

סיכום שיעור

כתבנו תוכנית שמוצאת אם מספר הוא ראשוני או לא. הרעיון היה לנסות ולחלק את המספר שקלטנו במספרים הולכים וגדלים מ-2 ועד לאחד פחות המספר עצמו. אם לא התחלק באף אחד \leq ראשוני. אם התחלק בדרך: לא ראשוני.

ראשוני - prime
לא ראשוני - composite

שיכללנו את התוכנית בכך שהגענו לבדוק עד לשורש של המספר + 1 (כי אם לא התחלק עד לשורש, גם הלאה לא יתחלק) - למשל אם 1,000,003 לא התחלק עד 1001 - הוא לא יתחלק במספרים גדולים ממנו (חוק הסימטריה בכפל).

הנה אפשרות לכתוב את התוכנית

```
using System;

namespace prime
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Enter a number to check");
            int input = int.Parse(Console.ReadLine());
            int upper = (int)Math.Sqrt(input) + 1;
            int div = 2;
            int composite = 0;
            if (input == div)
                Console.WriteLine("prime");
            else
            {
                while(div<=upper && composite==0)
                {
                    if (input % div == 0)
                    {
                        Console.WriteLine("Composite");
                        composite = 1;
                    }
                    ++div;
                }
                if(composite==0)
                    Console.WriteLine("prime");
            }
        }
    }
}
```

בשורה: `int upper = (int)Math.Sqrt(input) + 1`

ביצעתי הסבה של double ל-int. זה משהו שעדיין לא דיברנו עליו ונקרא: Data casting.

בהמשך דיברנו על מושגים בסיסיים לגבי **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 ופעולות. משתנים כבר ראינו ופעולות אל קיטעי תוכנה שעושים 'משהו'.

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

צורת יצירת עצם חדש ממחלקה שניקראת: Student

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

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

ראינו דוגמה להגדרת מחלקה עבור עיגול, בשם Circle

המחלקה של העיגול: לעיגול תהיינה 3 תכונות - קואורדינטת x, קואורדינטת y 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 המעגל 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 מייד במחלקה שהגדרנו בדוגמה של תוכנית:

```
using System;
using System.Collections.Generic;
namespace Circles
{
    /* Here comes the definition of Circle as we outlined above */
    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);
        }
    }
}
```

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 פעולת ה-Main תוכל 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

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.