Esame di Programmazione II, 1 febbraio 2024

(si consegnino `Sudoku.java`, `Main.java` ed eventuali classi ausiliarie che avete scritto)

Si crei un progetto Eclipse e il package `it.univr.sudoku`. Si copino al suo interno le classi del compito. Non si modifichino le dichiarazioni dei metodi e delle classi. Si possono definire altri campi, metodi o costruttori, ma devono essere `private`. Si possono definire altre classi, che in tal caso vanno consegnate. La soluzione che verrà consegnata dovrà compilare, altrimenti non verrà corretta.

Un sudoku è un gioco consistente in una griglia 9x9, divisa internamente in nove regioni 3x3, in cui sono posizionati dei numeri da 1 a 9. Alcune caselle possono essere lasciate vuote. Per esempio, questo è un sudoku:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Si richiede inoltre che i numeri in un sudoku siano unici all'interno di ciascuna riga, di ciascuna colonna e di ciascuna regione. Si noti che, tradizionalmente, i sudoku contengono numeri da 1 a 9, ma nessuno impedisce l'esistenza di sudoku in cui le nove alternative vengano rappresentate con nove oggetti di un tipo generico `E`. Per questo, la classe di partenza di questo compito è generica: `Sudoku⟨E⟩`. Tale classe possiede un costruttore che genera un sudoku casuale e che riceve:

1. il numero `empty` di caselle che devono essere lasciate vuote nel sudoku;
2. una funzione che indica come rappresentare le nove alternative con un oggetto di classe `E`. Tale funzione è data come un'implementazione dell'interfaccia di libreria `IntFunction⟨E⟩`: ha solo un metodo che dato un intero tra 1 e 9 restituisce un oggetto di tipo `E` che indica come rappresentare tale numero nel sudoku.

Per esempio, un sudoku tradizionale avrebbe tipo `Sudoku⟨Integer⟩` e il suo costruttore riceve una funzione che mappa 1 in `Integer.valueOf(1)`, 2 in `Integer.valueOf(2)`, ..., 9 in `Integer.valueOf(9)`.

Esercizio 1 (2 punti). Si completi il costruttore di `Sudoku` in modo da lanciare un'eccezione di tipo `IllegalArgumentException` se `empty` non fosse tra 0 e 61 inclusi.

Esercizio 2 (7 punti). Si completino i metodi `isHorizontallyUnique`, `isVerticallyUnique` e `isUniqueInRegion` di `Sudoku`, che determinano se un elemento del sudoku è unico nella sua riga, nella sua colonna e nella sua regione, rispettivamente.

Esercizio 3 (7 punti). Si completi il metodo `hide` di `Sudoku`, che rende vuote esattamente `howMany` caselle a caso del sudoku che non siano già vuote. Si noti che, nella matrice con cui è implementato il sudoku, le caselle vuote vengono rappresentate con il numero 0.

Esercizio 4 (7 punti). Si completi il metodo `toString` di `Sudoku`, che restituisce una stringa che

descrive il sudoku, come quelle che vedete nelle stampe che seguono. **Suggerimento:** potrebbe esservi utile il metodo `repeat` delle stringhe, che restituisce una stringa ripetuta più volte.

Esercizio 5 (8 **punti).** Si completi la classe `Main` in modo da creare i cinque sudoku indicati nel suo codice prima delle loro cinque stampe.

Se tutto è corretto, l'esecuzione del `Main` stamperà qualcosa del tipo:

```
Un sudoku di interi (1-9) con 61 caselle nascoste
 6 |5 9|
  8| 12|
 1 |  6|  8
-----------
  1|   |
2  |   |  4
   |8  |
-----------
   | 3 |
3  |  4|86
 2 |   |4

Un sudoku di interi (1-9) con 0 caselle nascoste
134|692|578
597|148|263
268|735|491
-----------
649|371|825
381|256|749
725|489|136
-----------
452|863|917
873|914|652
916|527|384

Un sudoku di caratteri (A-I) con 30 caselle nascoste
H I| CE|DFG
  B|HDG|
   |FIA|EHB
-----------
 HG| E |ACD
IB |DFC| E
C  |A  |B F
-----------
  F|CA |H
BCD|EHI|F
 EH| BF|CDI

Un sudoku di emoji [😀, 🤑, 🤔, 🤣, 😛, 😍, 🤫, 😇, 🙃] con 20 caselle nascoste
🤫😇  |😀🤑🤣|😛
😀🤔🤣|🙃   |😇🤑🤫
😍    |🤫🤔😇|🙃😀🤣
--------------------
😛🤫  |🤑😍  |   🙃
😇🤣😍|😛😀🙃|🤑
🤑🙃🤔|😇  🤫|😍😛😀
--------------------
🤔😍  |🤣😇😛|😀🙃🤑
🙃😀😇|😍🤫  |  🤣😛
🤣😛🤑|  🙃😀|🤫

Un sudoku di interi (1-9) con 62 caselle nascoste
Exception in thread "main" java.lang.IllegalArgumentException: empty deve essere tra 0 e 61 inclusi
```