שלב 1 ג' – סיום מימוש שכבת Dal ע"י רשימות

1. הגדרת הממשקים בתוך פרויקט DalFacade

בשלב זה, נגדיר את אוסף המתודות (API) שניתן לבצע על כל ישות נתונים.

נגדיר Interface עבור כל ישות בפרויקט. נמקם את כל הממשקים תחת הספריה DalApi.

1. יצירת תת ספריה בשם DalApi תחת פרויקט DalFacade:

לחצו לחיצה ימנית על הפרויקט DalFacade, בחרו Add 🡨 New Folder וקראו לספריה בשם DalApi.

1. הגדרת ממשקים בספריה DalApi:

בלחיצה ימנית על הספריה DalApi בחרו Add 🡨 Class

מתוך הרשימה שבחלון שנפתח, בחרו באפשרות Interface

כתבו את שם המשק כשם הישות המקושרת אליו, בתוספת I בתחילתו.

הסירו מראש הקובץ את רשימת ה using,

הוסיפו ";" בסוף השורה של ה namespace

שנו את הרשאת הגישה ל public.

1. הגדרת מתודות לממשקים בספריה DalApi

עבור כל ישות נרצה לאפשר את אותן המתודות. לכן, נגדיר בתוך כל ממשק סט זהה של מתודות על ישות **CRUD** **C**reate, **R**ead, **U**pdate, **D**elete.

int Create(T item); //Creates new entity object in DAL

T? Read(int id); //Reads entity object by its ID

List<T> ReadAll(); //stage 1 only, Reads all entity objects

void Update(T item); //Updates entity object

void Delete(int id); //Deletes an object by its Id

יש להגדיר לכל ישות את המתודות הנ"ל כאשר T מוחלף בטיפוס הישות שהוגדרה תחת DO (לצורך כך תצטרכו להוסיף using DO מתחת לשורת ה namespace)

חשוב מאוד להקפיד שהכותרות של כל אחת מהמתודות יהיו מדויקות וזהות בין כל הישויות. מכיוון שבעתיד כל הממשקים ירשו מממשק גנרי בשם ICrud.

1. יצירת "בסיס הנתונים" – יצירת רשימות בפרויקט DalList
2. מחלקת DataSource תכיל את רשימות הנתונים.

הוסיפו לפרויקט DalList מחלקה חדשה בשם DataSource (מחקו את ה using והזיחו את ה namespace)

שימו לב שמחלקה נוצרה תחת namespace **Dal** וכך כל המחלקות שיווצרו בפרויקט זה.

הגדירו את המחלקה כ static ובהרשאת internal.

1. עבור כל אחת מישויות DO שהגדרתם, הוסיפו למחלקה DataSource רשימה שתכיל את כל ההפניות לישות מאותו הטיפוס.

שם הרשימה יהיה כשם הישות (record) בתוספת s.

האיברים ברשימה יהיו מטיפוס מתאפס.

הרשימות יהיו static ובהרשאת internal.

אתחלו את הרשימות ברשימות ריקות (ע"י new).

1. הגדרת מחלקת קונפיגורציה שתכיל מספרים רצים עבור הישויות:

נגדיר מחלקת קונפיגורציה שתייצר לנו מספרים אוטומטיים רצים עבור השדות שמוגדרים כ"מזהה מספר רץ". לצורך כך, נגדיר מחלקה סטטית פנימית בשם Config, בתוך מחלקת DataSource ובה נגדיר את המספרים הרצים:

בתוך המחלקה DataSource, הגדירו מחלקה סטטית פנימית בשם Config, בהרשאת internal.

