מערכי עצמים ב-C# (Array of Objects)

פרק ראשון: יצירת מערך עצמים, הקצאת זיכרון ועבודה עם תאים

מהו מערך של עצמים?

כפי שאנו כבר מכירות מערך של טיפוסים פשוטים כגון `int[]` או `string[]`, ניתן ליצור מערך שבו כל איבר הוא **עצם (Object)** ממחלקה שהגדרנו.

> **■ דימוי אינטואיטיבי:**
> אם עצם יחיד מטיפוס `Student` הוא תיק אישי של תלמיד, אז מערך `Student[]` הוא כמו ארון תאים שלם בכיתה, שבו בכל תא מונח תיק של תלמיד אחר.

תהליך שני השלבים – החלק הקריטי!

ביצירת מערך של עצמים יש לבצע **שני שלבים נפרדים** בזיכרון המחשב. שכחה של השלב השני היא הטעות הנפוצה ביותר:

שלב א': יצירת המערך ("הארון")
מקצים מקום למערך שמכיל הפניות (References). בשלב זה כל התאים מכילים ערך `null`!

שלב ב': יצירת כל עצם בנפרד ("החפצים")
עבור כל תא במערך, חובה להפעיל פעולה בונה (`new Student(...)`) כדי ליצור עצם ממשי.

> **▲ זהירות מחריגת NullReferenceException:**
> אם ננסה לפנות לתא במערך שטרם יצרנו בו עצם (למשל: `students[0].GetName()`) התוכנית תקרוס בזמן ריצה, כיוון שהתא מצביע כרגע על "כלום" (`null`).

תכל'ס בקוד: הגדרה, אתחול וגישה לעצמים במערך

```
// יצירת מערך בגודל 3 תאים (שלב א': כרגע כולם מכילים null)
Student[] students = new Student[3];

// שלב ב': יצירת עצם ממשי בכל תא באמצעות הפעולה הבונה
students[0] = new Student("נועה", 95);
students[1] = new Student("תמר", 88);
students[2] = new Student("מיכל", 92);

// קריאה לפעולות של העצם:
Console.WriteLine(students[0].GetName());  // פלט: נועה
Console.WriteLine(students[1].GetGrade()); // פלט: 88
```

סריקת מערך עצמים באמצעות לולאת for

כאשר נרצה לחשב ממוצע או להדפיס את נתוני כל העצמים, נסרוק את המערך בלולאה:

```
double sum = 0;
for (int i = 0; i < students.Length; i++)
{
    sum += students[i].GetGrade();
    Console.WriteLine(students[i].ToString());
}
Console.WriteLine($"ממוצע הציונים: {sum / students.Length}");
```

מערך כתכונה במחלקה (Array as a Field)

פרק שני: ניהול אוסף נתונים פנימי בתוך עצם יחיד

הרעיון: עצם המכיל בתוכו רשימה

במקרים רבים עצם בודד נדרש לשמור כמות מרובה של נתונים מאותו סוג. במקום להגדיר משתנים נפרדים רבים, נגדיר **מערך כתכונה פנימית (Field)** של המחלקה.

- **דוגמה 1:** תלמידה (`Student`) שמחזיקה מערך ציונים פנימי (`int[] grades`).
- **דוגמה 2:** קבוצה (`Team`) שמחזיקה מערך שמות משתתפות (`string[] members`).
- **משתנה מונה מלווה:** לעיתים קרובות נלווה למערך משתנה עזר (כגון `count`), שמציין כמה איברים הוכנסו בפועל למערך עד כה.

דוגמת קוד מלאה: מחלקת Student המנהלת מערך ציונים

```
public class Student
{
    private string name;
    private int[] grades;      // מערך כתכונה פנימית
    private int gradesCount;   // מונה כמות הציונים שהוזנו בפועל

    // פעולה בונה המקבלת שם וכמות מקסימלית של ציונים
    public Student(string name, int maxGrades)
    {
        this.name = name;
        this.grades = new int[maxGrades]; // הקצאת המערך בבנאי
        this.gradesCount = 0;
    }

    // פעולה להוספת ציון למערך התכונה
    public bool AddGrade(int grade)
    {
        if (this.gradesCount < this.grades.Length)
        {
            this.grades[this.gradesCount] = grade;
            this.gradesCount++;
            return true;
        }
        return false; // אין מקום פנוי במערך
    }

    // פעולה לחישוב ממוצע הציונים מתוך המערך הפנימי
    public double GetAverage()
    {
        if (this.gradesCount == 0) return 0;
        double sum = 0;
        for (int i = 0; i < this.gradesCount; i++)
        {
            sum += this.grades[i];
        }
        return sum / this.gradesCount;
    }
}
```

■ **נקודה חשובה להבנה:**
הקצאת הזיכרון של המערך מתבצעת **בתוך הפעולה הבונה**. כל מופע חדש של `Student` שנוצר מקבל מערך ציונים אישי ועצמאי לחלוטין בזיכרון.

