ESAME DI FONDAMENTI DI INFORMATICA T-2 del 4/2/2021
Proff. E. Denti – R. Calegari – A. Molesini
Tempo a disposizione: 3 ore

NOME PROGETTO ECLIPSE: CognomeNome-matricola (es. RossiMario-0000123456)
NOME CARTELLA PROGETTO: CognomeNome-matricola (es. RossiMario-0000123456)
NOME ZIP DA CONSEGNARE: CognomeNome-matricola.zip (es. RossiMario-0000123456.zip)
NOME JAR DA CONSEGNARE: CognomeNome-matricola.jar (es. RossiMario-0000123456.jar)

Si devono consegnare DUE FILE: *l'intero progetto Eclipse* e *il JAR eseguibile*

Si ricorda che compiti *non compilabili* o *palesemente lontani da 18/30* NON SARANNO CORRETTI e causeranno la verbalizzazione del giudizio "RESPINTO"

Si vuole sviluppare un'app per simulare il gioco della Ghigliottina del noto show preserale "L'Eredità"

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Scopo del gioco è indovinare una parola a partire da altre parole (*indizi*, tipicamente 5), che si collegano ad essa per analogie, modi di dire, etc. Tali indizi però devono essere conquistati: a ogni manche, al concorrente vengono proposte due parole, di cui una sola è l'indizio giusto. Appena il concorrente fa la sua scelta, scatta la Ghigliottina: se la parola scelta è l'indizio corretto, il montepremi rimane intoccato, altrimenti viene dimezzato. L'indizio corretto viene poi comunque mostrato e lasciato visibile. A seconda delle scelte (e della fortuna) del concorrente si può quindi dimezzare da 0 a 5 volte.

Una volta accumulati i 5 indizi, il concorrente ha un minuto di tempo per pensare alla parola che li accomuna tutti: allo scadere del tempo deve scriverla su un cartoncino. Successivamente si procede alla verifica: se la parola scritta risulta la risposta esatta, il concorrente vince il montepremi; altrimenti, non vince nulla.

ESEMPIO

Si supponga che vengano via via proposte al concorrente le seguenti coppie di parole:

1. Lordo / Lardo (indizio corretto: Lardo)
2. Zampino / Zampone (indizio corretto: Zampino)
3. Tetto / Veranda (indizio corretto: Tetto)
4. Lenta / Frettolosa (indizio corretto: Frettolosa)
5. Gino Paoli / Mina (indizio corretto: Gino Paoli)

A seconda delle scelte che via via effettua, il concorrente può dimezzare zero o più volte il montepremi iniziale (si supponga siano € 100.000), giocando quindi per una cifra compresa fra €100.000 e € 3.125.

Allo scadere del tempo, il concorrente scrive la parola che ritiene accomuni i cinque indizi: se, alla successiva verifica, essa risulta quella corretta, il concorrente vince il montepremi; altrimenti non vince nulla.

In questo caso, la risposta esatta è "Gatta", poiché:

1. Lardo → proverbio: "Tanto va la gatta al lardo che ci lascia lo zampino"
2. Zampino → proverbio: "Tanto va la gatta al lardo che ci lascia lo zampino"
3. Tetto → titolo di film e opera teatrale: "La gatta sul tetto che scotta"
4. Frettolosa → proverbio: "La gatta frettolosa fa i gattini ciechi"
5. Gino Paoli → titolo di canzone: "La gatta"

Nell'applicazione da sviluppare, per semplicità, non vi sarà alcun timer a scandire lo scorrere del tempo dopo la scelta degli indizi: il giocatore avrà tutto il tempo che vuole per pensare e scrivere la sua risposta. Si potrà poi verificare il risultato premendo un apposito pulsante ("SVELA"): un dialogo informerà del risultato (vedere figure in coda al testo).

NB: le animazioni sono fornite già realizzate nell'apposito pannello *GhigliottinaPanel*.

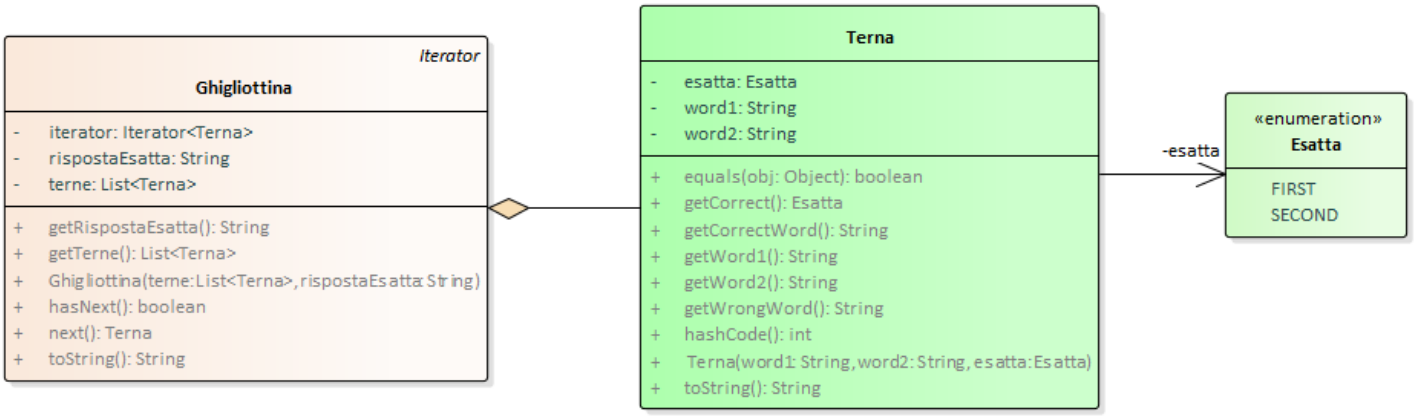
Cose da ricordare

- salva costantemente il tuo lavoro: l'informatica a volte può essere "subdolamente ostile"..
- in particolare: se ora compila e stai per fare modifiche, salva la versione attuale (non si sa mai)

