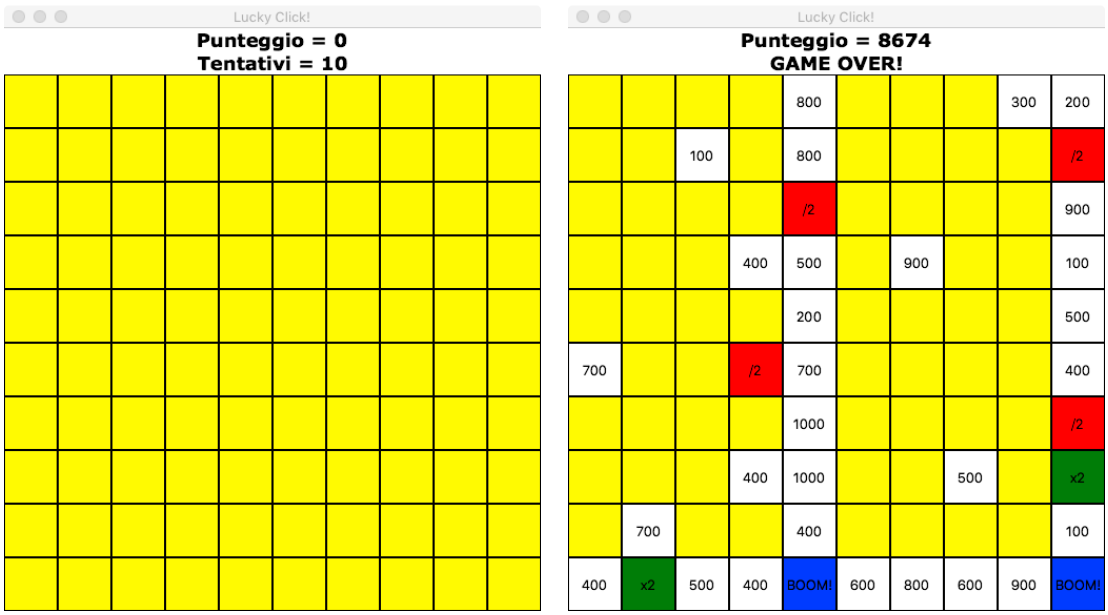
Esame di Linguaggi di Programmazione (prof. Picco) – 15 giugno 2018
Prova al calcolatore

Si vuole costruire il videogioco "Lucky Click!" in cui l'utente colleziona punti cliccando su una griglia le cui celle sono inizialmente coperte. Il click "scopre" il contenuto della cella, in base al quale il giocatore guadagna o perde punti. Il gioco termina dopo che il giocatore ha effettuato un numero predefinito di tentativi a sua disposizione.

L'area di gioco è composta da:

- una parte superiore in cui sono visualizzati due elementi testuali contenenti il punteggio corrente e il numero di tentativi
- una griglia 10x10 di celle quadrate di identica dimensione (lato di 50 pixel)

Di seguito viene visualizzata la schermata iniziale (a sinistra) e la schermata finale di una partita (a destra).



Le celle possono essere di vari tipi:

- Cella *base*: contiene un valore casuale multiplo di 100, compreso fra 100 e 1000, su sfondo bianco. Il valore della cella viene sommato al punteggio corrente.
- Cella *moltiplicatore*: contiene la scritta "x2" su sfondo verde. Raddoppia il punteggio attuale.
- Cella *divisore*: contiene la scritta "/2" su sfondo rosso. Dimezza il punteggio attuale.
- Cella *bomba*: contiene la scritta "BOOM!" su sfondo blu. Il suo effetto è quello di "scoprire" tutte le celle ancora coperte che si trovano sulla stessa riga e colonna, generandone gli effetti conseguenti. Ad esempio, nella figura a destra si nota come l'effetto della cella bomba al centro sia stato di scoprire le celle base, moltiplicatore, e divisore sulla riga e colonna (modificando il punteggio di conseguenza per ciascuna) ma anche di

scoprire la bomba nell'angolo a destra, che a sua volta ha scoperto la propria colonna.
- All'interno della griglia vi sono 10 celle moltiplicatore, 10 celle divisore, e 5 celle bomba, tutte disposte in maniera casuale all'interno della griglia. Le rimanenti 75 celle sono tutte celle base.

Inizialmente, tutte le celle sono "coperte": hanno un colore giallo e nessuna scritta al loro interno. Quando l'utente clicca su una cella coperta:
- la cella cambia aspetto e l'azione corrispondente viene eseguita, secondo quanto descritto sopra per ogni tipologia di cella;
- il punteggio corrente, modificato dalla cella scoperta, viene aggiornato nell'area superiore della finestra;
- il numero di tentativi viene decrementato di una unità e aggiornato nell'area superiore della finestra.

Al contrario, un click su una cella già scoperta non ha alcun effetto.

Il gioco termina quando il giocatore ha esaurito i dispositivi a disposizione:
- viene visualizzata la scritta "GAME OVER!" al posto dei tentativi a disposizione;
- l'applicazione non reagisce più ai click del mouse, ma rimane attiva.

Requisiti:
- Si usi una gerarchia di ereditarietà per rappresentare i diversi tipi di celle.
- È richiesto il diagramma UML delle classi sviluppate. Questo deve essere consegnato su un foglio di protocollo su cui sono indicati nome, cognome, numero di matricola.
- È possibile ottenere il massimo dei punti anche senza implementare l'intera specifica.

Suggerimenti:
- Può essere utile definire una classe astratta per le celle. Può essere utile includere in tale classe i corrispondenti elementi dell'interfaccia grafica. Attenzione alla definizione dei costruttori.
- Layout: la finestra principale si presta a essere implementata combinando un layout verticale e uno a griglia. Per le celle, si usi uno **`StackPane`** che combina un rettangolo e il testo necessario, come visto negli esempi.
- **`GridPane`** fornisce i metodi statici **`getRowIndex`** e **`getColumnIndex`** che ritornano la posizione dell'oggetto fornito come parametro. Quindi, passando l'oggetto su cui si è generato un evento (ottenuto con il metodo **`getSource`** sull'evento) si può ottenerne la posizione nella griglia.
- Può convenire lasciare l'implementazione della cella bomba per ultima, in quanto un più complessa.