הכלה ב-C# ("Has-A" / "Composition")

פרק שלישי: עצם כתכונה בתוך עצם אחר, ויחסי גומלין

מהי הכלה?

הכלה היא מצב שבו מחלקה אחת מכילה כתכונה (שדה) עצם מטיפוס של מחלקה אחרת. קשר זה נקרא בעולם התכנות קשר של **"יש ל-" ("Has-A")**.

- למכונית **יש** מנוע (מחלקה `Car` מכילה תכונה מטיפוס `Engine`).
- לאדם **יש** תאריך לידה (מחלקה `Person` מכילה תכונה מטיפוס `Date`).
- לכיתה **יש** מורה (מחלקה `Classroom` מכילה תכונה מטיפוס `Teacher`).

דוגמה מעשית: תאריך ועובד

```
// 1. המחלקה המוכלת - Date
public class Date
{
    private int day, month, year;

    public Date(int d, int m, int y)
    {
        this.day = d; this.month = m; this.year = y;
    }

    public string ToString()
    {
        return $"{day:D2}/{month:D2}/{year}";
    }
}

// 2. המחלקה המכילה - Employee (מכילה עצם מסוג Date)
public class Employee
{
    private string name;
    private Date hireDate; // הכלה! תכונה שהיא עצם בפני עצמו

    public Employee(string name, Date hireDate)
    {
        this.name = name;
        this.hireDate = hireDate;
    }

    public string ToString()
    {
        // הפעלת פעולה של האוביקט המוכל
        return $"עובד: {this.name}, תאריך קבלה: {this.hireDate.ToString()}";
    }
}
```

שילוב כוחות: הכלה ומערך עצמים יחד

כאשר מחלקה מכילה **מערך של עצמים**, נוצר שילוב עוצמתי של שני הנושאים. למשל: מחלקה `School` שמכילה מערך של תלמידים (`Student[]`).

> **■ שימי לב לעקרון הכימוס:**
> המחלקה המכילה (למשל בית ספר) לעולם אינה ניגשת ישירות לשדות הפרטיים של העצם המוכל (התלמיד), אלא רק דרך הפעולות הציבוריות שלו (כגון `student.GetAverage()`).

דוגמה מסכמת וטיפים לבחינה

פרק רביעי: מחלקת כיתה (Classroom) המנהלת מערך תלמידים + טעויות נפוצות

קוד מסכם: מחלקה המכילה מערך עצמים ופעולות עיבוד

```
public class Classroom
{
    private string className;
    private Student[] students;
    private int studentCount;

    public Classroom(string name, int maxStudents)
    {
        this.className = name;
        this.students = new Student[maxStudents];
        this.studentCount = 0;
    }

    // פעולה להוספת תלמיד לכיתה
    public void AddStudent(Student s)
    {
        if (this.studentCount < this.students.Length)
        {
            this.students[this.studentCount] = s;
            this.studentCount++;
        }
    }

    // מציאת התלמיד בעל הממוצע הגבוה ביותר
    public Student GetTopStudent()
    {
        if (this.studentCount == 0) return null;
        Student top = this.students[0];
        for (int i = 1; i < this.studentCount; i++)
        {
            if (this.students[i].GetAverage() > top.GetAverage())
                top = this.students[i];
        }
        return top;
    }
}
```

טעויות נפוצות בבחינות וכיצד להימנע מהן

- **ריצה עד `Length` במקום עד המונה (count):**
  אם המערך הוקצה לגודל 30 אך נרשמו רק 10 תלמידים, לולאה שתרוץ עד `Length` תפגוש תאי `null` ותגרום להתרסקות התוכנית. תמיד רצים בלולאה עד `studentCount`!
- **העתקת מצביעים במקום ערכים (Aliasing):**
  השמה פשוטה `students[0] = s1` מעתיקה את ההפניה (הכתובת בזיכרון) ולא יוצרת עותק חדש. שינוי דרך `s1` ישפיע ישירות על התא במערך.
- **שכחת הקצאת המערך בבנאי:**
  הגדרת `private Student[] students;` ללא `new Student[size]` בבנאי משאירה את המערך עצמו כ-`null`.

> ■ **כלל הברזל לסיכום:**
> **מערך רגיל:** שומר ערכים ישירים (מספרים, תווים).
> **מערך עצמים:** שומר *מצביעים* לעצמים. חובה לייצר את המערך, וגם לייצר כל עצם בנפרד!