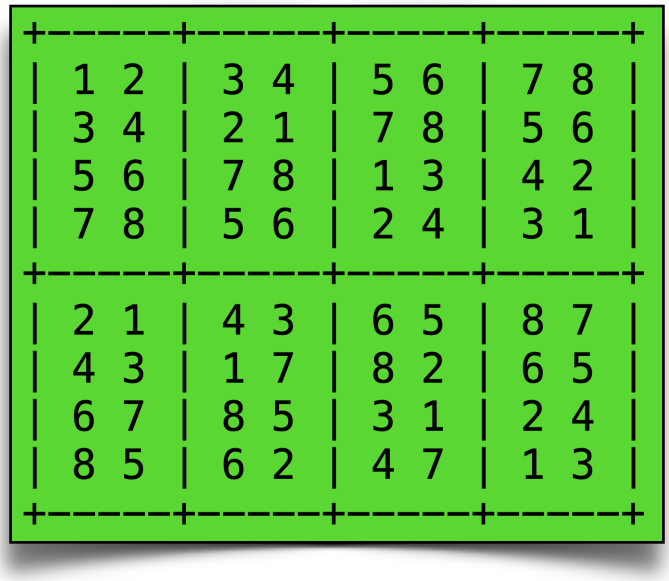
5. Praktikum: *Oktadoku*



数独

Bearbeitungszeitraum:	25.01.21 - 10.02.21, 9.15 Uhr
Gesamtpunktzahl:	5 Abgabepunkte + 25 Leistungspunkte
Praktikumsziel:	Sie implementieren abschließend einen Sudoku-Solver in Java, d.h. eine Klasse, mit der man (potentiell beliebige) spezielle Sudokus lösen kann. Da es sich um ungewöhnliche 8x8 Sudokus (mit der oben beispielhaft gegebenen Struktur) handelt, sollen diese hier ***Oktadokus*** heißen.
Voraussetzungen:	Alles was sie bisher zu Java gelernt haben.
Aufgabe:	Erweitern Sie das vorgegebene Fragment der Klasse **Oktadoku** so, dass man damit 8x8-Sudokus lösen kann.
FAQ:	Am Ende dieses Dokuments finden Sie die am häufigsten gestellten Fragen. Sie sind Teil der Aufgabenstellung und könnten sich während des Bearbeitungszeitraumes ändern, schauen Sie deshalb bitte öfter mal vorbei. **Sollten Sie sich die Aufgabenstellung ausdrucken, so achten Sie bitte darauf, dass sich die FAQ weiterhin ändern können.**
Abgabe:	Die vollständig implementierte Klasse **Oktadoku.java** senden Sie mit Hilfe von *Moodle* als Lösung ein. Implementieren Sie alle vorgegebenen Methoden sinnvoll. *Verwenden Sie genau diesen Programmnamen und achten Sie auf Groß- und Kleinschreibung.*

Aufgabenstellung

Implementieren Sie die Klasse **Oktadoku** mit (mindestens) den folgenden Methoden. **solve** soll dabei eine (beliebige) Lösung bestimmen und ausgeben oder aber die Unlösbarkeit eines **Oktadokus** ermitteln. In einem solchen **Oktadoku** müssen in jeder Zeile, in jeder Spalte und in jedem der 4x2-Felder jede der Ziffern 1 bis 8 genau einmal vorkommen.

Die Schnittstelle der Klasse ist folgende:
Oktadoku.solve() soll im Erfolgsfall die Zeichenkette `Oktadoku` bzw. `Oktadoku mit Diagonalen` gefolgt von der gefundenen Lösung ausgeben, andernfalls die Zeichenkette `nicht loesbar :-(` .

```
class Oktadoku {
  public enum Variante { normal, mitDiagonalen };
  private Variante v;
  // was sonst noch noetig ist …

  public Oktadoku(Variante var)  {  /* TODO */  }
  public void read()             {  /* TODO */  }
  public void write()            {  /* TODO */  }
  public boolean check()         {  /* TODO */  }
  public void solve()            {  /* TODO */  }
}
```

Das (verbindlich zu benutzende) Testprogramm ist unverändert zu übernehmen:

```
class testOktadoku {
        public static void main (String args []) {
                Oktadoku.Variante v = Oktadoku.Variante.normal;
                if (args.length > 0 && args[0].equals("-x"))
                    v = Oktadoku.Variante.mitDiagonalen;
                Oktadoku s = new Oktadoku(v);

                s.read();
                s.write();
                if (s.check())
                        s.solve();
                else
                        System.out.println("nicht ok");
        }
}
```

Da es sich um die letzte Aufgabe (mit der höchsten Punktzahl) handelt, wollen wir Ihnen den Lösungsalgorithmus komplett selbst überlassen.

Für die Implementation der erforderlichen Tests auf Einhaltung der üblichen Sudoku-Regeln sollten Sie die Klasse **`BitSet`** aus der Java-Bibliothek verwenden (vgl. `https://docs.oracle.com/javase/8/docs/api/index.html?java/util/BitSet.html`).

