג עסק מסוים אשר יהפכו להיות זמינים רק כאשר ייטען קובץ המודול המתאים של אותו העסק.

## נספח: דיאגראמות

### דיאגראמת מסד הנתונים (Data Base Diagram)



## דיאגראמות מחלקות (Class Diagrams)

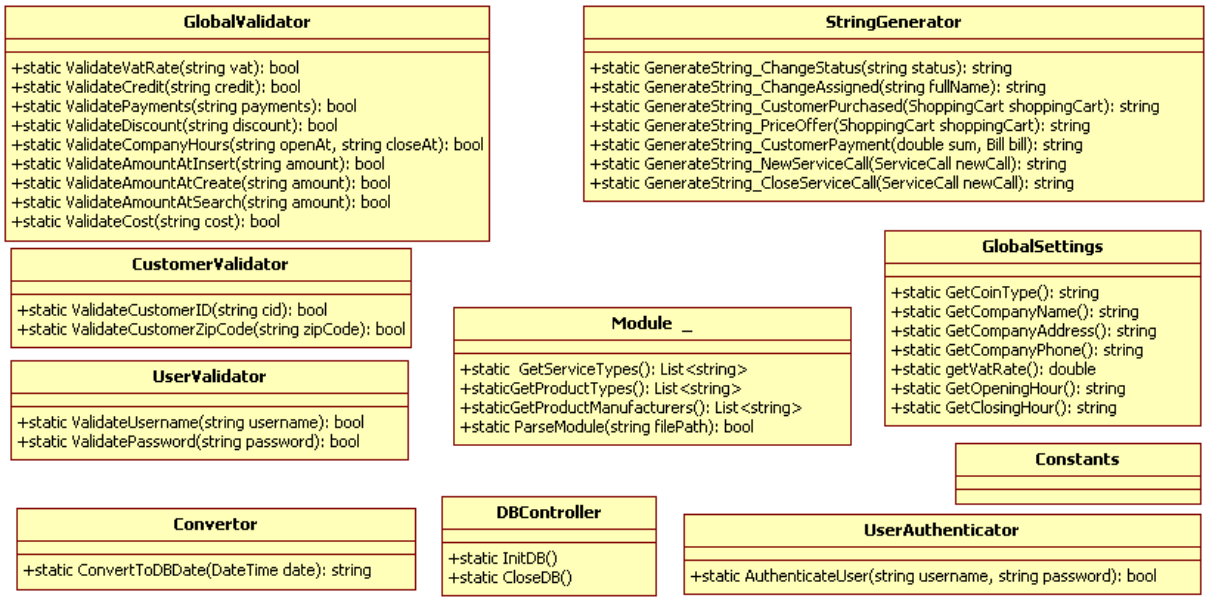
הערות כלליות עבור דיאגראמות המחלקות:

1. הסימן + מציין גישה ציבורית (Public).
2. הסימן – מציין גישה פרטית (Private).
3. הסימן # מציין גישה מוגנת (Protected).
4. הבנאים אינם מופיעים בדיאגראמות.
5. שיטות עזר פרטיות אינן מופיעות בדיאגראמות כדי לא להעמיס יתר על המידה.

### דיאגראמת מחלקות עבור המחלקות הכלליות בשכבת ה-Bl

הערות:

1. כל המחלקות אשר מופיעות בדיאגראמה זו הינן כלליות ומכילות שיטות סטטיות בלבד.



