uבס"ד תשפ"ג סמסטר א'

מיני פרויקט במערכות חלונות

**שלב 3**

# מטרות התרגיל

עבודה עם סוגי ערך והפניה **מתאפסים** (**Nullable** Value & Reference Types)

עבודה עם דלגטים של דוט-נט ולאמבדות (NET defined delegates and lambda functions.)

יצירת שכבת תצוגה (משתמש משתמש גרפי) בסיסית עבור ישות לוגית ראשית אחת

**הנחיות לביצוע התרגיל והגשתו**

* העבודה תתבצע בזוגות בלבד
* 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 השלב

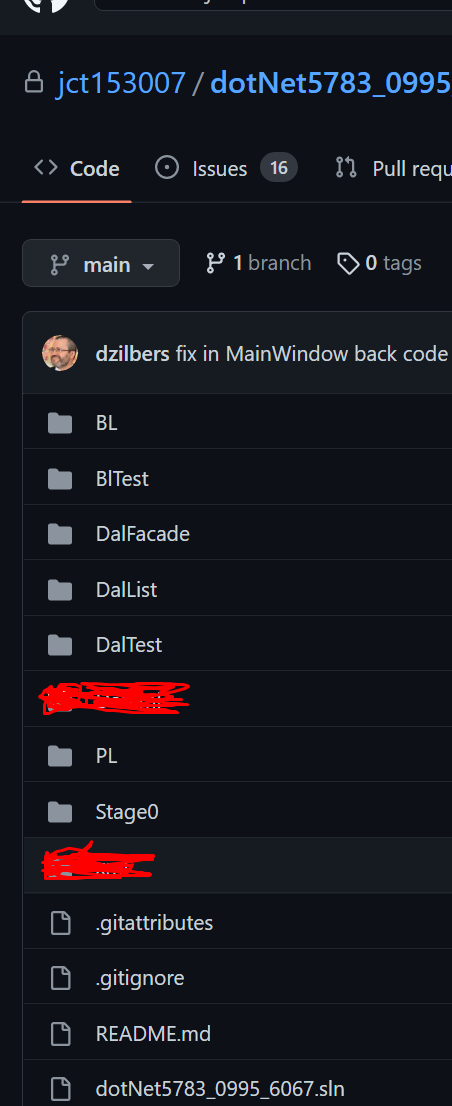
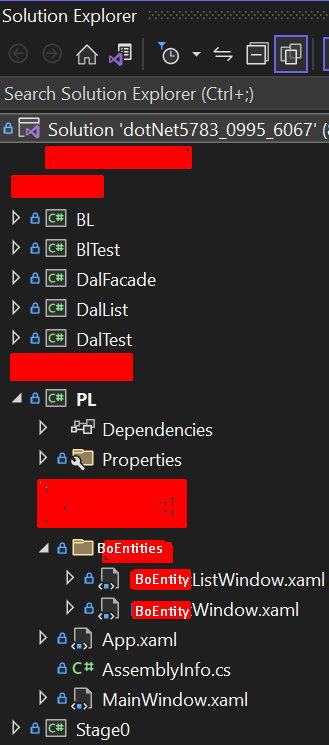
**הקדמה**

בשלבים 1 ו-2 התחלנו לבנות את שכבת הנתונים ואת השכבה הלוגית של הפרויקט וכתבנו תכניות ראשיות לבדיקת השכבות האלה.

בשלב 3:

1. **נעדכן את שתי השכבות האלה** עם הכלים הנוספים שנלמדו בינתיים
2. נוסיף שכבה תצוגה בסיסית
   1. ניצור חלון ראשי, חלון תצוגת רשימה וחלונות תצוגה\הוספה\עדכון לניהול אחת הישויות הלוגיות הראשיות
   2. נתפעל את החלונות הנ"ל ואת הנתונים המוצגים\המנוהלים בעזרת פניות לשכבה הלוגית

מבנה המאגר ב-github.com וה-Solution אחרי שלב 3:

בשמות של תת-התיקיות והחלונות בצילום המסך לעיל, BoEntitie**s** זהו שם ישות לוגית ראשית מתאימה **ברבים** (למשל Students), ו-BoEntity זהו שם אותה הישות הלוגית (למש

