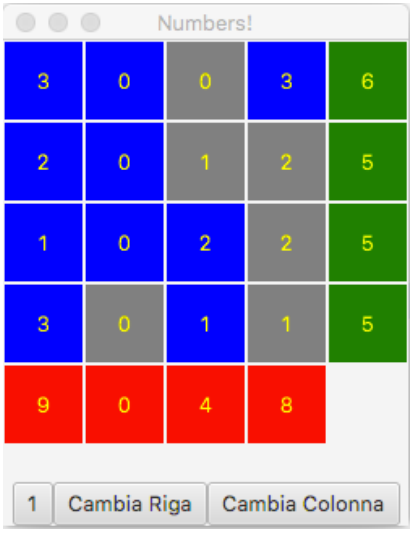
Compito di
Linguaggi di Programmazione – Programmazione 2 – prof. Picco
Linguaggi di Programmazione (Mod.1) - prof. Ronchetti

Prova al calcolatore – 5 febbraio 2020

1) E richiesto il class diagram UML. È sufficiente la vista di alto livello (senza attributi/metodi) ma devono essere mostrate le associazioni più importanti, oltre a quella di ereditarietà. Il diagramma UML deve essere consegnato su un foglio protocollo indicando nome, cognome, numero di matricola. Si prepari su un foglio il diagramma UML di tutte le classi coinvolte nel progetto sotto specificato. Non serve mostrare le classi di JavaFX, a parte quelle che hanno un ruolo diretto nel programma, e quelle da cui le classi scritte dal programmatore ereditano direttamente, se ve ne sono. **Al termine, il foglio (sul quale devono essere presenti nome, cognome, numero di matricola) dovrà essere consegnato anche se in bianco.**

2) Si costruisca un'interfaccia che presenta una scacchiera 5x5 e tre bottoni come da figura. La finestra avrà dimensione 250x300 pixel.



3) Il primo bottone contiene la scritta "x", dove x è un numero compreso tra 1 e 4 (inizialmente 1). Quando viene premuto, il suo valore viene incrementato ciclicamente (dopo 4 torna a 1).

4) La pressione dei tasti "1", "2", "3","4" sulla tastiera settano x al valore del tasto premuto e lo mostrano nella label del bottone.

5) Nella colonna di destra della scacchiera si trovano celle verdi, nella riga in basso si trovano celle rosse; fa eccezione la cella in basso a destra che non è occupata. Nelle restanti caselle della scacchiera può trovarsi una cella grigia o una cella blu. Tutte le celle contengono un numero.

6) Inizialmente le celle grigie e blu vengono create e collocate sulla scacchiera in modo casuale. Ciascuna contiene un numero compreso tra 0 e 3 (estremi inclusi) assegnato anch'esso in modo casuale.

7) I valori delle celle rosse vengono calcolati come somma dei valori presenti nel resto della colonna a cui esse appartengono, sommando quindi i valori delle corrispondenti celle blu e grigie.

8) I valori delle celle verdi vengono calcolati come somma dei valori presenti nel resto della riga a cui esse appartengono, analogamente al punto precedente.

9) Il bottone "Cambia riga" modifica le celle grigie e blu della riga x, dove x è il valore individuato dal primo bottone, dopodiché aggiorna i valori di tutte le celle verdi e rosse.

10) Il bottone "Cambia colonna" modifica le celle grigie e blu della colonna x, dove x è il valore individuato dal primo bottone, dopodiché aggiorna i valori di tutte le celle verdi e rosse.

11) Le operazioni di modifica delle celle blu e grigie menzionate ai punti 9 e 10 agiscono nel seguente modo:
    - celle blu: se ne decrementa il valore di 1 – se diventa minore di 0 torna a 3.
    - celle grigie: se ne incrementa il valore di 1 – se diventa maggiore 3 torna a 0;

12) Se (almeno) una cella verde assume il valore 10, viene visualizzato il messaggio "Hai vinto".

13) Se (almeno) una cella rossa assume il valore 10, viene visualizzato il messaggio "Hai perso".

14) Qualora simultaneamente si abbia (almeno) un 10 sia in una cella verde che in una cella rossa, viene visualizzato il messaggio "Parità".

15) I messaggi dei punti 12, 13, 14 devono essere visualizzati tramite pop-up; in alternativa, è possibile visualizzarli in console, con una piccola penalizzazione in termini di punteggio.

16) Contestualmente all'esecuzione dei punti 12, 13 e 14 vengono disabilitati i bottoni "Cambia riga" e "Cambia colonna".

17) Si documenti il codice prodotto con Javadoc (solo studenti del prof. Ronchetti).

Altri requisiti

- Il codice generato deve rispettare i principi della programmazione object-oriented. In particolare, si usi, quando evidente/utile, una gerarchia di ereditarietà, sfruttando ove possibile il polimorfismo.
- Si usino costanti ove ragionevole, e si badi alla pulizia del codice (es., linee guida Java) evitando duplicazioni e codice inutilmente complesso.