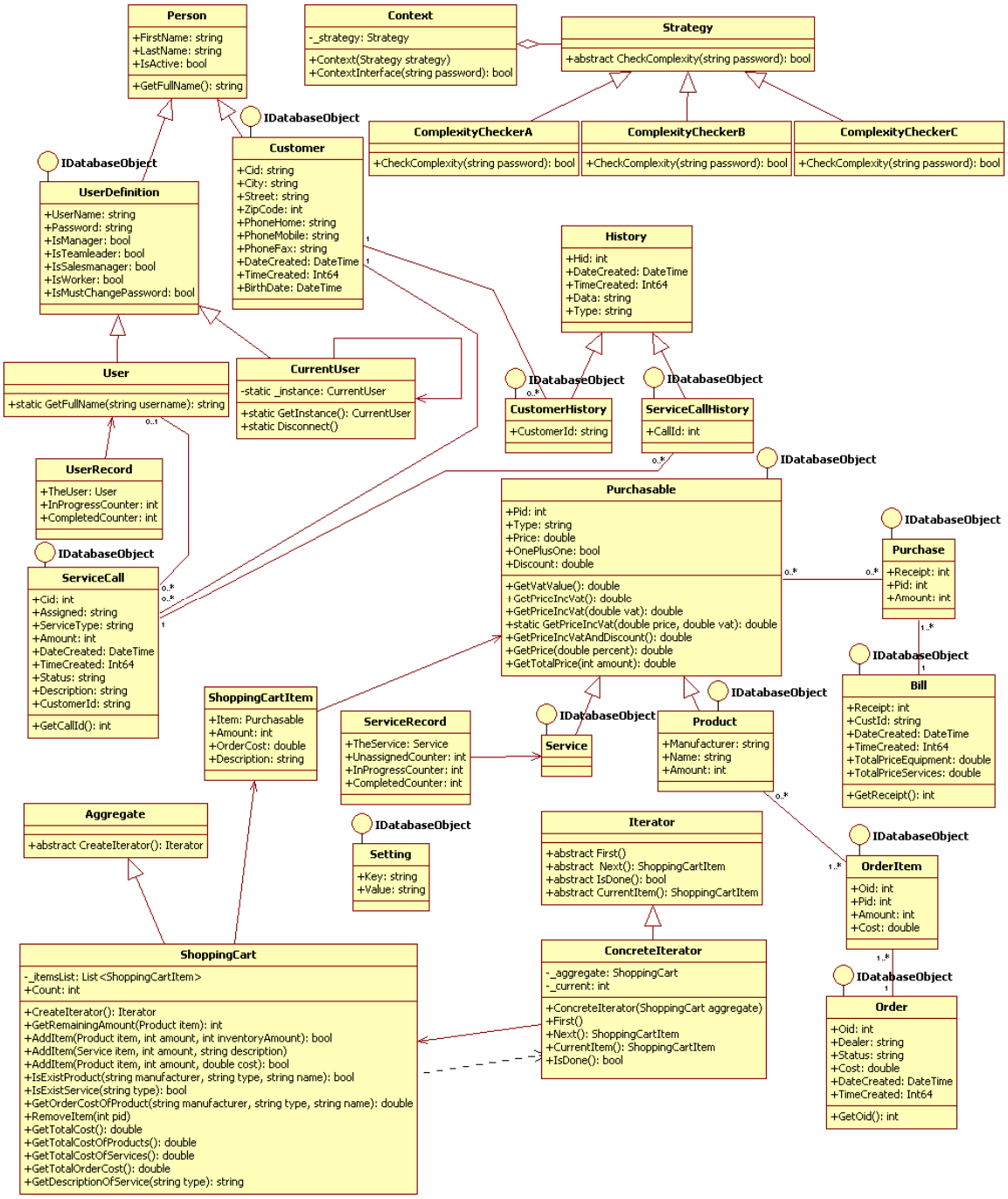
### דיאגראמת מחלקות עבור האובייקטים בשכבת ה-Bl

הערות:

1. כל מחלקה המממשת את הממשק IDatabaseObject, מכילה את השיטות הבאות:

bool InsertToDB(); bool RemoveFromDB(); bool UpdateDB(); bool IsExistInDB();

