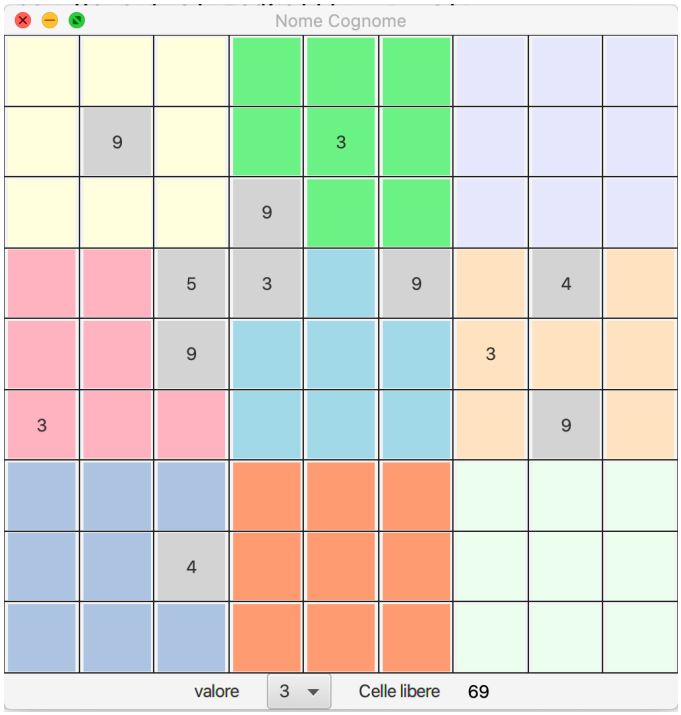
Compito di Programmazione 2/Linguaggi di Programmazione (Proff. Picco e Ronchetti)
Prova al calcolatore – gennaio 2020

Il Sudoku di ordine N è un gioco nel quale al giocatore viene proposta una griglia di $N^2 \times N^2$ celle, ciascuna delle quali può contenere un simbolo scelto da una lista di N^2 elementi differenti, oppure essere vuota. Per i Sudoku di ordine 2 e 3 in genere i simboli sono numeri compresi tra 1 ed N^2.
La griglia è suddivisa in N^2 righe orizzontali, N^2 colonne verticali e in N^2 "sottogriglie" quadrate non sovrapposte di N×N celle contigue. Queste sottogriglie (evidenziate graficamente, di solito delimitandole con bordi in neretto) sono dette "regioni".
La REGOLA è che in ogni riga, in ogni colonna e in ogni regione non vi siano simboli ripetuti.

Lo scopo del gioco è quello di riempire tutte le caselle rispettando la regola.
In tal modo in ogni riga, ogni colonna e ogni regione saranno presenti tutti i simboli, senza ripetizioni.



Si scriva un programma che gestisce un Sudoku di ordine N=2 o 3. In particolare:

1) Il programma deve riprodurre il più fedelmente possibile la figura sopra, con una scena di dimensione 500x525. Nel titolo devono figurare il vostro nome e cognome. La finestra comprende una griglia (GridPane) composta di N^4 celle, una ChoiceBox e un contatore. La ChoiceBox permette di scegliere un numero intero compreso tra 1 e N^2.
2) Le celle sono di tre tipi:
- NON INIZIALIZZATE (sfondo bianco);
- FISSE (sfondo grigio), con valori assegnabili una sola volta dall'utente;
- LIBERE (sfondo colorato), inizialmente vuote e modificabili dall'utente.
3) Le celle libere assumono un colore di sfondo che dipende dalla regione in cui si trovano. Si usino colori scelti tra i seguenti: LIGHTYELLOW, LIGHTPINK

LIGHTSTEELBLUE, LIGHTGREEN, LIGHTBLUE, LIGHTSALMON, LAVENDER, BISQUE, HONEYDEW.

4) Inizialmente all'utente viene chiesto l'ordine (N). I valori ammessi saranno N=2 ed N=3. (Si può eventualmente, con una piccola penalità, ammettere il solo valore 3 così da sviluppare il codice solo per un ordine fisso).
5) Ottenuto N, il sistema genera la griglia corrispondente, composta di celle non inizializzate.
6) Cliccando su una cella non inizializzata, questa viene sostituita da una cella fissa, il cui valore è quello mostrato al momento dalla ChoiceBox. L'operazione viene effettuata solo se la REGOLA non viene violata, altrimenti non accade nulla.
7) Al termine della generazione di N^2 celle fisse, tutte le rimanenti celle non inizializzate vengono sostituite da celle libere vuote e il gioco ha inizio.
8) Un click sulle celle fisse non avrà alcun effetto.
9) Un click sulle celle libere vuote proverà ad inserire il valore numerico mostrato al momento dalla ChoiceBox. L'inserimento avviene se la REGOLA non viene violata, altrimenti non accade nulla.
10) Un click sulle celle libere che abbiano già un valore assegnato rimuove tale valore.
11) Un contatore mostra in un campo di testo (non editabile manualmente) posto sotto la griglia il numero di celle libere attualmente vuote.
12) Quando il numero di celle libere vuote scende a zero, viene mostrata una finestra che dichiara la vittoria. (Nota: il test di questo caso è fattibile in tempi ragionevoli solo se l'ordine è 2).
13) Il programma termina (in ogni caso) quando l'utente chiude la finestra contenente la griglia.

Si richiede di usare ereditarietà ove possibile e di attenersi alle "buone pratiche" di programmazione.
Il compito è lungo: per ottenere la sufficienza non è indispensabile completare tutti i punti.
In ogni caso la consegna deve compilare senza errori, ed i punti implementati devono essere funzionanti correttamente.
Si vedano i suggerimenti alla pagina seguente.

1) Una ChoiceBox cb si popola inizialmente con

```
cb.getItems().addAll("A", "B", "C"); // lista ordinata dei valori possibili
cb.setValue("A"); // valore mostrato
```

Il valore corrente della ChoiceBox cb si ottiene con

```
String s=(String)cb.getValue();
```

2) Può risultare utile la seguente funzione per trovare un elemento in un GridPane a partire dalle sue coordinate i (indice di colonna) e j (indice di riga)

```
Node getElementAt(GridPane p, int i, int j) {
        for (Node x : p.getChildren()) {
            if ((GridPane.getColumnIndex(x) != null && GridPane.getColumnIndex(x) == i)
                    && (GridPane.getRowIndex(x) != null && GridPane.getRowIndex(x) == j)) {
                return x;
            }
        }
        return null;
    }
```

3) Per sostituire in un GridPane un oggetto A con un'oggetto B occorre rimuovere l'oggetto A (metodo *remove* applicato ai children della griglia) e aggiungere l'oggetto B (metodo *add* applicato alla griglia).

4) Potrebbe essere conveniente posticipare la colorazione delle regioni (punto 3).

5) La parte più complessa del compito è l'implementazione della REGOLA. Se progettata bene può semplificarsi parecchio.