בס"ד תשפ"ג סמסטר א'

מיני פרויקט במערכות חלונות

שלב 2

מטרות השלב

עבודה עם רשימות ואוספים כלליים

עבודה עם ממשקים

עבודה עם מחלקות\ממשקים גנריים

חריגות

יצירת שכבה לוגית בסיסית

הנחיות לביצוע התרגיל והגשתו

* חובה להשתמש בכלי לניהול גרסאות git ובאתר github.org
* חובה לבצע את השלב באותו המאגר ובאותו ה-Solution כמו שלב 0 (מקדים)
* חובה להגיש במודל קישור על פי הנחיות ההגשה
* חובה להקפיד על פורמט זה על מנת למנוע מצב של אי קבלת ציון על תרגיל מסוים
* חובה לקרוא את התיאור הכללי של הפרויקט לפני תחילת העבודה על כל שלב.

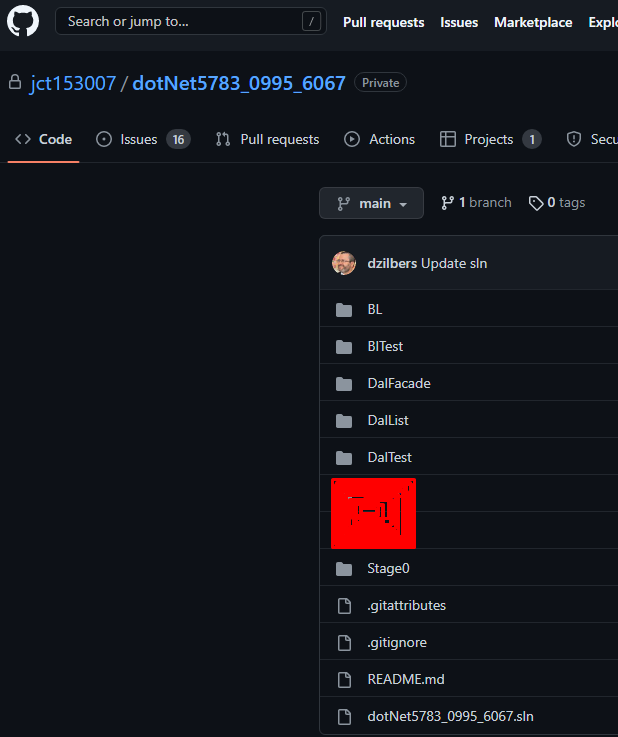
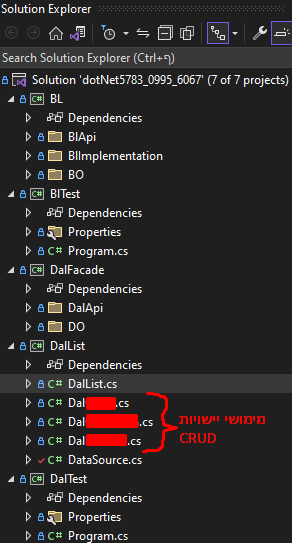
הקדמה

בשלב 1 התחלנו לבנות את שכבת הנתונים של הפרויקט וכתבנו תכנית ראשית לבדיקת השכבה.

בשלב 2:

1. **נעדכן את שכבת הנתונים** עם הכלים הנוספים שנלמדו בינתיים
2. נוסיף שכבה לוגית בסיסית
   1. נגדיר את ישויות שכבת הנתונים.
   2. נממש פעולות לעבודה מול שכבת הנתונים.
3. נבנה ממשק משתמש כקונסול טקסטואלי (CLI) לבדיקות

מבנה המאגר ב-github.com וה-Solution אחרי שלב 2:

המטלה

## עדכון שכבת הנתונים

* 1. ב"חזית" שכבת הנתונים (פרויקט DalFacade) נוסיף תת-תיקיה בשם DalApi ובתוכה:
     1. נוסיף ממשק **גנרי** בשם ICrud עם פרמטר גנרי אחד (נניח שנקרא לו T)
     2. בממשק הגנרי החדש הנ"ל, נוסיף הצהרות למתודות CRUD הבאות:

Add,Delete,Update,Get

כאשר טיפוס הנתונים המתקבל או המוחזר הינו T (הפרמטר הגנרי).

