Liebe Kolleginnen und Kollegen,

wir möchten euch zu einer spannenden **Coding Challenge** einladen! Die Teilnahme ist freiwillig ! Kein Zwang, es geht einfach darum Spaß am Coden zu haben.

Ziel ist es, einen **Sudoku Solver** zu entwickeln, der ein Sudoku-Rätsel effizient und fehlerfrei löst.

Aufgabe:
Entwickelt einen Algorithmus, der ein 9x9 Sudoku-Rätsel löst. Der Solver soll so schnell wie möglich arbeiten und dabei sicherstellen, dass das Sudoku korrekt ausgefüllt wird, d.h. jede Zahl von 1 bis 9 kommt in jeder Zeile, jeder Spalte und jedem 3x3-Untergitter nur einmal vor.

Anforderungen:

- **Programmiersprache:** Wählt eine objektorientierte Programmiersprache (z.B. Java, C++, Python, C#).
- **Lösungsqualität:** Das Sudoku muss korrekt gelöst werden.
- **Geschwindigkeit:** Die Lösung, die das Sudoku am schnellsten löst, gewinnt.
- **Codequalität:** Der Code sollte sauber, gut strukturiert und gut lesbar sein.

Bewertungskriterien:

- **Fehlerfreiheit:** Das Sudoku muss korrekt gelöst werden.
- **Geschwindigkeit:** Das Programm dass die Aufgabe am schnellsten, fehlerfrei löst gewinnt. Alle Programme werden final auf einen PC gemessen.
- **Code:** Objektorientiert, mehrere Klassen sollten verwendet werden
- **Bonuspunkte:** das Programm erkennt nicht ein-eindeutig lösbare Sudokus. Die Ausführungszeit wird dann künstlich um 5% reduziert.

Teilnahmeschluss:
Ihr habt bis zum **31. Januar 2025** Zeit, eure Lösung „einzureichen“.

Preis:
Für den Gewinner gibt es: Ruhm und Ehre im Entwicklerkaffee.

Beispiel:

```
def main():
    # Beispiel-Sudoku (0 bedeutet leeres Feld)
    sudoku_board = [
        [5, 3, 0, 0, 7, 0, 0, 0, 0],
        [6, 0, 0, 1, 9, 5, 0, 0, 0],
        [0, 9, 8, 0, 0, 0, 0, 6, 0],
        [8, 0, 0, 0, 6, 0, 0, 0, 3],
        [4, 0, 0, 8, 0, 3, 0, 0, 1],
        [7, 0, 0, 0, 2, 0, 0, 0, 6],
        [0, 6, 0, 0, 0, 0, 2, 8, 0],
        [0, 0, 0, 4, 1, 9, 0, 0, 5],
        [0, 0, 0, 0, 8, 0, 0, 7, 9]
    ]

    # Erstelle eine Instanz des SudokuSolvers
    solver = SudokuSolver(sudoku_board)

    # Zeitmessung starten
    start_time = time.time()

    # Lösungsaufruf
    solved_board = solver.solve()

    # Zeitmessung stoppen
    end_time = time.time()
```

Wir sind gespannt auf eure Lösungen und wünschen euch viel Spaß beim Coden!