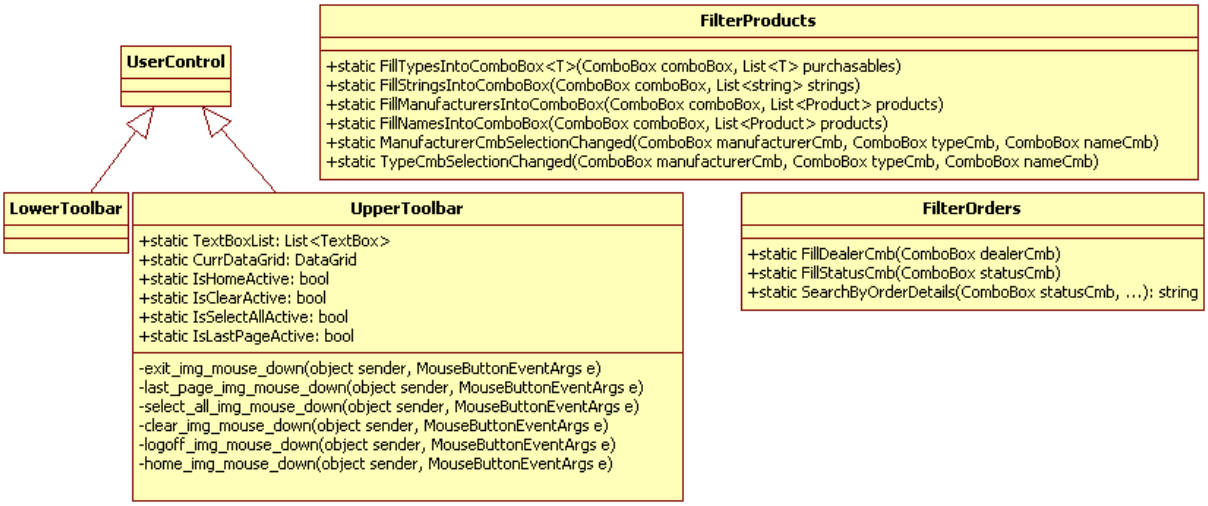
שיטות אלה לא מופיעות בדיאגראמה.



## דיאגראמת מחלקות עבור המחלקות הכלליות וה-UserControls בשכבת ה-Pl

הערות:

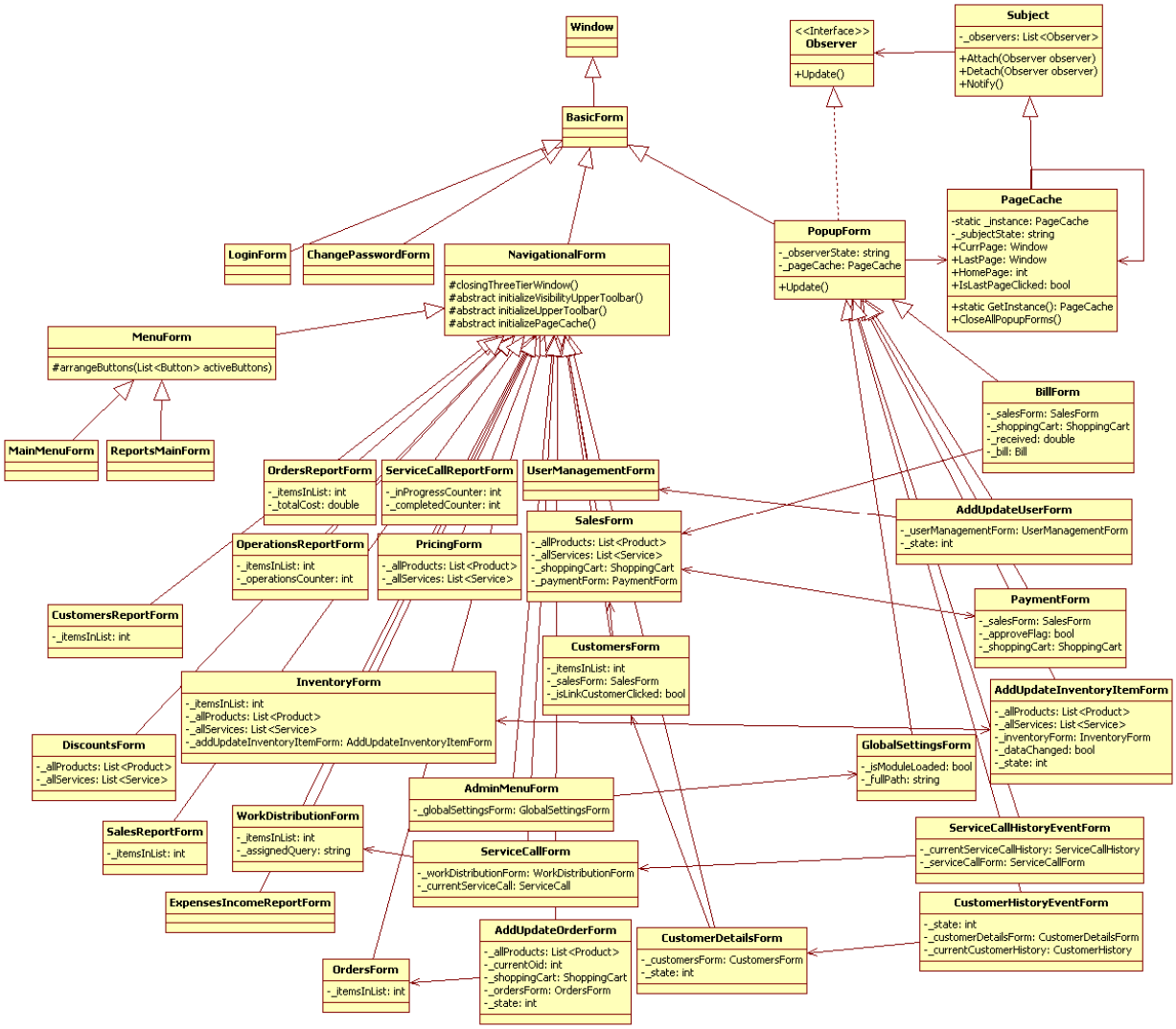
1. בדיאגראמה זו ניתן לראות ששתי המחלקות אשר מייצגות את סרגלי הכלים במערכת יורשות מהמחלקה UserControl.
2. המחלקות FilterOrders ו-FilterProducts הינן מחלקות כלליות המכילות שיטות סטטיות בלבד.