**הערה**: המתודה המחזירה רשימת אובייקטים של ישות תחזיר סוג <IEnumerable<T

* + 1. נוסיף ממשקים עבור כל אחת מישויות הנתונים:
       1. הרשאת הממשקים הינה public
       2. הממשקים מרחיבים את הממשק ICrud, כאשר בארגומנט הגנרי נשתמש בטיפוס ישות ראשית של DO המתאימה:
       3. שם כל ממשק הינו **I***Entity* (עבור כל ישות יש להחליף את המילה *Entity* בשם הישות) - נשתמש בסוג ישות הנתונים הראשית של BO המתאימה

**לדוגמה**:

**using DO;**

**namespace DalApi;**

**public interface I*Entity* : ICrud<*Entity*> { ... }**

* + 1. אם בממשק של ישות מסוימת יש צורך במתודות נוספות, נוסיף אותן לממשק,אך בשום פנים אין להוסיף הצהרות של מתודות CRUD בסיסיות (הן כבר מוגדרות ב-ICrud),   
       **כלומר** - יתכן שרוב או כל הממשקים יהיו "ריקים" (**המשמעות**: הם מגדירים ממשק CRUD עבור ישות **מסוימת**)
    2. נוסיף ממשק בשם IDal ובו נגדיר תכונות (properties) עבור כל ישות:
       1. שם התכונה יהיה כשם הישות
       2. טיפוס התכונה יהיה כשם הממשק המתאים כנ"ל
       3. בתכונה תהיה רק מתודה get (זאת אומרת - כל התכונות הן לקריאה בלבד)
    3. נוסיף בתת-תיקיה DalApi מחלקה חדשה בשם Exceptions על מנת להגדיר חריגות מתאימות לפי הכללים שנלמדו בקורס
       1. בתוך הקובץ נמחק את המחלקה Exceptions כליל
       2. נוסיף מחלקות עבור חריגות מתאימות שנרצה לזרוק משכבת הנתונים
       3. החריגות אמורות להיות כלליות לפי סוג בעיה - אין לעשות חריגות מיוחדות לכל ישות בנפרד.  
          כלומר תהיינה חריגות כלליות לכל הישויות. מומלצות בשכבת הנתונים שתי החריגות הבאות (כי בעצם אלו הדברים היחידים שנבדקים בלוגיקת השכבה):
          1. חריגה עבור ישות שלא נמצאה או מזהה חסר (עבור עדכון, מחיקה או בקשה)
          2. חריגה של כפילות מזהה (עבור הוספה של אובייקט עם מזהה שכבר קיים)
       4. ניתן להוסיף עוד חריגות לפי הצורך בתנאי שהצורך ינומק.
  1. במודול **DataSource**:
     1. **נהפוך** את כל המערכים לאוספים של מערך דינמי - <>List
     2. נבטל את כל איברי (שדות\תכונות) מעקב האינדקסים של המערכים ("מצייני המקום הפנוי הראשון") שהשתמשתם בהם בשלב 1
     3. נעדכן את קוד האתחול בהתאם
  2. מימוש שכבת הנתונים **DalList**:
     1. עבור כל המחלקות למימוש CRUD שכתבתם בשלב 1:
        1. נוסיף יבוא של מרחב השמות DalApi :

**using DalApi;**

* + - 1. נשנה את הרשאת המחלקה ל-internal ונוסיף הצהרה על מימוש ממשק הישות המתאימה כפי שהוגדר ב-DalApi, למשל:

using DalApi;

using DO;

namespace Dal;

