```
Class TestDaysOfWeek
{
      Static void Main()
      {
            DayesOfWeek week = new DayesOfWeek();
            Foreach ( string day in week )
            {
                  Console.Write(day+ " ");
            }

            IEnumerator myEnumerator = week.GetEnumerator();
            While ( myEnumerator.MoveNext() == true )
            {
                  Console.Write(myEnumerator.Current + " " );
            }
      }
}
```

فى هذا الكود قمنا بإنشاء فئة تسمى DaysOfWeek والتى تقوم بتطبيق الـ interface التى تدعى IEnumerable. (يرجى العودة الى أى مرجع لفهم الـ Interfaces). بتطبيق تلك الـ interface على تلك الفئة, أصبح لدى الفئة دالة تسمى GetEnumerator(). إذا كان بداخل تلك الدالة كلمة yield فإن المترجم يحدد تلك الدالة على إنها iterator ويقوم بتوليد فئة جديدة تقوم بتطبيق الـ interface التى تدعى IEnumerator وايضاً يقوم بتوليد الدوال التى تحتويها تلك الـ interface و هما MoveNext و Dispose. ثم بعد ذلك قمنا بعمل Loop باستخدام الـ iterator. فى كل مرة يتم فيها تنفيذ دالة MoveNext الموجودة بداخل الـ iterator يتم تنفيذ جملة yield واحدة ثم يتوقف. ثم فى الدورة الثانية يتم تنفيذ الجملة الثانية ثم يتوقف وهكذا حتى يصل الى جملة yield break وعندها يتوقف نهائياً.

دعنا نشرح الكود شرحاً أكثر تفصيلاً:

```
IEnumerator myEnumerator = week.GetEnumerator();
```

هنا قمنا باشتقاق الكائن myEnumerator من الفئة التى يقوم المترجم بتوليدها.

```
myEnumerator.MoveNext()
```

هنا يتم ايجاد أول جملة yield. إذا وجد المترجم جملة yield break أو انتهى المجال الذى يدور فيه فسوف تعود دالة MoveNext بالقيمة false وتنتهى جملة while.

```
myEnumerator.Current
```

تلك الجملة تحتفظ بالقيمة التى تعود بها جملة yield.

أما نتيجة تنفيذ هذا البرنامج فهى كالتالى:

```
Sun Tue Thr
```