בס"ד

שלב 1 ב' – יצירת ישויות הנתונים

הנחיות כלליות:

- חובה לפרמט את הקוד בהתאם - הזחות, שורות רווח, ריווח בתוך השורות
- כל השמות (של המחלקות, השדות, התכונות, המתודות) חייבות להיות באנגלית ובעלי משמעות בהתאם לתפקיד המחלקה, המבנה, התכונה, המתודה
- שמות כל הסוגים, כמו כן השדות, התכונות, והמתודות עם הרשאה public יש להגדיר בפורמט PascalCase
- שמות כל השדות והתכונות בהרשאה private או internal יש להגדיר בפורמט _camelCase
- שמות כל המתודות בהרשאה private או internal, הפרמטרים, והמשתנים המקומיים יש להגדיר בפורמט camelCase
- חובה לתעד את כל הסוגים, המתודות, והתכונות בעזרת תיעוד מפורמט (///)
- חובה לתעד את הקוד אם איננו ברור מאליו ממבנה הקוד (למשל במקרה של חישובים מתמטיים שאינם בסיסיים)

ישויות הנתונים:

צרו בפרויקט DalFacade תת-תיקיה בשם DO:

לחיצה ימנית על הפרויקט -> Add -> New Folder

מלבד קבצי הגדרת הישויות, בספריה זו גם נשמור גם את קובץ האנומרציות הנחוצות בהגדרת ישויות הנתונים. בשלב זה נוסיף קובץ אנומרציות ריק בשם :**Enums.cs**

להוספת קובץ אנומרציות ריק לספריית **DO**:

1. עמדו עם עכבר ימני על הספריה **DO**
2. לחצו על **Add → Class**
3. כתבו את שם הקובץ **Enums.cs** ולחצו על **Add**
4. יווצר לכם קובץ cs בשם **Enums.cs**
5. הסירו מראש הקובץ את רשימת הusing- הקיימת, משום שאין צורך לציין using גלובליים. והם יצוינו אוטומטית בקובץ נפרד.
6. שימו לב שמרחב השמות של הקובץ נקרא באופן אוטומטי **DO** כמו שם הספריה שייצרתם.
7. הוסיפו סימן נקודה-פסיק "**;**" בסוף השורה: "namespace DO;"
   a. שימו לב שהצומדיים של הבלוק {...} של מרחב השמות נמחקים אוטומטית ע"י ויז'ואל סטודיו, וכל התוכן של הבלוק מוזח שמאלה בהתאם.
   b. וודאו שהשורה הזו נמצאת בראש הקובץ (ראשונה בקובץ)
8. רוקנו את הקובץ מהגדרת המחלקה שנוצרה. בהמשך, כאשר נגדיר את הישויות עצמן, נמלא את הקובץ בהגדרות של enums על פי הצורך.

כעת ניגש ליצירת טיפוס חדש עבור כל ישות נתונים (ישות DO) נדרשת.

הנחיות כלליות להגדרת ישות ותכונותיה:

נרצה לחסוך בהעתקות במעבר של ישות משכבה לשכבה וכן להגן על הישויות כך שישמרו בבסיס הנתונים בצורה אותנטית. כלומר, לוודא שכאשר ישות עוברת כהפניה משכבה לשכבה, היא תעבור בצורה שלא ניתנת לשינוי (**immutable**). לכן:

1. נגדיר כל ישות כטיפוס מסוג **record** (לא class ולא struct)
   במידה ובכל זאת נצטרך להעתיק ישות משכבה לשכבה, נרצה שהעותק יהיה זהה בערכיו וללא הפניות.
   לכן נגדיר כל ישות כ **PDS**, ראשי תיבות של **Passive Data Structure**, כלומר ישות בעלת מבנה שטוח ופסיבי. כדי שישות תהיה **PDS** אזי כל התכונות שלה צריכות להיות מסוג **value type**. (הרחבה והסבר נוסף בסוף ההנחיה)
2. לגבי כל אחת מהתכונות עליכם להחליט אם היא צריכה להיות **value type** רגיל או **nullable value type**
3. כאמור, לישות מסוג **record**, מוגדרת באופן אוטומטי העמסה למתודה **ToString**. לכן אין צורך להעמיס אותה מחדש. אלא אם כן, אתם רוצים לשנות אותה (לקצר אותה וכדומה).

4. אין להוסיף מתודות נוספות להגדרת הישות - כי זו ישות PDS!
5. המתודות היחידות שעליכם להוסיף הן 2 בנאים:

א. בנאי עם פרמטרים - שמקבל כפרמטרים, את כל ערכי התכונות (בהתחשב גם בערכי ברירת מחדל) ← אם הגדרתם את ה record בצורה שלמדנו, כבר קיים בנאי כזה אוטומטי ואין צורך להוסיף שוב!
ב. בנאי ריק (שלא מקבל פרמטרים) - נועד לצורך עתידי שנגיע אליו בהמשך.

