

מבני נתונים דינמיים ב- C# / טיפוסים גנריים וגישה למאפיינים פרטיים

רשימה - כמו מערך דינמי List - זוהי מחלקה קיימת בשפה, שאפשר להשתמש בה לצרכים שונים. הספרייה שמכילה מבנים כאלה היא: System.Collections.Generic

נניח שיש לנו תוכנית שאמורה לקלוט רשימה של כל התלמידים של בית ספר. כל תלמיד יכול להיות עצם. ישנם בתי ספר עם 500 תלמידים וכאלה עם 5000. כיצד נגדיר את המערך, כך שיתאים לכולם?

אין לנו ברירה אלא להגדירו כמקסימום. מה יקרה בעתיד אם נוסף בית ספר עם 6000 תלמידים - התוכנית שלנו תיכשל.

לשם כך נועדו רשימות דינמיות שאותן אפשר להגדיל תוך כדי ריצת התוכנית. זוהי רשימה, שמתנהגת בדומה למערך, אבל יכולה לגדול ולקטון על פי הצורך.

בנוסף, למדנו על הגדרות גנריות בעיקר תוך שימוש במחלקות. זה אומר שניתן להגדיר מחלקה בעזרת תבנית שתשמש "שומר מקום" עבור טיפוס (או סוג) כלשהו כאשר רוצים להגדיר עצמים דומים, בעלי סוגי נתונים שונים.

כדי לשלב בין שני הנושאים דיברנו למשל על מחלקה קיימת שמוגדרת בסיפריית האוספים (Collections) של C#, מחלקה בשם List, (רשימה) שבעצם מוגדרת כ: List<T> כאשר האות T מסמלת סוג, ויכולה להכיל כל סוג של נתונים, גם סוגים מובנים בשפה (כמו למשל: string, int, double וכו') וגם סוגים שהם מחלקות שאנו הגדרנו.

עשינו גם 2 שימושים במחלקה זו. שימוש ראשון היה להגדיר רשימה של שלמים ולבצע בה מספר פעולות על פי המתודות/ פעולות שקיימות במחלקה זו: (למשל Add מוסיפה תא לרשימה)

```
List<int> a = new List<int>();
```

```
List<Students> school = new List<Student>();
```

```
//Add students to the list
```

```
Student st1= new Student("Salina Gomez", "1995-12-31");
school.Add(st1);
```

הרשימה school נוצרה ללא כל תאים. בעזרת הפעולה (מתודה Add) ניתן להגדיל את מספר התאים ברשימה. ישנן עוד מספר מתודות מועילות שמוגדרות במחלקה List.

להתייחס לתא ברשימה נישתמש בסוגריים המרובעות, כמו ובמערך עם אינדקס שרוצים.

המתודה: Contains מאפשרת בדיקה אם אובייקט מסוים נימצא ברשימה, למשל:

```
if (school.Contains(st1))
    Console.WriteLine("Student:" + st1.print() + " is in the school");
```

כלומר הפעולה Contains מחזירה ערך בוליאני (אמת או שקר).

הפעולה RemoveAt לוקחת פרמטר שהוא אינדקס ברשימה ומוחקת את התא הפעולה: IndexOf לוקחת פרמטר שהוא עצם מהסוג שמוגדר ברשימה ומחזירה את האינדקס שלו אם הוא נימצא, או מינוס 1 אחרת. העולה: Clear() מוחקת את כל התאים ברשימה

בנוסף למדנו איך לגשת למאפיין שמוגדר פרטי בתוך מחלקה, מחוץ למחלקה, על ידי הגדרת getter ו-setter.

בדף הבא ישנה דוגמה מלאה שכדאי להריץ ולהיווכח בתוצאות. זו דוגמה פשוטה, אבל מראה תוצאות כמצופה.

```

using System.Collections.Generic;
namespace listactions
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            List<Student> list1 = new List<Student>();
            Student a= new Student("Jerry","2000-10-26");
            Student b= new Student("Johny","2001-05-23");
            Student c= new Student("Sunny","1999-08-10");
            list1.Add(a);
            list1.Add(b);
            foreach (Student s in list1)
                s.print();
            if (list1.Contains(a))
                Console.WriteLine("a is in list1");
            else
                Console.WriteLine("a is NOT in list1");
            if (list1.Contains(c))
                Console.WriteLine("c is in list1");
            else
                Console.WriteLine("c is NOT in list1");
            int locationb = list1.IndexOf(b);
            int locationc = list1.IndexOf(c);
            Console.WriteLine("Location of b in list1 is:
                "+locationb);
            Console.WriteLine("Location of c in list1 is:
                "+locationc);
            list1.RemoveAt(0);
            Console.WriteLine("After removing the first cell of
                list1");
            if (list1.Contains(a))
                Console.WriteLine("a is in list1");
            else
                Console.WriteLine("a is NOT in list1");
        }

        public class Student{
            private string name;
            private string dob;
            public Student(string sname, string dbirth){
                name = sname;
                dob = dbirth;
            }
            public void print(){
                Console.WriteLine("Student name is: " + name+ " date of
                    birth:"+dob);
            }
        }
    }
}

```

גישה למשתנים (מאפיינים) פרטיים של מחלקה – דרך עקיפה גם לשמור על עיקרון 'החבאת' המידע, וגם אפשרות לדלות או לשנות אותו. לדוגמה מחלקה של 'זמן' עם מאפייני תאריך:

```
public class Time
{
    private int year;
    private int month;
    private int day;

    public Time(y,m,d) //constructor
    {
        year=y; month=m; day=d;
    }
    //Create property – pay attention the property's first letter is capitalized
    public int Year
    {
        get
        {
            return year;
        }
        set
        {
            year=value;
        }
    }
}
```

השימוש במאפיינים נעשה על ידי שם העצם, נקודה, שם המאפיין, למשל:

```
Time t= new Time(2014,10,27);
t.Year=2013; //setting the year private variable indirectly
```

לסיכום, הגדר מאפיין - סוג ושם (כמו: public int Year) ולאחריו צמד סוגריים מסולסלות שבתוכן יוגדרו פעולות ה-set וה-get (ניקראים גם accessors) - אין להם פרמטרים פורמליים. ה-get מחזיר ערך של משתנה פרטי כאשר מציינים את העצם עם הנקודה ושם המאפיין. פעולת ה-set ניקראת כאשר עושים השמה למאפיין ומה שאנו שמים, ניכנס לשדה value (מילת מפתח כמו משתנה פנימי של השפה). שימו לב שלמאפיינים שמות כמו למשתנים, אבל עם אות גדולה תחילה.

