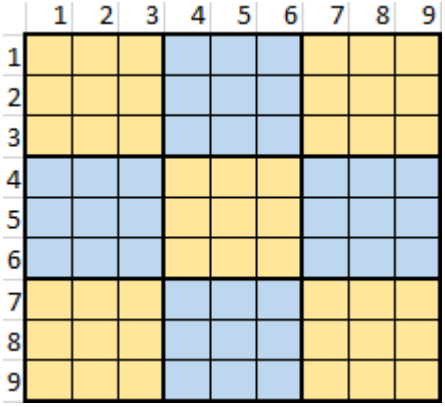
Klasse Sudoku

Sudoku sind Rätsel auf Basis einer 9x9 Felder großen Matrix. In jedem Feld können die Ziffern 1 bis 9 stehen. Gegeben werden ein paar vorbelegte Felder und ein Satz von Regeln:

- In der Zeile darf jede Ziffer nur genau einmal vorkommen.
- In jeder Spalte darf jede Ziffer nur genau einmal vorkommen
- Die 9x9 Matrix ist in 9 3x3 Blöcke aufgeteilt, hier gelb und blau hervorgehoben.



- Auch pro Block darf jede Ziffer nur jeweils einmal vorkommen.

Ihre Aufgabe ist es, die Klasse Sudoku zu implementieren. Diese Klasse soll mit dem gegebenen Testcode lauffähig sein.

Die Reihenfolge der folgenden Teile ist mit Bedacht gewählt. Sie soll Ihnen helfen. Wir empfehlen Ihnen diese beizubehalten.

Teil 1:

Beginnen Sie die Klasse Sudoku. Die öffentlichen Schnittstellen sollen so gestaltet werden, dass Zeilen und Spalten von jeweils 1 bis 9 adressiert werden.

Erstellen Sie einen Konstruktor, der ein 2-dimensionales Array von byte-Werten annimmt. Dieser soll die Inhalte in eine klasseninterne Variable kopieren. Dabei sollen höchstens 9 auf 9 Felder kopiert werden. Ist das übergebene Feld kleiner, sollen nur die ersten 9 Spalten und/oder 9 Zeilen kopiert werden.

Sie dürfen davon ausgehen, dass nur gültige Werte in dem Feld gespeichert sind. Das sind die Ziffern 1 bis 9 und für leere Felder die Zahl 0.

Erlauben Sie auch die Anweisung

```
Sudoku t = new Sudoku();
```

Erstellen Sie eine öffentliche Methode `Print()`, welche das Sudoku auf der Console ausgibt. Hier wie es aussehen soll:

```
. 3 .  . . .  . . .
. . .  1 9 5  . . .
. . 8  . . .  . 6 .  _
8 . .  . 6 .  . . .
4 . .  8 . .  . . 1
. . .  . 2 .  . . .  _
. 6 .  . . .  2 8 .
. . .  4 1 9  . . 5
. . .  . . .  . 7 .  _
```

Damit sollte nun auch der Teil 1 in der Beispiel Main() funktionieren!

Teil 2:

Erstellen Sie eine öffentliche Methode `SetzeFeld(…)`. Diese Methode soll 3 Parameter erhalten: Zunächst die x und y Koordinaten des Feldes und als drittes einen Wert vom Typ byte. Wenn x oder y nicht im Bereich von 1 bis 9 liegen oder der Wert nicht im Bereich 1 bis 9 liegt, soll eine `ArgumentException` ausgelöst werden. Ansonsten soll der Wert in das Feld übernommen werden und true zurückgegeben werden.

Jetzt sollte auch der Teil 2 des Beispielcodes in Main() funktionieren. Die Ausgabe sollte danach zusätzlich dies enthalten:

```
Setze 4 in das 3. Feld der ersten Zeile:
. 3 4  . . .  . . .
. . .  1 9 5  . . .
. . 8  . . .  . 6 .  _
8 . .  . 6 .  . . .
4 . .  8 . .  . . 1
. . .  . 2 .  . . .  _
. 6 .  . . .  2 8 .
. . .  4 1 9  . . 5
. . .  . . .  . 7 .  _
```

Teil 3:

Erstellen Sie die private Methode `zeilenZiffernZaehlen`, welcher der Index einer Zeile übergeben wird. Die Methode soll ein Array zurückgeben, in dem die Häufigkeit der Ziffern stehen. Dabei sollen die Häufigkeiten der Ziffer 1 am Index 0 und die der Ziffer 9 am Index 8 zu finden sein. Leerstellen brauchen nicht gezählt zu werden.

