

תכנות מונחה עצמים

מחלקה - class הגדרה של סוג חדש (טיפוס) של מבנים שיקראו **עצמים**

כלומר ממחלקה אפשר ליצור הרבה עצמים, כמו שם- int אפשר להגדר הרבה משתנים שיכילו מספר שלם.

המחלקה מאגדת הגדרות של משתנים (מאפיינים) ופעולות (מתודות). משתנים כבר ראינו ופעולות אלה קיטעי תוכנה שעושים 'משהו'.

בנאי (או בונה) - **Constructor** - הפעולה שיוצרת את העצם הראשוני.

צורת יצירת עצם חדש ממחלקה שניקראת: Student שלה 2 תכונות - שם תלמיד וגיל.

```
Student s1 = new Student("Binyamin", 10)
```

במקרה זה במחלקה Student ישנם משתנים (או מאפיינים) עבור שם וגיל. הבנאי שלה מצפה לקבל אותם בסדר זה (קודם שם ואח"כ גיל).

דוגמה נוספת: **מחלקה עבור עיגול, בשם Circle**

המחלקה של העיגול: לעיגול תהיינה 3 תכונות - קואורדינטת x, קואורדינטת y במרחב ורדיוס r. הפעולות יהיו: שינוי רדיוס וציור העיגול.

```
public class Circle {
    private int r, x, y;
    public Circle(int radius, int xcoord, int ycoord) //Constructor
    {
        r=radius;
        x=xcoord;
        y=ycoord;
    }
    public void draw()
    {
        Console.WriteLine("Circle with radius: {0} is drawn at: {1}:{2}", r,x,y);
    }
    public void change_radius(int c)
    {
        r=c;
    }
}
```

שימו לב שתכונות העיגול הוגדרו כמשתנים מסוג private. זה כדי להגן עליהם מפני שינוי מחוץ למחלקה - מה שניקרא: Data encapsulation - אחת ממטרות התכנות מונחה עצמים זה שיהיו כמה שפחות

אפשרויות לטעות ולכן אם מגינים על משתנים ונותנים להם תכונת private - באם יהיה בהם שינוי בלתי מובן, נוכל לחפשו רק בתוך המחלקה שבה הוגדר ולא בכל התוכנית.

שימו לב שפעולת הציור היא רק מתודית ולא ממש ציור (כי לא למדנו גרפיקה בשפה).

בואו נעשה שימוש מיידי במחלקה שהגדרנו בדוגמה של תוכנית:

```
using System;
using System.Collections.Generic;
namespace Circles
{
    /* כאן תכנס ההגדרה של המחלקה עיגול שכתבנו קודם, ללא תווי ההערה כמובן */

    class MainClass
    {
        public static void Main(string[] args)
        {
            Circle c1= new Circle(4,2,3); //Instantiating one object
            Circle c2=new Circle(7,5,2); //Instantiating second object
            c1.draw();
            c2.draw();
            c1.change_radius(6);
        }
    }
}
```

שימו לב שהמחלקה MainClass זה מה שסביבת העבודה יצרה. זוהי הייתה עד היום המחלקה היחידה שבה גם נימצאת הפעולה (מתודה) Main ששם בעצם מתחילה התוכנית. כלומר המחלקה החדשה Circle אינה ברת ביצוע, אלא אם קוראים לה.

צעד אחד אחורה - עבור כל פעולה או method שניכתוב בשפת C# בד"כ בכותרת של הפעולה נראה מילות מפתח כמו: void, public, int וכד'. כעת הגיע הזמן להסביר מה הכוונה.

public - יש גם private כפי שראינו בעיגול. ב-public כל פעולה אחרת יכולה להשתמש בפעולה שמוגדרת עם public. למשל בתוך פעולת ה-Main נוכל לקרוא לפעולה אחרת אם האחרת מוגדרת כ-public.

void או טיפוס משתנה כמו int או string או long או כל טיפוס אחר - מקדים הגדרה של פעולה באופן שהוא מציין איזה סוג ערך 'מחזירה' הפעולה.

פעולה מחזירה ערך על ידי מילת המפתח: return. פעולה זו בעצם פונקציה, למי שמכיר מושג זה, אבל פונקציה שמוגדרת בתוך מחלקה ואפשר להפעילה גם מחוצה לה.

דוגמה לפעולה שלא מחזירה שום ערך: `public void print()` - זוהי כותרת של פעולה בשם `print` שניתן להשתמש בה על ידי פעולות אחרות כי היא `public` והיא לא מחזירה שום ערך ולכן: `void`

כעת נוסיף למחלקה `Circle` פעולה בשם `GetRadius`, אשר תחזיר את ערך הרדיוס שלו, וכך נכתוב אותה:

```
public int GetRadius()
{
    return r;
}
```

כעת בתוכנית ה- `Main` שלנו, לאחר שניצור עיגול כלשהו, אפשר לאחזר את רדיוסו בעזרת הפעולה: `GetRadius` (זיכרו שלא ניתן לגשת ישירות ל- `r` ב-`Main` כי `r` מוגדר כ- `private`)

```
Circle c1= new Circle(4,2,3);
x=c1.GetRadius();           // x will get the value 3 here
```

עשינו 'זימון' של פעולת ה- `GetRadius` על העצם שהוגדר כ- `c1`. זה בעצם אומר להפעיל את פעולת ה- `GetRadius` על העצם `c1`. ומכיוון שבנינו אותו עם רדיוס של 3, הפעולה `GetRadius` תחזיר 3.

תרגיל

יש להגדיר מחלקה בשם `Point` שלה 2 מאפיינים. קואורדינטת `x` וקואורדינטת `y`. שתיהן מטיפוס `double`.
הבנאי יקבל 2 ערכים מסוג `double` הראשון יכנס ל- `x` והשני ל- `y`.
יש להוסיף פעולות שמאחזרות את ערכי המאפיינים, פעולה לכל מאפיין. `GetX` ו- `GetY`
יש להוסיף פעולות שמשנות את המאפיינים: `SetX` ו- `SetY`
יש להוסיף פעולה בשם: `Print` שתדפיס את הנקודה בצורה יפה, למשל: `(X=... Y=...)`

בתוכנית הראשית נקלוט 4 מספרים, 2 לכל נקודה, ניצור מהן 2 נקודות בעזרת המחלקה הנ"ל. נחשב את נקודת האמצע וניצור אותה (ממוצע `x` וממוצע `y` של שתי הראשונות), ואז נדפיס את 3 הנקודות. הכל בעזרת הפעולות שהגדרנו במחלקה.