**המטלה**

## עדכון שכבת הנתונים ושכבה לוגית

* 1. הן בישויות הנתונים (DO) והן בישויות לוגיות (BO) נהפוך לטיפוסי מתאפסים (בעזרת הוספת "?" - Nullable Value/Reference types) את טיפוסי כל התכונות **למעט** מזהים ותכונות מהטיפוסים הבאים: int, double, bool. למשל:

**// Data Object/Entity:**

**public struct Person**

**{**

**public int ID { set; get; }**

**public string? Name { set; get; }**

**public DateTime? BirthDate { set; get; }**

**public Gender? Gender { set; get; }**

**}**

**// Business Object/Entity:**

**public class Student**

**{**

**public int ID { set; get; }**

**public string? Name { set; get; }**

**public double Year { set; get; }**

**public List<Course?>? Courses { set; get; }**

**}**

**נ.ב.** אם לדעת הסטודנטים כדאי להפוך לטיפוסים מתאפסים גם שדות מהטיפוסים המוזכרים לעיל (int, וכו') - מותר לעשות זאת, למעט תכונות של מזהים.

* 1. נהפוך את סוג האלמנט בכל האוספים ו-\או את האוספים עצמם לטיפוסים מתאפסים (ע"י הוספת "?")
     1. רשימות ב-DataSource - רק טיפוסי האלמנטים, למשל: **internal List<Student?> s\_students**
     2. סוג ערך מוחזר של מתודות ב-DalApi.ICrud שמחזירות אוספים (רק טיפוסי אלמנטים), למשל: **<?IEnumerable<T**
     3. סוג ערך מוחזר במתודות של BlApi המחזירות אוספים (רק טיפוסי אלמנטים) - כנ"ל לגבי מתודות של DalApi.ICrud
  2. בממשק DalApi.ICrud, במתודת בקשת אוסף, נוסיף פרמטר **אופציונלי** עבור פרדיקט - דלגט מטיפוס **?<Func<T?, bool**, עם ערך ברירת מחדל של null
  3. נעדכן את כל המימושים של המתודה הנ"ל (עבור כל ישויות הנתונים) כך שתביא אוסף מופעים מלא, אם התקבל null בפרמטר הנ"ל, אחרת היא תחזיר אוסף "מסונן" ע"י המתודה שבפרמטר הנ"ל
  4. נוסיף ב-DalApi.ICrud מתודת בקשה של אובייקט בודד ע"פ תנאי שתקבל פרמטר כמו בסעיף **c.** לעיל (אך הפרמטר לא אופציונלי - ללא ערך ברירת מחדל), ונממש את המתודה עבור כל ישויות הנתונים בהתאם
  5. אם בשלבים הקודמים הוספנו מתודות מיוחדות במימושי מתודות של חלק מהישויות - המחזירות אוספים "מסוננים" או מתודות שמחזירות אובייקט לפי תנאי מסוים - נמחק אותם ונחליף את כל הזימונים שלהן בזימון המתודות בסעיפים **c.** או **e.** הנ"ל

