שלב 2 ב' – שכלול שכבת DAL ע"י linq

1. **הגדרת חריגות עבור שכבת הנתונים בפרויקט DalFacade**

חריגה היא אובייקט שנזרק מתוך קוד כאשר יש שגיאה בזמן ריצה, ואמורה להיתפס בבלוק catch מתאים.  
עד כה נתבקשתם לזרוק, במקומות הנצרכים, חריגות פשוטות מטיפוס Exception. כעת נרצה להשתכלל ולזרוק חריגות מטיפוסים שונים בהתאם לסוג החריגה הנזרקת.

**לצורך כך, עליכם להגדיר טיפוסי חריגות חדשים בפרויקט חזית הנתונים (DalFacade**).  
בשלב זה, החריגות השונות **נזרקות** מתוך המתודות שמממשות את הממשקים (בפרויקט DalList) והן **נתפסות** בפרויקט DalTest. בשלב הבא, החריגות **תתפסנה** בשכבה עליונה יותר (בפרויקט BL).

**כללים ליצירת טיפוס חדש של חריגה:**

* כל חריגה תוגדר **במחלקה נפרדת** וכל מחלקות החריגה **יופיעו באותו קובץ, אחת אחרי השניה**.
* תוגדר מחלקה נפרדת עבור **כל חריגה מסוג כללי** שנרצה לזרוק משכבת נתונים  
  החריגות אמורות להיות **כלליות לפי סוג תקלה אפשרית** - אין להגדיר חריגות מיוחדות לכל ישות בנפרד. כלומר תהיינה חריגות כלליות שמשותפות לכל הישויות.
* מכיוון שהחריגות שתגדירו כעת שייכות לשכבת ה Dal, **שם הטיפוס החדש של החריגה יתחיל ב- Dal**
* טיפוס החריגה החדש יירש ממחלקת Exception
* מעל כל חריגה נוסיף את האטריביוט [Serializable]
* לחריגה יוגדר בנאי שמקבל כפרמטר את הודעת השגיאה

**דוגמאות לטיפוסי חריגות אפשריים שתצטרכו להגדיר/לזרוק/לתפוס בשכבת הנתונים:**

* חריגה עבור ישות עם מספר מזהה שלא קיים ברשימה

תתרחש בניסיון לעדכון, מחיקה או בקשה של אובייקט עם מספר מזהה שלא קיים ברשימה

* חריגה עבור ישות עם מספר מזהה שכבר קיים ברשימה

תתרחש בניסיון להוספה של אובייקט חדש עם מספר מזהה שכבר קיים ברשימה

* ניתן להוסיף עוד חריגות כלליות לפי הצורך בתנאי שהצורך ינומק.

**לצורך הגדרת טיפוסי חריגות בשכבת הנתונים:**

1. תחת ספריית **DO** בפרויקט DalFacade, צרו קובץ בשם **Exceptions.cs**
2. הסירו מראש הקובץ את רשימת ה-using הקיימת
3. שימו לב שמרחב השמות של הקובץ נקרא באופן אוטומטי **DO** כמו שם הספריה.
4. הוסיפו סימן נקודה-פסיק "**;**" בסוף השורה: "**;**namespace DO"
5. רוקנו את הקובץ מהגדרת המחלקה שנוצרה, כיוון שאתם הולכים להגדיר בתוכו מחלקות חדשות אחרות.
6. **עבור כל טיפוס חדש של חריגה, הגדירו מחלקה נפרדת בעזרת הכללים שציינו לעיל**. וכל מחלקות החריגה **יופיעו בקובץ הזה אחת אחרי השניה**.