## דיאגראמת מחלקות עבור הטפסים בשכבת ה-Pl

הערות:

1. עקב ריבוי השיטות בכל מחלקה של טופס קונקרטי, השיטות של מחלקות אלה אינן מופיעות בדיאגראמה כדי לא להעמיס יתר על המידה.
2. בדיאגראמה זו ניתן לראות את היררכיית הירושה בין כל הטפסים במערכת.
3. בדיאגראמה זו ניתן לראות את הצמידות בין טפסים מסוימים הקשורים ביניהם מבחינה לוגית, דבר זה בא לידי ביטוי כאשר מחלקה אחת מחזיקה הפניה (Reference) למחלקה השנייה.
4. כל הטפסים במערכת יורשים מהמחלקה האבסטרקטית BasicForm היורשת מהמחלקה האבסטרקטית Window.



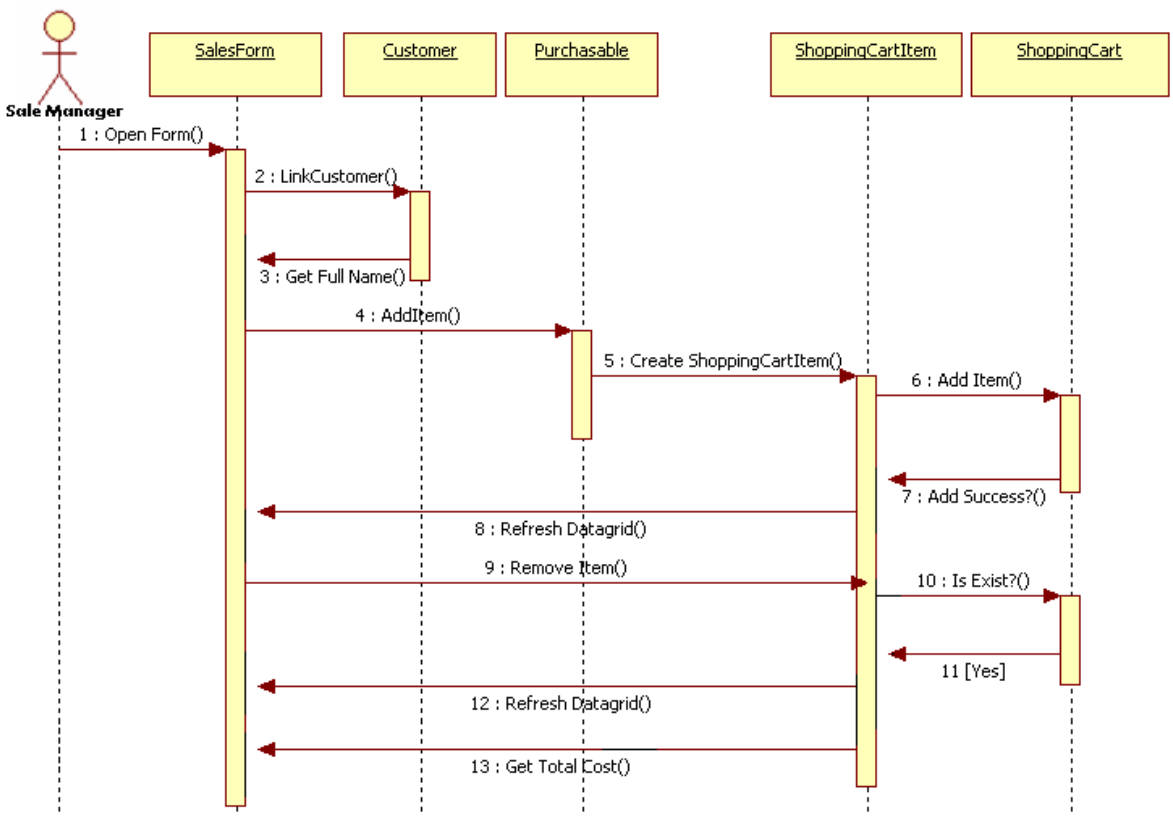
## דיאגראמות רצף (Sequence Diagrams)

