Esame di Programmazione II, 24 febbraio 2023

Si crei un progetto Eclipse e il package `it.univr.quindici`*. 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, costruttori e classi, ma devono essere* `private`*. La soluzione che verrà consegnata dovrà compilare, altrimenti non verrà corretta.*

Il gioco del 15 consiste in una matrice quadrata di 4x4 tessere di cui una sola è vuota e le altre 15 sono riempite con i numeri tra 1 e 15. Il gioco è risolto se la casella vuota è in basso a destra e le tessere sono ordinate in senso crescente secondo una lettura per righe, cioè se la matrice è nella configurazione:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Si intende adesso generalizzare questo gioco a una matrice di dimensioni generiche *width* × *height* (*width* righe e *height* colonne). Le tessere non sono più solo numeriche, ma possono essere degli oggetti di tipo `T` con una relazione di ordinamento fra di loro. Ad esempio, un gioco 4×3 con tessere alfabetiche di lunghezza fra 1 e 5 è il seguente:

```
 swhv    kt     g  wohp
sqvtc      nzwuo   evs
  hkf    qb    lt    me
```

Si noti che tale gioco non è risolto poiché le stringhe non sono in ordine alfabetico secondo una lettura per righe, né la casella vuota è in basso a destra.

Esercizio 1 (7 punti)

(si consegni `Tessera.java`)

Una tessera è rappresentata dalla classe generica `Tessera⟨T⟩`, dove `T` è il tipo del valore contenuto nella tessera (una stringa, un intero, o altro tipo comparabile). La classe, per come la trovate già scritta, vincola `T` a essere comparabile con se stesso, in modo che le tessere si possano mettere in ordine rispetto al valore contenuto:

```
class Tessera<T extends Comparable<T>> implements Comparable<Tessera<T>>
```

Si completi la classe `Tessera⟨T⟩` (metodi `hashCode`, `toString` e `compareTo`).

Esercizio 2 (5 punti)

(si consegni `FattoriaDiTessereNumeriche.java`)

La classe astratta e generica `FattoriaDiTessere⟨T⟩` rappresenta oggetti che generano tessere di tipo `Tessera⟨T⟩`, casuali, ogni volta che si chiama il metodo `get` della fattoria. Tale classe (già completa) estende la classe di libreria `java.util.function.Supplier`. Si definisca una sottoclasse `FattoriaDiTessereNumeriche` di `FattoriaDiTessere⟨Integer⟩` che definisce `get` in modo che a ogni chiamata si ottenga una `Tessera⟨Integer⟩` casuale, con valore numerico contenuto tra 1 e `max` inclusi, dove `max` viene specificato al costruttore di `FattoriaDiTessereNumeriche` (si veda il `Main.java` per un esempio di utilizzo).

Esercizio 3 (6 punti)

(si consegni `FattoriaDiTessereAlfabetiche.java`)

Si definisca una sottoclasse `FattoriaDiTessereAlfabetiche` di `FattoriaDiTessere⟨String⟩` che definisce `get` in modo che a ogni chiamata si ottenga una `Tessera⟨String⟩` casuale, con valore stringa fatto da 1 a 5 lettere alfabetiche minuscole (si veda il `Main.java` per un esempio di utilizzo).

Esercizio 4 (13 punti)

(si consegni `Gioco.java`)

Si completi la classe `Gioco.java` che implementa un gioco di dimensione `width` x `height` (`width` colonne, `height` righe). Il suo costruttore (che dovete completare) crea il gioco casualmente (tessere casuali *tutte distinte*, casella vuota posizionata casualmente). Il costruttore sa il modo per costruire tessere casuali poiché gli viene passata una fattoria:

```
public Gioco(FattoriaDiTessere<T> fattoria, int width, int height)
```

e può quindi chiamarne il metodo `get` ripetutamente per ottenere tessere casuali (dovrà però in qualche modo garantire che risultino tutte distinte). Inoltre dovrete completare il metodo `risolto` che determina se il gioco è risolto (casella vuota in basso a destra, tessere ordinate crescenti in una lettura per righe).

Se tutto è corretto, l'esecuzione del `Main.java` (già fatto, da non modificare) stamperà qualcosa del tipo:

```
 ttno         lzp    qy
    h   zte    ta jigbz
  zve     k xhjmf    at
galtp    mj    yo    pd

   1     8     3
         5     2

         8     4
   1     2     7

   // molti tentativi qui omessi per brevità, fino a un gioco risolto (sotto)

   1     2     3
   5     7
```

Come si vede dalla stampa (e dal codice), il `Main.java` prima genera un gioco casuale 4x4 con tessere alfabetiche, poi genera dei giochi casuali 3x2 con tessere numeriche tra 1 e 8, finché non ottiene un gioco risolto e a quel punto termina.