**נ.ב.** תנאי הסינון בזימוני המתודות האלה יינתנו ע"י מתודות אנונימיות בתחביר למבדה

* 1. עבור התכונות שהטיפוס שלהן השתנה לטיפוס מתאפס ואותחל קודם עם ערך ברירת מחדל או ערך מיוחד (כמו למשל DateTime.MinValue עבור תאריכים "חסרים") - נשנה את האתחול ל-null, ונתאים את כל הבדיקות של התכונה לערך "חסר" להשוואה עם null או לשימוש באופרטור null coalescing ("**??**") או לשימוש באופרטור "**?**"
  2. נבצע את העדכונים הנדרשים לתוכניות בדיקה של שתי השכבות DalTest ו-BlTest בהתאם לשינויים הנ"ל על מנת שהתוכניות האלה תעבודנה בצורה תקינה לצורך חזרה על הבדיקות של השכבות
  3. **לבונוס**: ב-BL - שימוש במתודה\-ות גנרית\-יות הרחבה עם reflection על מנת להעתיק נתונים בין אובייקטים של ישויות נתונים ושל ישויות לוגיות עבור תכונות עם אותו שם ואותו טיפוס
  4. **לבונוס**: בתוכניות בדיקה של שכבת נתונים ושכבה לוגית - שימוש במתודה\-ות גנרית\-יות הרחבה עם reflection על מנת להדפיס את האובייקטים של ישויות הנתונים והישויות הלוגיות בהתאם
  5. **לבונוס**: כל שימוש מיוחד ויצירתי בדלגטים, מתודות הרחבה, reflection, סוגים\טכנולוגיות שלא נלמדו (למשל RegExp) בכל מקום מתאים המאפשר לשמור על רמה נמוכה של מורכבות מיותרת בקוד ורמה גבוהה של בהירות הקוד

**הערה חשובה בעניין הבונוסים**: כל ניקוד הבונוסים יינתן בבדיקה הסופית של הפרויקט ולא בניקוד השלבים. על מנת לאפשר זאת, כבר בשלב הזה צריך להוסיף קובץ דו"ח פרויקט בכל פורמט מתאים (למשל קובץ Word, או ב-README.md) שיכיל רשימת הבונוסים שהסטודנטים בצעו. הקובץ חייב להיות בתיקיה של ה-Solution של הפרויקט, ובכך גם הוא יעלה אוטומטית למאגר גיט משותף (בענן - github.com) של כל זוג סטודנטים. בלי זה אי אפשר יהיה לתת תוספת ניקוד לציון הפרויקט עבור הבונוסים. ניקוד שכל בונוס ובונוס יפורט ויפורסם בהמשך. על מנת לאפשר לרכזי הקורס לאשר ולקבוע ניקוד לבונוסים ספציפיים - אנא להעביר את הרעיונות שלכם בצורה כתובה דרך המרצים שלכם.

## הכנת הפרויקטים לשלב הנוכחי

* 1. באותו Solution, נוסיף פרויקט חדש בשם **PL** מסוג "**WPF Application**":
  2. עשו לחיצה כפולה על שם הפרויקט החדש 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>WinExe</OutputType>*

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

*<Nullable>enable</Nullable>*

*<UseWPF>true</UseWPF>*

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

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

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

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

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

*</PropertyGroup>*

*</Project>*

* + 1. בתפריט ההקשר(context menu) של הפרויקט (לחיצה על כפתור ימני כאשר הסמן מעל שם הפרויקט) בחרו מאפיינים (Properties) וודאו 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 (...Add→Project Reference), ובו סמנו את הפרויקטים BL (לחצו OK לאחר מכן)
    3. בתפריט ההקשר של הפרויקט בחרו …Build Dependencies>Project Dependencies וודאו שהסימון דומה לסימון ההפניות כנ"ל.

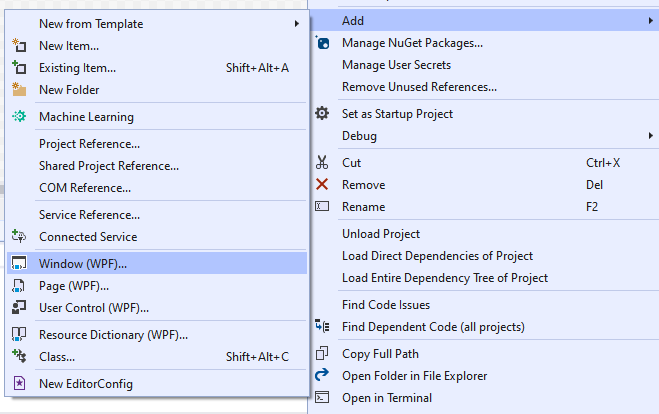
## 

## הנחיות מפורטות ליצירת ממשק גרפי ראשוני

**הערה חשובה**: סטודנטים המעדיפים להשתמש בעיצוב ובפקדים שונים ממה שמופיע להלן - רשאים לעשות זאת, בתנאי שאיכות ויכולות התצוגה הנוצרים משתפרים בהשוואה למה שנוצר ע"פ ההנחיות בהמשך.