6. חשוב מאוד! <u>הוספת תיעוד</u>**:** הוסיפו הערות לפני הגדרת כל ישות, בהערות יש לתאר בקצרה את משמעות הישות וכן הסבר על כל אחד ממאפייניה.

נגדיר שלוש ישויות נתונים : מוצר, מבצע ולקוח.
רשימת הישויות והנתונים שבהם:

- מוצר
  - מספר מזהה ייחודי prodId
    (כמו מספר הברקוד של המוצר)
  - שם המוצר prodName
  - קטגוריה prodCategory
  - מחיר price
  - כמות במלאי quantity

- לקוח
  - תעודת זהות customerId
  - שם הלקוח customerName
  - כתובת customerAddress
  - טלפון customerPhone

- מבצע
  - מספר מזהה ייחודי (מספר רץ אוטומטי) saleId
  - מספר מזהה של המוצר prodId
  - כמות נדרשת לקבלת המבצע quentityForSale
  - מחיר כולל במבצע totalPriceSale
  - האם המבצע מיועד לכלל הלקוחות או רק ללקוחות מועדון isAlCustomer
  - תאריך תחילת המבצע startDate
  - תאריך סיום המבצע endDate

טיפוסי הנתונים בישויות יהיו:

- מספרים מזהים (ID) - מסוג int
- שמות, כתובות - מסוג string
- עבור קטגוריית מוצר תוגדר אנומרציה (enum)
- כמויות - מסוג int
- מחירים - מסוג double
- תאריכים - מסוג DateTime

בס"ד

<u>ישות PDS – ישות שטוחה ופסיבית</u>

כל אחת מהישויות שנגדיר צריכה להיות **PDS**, ראשי תיבות של **Passive Data Structure**, כלומר ישות בעלת מבנה שטוח ופסיבי. על מנת שההעתקה של ישות PDS תיצור עותק זהה בערכיו וללא הפניות.

<u>מהי ישות **PDS**?</u>

- תכיל רק תכונות ולא תכיל מתודות (למעט העמסת של מתודה ToString לצורכי debug)
- תכונות אלו יהיו מטיפוסים שהם ValueType בלבד ולא ReferenceType (למעט מחרוזות - string), כלומר:
  - טיפוסים פשוטים כמו int, char, bool וכדומה
  - DateTime, TimeSpan
  - טיפוסים מסוג של אנומרציות מוגדרות - סוגי enum שיוגדרו ע"י המתכנת
  - מחרוזות (string) - היוצא מן הכלל שהינו Reference Type
    - מחרוזת מסוג string הינו סוג הפניה ש<u>לא ניתן לשינוי</u>(immutable)
- לא תכיל תכונות מטיפוס רשימה או אוסף מכל סוג שהוא
- לא תירש מאף מחלקה אחרת

אובייקטים של הישויות (הנתונים) יעברו בין השכבות של הפרויקט:

- בפרמטרים של מתודות הממשק
- בערך חוזר בודד של מתודת ממשק
- באוסף אובייקטים של ישות שמוחזר ממתודת ממשק

למשל, שכבת DAL מחזירה לשכבת שמעליה (BL - שכבה לוגית) את אוסף האובייקטים של ישות מסוימת, או ההיפך.

חשוב - מכיוון שהעתקות של מבנים (**struct**) ושל רשומות (**record**) הינן העתקות רדודות (**shallow copy**) - בזכות העובדה שהישות היא **PDS**, אזי כאשר יוצרים ממנה העתק, הוא זהה בערכיו ממש ולא מכיל כתובות/הפניות (למעט הפניות למחרוזות, שלא ניתנות לשינוי).

השילוב של ישות שמוגדרת כ-**record** ושהיא גם **PDS** יוצר את <u>היתרונות</u> הבאים לצורכי הפרויקט שלנו:

- ה-record הוא טיפוס הפניה, לכן כשמעבירים אובייקט בין שכבות, בעצם מעבירים הפניה ולא אובייקט שלם וזה יותר יעיל מבחינת ביצועי מעבד.
- מצד שני, בזכות העובדה שניצור record עם תכונות שאינן ניתנות לשינוי (immutable), אזי האובייקט שעובר בין השכבות יהיה מוגן וישמור על ערכיו
- במידה ובכל זאת רוצים לשנות שדות של אובייקט (למשל במתודת עדכון), אז בשכבה הרלוונטית ניצור העתק שלו. ומכיוון שהישות היא PDS ולא מכילה הפניות או רשימות אזי ההעתק שטוח וזהה בערכיו לאובייקט המקורי.