

שרשרת חוליות אנרית

יצירת שרשרת והצגתה

לעיתים קרובות מתחיל התרגיל במשפט:

"נתונה שרשרת חוליות ... כתוב פעולה שתקבל את השרשרת ותבצע ..."

מטרת התרגיל: לכתוב את הפעולה הנדרשת.

הבעיה - כדי לבדוק את נכונות הפתרון שכתבנו, יש לבנות את השרשרת.

פעילות מייגעת החוזרת על עצמה עבור כל תרגיל שנרצה לפתור ולהריץ במחשב.

ואם התכנית מכילה שרשרת חוליות מטיפוסים שונים, נצטרך לכתוב את הקוד עבור כל שרשרת בנפרד.

כדי לחסוך בשכפול קוד, נוכל להמיר את הפעולות הקונקרטיות בפעולות גנריות.

שים לב: אין מחלקה המגדירה שרשרת חוליות. השרשרת הינה מבנה נתונים, ובמבנה נתונים מטפלים באמצעות פעולות סטטיות.

כדי להפוך פעולה קונקרטית (`Node<Integer>`) לפעולה גנרית: נוסיף למילה `static` את הסימון `<T>`:

`public static` (רשימת הפרמטרים שהפעולה מקבלת) `<T>` methodName טיפוס-ערך-מוחזר

אם הפרמטר מתייחס לעצמים, נרשום את טיפוס העצם כ- `T`:

- עצם יחיד `T`
- מערך מסוג העצם `T []`
- שרשרת או הפנייה `Node<T>`

למשל: הפעולה `Show` מקבלת שרשרת חוליות גנרית `chain`

הפעולה `GetChain` מקבלת מערך גנרי `arr`

שים לב: כל האמור בדף זה נכון לתרגול ליד המחשב. בבחינת הברורות נכתוב את הפעולה הקונקרטית (המשתמשת בטיפוס הנתון בשאלה)

התכנית שלהלן יוצרת שתי שרשראות של חוליות, האחת שרשרת של מספרים שלמים והאחרת של מחרוזות. הבנייה וההצגה נעשות באמצעות פעולות סטטיות גנריות:

```
public class ChkNodeT
{
    /*          שימוש בפעולות סטטיות גנריות          */
    public static void Main(string[] args)
    {
        //--- המערך המכיל את הערכים כפי שניתנו בתרגיל ---
        int [] arr1 = {3, 3, 4, 5, 3, 7, 3, 3, 5, 8, 3};
        string [] arr2 = {"aaa", "bbb", "ccc", "ddd", "eee"};

        Node <int> p1 = getChain(arr1);
        Node <string> p2 = getChain(arr2);

        show (p1);
        show (p2);
    }

    //--- פעולה המציגה את השרשרת ---
    public static void Show <T> (Node<T> chain)
    {
        Console.WriteLine("[");
        while (chain != null)
        {
            Console.WriteLine(chain.GetValue().ToString());
            if (chain.HasNext())
                Console.Write(", ");
            else
                Console.WriteLine("]");
            chain = chain.GetNext();
        }
    }

    //--- הפעולה מקבלת מערך מטיפוס גנרי, יוצרת ומחזירה שרשרת חוליות המכילות ערכים אלו ---
    public static Node<T> GetChain <T> (T [] arr)
    {
        int n = arr.Length;

        //--- בניית השרשרת ---
        Node<T> chain = new Node<T> (arr[0]);
        Node<T> pos = chain;

        for (int i = 1 ; i < n ; i++)
        {
            pos.SetNext(new Node<T> (arr[i]));
            pos = pos.GetNext();
        }
        return chain;
    }
}
```

הסימון הצהוב נועד לצרכי הדגשה בלבד ואינו מופיע בקוד

מחלקת שרשרת אנרית

נוכל ליצור מחלקת שרשרת שתכיל את הפעולות הגנריות, ולקרוא לפעולות שלה מתוך התכנית:

```

/* מחלקת שרשרת חוליות גנרית */
public class Chain
{
    //--- פעולה המציגה את השרשרת ---
    public static void Show <T> (Node<T> chain)
    {
        Console.WriteLine("[");
        while (chain != null)
        {
            Console.WriteLine(chain.GetValue().ToString());
            if (chain.HasNext())
                Console.WriteLine(", ");
            else
                Console.WriteLine("]");
            chain = chain.GetNext();
        }
    }
    //--- הפעולה מקבלת מערך מטיפוס גנרי, יוצרת ומחזירה שרשרת חוליות המכילות ערכים אלו ---
    public static Node<T> GetChain <T> (T [] arr)
    {
        int n = arr.Length;
        //--- בניית השרשרת ---
        Node<T> chain = new Node<T> (arr[0]);
        Node<T> pos = chain;
        for (int i = 1 ; i < n ; i++)
        {
            pos.SetNext(new Node<T> (arr[i]));
            pos = pos.GetNext();
        }
        return chain;
    }
}

```

התכנית:

```

public class ChkGenericChain
{
    public static void Main(string[] args)
    {
        int[] arr1 = { 3, 3, 4, 5, 3, 7, 3, 3, 5, 8, 3 };
        string[] arr2 = { "aaa", "bbb", "ccc", "ddd", "eee" };
        Node<int> p1 = Chain.GetChain(arr1);
        Node<string> p2 = Chain.GetChain(arr2);
        Chain.Show(p1);
        Chain.Show(p2);
    }
}

```

את הפעולות מזמנים ע"י ציון שם המחלקה המכילה אותן.

הסימון הצהוב נועד לצרכי הדגשה בלבד ואינו מופיע בקוד

פעולות אנריות התלויות בסוג האיבר

יש פעולות העוסקות בתוכן החוליה: הימצאות x בשרשרת, מיון הערכים בשרשרת וכד'. כדי להשוות את תוכן החוליה לערך, נשתמש בפעולה **Equals** המחזירה אמת אם הערכים שווים ושקר אחרת. כדי למיין את החוליות לפי סדר מסוים, נשתמש בפעולה **CompareTo** המחזירה מספר חיובי אם הערך המשווה (המפעיל) גדול מהערך המשווה (הפרמטר), מספר שלילי אם הוא קטן ממנו ו-0 (אפס) אם הם שווים:

```
//--- פעולה המחזירה אמת אם הערך x נמצא בשרשרת, ושקר אחרת ---
public static bool Exist<T>(Node<T> chain, T x)
{
    while (chain != null)
    {
        if (chain.GetValue().Equals(x))
            return true;

        chain = chain.GetNext();
    }
    return false;
}
```

זימון הפעולה בתכנית:

```
Console.WriteLine ("Exist(p1, 8) : " + Chain.Exist(p1, 8));
Console.WriteLine ("Exist(p2, 'ggg') : " + Chain.Exist(p2, "ggg"));
```

הפלט:

```
Exist(p1, 8) : true
Exist(p2, 'ggg') : false
```

שתי הפעולות מוגדרות בכל הטיפוסים הבסיסיים.

הפעולה **Equals** מוגדרת באופן אוטומטי (הורשה) בכל מחלקה שנכתוב, אבל משווה הפניות ולא את מצב העצם (הפעולה תחזיר אמת אם שתי ההפניות מפנות לאותו עצם, ושקר אחרת, גם אם ערך התכונות זהה).

הפעולה **CompareTo** אינה מוגדרת במחלקה שכתבנו, ויהיה עלינו לכתוב אותה במחלקה.

כדי להשתמש בפעולות הגנריות בהצלחה, נוסיף את שתי הפעולות למחלקות המגדירות את העצם.