## צעד 1: יצירת פרויקט חדש מסוג WPF.

1. את החלון הראשי (MainWindow) נרצה להתאים בהמשך להיות מסך כניסה ראשי למערכת
2. לכן נוסיף עוד שני חלונות (ראה צילום מסך בתיאור הכללי):
   1. **כדאי** להוסיף תת-תיקייה עבור חלונות הקשורים לכל **ישות ראשית** - בשלב הזה תת-תיקיה עבור הישות שעבורה אנו מוסיפים חלונות
   2. **חלון לתצוגת רשימה של ישות העזר המתאימה ליישות הראשית**, כפי שמופיע בתיאור הכללי של המערכת - בתפריט ההקשר (context menu) של פרויקט WPF או תת-התיקייה (ע"י לחיצה על כפתור ימין כאשר הסמן מעל שם הפרויקט או מעל שם תת-התיקייה ב-Solution Explorer), בחרו Add ולאחר מכן בחרו (Window (WPF, לאחר מכן תנו שם לחלון לפני אישור ההוספה).



נתבונן למשל בקוד ה-xaml שנוצר עבור חלון הצגת רשימה:

**<Window x:Class="PL.Students.StudentListWindow"**

**xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"**

**xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"**

**xmlns:d="http://schemas.microsoft.com/expression/blend/2008"**

**xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"**

**xmlns:local="clr-namespace:PL.Products"**

**mc:Ignorable="d"**

**Title="StudentListWindow" Height="450" Width="800">**

**<Grid>**

**</Grid>**

**</Window>**

אפשר לראות שהחלון נוצר עם רכיב של תסדיר Grid בתוכו.

* 1. **חלון לתצוגת** **הישות הלוגית הראשית** (בדומה לסעיף הקודם), חלון זה ישמש עבור:
     1. הוספת מופע חדש (Add)
     2. עדכון פרטי הישות (Update) וביצוע פעולות נוספות בישות.

## 

## צעד 2: עיצוב חלון תצוגת הרשימה - חלוקת ה Grid ל-3 שורות

נתחיל בעיצוב חלון תצוגת הרשימה.

1. דרך ה-XAML **שנו** את גובהו של החלון בתוך התג <Window> ל-**"Height=640"**.
2. **שנו** את שם ה Grid ל-**<"Grid Name="MainGrid>**
3. חלקו את הGrid ל 3 שורות ביחס של 40\*:500\*. תוכלו לעשות זאת ב 2 דרכים (בחרו את הדרך המועדפת עליכם):
   * דרך 1: כתבו את השינויים ידנית ל-XAML בתוך התג <Grid>:

**<Grid Name="MainGrid">**

**<Grid.RowDefinitions>**

**<RowDefinition Height="40\*"/>**

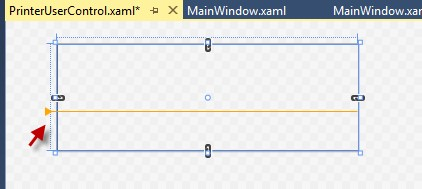
**<RowDefinition Height="500\*"/>**

**<RowDefinition Height="Auto"/>**

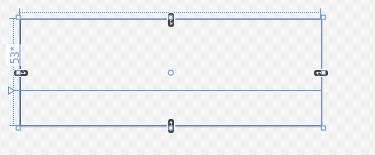
**</Grid.RowDefinitions>**

**</Grid>**

* + דרך 2: בחלון הגרפי (חלון 2) עברו עם העכבר בשוליו של ה grid, ראו קו כתום שמופיע באופן זמני וזז בהתאם עם העכבר. כל עוד העכבר נמצא במסגרת של ה grid



במידה ונלחץ מקש שמאלי על העכבר, קו זה יישאר ויהווה חלוקה של ה-grid:



**הערה**: בגרסאות החדשות של ויזואל סטודיו במקום הצגת הקו הכתום, סמן העכבר ישתנה לחץ עם פלוס קטן.

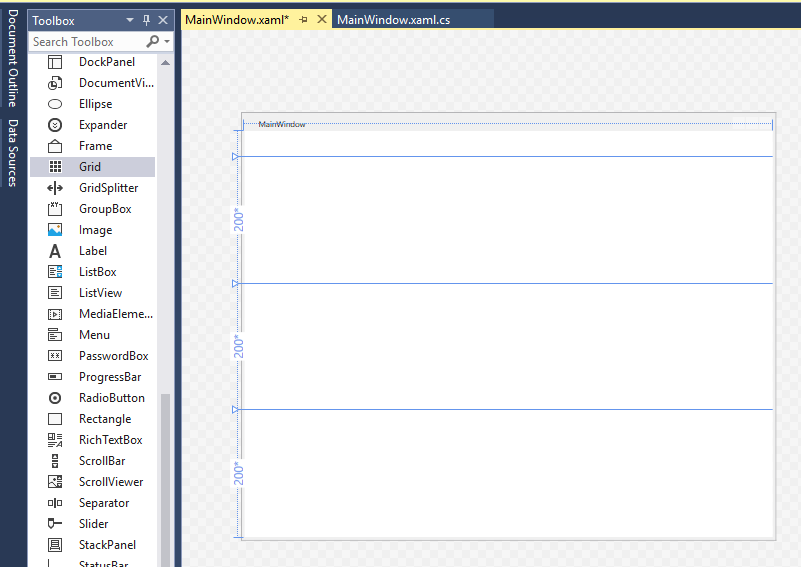
עיינו בקוד שהתקבל בחלונית ה XAML: קיבלתם שלוש שורות של RowDefinition. מה מסמל הערך שנתנו לאטריביוט Height?

* מספרים בלבד - הערכים ישארו קבועים בשינוי גודל החלון
* מספרים עם כוכבית - ישמרו היחסים בשינוי גודל החלון
* כוכבית בלבד - ישמרו יחסים זהים בשינוי גודל החלון
* פירוש ערך Auto באטריבוטים של גובה השורה או רוחב העמודה של גריד - הערך ייקבע באופן אוטומטי בהתאם לגדלים של הפקדים שהשורה או העמודה יכילו בתוך תאים שלה (אם אין עדיין פקדים, השורה או העמודה לא תוצג, אך היא עדיין קיימת)

## צעד 3: חלון תצוגת הרשימה - יצירת Grid פנימי.

1. **צרו** Grid פנימי שיופיע בתוך השורה העליונה של ה Grid הראשי, באופן הבא:

גררו עם העכבר מחלונית (Toolbox) את הרכיב הגרפי Grid. גררו אותו לשורה העליונה של ה Grid הראשי



Grid זה משמש להכיל את החלק העליון של חלון תצוגת הרשימה.

1. כתוצאה מהגרירה, נוצר תג xml חדש בשם <Grid> בתוך תג ה-<Grid> החיצוני. **החליפו** אותו בהגדרה הבאה:

**<Grid Name="UpGrid" HorizontalAlignment="Stretch" Height="auto" Grid.Row="0"**

**VerticalAlignment="Stretch" Width="auto">**

**<Grid.ColumnDefinitions>**

**<ColumnDefinition Width="\*"/>**

**<ColumnDefinition Width="\*"/>**

**</Grid.ColumnDefinitions >**

**</Grid>**

**הסבר**: ה-grid הפנימי החדש ממוקם בשורה 0 של ה-grid שבו הוא מוכל (הספירה מתחילה ב-0), הוא מתוח לרוחב ולגובה ומקבל את הגובה והרוחב שלו באופן אוטומטי. הוא מחולק ל 2 עמודות שוות רוחב (\*).

## צעד 4: יצירת פקדים בחלון תצוגת הרשימה

נרצה לאפשר סינון בתצוגת הרשימה:

הסינון מבוצע ע"פ המוגדר במסמך תיאור הכללי. לשם כך נשתמש בפקד ComboBox.