להלן מספר דיאגראמות רצף המתארות חלק מהתהליכים המרכזיים במערכת.

דיאגראמות אלה מתארות באופן כללי תהליכים מרכזיים במערכת, ייתכנו שינויים והבדלים בין דיאגראמות אלה לבין המימוש הסופי מכיוון שבפועל רצף האירועים ושליחת ההודעות בין האובייקטים השונים הינו מורכב יותר. לפיכך, דיאגראמות אלה מכילות אך ורק את האובייקטים וההודעות המשמעותיים ביותר בתהליך וזאת מתוך מטרה להמחיש את עיקרי הדברים ללא העמסת מידע יתר על המידה.

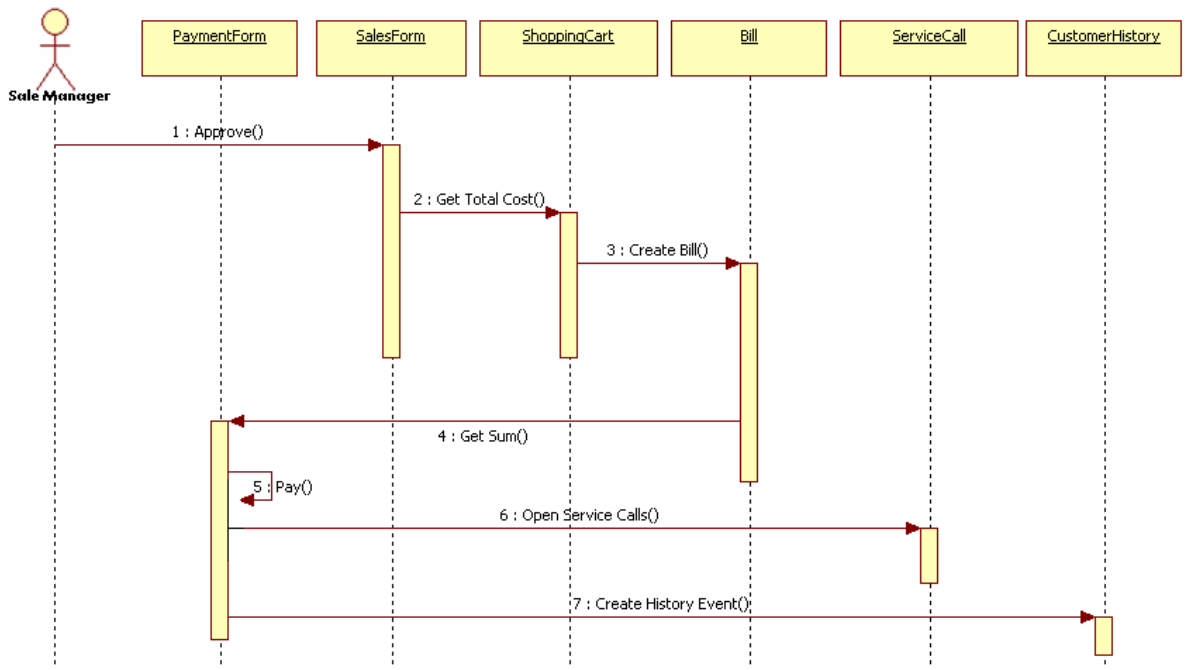
### דיאגראמת רצף עבור הפקת הצעת מחיר ללקוח

דיאגראמה זו מתארת את תהליך הפקת הצעת המחיר ללקוח. מנהל המכירות פותח את טופס המכירות, מקשר אליו את הלקוח המבוקש (אופציונאלי), מוסיף פריטים (מוצרים ושירותים) לעגלת הקניות, מסיר פריטים מסוימים מעגלת הקניות. לבסוף, טופס המכירות יציג למנהל המכירות את תוכן העגלה ואת העלות הכוללת.



