IComparer & ICompareable

עד היום מיינו ערכים שלמים, וממשיים, תווים ומחרוזות במערכים וב List.
מיינו ערכים מהקטן לגדול (והפוך).
מה קורה כשנרצה למיין רשימה של אובייקטים מסוגים אחרים?
ניקח לדוגמה רשימת סטודנטים. לסטודנט יש שם פרטי, משפחה וציון.
מה המשמעות של למיין רשימת סטודנטים? אין לזה משמעות, בלי להגדיר איך רוצים למיין! האם נרצה למיין לפי שם משפחה, או אולי לפי ציון או שם פרטי. כלומר, יש להחליט, מי נחשב הקטן יותר בסדר, ומי גדול יותר שיש לסדר אחריו.
כשנגדיר איך אנחנו רוצים למיין, מה נחשב קטן וגדול אצל סטודנט, האם נצטרך בכל פעם לכתוב לוגיקה שלימה של מיון מערך, כאשר הלוגיקה של מיון חוזרת על עצמה? יהיה חבל.

אם למשל Array.Sort מממש מיון בועות, אז התהליך משווה שוב ושוב בין האובייקטים במערך, כאשר כל פעם האובייקט מושווה לאובייקט אחר, ואז לזה שאחריו, כך שהאיבר הגדול ביותר נדחף לסוף המערך, ואחר כך הבא אחריו עד איבר אחד לפני סוף המערך, וכן על זה הדרך, עד שהכל ממויין.

אם נוכל לשמר את הלוגיקה שעוברת על האיברים, ולהחליף רק את הלוגיקה שמשווה בין האובייקטים ומחליטה מי ביניהם נחשב ל"גדול יותר", היינו חוסכים מאמץ רב. באמצעות Interfaces נוכל לממש דבר כזה. וגם אין צורך להמציא את הגלגל לבד (אם כי זה מעולה לטובת תרגול של Interfaces ומיונים!).
אנחנו נשתמש ב Interfaces מובנים של דוטנט IComparer<T>, IComparable<T>.

נציין, ששימוש כזה ב interface, שמעביר אלי את השליטה על הלוגיקה שתתבצע בזמן אמת, הוא מימוש של תבנית העיצוב הידועה והמוכרת IoC.
כשמפתחי הדוטנט פתחו את Array או List הם לא ידעו מה המימוש שאני אזריק לטובת המיון של סטודנטים, אך העבירו לי ואפשרו לי שליטה על התהליך.

ישנה גרסה לא גנרית לממשקים הנ"ל, כמובן, שאם אין סיבה מיוחדת, נקפיד להשתמש ב בגרסה הגנרית של הממשקים.

ה IComparer תומך בלוגיקה של מיון בין שני אובייקטים שמועברים אליו, לעומת IComparable התומך במיון בין בין המופע של האובייקט שמפעיל אותו לאובייקט אחר (בגרסה הגנרית מדובר באובייקטים מאותו סוג ובזה נתמקד). זה יובן מיד כשנראה דוגמאות.

המתודה CompareTo מחזירה ערך גדול מ 0 אם המופע גדול מערך אליו משווים, 0 כאשר הם שווים, וערך נמוך מ 0 אם הערך אליו משווים גדול יותר.

```
int x = 200;
Console.WriteLine(x.CompareTo(100)); //1
Console.WriteLine(x.CompareTo(200)); //0
Console.WriteLine(x.CompareTo(300)); //-1
```

תרגילי כיתה ובית

1. צור רשימת סטודנטים. לסטודנט יש שם פרטי, משפחה וציון.
   a. מיין את הסטודנטים לפי שם משפחה.
   b. מיין את הסטודנטים לפי הציון בסדר יורד.
   c. מיין לפי שם משפחה ואחר כך שם פרטי.
   d. הוסף למיון הקודם, מיון בסדר יורד על פי הציון.

2. צור מחלקה עם פרטי מדידה של טמפרטורה (עיר, תאריך, מעלות צלזיוס, רמת לחות), הזן רשימת מדידות.
   a. מיין לפי שם עיר.
   b. מיין לפי שם עיר, תאריך.
   c. מיין לפי טמפרטורות בסדר יורד, ושם עיר בסדר עולה.
   d. מיין לפי רמת לחות בסדר יורד, טמפרטורה יורד, עיר עולה.

Extensions Methods

1. צור הרחבה ל string בשם HasLowerChars שמחזיר true אם יש אותיות קטנות באנגלית.

2. צור הרחבה ל string בשם ReverseCase, המתודה תחליף כל אות קטנה לגדולה וכל גדולה לקטנה.

3. צור הרחבה ל int בשם Print. המתודה תבדוק אם המספר גדול מ 100, נחזיר "Ad Kan!", אחרת, המתודה מחזירה במילים את המספר. למשל:

```
Int x = 101;
x.Print(); // returns Ad Kan!
X = 97;
x.Print(); // returns “Ninety Seven”
```

4. צור הרחבה ל int – בשם MultipleBy שמקבלת פרמטר מספר שלם, ומחזירה את הערך שב int * המספר שקיבלנו.
   למשל:

```
Int x=2;
x.MultipleBy(10); //returns 20
```

5. צור הרחבה לסוג object בשם IsNull שבודקת אם הערך של האובייקט הוא null.
   מי שיממש את המתודה הזו יוכל לכתוב

```
If(x.IsNull())
```

במקום לכתוב

```
If(x==null)
```

6. צור הרחבה ל string בשם IsValidIsraeliIdNumber החזר true אם הערך במחרוזת מייצג מספר זהות ישראלית תקין (זה תרגיל חיפוש באינטרנט! חפשו את הלוגיקה לבדיקת ספרת הביקורת של מספר הזהות).