1. גררו עם העכבר מחלונית ה- Toolbox, לתוך העמודות המתאימות של ה Grid **הפנימי**, את הרכיבים הגרפיים הנחוצים. לחילופין, אפשרי לערוך את קוד ה-XAML.
2. כתוצאה מהגרירה, יווצרו בתוך תג ה <Grid> הפנימי, תגי xml חדשים מסוג <ComboBox>
3. מכיוון שבהמשך נרצה להתייחס לרכיבים האלו בקוד cs אזי כעת **שנו** לכל אחד מהרכיבים את שמו ועוד כמה מאפיינים נצרכים, באופן הבא:

| סוג הפקד | מאפיינים לדוגמא | הסבר |
| --- | --- | --- |
|  | Name="**Attribute**Selector"  HorizontalAlignment= "Stretch"  VerticalAlignment="Stretch"  Grid.Column="1"  הסירו את שאר האטריבוטים (אם קיימים) - כגון: Margin, Width | ניצור ComboBox שיופיע בתוך העמודה השניה של ה Grid הפנימי. ComboBox הוא רכיב גרפי המאפשר בחירה מתוך רשימה נפתחת, ממוקם בעמודה השניה משמאל |
|  | Grid.Column="0" | ניצור Label שיופיע בעמודה הראשונה. תגית תתן תיאור לשדה שלידו. |

**הערה:** בחרו את שמות הפקדים בצורה משמעותית כך שיתאימו לתכונה על פיה נעשה הסינון, למשל בדוגמא **Attribute**Selector נרצה לסנן על פי מאפיין **Attribute** - לדוגמא אם הישות היא רשימת סטודנטים, אזי ייתכן ויש לו תכונה Age ונרצה לסנן סטודנטים על פי גיל אזי נקרא לפקד AgeSelector.

1. לאחר מכן, התבוננו בקוד ה XAML וראו דרכו את תוצאות השינויים שערכתם.

צעד 5: הוספת פקד ListView עבור תצוגת הרשימה

1. גררו עם העכבר מחלונית ה- Toolbox, לתוך ה-Grid החיצוני (MainGrid) רכיב תצוגה מסוג ListView, התאימו את הרכיב לחלון.

שימו לב שיש להגדיר את השורה שבה הפקד יופיע: **"Grid.Row="1**, שורה זו מתייחסת לשורה בתוך הGrid החיצוני. כלומר רכיב ה-ListView יוצב בשורה השניה של הגריד החיצוני.

1. מכיוון שבהמשך נרצה להתייחס לרכיב זה נקבע את שמו לשם משמעותי, לדוגמא: **"x:Name="StudentListview**

## צעד 6: יצירת חלון ראשי זמני

כפי שאמרנו למעלה, אנו רוצים שהחלון הראשי ישמש כנקודת כניסה למערכת. בשלב זה **נעצב את החלון הראשי** כך שיוביל לחלון **תצוגת הרשימה**.

1. צרו פקד מסוג **כפתור** בקוד xaml של החלון הראשי (תנו לו שם משמעותי!) ע"י אחת הדרכים שתוארו.
2. השתמשו באירוע ברירת מחדל של לחיצה על הכפתור על מנת להציג את חלון תצוגת הרשימה
3. במחלקה של החלון הראשי (בקוד האחורי בחלק של מפתח) הוסיפו שדה פרטי בשם bl מטיפוס IBL שיאותחל באובייקט של מחלקה BL המממשת את הממשק IBL.
4. מכיוון שגם חלון תצוגת הרשימה צריך גישה ל-BL, לא נשלח את האובייקט לבנאי של חלון תצוגת הרשימה אלא גם למחלקה של חלון הרשימה יתווסף שדה bl כמו בסעיף 3).
5. אם בחרתם את השם *BoEntity***ListWindow** לחלון של תצוגת הרשימה, הפעולה שמתבצעת בלחיצה על הכפתור יכולה להראות כך (עבור רשימת סטודנטים לדוגמא):