Erstellen Sie eine private Methode `ziffernFehler`. Sie soll ein Array, wie es `zeilenZiffernZaehlen` liefert, prüfen. Sollte eine Ziffer öfter als einmal vorkommen, soll sie false zurückliefern.

Erstellen Sie eine öffentliche Methode `ZeilenFehler` die zu einer Zeile 1 bis 9 true liefert, falls Ziffern mehrmals vorkommen. Leerstellen sind natürlich egal. Nutzen Sie hierzu die vorhandenen privaten Methoden.

Teil 4:

Erstellen Sie zwei öffentliche Methoden `SpaltenFehler` und `BlockFehler`. `SpaltenFehler` soll zu einer Zeile (1 bis 9) bei einem Fehler true liefern. `BlockFehler` soll zu einem Block bei einem Fehler true liefern. Der Block wird per x und y, je 1 bis 3 identifiziert. Der Block oben links hat demnach die Indizes x = 1 und y= 1, der unten in der Mitte x = 2 und y = 3.

Reduzieren Sie die Funktionen auf sinnvolle Inhalte. Eventuell lohnt es sich vorhandene Funktionen zu nutzen oder private Funktionen anzulegen.

Teil 5:

Erstellen Sie ein Property `Fehler`, welches true liefert, sobald eine Zeile, eine Spalte oder ein Block Fehler enthält.

Erstellen Sie ein Property `Unvollstaendig`, welches true liefert solange noch Felder leer sind.

Teil 6:

Ergänzen Sie einen Konstruktor, der eine Instanz von Sudoku erhält und das Spielfeld kopiert.

Implementieren Sie eine Methode `IstGleich`, welche eine Instanz von Sudoku übergeben bekommt und dies mit der Instanz selbst vergleicht. Dabei sollen alle Felder auf Gleichheit geprüft werden. Sind alle Felder gleich gefüllt, liefert IstGleich true sonst false.

Teil 7:

Erstellen Sie eine statische Methode `LeseDatei` welche ein Spielfeld aus einer Datei ausließt und als neues Sudoku Objekt zurückgibt. Eine Beispieldatei beispiel.sudoku ist gegeben.

Teil 8:

Erstellen Sie eine Methode `SpeichernUnter` welche Dateien wie aus Teil 7 schreiben kann.

```
static void Main(string[] args)
{
	// Teil 1
	byte[,] Beispiel = {
		{ 0,3,0, 0,0,0, 0,0,0 }, // 0
		{ 0,0,0, 1,9,5, 0,0,0 },
		{ 0,0,8, 0,0,0, 0,6,0 }, // 2
		{ 8,0,0, 0,6,0, 0,0,0 }, // 3
		{ 4,0,0, 8,0,0, 0,0,1 },
		{ 0,0,0, 0,2,0, 0,0,0 }, // 5
		{ 0,6,0, 0,0,0, 2,8,0 }, // 6
		{ 0,0,0, 4,1,9, 0,0,5 },
		{ 0,0,0, 0,0,0, 0,7,0 } // 8
	};

	Sudoku s = new Sudoku(Beispiel);
	Sudoku t = new Sudoku();

	s.Print();
	t.Print();

	// Teil 2
	Console.WriteLine("Setze 4 in das 3. Feld der ersten Zeile:");
	s.SetzeFeld(3, 1, 4);
	s.Print();

	// Teil 3
	Console.WriteLine("Zeilen:");
	for (int i = 1; i <= 9; i++)
		Console.WriteLine("{0} {1}", i, s.ZeilenFehler(i) ? "Fehler" : "OK");

	// Teil 4
	Console.WriteLine("Spalten");
	for (int i = 1; i <= 9; i++)
		Console.WriteLine("{0} {1}", i, s.SpaltenFehler(i) ? "Fehler" : "OK");
	Console.WriteLine("Blöcke:");
	for (int i = 1; i <= 3; i++)
		for (int j = 1; j <= 3; j++)
		Console.WriteLine("{0}|{1} {2}", i, j, s.BlockFehler(i, j) ? "Fehler" : "OK");

	// Teil 5
	if (s.Fehler)
		Console.WriteLine("Fehlerhaft!");
	else
		Console.WriteLine("Fehlerfrei");

	// Teil 6
	t = new Sudoku(s);

	Console.WriteLine("t == s         : {0}", (t == s) ? "JA" : "NEIN");
	Console.WriteLine("t.IstGleich(s) : {0}", (t.IstGleich(s)) ? "JA" : "NEIN");


	t.SetzeFeld(1, 1, 5);
	Console.WriteLine("t neu:");
	t.Print();
	Console.WriteLine("t.IstGleich(s) : {0}", (t.IstGleich(s)) ? "JA" : "NEIN");
}
```