E04		erhalten am:
	Konsole	

Einheit 04a: Spielregeln in Testumgebung testen

Nun fehlt noch die Auswertung des aktuellen Spielstands.
Dazu soll eine Methode `spielAuswerten()` erstellt werden, die den aktuellen Spielstand in unserem Modell-Array auswertet und mit einem Rückgabewert das Ergebnis übermittelt.
Folgende Rückgabewerte sollen verwendet werden...

1,2 der betreffende Spieler hat gewonnen
0 es gibt keine vier Steine nebeneinander, keiner hat gewonnen, das Spiel läuft weiter.
-1 unentschieden, keiner hat gewonnen und alle Felder sind belegt.

Wenn alle möglichen Kombinationen ausgewertet werden sollen, führt das selbst in unseren 6 x 7er Array zu unzähligen, länglichen Auswahlanweisungen...
Da wir das Programm weiterhin flexibel halten wollen, also auch größere Arrays möglich sind, mach die Geschichte nicht gerade einfacher...

Wie verteilen die Auswertung deshalb auf zwei Methoden, eine Methode `viererBlock(...)` und die eigentliche Methode `spielAuswerten()`, welche `viererBlock(...)` „verwendet“...

Eine Gewinnkonstellation muss in einem Block von 4 x 4 Elementen liegen. Die Methode `viererBlock(...)` erhält Zeilen- und Spaltennummer der linken, oberen Ecke eines 4x4 Element großen Bereichs. In diesem Bereich prüft sie nun horizontal, vertikal und diagonal auf vier gleiche Spielerkennungen. Dabei sollen folgende Rückgabewert das Ergebnis liefern…

`viererBlock(...)` liefert folgende Rückgabewerte

1,2 der betreffende Spieler hat gewonnen
0 es gibt keine vier Steine nebeneinander, keiner hat gewonnen, das Spiel läuft weiter.

Testumgebung

Zunächst basteln wir uns eine Testumgebung, um die Methoden zu entwickeln und zu testen..
Erstellen Sie ein Konsolen-App (.Net Framework) Projekt mit Namen **Testumgebung** und kopieren Sie sich die `Main`-Methode der vorhergehenden Einheit in die Datei `Program.cs`.

- Wir initialisieren das Array bei der Erzeugung, so dass wir verschiedene Konstellationen eintragen und mit den Methoden zum Auswerten testen können.
  Die anderen Attribute / Methoden löschen, oder ignorieren Sie.

```
public static int[,] spielbrett = new int[6, 7] { {0,0,0,0,0,0,0},
                                                  {0,0,2,0,0,0,0},
                                                  {0,0,0,2,0,0,0},
                                                  {0,0,0,0,2,0,0},
                                                  {0,0,0,0,0,2,0},
                                                  {0,0,0,0,0,0,0} };
```

- Erstellen Sie eine private (Hilfs-)Methode `viererBlock(…)` welche wie besprochen die Indizes der linken, obereren Ecke eines Blocks aus 4 x 4 Elementen erhält.
  Tipp: lassen Sie einen neuen Verweis `sb` auf unser Array `spielbrett` verweisen, das spart Tipparbeit!!
  Die Methode prüft Zeilen, Spalten und Diagonalen auf 4 gleiche, **von 0 verschiedene** Elemente und liefert die oben besprochenen Rückgabewerte. Der Ausruf von `viererBlock(1,2)` müsste für die gezeigte Belegung 2 liefern! Wichtig ist dabei nur, sich bei den Auswahlanweisungen immer auf die übergebenen Indizes zu beziehen. z.B. könnte die Prüfung er ersten Zeile eines Viererblocks so aussehen:

  ```
  if (sb[z, s] == sb[z, s + 1] && sb[z, s] == sb[z, s + 2] && ...
  ```

  Dabei lassen sich mache gleichartigen Überprüfungen mit Schleifen zusammenfassen…
  Rufen Sie im Hauptprogramm die Methode auf und geben Sie den Rückgabewert zur Kontrolle aus.
  Testen Sie durch entsprechende Initialisierung des Arrays die Funktion der Methode.

Wenn `viererBlock(…)` funktioniert…

...erstellen Sie eine Methode `spielAuswerten()`. Die Methode „schiebt“ mit geeigneten Schleifen `viererBlock(…)` über das Array. Liefert die Hilfsmethode einen Gewinner, wird dieser Wert zurückgeliefert und die Auswertung ist beendet.
Liefern alle Aufrufe keine Gewinner, muss abschließend noch geprüft werden, ob das Spiel weiterlaufen kann oder unentschieden endet. Durchsuchen Sie dazu das Array mit geeigneten Schleifen: Gibt es kein einziges, freies Feld mehr, endet das Spiel unentschieden. Gibt es noch freie Felder, läuft das Spiel weiter. Liefern Sie entsprechend dieser Möglichkeiten 0 oder -1 zurück, wie anfangs besprochen.
Rufen Sie in der `Main`-Methode `spielAuswerten()` anstelle von `viererBlock(…)` auf und testen Sie die Funktion durch entsprechende Initialisierungen des Arrays.