**private void ShowStudentsButton\_Click(object sender, RoutedEventArgs e) => new StudentListWindow().Show();**

**הריצו את הפרויקט ובדקו את התוצאות המתקבלות.**

## צעד 7: הצגת הרשימה בחלון

**נחזור לחלון תצוגת הרשימה**

1. צרו במחלקה של חלון תצוגת הרשימה (בדוגמא : *BoEntity***ListWindow**) שדה פרטי מסוג IBL בשם bl, השדה יאותחל למופע של BL, כנ"ל.
2. על מנת לראות את הרשימה בפקד ListView שיצרנו בצעד 5 נוסיף בגוף הבנאי הוראה שתבצע השמה של האוסף החוזר משכבת הBL לתכונה ListView.ItemsSource. לדוגמא אם אנו רוצים לראות את רשימת הסטודנטים נכתוב כך:

**StudentsListView.ItmesSource = bl.Student.GetStudents();**

המתודה המזומנת מחזירה את אוסף ישויות העזר לתצוגה (למשל StudentsForList)

1. הריצו את התוכנית וקבלו את חלון תצוגת הרשימה. מה התצוגה המתקבלת בתוך ה-ListView? מדוע מתקבלת תצוגה זו?

## צעד 8 : סינון תצוגת הרשימה ע"י ComboBox

בצעד 4 יצרתם פקד ComboBox, נשתמש בו על מנת לסנן את הרשימה המוצגת בחלון ע"פ הקריטריון כפי שמוגדר במסמך התיאור הכללי.(לדוגמא כפי שהצענו לסנן סטודנטים על פי גיל).

1. נקבל את מקור הנתונים של הפקד ComboBox ע"י Enums שהגדרנו בשכבת ה-BL - נוסיף השמה כדלהלן בגוף הבנאי של חלון הרשימה:

**AgeSelector.ItemsSource = Enum.GetValues(typeof(BO.StudentAges));**

1. נעשה לחיצה כפולה על הפקד ComboBox על מנת להוסיף טיפול באירוע ברירת המחדל של הפקד (*אירוע ברירת המחדל של פקד ComboBox הוא SelectionChanged*)
2. נממש את הפונקציה שנרשמה לאירוע זה:
   1. כדי למצוא את התנאי לסינון, נשתמש בתכונה SelectedItem של הפקד ComboBox על מנת לקבל את הערך שנבחר מהרשימה (שימו לב שיש להשתמש בהמרות (casting) מתאימות)
   2. **יש להשתמש בקריאה למתודה של שכבת ה-BL** המחזירה רשימה מסוננת **ולעדכן את מקור הנתונים של התצוגה (בתכונה ItemsSource של ה-ListView, כנ"ל)**
      1. אם אין ב-BL מתודה כזו - יש להוסיפה לממשק המתאים ולממש במקום המתאים
      2. ניתן גם להוסיף למתודה בשכבת BL שמחזיר רשימה רצויה, פרמטר דלגט מטיפוס Func המקבל פרמטר של ישות העזר המתאימה ומחזיר ערך בולאני - וביזמון המתודה מקוד חלון הרשימה להעביר בארגומנט מתודה אנונימת (בתחביר למבדה) שתבדוק את תנאי הסינון; הסינון יתבצע בפועל בתוך מימוש המתודה בשכבת BL

**שימו לב**: לאחר הבחירה ב-ComboBox אפשרי לבחור רק את אחד הערכים האחרים.

**אתגר**: כיצד לנקות את הבחירה ולחזור למצב בו רואים את כל הרשימה?

צעד 9 : עיצוב חלון הישות הלוגית הראשית : מצב הוספה

1. **בצעד 2 , בחלון תצוגת הרשימה**, חילקנו את ה-Grid ל-3 שורות. **ניצור בחלון תצוגה הרשימה כפתור נוסף.** יש ליצור אותו בשורה השלישית **"Grid.Row="2**, מכיוון שהשורה השלישית של הגריד הוגדרה עם גובה Auto, קשה לגרור כפתור לתוכה בדיזיינר, לכן הוסיפו ישירות ב-xaml של חלון תצוגת הרשימה את השורה הבאה:

**<Button Content="Add new Product" Grid.Row="2" HorizontalAlignment="Right" Margin="5" Padding="5"/>**

1. נרצה שבלחיצה על הכפתור הנ"ל יִפַּתָּח חלון חדש שמייצג את הישות הלוגית הראשית, **במצב הוספה**. כלומר החלון יהיה במצב שהוא מאפשר למשתמש להכניס בו נתונים על מנת להוסיף עוד ישות לוגית ראשית.