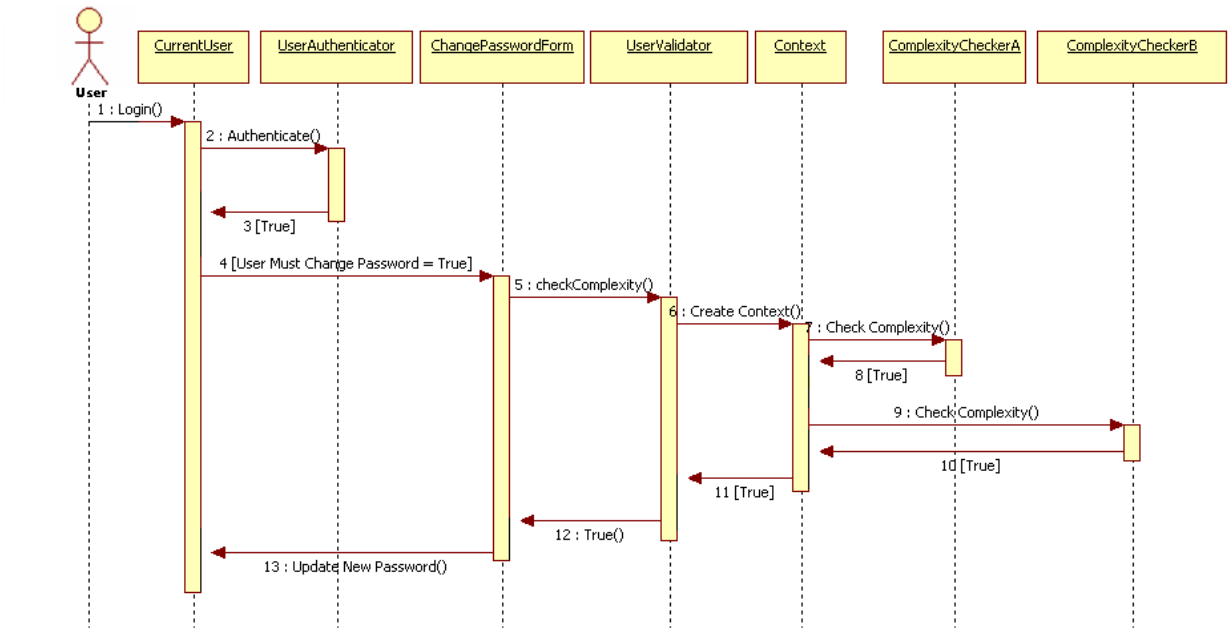
## דיאגראמת רצף עבור ביצוע רכישה ע"י לקוח

דיאגראמה זו מתארת את תהליך הרכישה. מנהל המכירות מאשר את הצעת המחיר ע"י לחיצה על כפתור ה-Approve, האובייקט Bill נוצר בהתאם לתוכן העגלה, טופס קבלת התקבולים נפתח, מנהל המכירות מזין את אמצעי התשלום, ולבסוף אישור התשלום יגרום לפתיחת קריאות השירות הרלוונטיות וליצירת אירוע היסטוריית לקוח מתאים.