Es folgen noch einige Beispiele für die erwarteten Ausgaben:

```
$ java testOktadoku < in1
Oktadoku
 +-----+-----+-----+-----+
 | 3   |     |     |   5 |
 |     | 7 6 |     |     |
 |   2 |     |   7 |     |
 |   5 |   4 |     | 6 1 |
 +-----+-----+-----+-----+
 | 6 1 |     | 8   | 2   |
 |     | 4   |     | 8   |
 |     |     | 5 3 |     |
 | 2   |     |     |   6 |
 +-----+-----+-----+-----+
Oktadoku
 +-----+-----+-----+-----+
 | 3 6 | 1 8 | 4 2 | 7 5 |
 | 4 8 | 7 6 | 1 5 | 3 2 |
 | 1 2 | 3 5 | 6 7 | 4 8 |
 | 7 5 | 2 4 | 3 8 | 6 1 |
 +-----+-----+-----+-----+
 | 6 1 | 5 7 | 8 4 | 2 3 |
 | 5 3 | 4 1 | 2 6 | 8 7 |
 | 8 7 | 6 2 | 5 3 | 1 4 |
 | 2 4 | 8 3 | 7 1 | 5 6 |
 +-----+-----+-----+-----+
```

```
$ java testOktadoku -x < in1
Oktadoku mit Diagonalen
 +-----+-----+-----+-----+
 | 3   |     |     |   5 |
 |     | 7 6 |     |     |
 |   2 |     |   7 |     |
 |   5 |   4 |     | 6 1 |
 +-----+-----+-----+-----+
 | 6 1 |     | 8   | 2   |
 |     | 4   |     | 8   |
 |     |     | 5 3 |     |
 | 2   |     |     |   6 |
 +-----+-----+-----+-----+
nicht loesbar :-(
```

```
$ java testOktadoku -x < in7
Oktadoku mit Diagonalen
 +-----+-----+-----+-----+
 | 1 2 | 3 4 |     |     |
 |     |     |     |     |
 |     |     |     |     |
 |     |     |     |     |
 +-----+-----+-----+-----+
 |     |     |     |     |
 |     |     |     |     |
 |     |     |     |     |
 |     |     | 5 6 | 7 8 |
 +-----+-----+-----+-----+
Oktadoku mit Diagonalen
 +-----+-----+-----+-----+
 | 1 2 | 3 4 | 6 5 | 8 7 |
 | 3 4 | 1 2 | 7 8 | 5 6 |
 | 5 6 | 7 8 | 1 2 | 3 4 |
 | 7 8 | 6 5 | 3 4 | 1 2 |
 +-----+-----+-----+-----+
 | 8 7 | 5 6 | 2 1 | 4 3 |
 | 6 5 | 8 7 | 4 3 | 2 1 |
 | 2 1 | 4 3 | 8 7 | 6 5 |
 | 4 3 | 2 1 | 5 6 | 7 8 |
 +-----+-----+-----+-----+
```

```
$ java testOktadoku < in4
Oktadoku
 +-----+-----+-----+-----+
 |   6 |   1 |     |     |
 | 3 1 |     |   6 |   2 |
 | 4   | 3   | 5   |     |
 |     |     | 4   |     |
 +-----+-----+-----+-----+
 |     |   4 |     |     |
 |     |   8 |   3 |   6 |
 | 7   | 1   |     | 2 8 |
 |     |     | 7   | 4   |
 +-----+-----+-----+-----+
Oktadoku
 +-----+-----+-----+-----+
 | 5 6 | 2 1 | 3 7 | 8 4 |
 | 3 1 | 4 7 | 8 6 | 5 2 |
 | 4 8 | 3 6 | 5 2 | 1 7 |
 | 2 7 | 8 5 | 4 1 | 6 3 |
 +-----+-----+-----+-----+
 | 6 2 | 7 4 | 1 8 | 3 5 |
 | 1 4 | 5 8 | 2 3 | 7 6 |
 | 7 5 | 1 3 | 6 4 | 2 8 |
 | 8 3 | 6 2 | 7 5 | 4 1 |
 +-----+-----+-----+-----+
```

```
$ java testOktadoku -x < in4
Oktadoku mit Diagonalen
 +-----+-----+-----+-----+
 |   6 |   1 |     |     |
 | 3 1 |     |   6 |   2 |
 | 4   | 3   | 5   |     |
 |     |     | 4   |     |
 +-----+-----+-----+-----+
 |     |   4 |     |     |
 |     |   8 |   3 |   6 |
 | 7   | 1   |     | 2 8 |
 |     |     | 7   | 4   |
 +-----+-----+-----+-----+
nicht ok
```

```
$ java testOktadoku < in6
Oktadoku
 +-----+-----+-----+-----+
 |   2 |     |     |     |
 |   0 | 2   |     |     |
 |     | 0   |     |     |
 |     |     |     |     |
 +-----+-----+-----+-----+
 |     |     |     |     |
 |     |     |   2 |     |
 |     |     |   0 | 2   |
 |     |     |     | 1   |
 +-----+-----+-----+-----+
Oktadoku
 +-----+-----+-----+-----+
 | 1 2 | 3 4 | 5 6 | 7 8 |
 | 3 4 | 2 1 | 7 8 | 5 6 |
 | 5 6 | 7 8 | 1 3 | 4 2 |
 | 7 8 | 5 6 | 2 4 | 3 1 |
 +-----+-----+-----+-----+
 | 2 1 | 4 3 | 6 5 | 8 7 |
 | 4 3 | 1 7 | 8 2 | 6 5 |
 | 6 7 | 8 5 | 3 1 | 2 4 |
 | 8 5 | 6 2 | 4 7 | 1 3 |
 +-----+-----+-----+-----+
```