**לאחר שהגדרתם טיפוסים חדשים של חריגה:**

1. עליכם לעדכן את כל המקומות שבהם זרקתם חריגות פשוטות מסוג Exception שיזרקו חריגות מהטיפוסים החדשים שהגדרתם.
2. החריגה תיזרק עם טקסט שמתאר את הבעיה ולאיזו יישות היא שייכת.
3. שימו לב - מרגע זה **אסור** לזרוק יותר בקוד חריגות פשוטות מסוג Exception, אלא רק חריגות מהטיפוסים החדשים.
4. **עדכון של מימוש מתודות CRUD לעבודה עם linq**

בפרויקט DalList, נשכלל את המימוש הפנימי של כל מתודת CRUD בעזרת שימוש בשאילתות **linq** או מתודות הרחבה של המחלקה **Enumerable**.

עד כה בשלב 1, מימשתם את המתודות בעזרת עבודה עם הרשימות שלנו שהן מסוג **List**. ו**השתמשתם במתודות הקיימות של המחלקה List**, כגון Add, Find, FindAll, RemoveAll ועוד.

**כעת, בשלב 2 והלאה**, נשנה את העבודה עם הרשימות לעבוד עם שאילתות linq או עם מתודות ההרחבה של מחלקת Enumerable.

עדכנו את כל המימושים של מתודות CRUD בכל אחד מהממשקים שממשתם לעבודה כנ"ל.

לדוגמא עבדו עם מתודות ההרחבה: FirstOrDefault, Select, Where

1. **עדכון המתודה ReadAll להחזרת רשימת אובייקטים לפי פרמטר סינון**

**המטרה**: לעדכן את המתודה הקיימת ReadAll של הממשק CRUD כך שתחזיר רשימה מסוננת של אובייקטים, ע"פ פרמטר.

בפרויקט DalFacade, **בממשק הגנרי ICrud** עדכנו את הגדרת המתודה ReadAll כך: (מחקו את ההגדרה הקודמת):

List<T?> ReadAll(Func<T, bool>? filter = null); // stage 2

המתודה תקבל מצביע לפונקציה בוליאנית, delegate מסוג Func, שתפעל על אחד מאיברי הרשימה מסוג T ותחזיר את רשימת כל האובייקטים ברשימה שהפונקציה מחזירה עליהם True. במידה ולא ישלח מצביע, תוחזר כל הרשימה כמו בשלב הקודם (שלב 1).

ברגע שעדכנתם את הגדרת המתודה בממשק, אתם אמורים לראות שגיאות קומפילציה בכל אחת ממחלקות הגישה שממשות את הממשק. על מנת לתקן את השגיאה הזו, עדכנו את הכותרת והמימוש בכל אחת ממחלקות הגישה שהתחייבו לממש את הממשק, ממשו את המתודה כך שתבצע את מטרתה בעזרת שאילתת linq או אחת מתודות ההרחבה של המחלקה Enumerable. שימו לב שהמתודה אמורה להחזיר עותק של הרשימה וזה קורא אוטומטית כשעובדים עם שאילתות, לכן אין צורך לייצר את העותק.

1. **הוספת מתודה חדשה בשם Read להחזרת אובייקט לפי פרמטר סינון**

**המטרה**: להוסיף לממשק CRUD מתודה להחזרת אובייקט, לא רק על פי Id, אלא ע"פ פרמטר אחר.

בפרויקט DalFacade, **הוסיפו לממשק הגנרי ICrud** את המתודה הבאה (הגדרה בלבד ללא מימוש):

T? Read(Func<T, bool> filter); // stage 2

המתודה תקבל מצביע לפונקציה בוליאנית, delegate מסוג Func, שתפעל על אחד מאיברי הרשימה מסוג T ותחזיר את האובייקט הראשון ברשימה שהפונקציה מחזירה עליו True.

ברגע שהוספתם מתודה לממשק, אתם אמורים לראות שגיאות קומפילציה בכל אחת ממחלקות הגישה שממשות את הממשק. על מנת לתקן את השגיאה הזו, ממשו בכל אחת ממחלקות הגישה שהתחייבו לממש את הממשק, את המתודה החדשה כך שתבצע את מטרתה בעזרת שאילתת linq או אחת מתודות ההרחבה של המחלקה Enumerable.