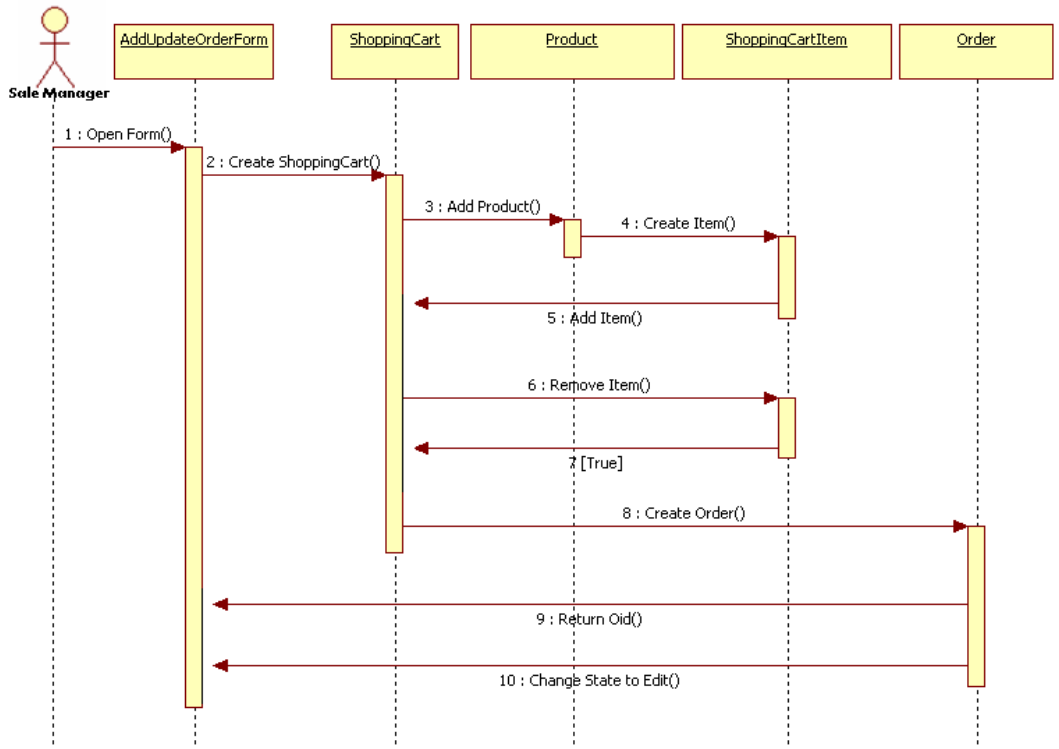
## דיאגראמת רצף עבור החלפת סיסמא של משתמש

דיאגראמה זו מתארת את החלפת הסיסמא במערכת. משתמש כלשהו מקליד שם משתמש וסיסמא בטופס ההתחברות, נוצר אובייקט מסוג CurrentUser, מתבצע אימות של שם המשתמש והסיסמא, אם ערכו של הדגל "המשתמש חייב לשנות את סיסמתו" הינו 'True' אז ייפתח טופס שינוי הסיסמא, המשתמש מקליד את סיסמתו החדשה, מתבצעות מספר בדיקות מורכבות, ולבסוף אם הבדיקות עברו בהצלחה אז סיסמת המשתמש תתעדכן במסד הנתונים וטופס המסך הראשי ייפתח.



## דיאגראמת רצף עבור יצירת הזמנה חדשה במערכת

דיאגראמה זו מתארת את תהליך יצירת הזמנה חדשה במערכת. מנהל המכירות פותח את הטופס AddUpdateOrderForm, נוצר אובייקט מסוג ShoppingCart, מנהל המכירות מוסיף מוצרים אל עגלת הקניות תוך כדי שהוא קובע את עלות המוצר, מנהל המכירות מסיר מוצרים מעגלת הקניות, לחיצה על כפתור ה-Save יוצרת אובייקט מסוג Order, הטופס AddUpdateOrderForm משנה את מצבו למצב עריכה כאשר כל פרטי ההזמנה שנוצרה מופיעים בטופס.



## דיאגראמת רצף עבור הקצאה וטיפול בקריאת שירות

דיאגראמה זו מתארת את תהליך ההקצאה והטיפול בקריאות שירות לאחר שנפתחו במערכת. ראש הצוות פותח את הטופס ServiceCallFrom ע"י לחיצה כפולה על אחת הקריאות שטרם שויכו, ראש הצוות קורא את פרטי הקריאה המופיעים בטופס, ראש הצוות משייך את הקריאה לאחד העובדים, נוצר אירוע היסטוריה מסוג שיוך לעובד, ראש הצוות משנה את הסטאטוס של הקריאה, נוצר אירוע היסטוריה מסוג שינוי סטאטוס. עובד הצוות פותח את הטופס ServiceCallFrom ע"י לחיצה כפולה על אחת הקריאות אשר שויכו אליו, עובד הצוות קורא את פרטי הקריאה המופיעים בטופס, מטפל בקריאה, עובד הצוות משנה את הסטאטוס של הקריאה בהתאם לצורך, נוצר אירוע היסטוריה מסוג שינוי סטאטוס.