דוגמה לשאלה ופתרון (מועד א' תשע"ב)

התרגיל הינו לתכנן מבנה נתונים לשמירת איברים במחסנית, כאשר ההחלטה על איזה מימוש של מחסנית (דהיינו איזה אובייקט מחסנית ייווצר) מתבצעת על ידי פונקציה נפרדת.
לשם כך הגדר את המחלקות הבאות:

א. הגדר ממשק (interface) ג'נרי (generics) בשם **IStack** הכולל

1. **פונקציה בשם Push**
פונקציה זו מכניסה איבר לראש מחסנית, (פונקציה זו אמורה תמיד להצליח להכניס איבר לראש המחסנית)

2. **פונקציה בשם Pop**
פונקציה זו מוציאה את האיבר הראשון מהמחסנית ומחזירה אתו, במידה והמחסנית ריקה יוחזר ערך ברירת מחדל (defult) של אותו איבר (לדוגמה עבור אובייקט מחלקה יוחזר null, עבור int יוחזר 0 וכו ...)

3. **מאפיין (property) בשם Count**
שיחזיר את מספר האיברים שכרגע במחסנית.

4. **פונקציה בשם Clear**
שמרוקנת את כל המחסנית, הפונקציה לא מחזירה כלום ולאחר הפעלתה המחסנית תהיה ריקה.

ב. הגדר מחלקה ג'נרית בשם StackByArray
מחלקה זו תממש את הממשק IStack באמצעות מערך, כלומר כל איברי המחסנית ישמרו במערך.
עליך לדאוג שהמערך יגדל במידת הצורך – לצורך כך הגדר פונקציה נוספת שתטפל בצורך הזה ותפעיל אותה במיקום המתאים.

ג. הגדר מחלקה ג'נרית בשם StackByList
מחלקה זו תממש את הממשק IStack באמצעות רשימה (List<T>) כלומר כל איברי המחסנית ישמרו ברשימה.

ד. הגדר פונקציה שתהווה factory method ותחזיר מופע של המחסנית הרצויה.

פתרון:

הערה: בפתרון השתמשתי גם ב IEnumerable לצורך הנוחות בבדיקה של הקוד
במבחן זה לא נדרש !

סעיף א:

```
public interface IStack<T> : IEnumerable
{
    void Push(T obj);
    T Pop();
    int Count { get; }
    void Clear();
}
```

סעיף ג:

```
public class StackByList<T> : IStack<T>
{
    List<T> list;
    public StackByList()
    {
        list = new List<T>();
    }
    public void Push(T obj)
    {
        list.Add(obj);
    }

    public T Pop()
    {
        T temp = default(T);
        if (this.Count > 0)
        {
            temp = list[Count - 1];
            list.Remove(temp);
        }
        return temp;
    }
    public int Count{get { return list.Count; }}

    public void Clear()
    {
        list.Clear();
    }

    public IEnumerator GetEnumerator()
    {
        return list.GetEnumerator();
    }
}
```

סעיף ב:

```
public class StackByArray<T> : IStack<T>
{
    T[] arr;
    int index = 0;
    int capacity = 2;

    public StackByArray()
    {
        arr = new T[capacity];
    }

    private void growingArray()
    {
            capacity += capacity;
            T[] temp = new T[capacity];
            for (int i = 0; i < arr.Length; i++)
                temp[i] = arr[i];
            arr = temp;
    }

    public void Push(T obj)
    {
        if (index >= capacity)
            growingArray();

        arr[index] = obj;
        index++;
    }

    public T Pop()
    {
        if (index-1 >= 0)
            return arr[--index];
        return default(T);
    }

    public int Count{get { return index; }}

    public void Clear()
    {
        index = 0;
    }

    public IEnumerator GetEnumerator()
    {
        for (int i = 0; i < index; i++)
            yield return arr[i];
    }
}
```

סעיף ד:

```
public class Factory
{
    public static IStack<T> FactoryStack<T>()
    {
        return new StackByArray<T>();
    }
}
```

דוגמה להרצה (לא נדרש בבחינה)

```
class Program
{
    static void Main(string[] args)
    {
        IStack<int> stack = Factory.FactoryStack<int>();
        stack.Push(12);
        stack.Push(11);
        stack.Push(10);
        int x = stack.Pop();
        foreach (var item in stack)
            Console.WriteLine(item);
        Console.WriteLine("-----------");
        stack.Clear();

        stack.Push(9);
        stack.Push(8);
        stack.Push(7);
        foreach (var item in stack)
            Console.WriteLine(item);

        int count = stack.Count;
        Console.WriteLine("count: " + count);
    }
}
```