עבור כל מספר מזהה רץ שנחוץ לצורך הישויות, הוסיפו למחלקת Config שלוש הגדרות:

1. שדה מספרי קבוע (const) בהרשאת internal, שיקבל ערך התחלתי למספר הרץ: המספר המזהה הקטן ביותר לפי דרישת כל ישות.
2. שדה מספרי סטטי בהרשאת private שיקבל כערך התחלתי את השדה הקבוע הקודם.
3. מאפיין עם get בלבד שיחזיר את ערך השדה הסטטי ויקדם אותו אוטומטית.
4. מימוש מחלקות הגישה לנתונים בפרויקט DalList:

עבור כל ישות, הוסיפו לפרויקט DalList מחלקה חדשה כשם הישות בתוספת המילה Implementation (מחקו את ה using והזיחו את ה namespace)

הוסיפו using עבור DalApi, DO

הוסיפו למחלקה הגדרת ירושה לממשק המתאים לה (שהגדרתם בספרית DalApi)

ממשו את המתודות המוגדרות בממשק.

**שימו לב! תפקיד מחלקות הגישה בשכבת DAL הוא לאפשר גישה לנתונים בלבד, ע"י פעולות CRUD.**

**הלוגיקה היחידה שתהיה מותרת בשכבה זו היא:**

1. **בדיקת תקינות של מזהה (וידוא שלא קיים בהוספה או וידוא שקיים בשליפה, עדכון ומחיקה)**
2. **~~סינון לפי פילטר המועבר משכבה עליונה (שייך לשלב מתקדם יותר)~~**

**אסור לבצע בשכבה זו שום לוגיקה נוספת, כגון בדיקות קלט, או פעולות קלט/פלט.**

\*\*\*

1. זריקת חריגות מתוך פרויקט DalList:

עליכם לזרוק חריגות במקומות המתאימים בקוד שבו מימשתם את הממשקים.

דוגמאות:

מתודת עדכון או מחיקה שקיבלה מזהה ישות שלא קיים ברשימה.

מתודת הוספה שקיבלה ישות עם מזהה שכבר קיים ברשימה (כשלא מדובר במזהה רץ)

החריגות יתפסו בשכבה שמעל. בשלב זה זהו פרויקט DalTest.

1. אתחול בסיס נתונים בערכים, בפרויקט DalTest:

נרצה לאתחל את בסיס הנתונים בערכים. נרצה למלא את הרשימות בנתונים הגיוניים שיש ביניהם קשר והם יוצרים מערכת מתפקדת שאפשר להתחיל לעבוד איתה ולבדוק אותה.

1. בפרויקט DalTest, נגדיר מחלקה **סטטית** בשם **Initialization**, בהרשאת **public**

(מחקו את ה using והזיחו את ה namespace), הוסיפו using עבור DalApi, DO

1. כדי לגשת לרשימות הנתונים של כל ישות, נשתמש בממשקים שהגדרנו בפרקים הקודמים. נגדיר שדה עבור כל ממשק לצורך גישה למתודות הממשק.

עבור כל אחד מהממשקים: הגדירו בתוך המחלקה Initialization, שדה פרטי, סטטי, מתאפס, מטיפוס הממשק המתאים. (למשל עבור הממשק IStudent, הגדירו שדה בשם s\_dalStudent)

1. לאחר מכן נרצה לאתחל בנתונים את כל אחת מרשימת הישויות המוגדרות במחלקה DataSource.

עבור כל אחת מרשימות הנתונים, הוסיפו למחלקה Initialization מתודה create, סטטית פרטית לאתחול הרשימה. (למשל, אם שם הישות הוא Student אז המתודה תקרא: createStudents) הפונקציה תייצר אובייקטים מטיפוס הישות ותוסיף אותם לרשימה ע"י מתודת CRUD המתאימה דרך אובייקט הגישה המתאים שהכנו.

1. נוסיף למחלקה Initialization מתודה ציבורית בשם Initialize, שתזמן את המתודות הפרטיות שהכנו ותחולל את האתחולים של הרשימות.

מתודה זו תופעל ע"י השכבה העליונה בשלבים הבאים, או ע"י התוכנית הראשית של פרויקט DalTest שנכתוב בפרק הבא של מסמך זה.

המתודה תקבל כפרמטרים את כל משתני הגישה לכל הרשימות (פרמטר עבור כל סוג ממשק שהגדרנו), שיגיעו מאותחלים כבר מהשכבה שמעל.