FAQ

1. **Q:** Wie schnell soll mein Programm eine Lösung finden, bzw. die Unlösbarkeit ermitteln?
   A: Auch mit einem *brute force* Ansatz (d.h. durch blindes Ausprobieren ALLER Möglichkeiten) sollten alle Eingaben aus der Beispielsammlung auf der Referenzplattform in unter 5 Sekunden gelöst werden.
2. **Q:** Muss für den Beispiel-Input `in6` und `in7` die Lösung am Anfang dieser Aufgabenstellung ermittelt werden?
   A: Nein, es reicht irgendeine (korrekte) Lösung.
3. **Q:** Sollen Fehler bei der Eingabe behandelt werden (weniger als 8 Zeichen auf der Zeile, weniger als 8 Zeilen)?
   A: Nein, Sie können davon ausgehen, dass wenigstens 8 Zeilen mit jeweils wenigstens 8 Zeichen in der Eingabe zur Verfügung stehen. Überzählige Zeilen/Zeichen sollen ignoriert werden. Nicht-Ziffern (auch 0 vgl. `in6`!) sollen als freie Position behandelt werden.

4. **Q:** Was hat es mit diesem *Oktadoku mit Diagonalen* auf sich?
   A: Bei einem solchen kommt auch auf den Hauptdiagonalen jede Ziffer nur einmal vor.
5. **Q:** Wo finde ich weitere Eingabebeispiele für Oktadokus unterschiedlicher Schwierigkeit?
   A: Schauen Sie auf http://menneske.no/sudoku/eng nach.
6. **Q:** Die Ausgabe beim vorletzten Beispiel ist `nicht ok`, ist das gleichzusetzen mit `nicht loesbar :-(` ?
   A: Nein, das meint, dass schon der Input die Sudoku-Regeln verletzt.

ACHTUNG

Um die Korrektur der Aufgaben durch die Tutoren zu erleichtern, gelten wie immer (d.h. auch bei allen weiteren Aufgaben) die folgenden Regeln:

- Auch wenn Java prinzipiell die Verwendung von Umlauten (und vielen anderen Sonderzeichen) erlaubt, sollten Sie im Praktikum generell darauf verzichten, um sich nicht vom sog. *encoding* abhängig zu machen. Das betrifft übrigens auch Texte in Kommentaren!

- Eingesandte Java-Programme, die nicht vom Java-(Referenz-)Compiler akzeptiert werden (d.h. Fehler bei der Übersetzung liefern) werden ohne weitere Inspektion mit 0 (Leistungs-) Punkten bewertet! Ob Sie die 5 Abgabepunkte erhalten, entscheiden wir in Abhängigkeit davon, ob Ihre Einsendung genügend mit der Aufgabe zu tun hat ;-)

- Halten Sie sich bei der Benennung von Klassen, Funktionen und Dateien immer an die Vorgaben und achten Sie darauf, dass Sie keine Packages verwenden (s.o.). Alle Abweichungen von diesen Regeln führen zu zusätzlichem Aufwand bei der Bewertung und daher auch zu Punktabzug.

- Die vom implementierten Programm erzeugten Ausgaben müssen das geforderte Format exakt einhalten (inkl. Zeilenstruktur und Leerzeichen). Soll und Ist werden automatisch verglichen. Sofern Abweichungen festgestellt werden, gibt es Punktabzug, über dessen Höhe dann die Tutoren entscheiden! Die einzig zulässige Abweichung (eine die man bei der Textansicht des Programms und seiner Ausgaben u.U. gar nicht feststellen kann) besteht in der Codierung der Zeilenenden, die unter Windows mit zwei Zeichen (carriage return [/x0d] + line feed [/x0a]) unter Unix dagegen nur mit einem Zeichen (newline [/x0a]) dargestellt werden.

Das war die 5. und letzte Praktikumsaufgabe! - Viel Erfolg

Letzte Änderung 26.1.2021
Inhaltliche Fragen/Korrekturen bitte an: Klaus Ahrens