Institut für Softwaretechnik und Programmiersprachen | Sommersemester 2024
Matthias Tichy, Raphael Straub und Florian Sihler

Abgabe (git) bis
14. Juli 2024

Threads and GUI

7 ★ 6 🟦 4 🟢

- Threads
- GUI

Aufgabe 1: Threads und Synchronisierung

Im Repository finden Sie eine Java-Klasse mit dem Namen `Deadlock`. Betrachten Sie den Code und führen Sie ihn aus. Beantworten Sie dann die folgenden Fragen.

a) Möglichkeiten der Nutzung von `synchronized` ★

Was ist der Unterschied zwischen einer `synchronized` Methode und den `synchronized` Blöcken in der `run` Methode in der Klasse?

b) Fehlerwahrscheinlichkeiten ★

Warum werden in dieser Aufgabe zwei Threads hunderte Male in einer Schleife gestartet? Würde der Fehler nicht auftreten, wenn die Schleife nicht da wäre?

c) Fehlerauftreten ★

Wenn das Programm ausgeführt wird, bleibt es in einem Deadlock stecken. An welcher Stelle im Code bleiben die Threads genau hängen?

d) To `join` or not to `join` ★

Wofür sind die `join` Aufrufe gut? Was passiert, wenn sie entfernt werden?

e) Gründe und Behebung der Deadlocks ★ 🟦

Wieso kommt es zum Deadlock?

Wie kann der Deadlock verhindert werden?

Aufgabe 2: Sudoku

In einer früheren Übung haben wir bereits Sudoku implementiert. Wir haben einen Solver geschrieben, der ein Sudoku lösen kann. Nun wollen wir den Solver nutzen, um neue Sudokus zu generieren.

Hierfür betrachten wir zunächst ein leeres Sudoku. Wir füllen eine beliebige Anzahl an Zellen mit Zahlen. Dann rufen wir den Solver auf, um das Sudoku zu lösen. Das Ergebnis ist ein gültiges Sudoku. Im letzten Schritt entfernen wir eine beliebige Anzahl an Zellen, um das Sudoku unvollständig zu machen.

a) The Application

Zunächst einmal wollen wir ein simples Fenster öffnen, das zunächst einmal nichts macht. Implementieren Sie hierfür in der Klasse `SudokuApp` die Methode `start`. Als Root können Sie eine leere `VBox` verwenden.

Falls Sie die Aufgabe korrekt lösen, sollte ein Start der Applikation ein leeres Fenster öffnen.

b) The Board

Nun wollen wir das Sudoku-Feld tatsächlich anzeigen.

Implementieren Sie hierfür die Methode `createUI` in der Klasse `SudokuController`. Passen Sie Ihren Code aus der vorherigen Teilaufgabe so an, dass die Rückgabe der Methode `createUI` als Root der Szene gesetzt wird.

Um die Zellen des Sudokus anzuzeigen, können Sie eine `GridPane` verwenden, die Ihnen *JavaFX* zur Verfügung stellt. Verwenden Sie für die Zellen ein `TextField`, um die Zahlen anzuzeigen.

c) Changing The Board

Nun wollen wir dafür sorgen, dass der Benutzer die Zahlen in den Zellen ändern kann. Hierfür erweitern wir die Methode `createUI` in der Klasse `SudokuController`, indem wir jedem `TextField` einen `ChangeListener` hinzufügen. Einen Listener können Sie einem `TextField cell` wie folgt hinzufügen: `cell.textProperty().addListener(...)`

Der Listener wird mit dem neu eingegebenen Wert aufgerufen und soll diesen dann als neuen Wert in der Zelle setzen. Falls der Nutzer keine gültige Zahl zwischen 1 und 9 eingibt, soll die Zelle leer bleiben.

d) Solving the Sudoku

Erweitern Sie nun erneut die Methode `createUI` in der Klasse `SudokuController` um einen Button, der das Sudoku löst. Verwenden Sie also erneut einen `ChangeListener`, um auf das Drücken des Buttons zu reagieren. Zum Lösen des Sudokus können Sie den mitgelieferten Solver verwenden.

Prinzipiell kann der Benutzer das Sudoku so bearbeiten, dass es nicht lösbar ist. Allerdings gehen wir davon aus, dass der Benutzer ein lösbares Sudoku eingibt und ignorieren diesen Fall. Es steht Ihnen frei, dies als Übung zu implementieren.

Beachten Sie, dass Sie die Werte der `TextField`s nach dem Lösen updaten müssen, damit das gelöste Sudoku angezeigt wird.

e) Generating the Sudoku to Solve

Jetzt haben wir ein gelöstes Sudoku. Um ein spielbares Sudoku zu erhalten, müssen wir nun Zellen entfernen. Erneut müssen Sie hierfür die Methode `createUI` in der Klasse `SudokuController` erweitern. Fügen Sie zunächst ein weiteres `TextField` hinzu, in dem der Benutzer die Anzahl der Zellen eingeben kann, die entfernt werden sollen. Anschließend fügen Sie einen Button hinzu, der die eingegebene Anzahl an Zellen entfernt und das Sudoku spielbar macht.