הפרמטרים שיתקבלו יוצבו בשדות שהכנו קודם לצורך גישה למתודות הממשק.

1. **תכנית ראשית לבדיקת שכבת הנתונים בפרויקט DalTest**

בתוך פרויקט DalTest, נרצה לבנות תוכנית ראשית לצורך בדיקה ידנית של תפקוד שכבת הנתונים שבנינו.

התוכנית תספק ממשק קונסול טקסטואלי שיציג תפריט לביצוע כל מתודות ה-CRUD של כל אחת ממחלקות הגישה (הממשקים) שהגדרנו. התוכנית תקלוט מידע מהמשתמש, תפעיל את המתודה הרצויה, ותדפיס את התוצאה למסך.

1. הגדרת שדות לצורך גישה לנתונים והפעלת האתחול של בסיס הנתונים:

במחלקת Program (שנוצרה אוטומטית בפרויקט DalTest) הוסיפו שדה סטטי פרטי עבור כל אחד מהממשקים שהגדרתם (בספריית AdlApi) ואתחלו אותם באובייקטים מסוג מחלקות הגישה המממשות המתאימות (שהגדרתם בפרויקט DalList).

לצורך אתחול בסיס הנתונים, זמנו בתחילת התוכנית הראשית את המתודה Initialize של המחלקה Initialization (שלחו לה כפרמטרים את כל אחד מהאובייקטים של מחלקות הגישה שאתחלתם כרגע).

1. התוכנית הראשית, תפעיל בסופו של דבר מתודות של הממשקים שממשנו בפרויקט שעלולות לזרוק מתוכן חריגות מסוגים שונים. לכן יש "לעטוף" את התוכנית הראשית (וגם חלק מהמתודות הפרטיות של המחלקה) בבלוק של try{}catch{} ולתפוס את כל החריגות. בתוך ה catch יש להדפיס למסך את החריגה שנתפסה.
2. הוסיפו הצגה של תפריט ראשי עם תתי תפריטים לצורך בדיקת כל אחת ממתודות ה-CRUD של כל אחד ממחלקות הגישה (הממשקים).

התפריט הראשי יאפשר בחירת ישות לבדיקה וכן אפשרות ליציאה (שמשמעותה סגירת התוכנית).

עבור כל ישות שנבחרה יודפסו אפשרויות בחירה לפעולות CRUD עבור אותה ישות,

בבחירת פעולה, תודפס בקשה להקשת הקלט הנדרש ותזומן מתודת CRUD המתאימה.

התוכנית צריכה להיות בעלת מבנה מסודר וקריא. צרו כמה שיותר מתודות פרטיות שמאגדות קוד חוזר. למשל:

* + מתודה פרטית שמציגה את התפריט הראשי
  + מתודה פרטית שמציגה את תת-תפריט של אפשריות CRUD, שמקבלת כפרמטר מחרוזת עם שם הישות
  + מתודה פרטית לקליטה ויצירת כל ישות

התוכנית הראשית, לא תכלול שום לוגיקה למעט קלט, פלט וזימון מתודות הממשק. כלומר, התוכנית לא תבצע שום בדיקת תקינות של קלט. למשל אין צורך לבדוק אם הת.ז תקינה או שהמספר חיובי וכדומה.

האחריות על שמירת שלמות הנתונים תוך הכנסת הנתונים בהרצת התוכנית הראשית - היא אך ורק של מריץ התוכנית הראשית (ה"משתמש"), הוא צריך לוודא שהנתונים שהוא מכניס הם תקינים, וגם שכבת הנתונים (DAL) לא הולכת לבדוק את זה כי אין בה לוגיקה.

כשנכתוב את שכבת ה BL שמעל ה DAL, היא זו שתבדוק תקינות. כרגע אנחנו כותבים תוכנית שבאה לבדוק את ה DAL ולא את תקינות הנתונים.

לצורך המרות של מספרים ותאריכים מהקלט - להשתמש במתודה TryParse