עליכם לחשוב מהם הפעולות הנדרשות לשם כך. חלקם נפרט להלן ואת היתר תסיקו ממה שעשינו עד כה.

1. יש להעביר בבנאי של החלון את אובייקט הBL כפי שעשינו עבור חלון תצוגת הרשימה. (לא לשכוח את כל השלבים)
2. יש ליצור פקדים מתאימים כדי למלא את פרטי הישות הלוגית הרלוונטיים (בתכונות המתאימות). (לכל תכונה, יהיה תיאור התכונה ופקד מתאים להכנסת ערך עבורה).  
   **שימו לב: בשלב זה יש לתת שמות משמעותיים לפקדים, זאת על מנת שיהיה אפשרי להתייחס אליהם ב code-behind !**

**בהמשך הקורס נלמד מנגנון נוסף כך שלא נצטרך להתייחס לפקדים בצורה מפורשת.**

1. **בחלון הישות הלוגית** נוסיף כפתור אישור הוספת מופע של הישות הלוגית הראשית - בלחיצה עליו תזומן מתודה של שכבת BL עבור הוספת אובייקט של ישות לוגית ראשית מתאימה, שכמובן תזמן בתוכה מתודות מתאימות של הוספות בשכבת הנתונים.

**שימו לב** לפקדים הנדרשים לפי הפירוט בתיאור הכללי.

## צעד 10 : חלון ישות לוגית ראשית - מצב "עדכון ופעולות"

**נחזור לחלון תצוגת הרשימה.**

נרצה בנוסף, **שבלחיצה כפולה על שורה ברשימה המוצגת** ייפתח חלון הישות הלוגית במצב **עדכון ופעולות**. לשם כך נעשה את סידרת הפעולות הבאה:

1. במסך העיצוב של **חלון תצוגת הרשימה** בחרו את ה-ListView שיצרתם בצעד 5
2. **בחלונית ה-Properties** בחרו את סמל הברק
3. התבוננו באירועים האפשריים לפקד זה, שימו לב שקיימת רשימה ארוכה של אירועים אפשריים , עליכם למצוא את האירוע המתאים ללחיצה כפולה על פריט ברשימה, להוסיף טיפול באירוע ולממש אותו
4. בקוד cs נוצרה פונקציה שנרשמה לאירוע המתאים - ממשו את תוכן הפונקציה ע"פ הדרישות הבאות:

* בלחיצה כפולה על פריט ברשימה ייפתח החלון של הישות הלוגית הראשית **במצב עדכון**, כאשר בחלון מופיעים כבר כל ערכי פרטי מופע הישות שנבחר
* הבנאי של החלון יקבל בפרמטר את מזהה של הפריט שנבחר ויבקש משכבה לוגית את האובייקט המתאים שפרטיו יוצגו בחלון

1. הוסיפו כפתור עבור אישור עדכון המופע של הישות הלוגית הראשית - בלחיצה על הכפתור תופעל מתודה מתאימה מהשכבה הלוגית
2. לפי הצורך, הוסיפו כפתורים לפעולות נוספות שניתן לבצע על הישות בחלון זה

שימו לב שאתם נדרשים להשתמש באותו מחלקה של חלון שיצרתם בצעד 9 וכן באותו חלון למעשה עם אותם פקדים.

אתם נדרשים להתאים את התצוגה של החלון לפי המצב בו נפתח החלון (הוספה / פעולות). (איך ניתן לממש את זה?)

במצב עדכון ופעולות יוצגו כפתורים לביצוע הפעולות הנדרשות לפי התיאור הכללי של הפרויקט.

## צעד 11 : התאמה אישית

כאן יש לכם יד חופשית למעשה לעצב את החלונות כיד הדמיון הטובה עליכם, ניתן לבחור צבעים, גופנים, כותרות ועוד.

**בהצלחה רבה!**