**internal** class DalMyDOClass **: IMyDOClass**

{

…

כאשר MyDOClass היא ישות נתונים שהוגדרה ב-DO

* + - 1. נעדכן את חתימות המתודות הקיימות בהתאם לממשקים הנ"ל
      2. נעדכן את מימושי המתודות בהתאם לשינוים שביצענו בין מערכים לרשימות כנ"ל
      3. נחליף את זריקות החריגה הכללית שהוספתם בשלב 1 בחריגות מתוך המודול Exceptions שיצרתם בסעיף הקודם.
    1. נוסיף מחלקה עם הרשאה public בשם DalList שתממש את הממשק IDal.

sealed public class DalList : IDal

נממש בה את התכונות (properties) שהוגדרו בממשק IDal כך שיחזירו אובייקטים מטיפוס שמממש את הממשק המתאים לכל ישות נתונים.  
נ.ב. יש להשתמש באתחול מקוצר של התכונה בעזרת האופרטור "<=, למשל:

public IMyDOClass MyDOClass => new DalMyDOClass();

* 1. **בתוכנית ראשית של בדיקות שכבת הנתונים** שכתבתם בשלב 1:
     1. נשנה את סוגי כל המשתנים המקבלים רשימות אובייקטים ממערך ל-<>IEnumerable המתאים לפי כל ישות וישות.
     2. כדי לבדוק את השינויים ( ICrud) שנעשו על שכבת הנתונים, יש לשנות את הגישה אל המתודות בצורה הבאה:
  + יש ליצור ולשמור אובייקט של DalList במשתנה מקומי או בשדה סטטי של מחלקת Program (הטיפוס של משתנה\שדה יהיה IDal)
  + הפניה לישויות המתאימות תהיה דרך התכונות של אובייקט שהוגדר לעיל

## השכבה הלוגית - תיאור הצעדים הבאים

* + נוסיף שני פרויקטים - אחד עבור שכבה לוגית ושני עבור תוכנית בדיקות של שכבה לוגית
  + נוסיף בפרויקט השכבה הלוגית תת-תיקיות עבור:
    1. ישויות לוגיות
    2. ממשק שכבה לוגית
    3. מימוש שכבה לוגית
  + נגדיר ישויות לוגיות (כולל חריגות מותאמות של שכבה לוגית)
  + נוסיף ממשקים של שכבה לוגית
  + נממש לוגיקה בסיסית של הפרויקט
  + נטפל בחריגות המתקבלות משכבת הנתונים (נתפוס אותן) ונזרוק חריגות חלופיות (של שכבה לוגית) איפה שנדרש

## 

## הכנת התיקיות והפרויקטים לשלב הנוכחי - הנחיות מפורטות:

* 1. תוסיפו פרויקט חדש BL מסוג Class Library:
     1. עשו לחיצה כפולה על שם הפרויקט ב-Solution Explorer (ייפתח לעריכה קובץ ניהול הפרויקט עם סיומת csproj) - יש להוסיף לו את השורות הבאות (המודגשות) בסוף האלמנט PropertyGroup:

*<Project Sdk="Microsoft.NET.Sdk">*

*<PropertyGroup>*

*<TargetFramework>net6.0</TargetFramework>*

*<ImplicitUsings>enable</ImplicitUsings>*

*<Nullable>enable</Nullable>*

**<BaseOutputPath>$(SolutionDir)\bin\</BaseOutputPath>**

**<OutputPath>$(SolutionDir)\bin\</OutputPath>**

**<AppendTargetFrameworkToOutputPath>false</AppendTargetFrameworkToOutputPath>**

**<AppendRuntimeIdentifierToOutputPath>false</AppendRuntimeIdentifierToOutputPath>**

**<UseCommonOutputDirectory>true</UseCommonOutputDirectory>**

**<RootNamespace />**

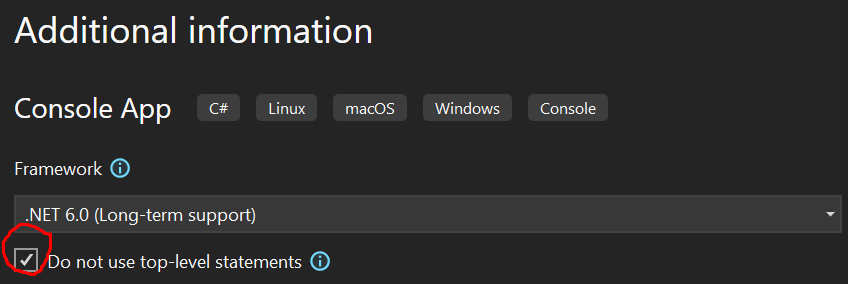
*</PropertyGroup>*

*</Project>*

* + 1. בתפריט ההקשר של הפרויקט (לחיצה על כפתור ימני כאשר הסמן מעל שם הפרויקט) בחרו מאפיינים (Properties) וודאו שהשדה Default namespace ריק (תחת כותרת Application>General, כתוצאה מהשורה המודגשת האחרונה לעיל), ובשדה Base output path (תחת כותרת Build>Output, כתוצאה מהשורה המודגשת הראשונה לעיל) מופיע:

**$(SolutionDir)\bin\**

* + 1. **נ.ב.** יתר ההוספות שעשינו לעיל לא משתקפות בממשק המאפיינים של פרויקט ולכן עשינו אותם ע"י עריכה ישירה בתוך הקובץ csproj
    2. בתפריט ההקשר של הפרויקט בחרו הוספת הפניות פרויקט (...Project Reference), ובו סמנו את הפרויקטים DalFacade ו-DalList (לחצו OK לאחר מכן)
    3. בתפריט ההקשר של הפרויקט בחרו …Build Dependencies>Project Dependencies וודאו שהסימון דומה לסימון ההפניות כנ"ל.

* 1. תוסיפו פרויקט BlTest
     1. תוסיפו ב-Solution פרויקט חדש מסוג Console Application
     2. בחלון הבא יש להקליד את שם הפרויקט BlTest
     3. בחלון הבא יש לסמן "Do not use top-level statements":
     4. עשו לחיצה כפולה על שם הפרויקט ב-Solution Explorer (ייפתח לעריכה קובץ ניהול הפרויקט עם סיומת csproj) - יש להוסיף לו את השורות הבאות (המודגשות) בסוף האלמנט PropertyGroup:

*<Project Sdk="Microsoft.NET.Sdk">*

*<PropertyGroup>*

*<OutputType>Exe</OutputType>*

*<TargetFramework>net6.0</TargetFramework>*

*<ImplicitUsings>enable</ImplicitUsings>*

*<Nullable>enable</Nullable>*

**<BaseOutputPath>$(SolutionDir)\bin\</BaseOutputPath>**

**<OutputPath>$(SolutionDir)\bin\</OutputPath>**

**<AppendTargetFrameworkToOutputPath>false</AppendTargetFrameworkToOutputPath>**

**<AppendRuntimeIdentifierToOutputPath>false</AppendRuntimeIdentifierToOutputPath>**

**<UseCommonOutputDirectory>true</UseCommonOutputDirectory>**

*</PropertyGroup>*

*</Project>*

* + 1. בתפריט ההקשר של הפרויקט בחרו מאפיינים (Properties) ותוודאו שבשדה Base output path (תחת כותרת Build>Output, כתוצאה מהשורה המודגשת הראשונה לעיל) מופיע:

**$(SolutionDir)\bin\**

* + 1. **נ.ב.** יתר ההוספות שעשינו לעיל לא משתקפות בממשק המאפיינים של פרויקט ולכן עשינו אותם ע"י עריכה ישירה בתוך הקובץ csproj
    2. בתפריט ההקשר של הפרויקט בחרו הוספת הפניות פרויקט (...Project Reference), ובו סמנו את הפרויקט BL
    3. בתפריט ההקשר של הפרויקט בחרו …Build Dependencies>Project Dependencies וודאו שהסימון דומה לסימון ההפניות כנ"ל (מסומן פרויקט BL בלבד)

## בניית ארכיטקטורה של השכבה לוגית - הנחיות מפורטות:

נוסיף שלוש תת-תיקיות בפרויקט BL:

1. תת-תיקייה עבור ישויות לוגיות BO
2. תת-תיקייה עבור ממשקי שכבה לוגית BlApi
3. תת-תיקייה עבור מימוש של שכבה לוגית BlImplementation

## בניית ישויות לוגיות - הנחיות מפורטות:

1. **כל** ההגדרות של ישויות לוגיות ומודולים נוספים המפורטים בהמשך בפרק הזה נוספים **בתת-תיקיה BO** של השכבה הלוגית (בפרוייקט BL)
2. **עבור כל ישות לוגית** המתוארת בתיאור כללי של הפרויקט, נוסיף **מחלקה** (במודול נפרד)
   1. המודול אמור להיווצר אוטומטית עם namespace BO בעת הוספת המחלקה
   2. שם המחלקה יהיה זהה לשם הישות
   3. הרשאת המחלקה public
   4. נגדיר את כל הנתונים כתכונות (properties):
      * תכונה אוטומטית
      * הרשאה public
      * ללא אתחול בהצהרה
   5. אין להוסיף מתודות למחלקה מלבד דריסת ToString
   6. אסור להוסיף כל לוגיקה\קוד ביצוע למעט דריסה של ToString, זאת אומרת שאסור להוסיף: בנאים מכל סוג, הורס, מתודות כלשהן
   7. לדוגמה:

**namespace BO;**

**public class** *Entity*

**{**

**public ... { set; get; }**

**...**

**}**

1. ניצור קובץ נוסף (מודול נפרד) ששמו Enums.cs ובו:
   1. עבור האנומרציות שמוגדרות עם ישויות הנתונים, נגדיר גם האנומרציות (enum) לוגיות בצורה זהה לאנומרציות ביושיות הנתונים  
      **נ.ב.** לא נוכל (**אסור**) להשתמש בטיפוסים של ישויות הנתונים עבור ישויות לוגיות
   2. נוסיף במודול גם אנומרציות נוספות כפי שנדרש בישויות הלוגית
2. נוסיף קובץ (מודול נפרד) עבור הגדרת חריגות של השכבה הלוגית
   1. בהתאם לסוגי התקלות בשכבה לוגית כפי שיתואר בהמשך
   2. החריגות מוגדרות ע"פ הכללים המתוארים במסמך החריגות בחומרי הקורס

## 

## בניית ממשקים של שכבה לוגית

1. **כל** הממשקים נוסיף **בתת-תיקיה BlApi**
2. עבור כל ההגדרות חובה להוסיף תיעוד "חיצוני" בעזרת הערה מפורמטת "///"
   1. לפני כל ממשק
   2. לפני כל תכונה
   3. לפני כל מתודה
3. נוסיף ממשקים עבור כל **ישות לוגית ראשית** (ראו בתיאור כללי של הפרויקט)
   1. הרשאה public
   2. שם ממשק בפורמט I*Entity*, כאשר *Entity* הינו שם ישות לוגית ראשית
   3. נוסיף הצהרות של מתודות לפי הצורך כפי שמופיע בתיאור כללי של הפרויקט
   4. במתודות שאמורות להחזיר רישמה כלשהי, סוג הערך המוחזר יהיה אוסף כללי גנרי: <IEnumerable<*Entity* בהתאם
   5. לדוגמה:

**using BO;**

**namespace BlApi;**

**public interface I***Entity*

**{**

**public ... Method1(...);**

**...**

**}**

1. נוסיף ממשק ראשי בשם IBl שירכז את כל הממשקים של השכבה
   1. הרשאה public
   2. נגדיר תכונות עבור כל ישות לוגית ראשית (ראו בתיאור כללי של הפרויקט)
      1. הרשאה public
      2. טיפוס - ממשק ישות לוגית ראשית מתאימה כנ"ל
      3. שם - כשם ישות לוגית ראשית *Entity*
      4. לקריאה בלבד (רק עם get)
      5. ללא אתחול בהצהרה

**לדוגמה**:

**public interface IBl**

**{**

**public I***Entity**Entity* **{ get; }**

**...**

**}**

## TBD

המשך ההנחיות יושלם בשעות\הימים הקרובים