Parte 1 (punti: 18)

Dati (namespace ghigliottina.model) *(punti: 5)*

Il modello dei dati deve essere organizzato secondo il diagramma UML di seguito riportato:



SEMANTICA:

a) L'enumerativo ***Esatta*** (fornito) definisce i due valori FIRST e SECOND per denotare quale delle due parole proposte sia l'indizio corretto

b) La classe ***Terna*** (fornita) modella una proposta di due parole come possibile indizio, con indicazione della relativa risposta esatta. Ad esempio, con riferimento all'esempio sopra, l'alternativa fra "Lordo" e "Lardo", in cui l'indizio corretto è la seconda parola ("Lardo"), sarà rappresentato dalla terna (*"Lordo", "Lardo", SECOND).* Il costruttore riceve le due parole e l'indicazione di quale sia l'indizio corretto (tramite uno dei due valori dell'enumerativo); appositi accessor consentono di recuperare le due parole sia in base all'ordine con cui compaiono nella terna (***getFirst***/***getSecond***), sia in base alla loro esattezza (***getCorrectWord***/***getWrongWord***); è anche possibile recuperare l'indicazione di quale sia quella esatta come valore dell'enumerativo (***getCorrect***). Completano la classe alcune implementazioni ovvie di ***toString***, ***equals*** e ***hashcode***.

c) La classe **Ghigliottina** (da realizzare) rappresenta il gioco e pertanto incapsula:

- una lista di ***Terna*** (tipicamente saranno 5, ma occorre essere generali) con le proposte di indizi
- la risposta esatta (una stringa)

Il costruttore deve controllare accuratamente la qualità degli argomenti ricevuti, in particolare che essi non siano null né vuoti (nel caso della risposta esatta, è richiesto che non sia *blank* ossia composta unicamente di spazi, tabulazioni, etc.): in tal caso dovrà essere lanciata apposita ***IllegalArgumentException***, con idoneo messaggio.

Non è invece richiesta alcuna particolare implementazione per ***toString***, ***equals*** e ***hashcode***.

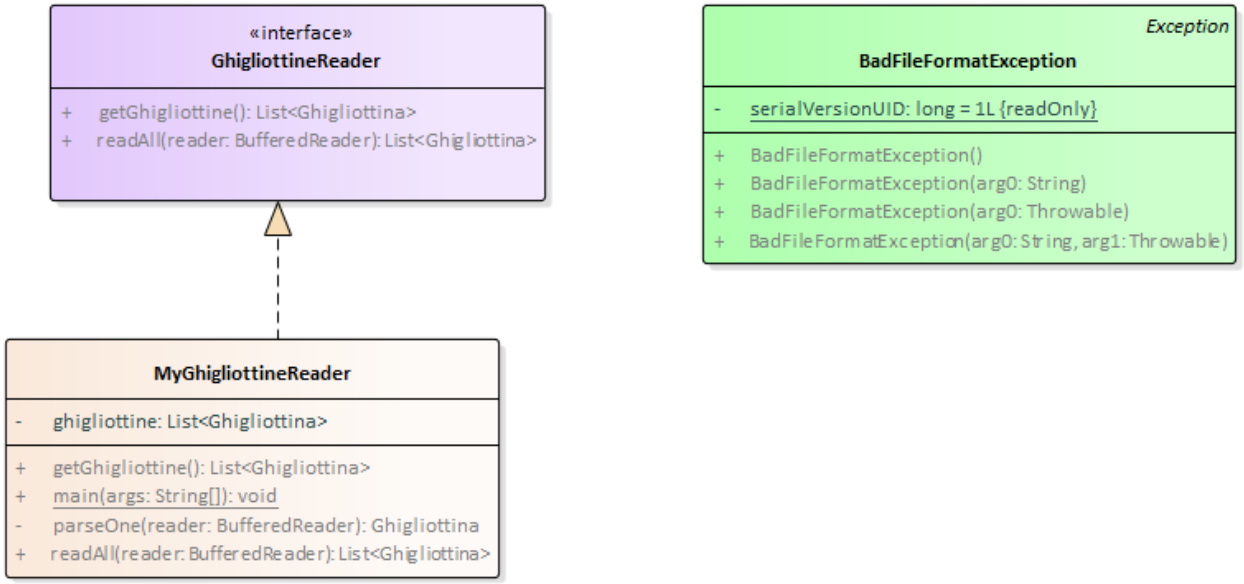
Persistenza (namespace ghigliottina.persistence) *(punti: 13)*

Questo package definisce il reader per leggere da file le possibili ghigliottine.

SEMANTICA:

a) l'interfaccia ***GhigliottineReader*** (fornita) dichiara i due metodi ***readAll*** e ***getGhigliottine***: il primo legge da un (buffered)reader una serie di ***Ghigliottina*** e le restituisce sotto forma di lista; il secondo restituisce semplicemente la lista già letta (o lancia ***NullPointerException*** se la lettura non è ancora avvenuta)

b) la classe **MyGhigliottineReader** (da realizzare) implementa tale interfaccia: si suggerisce di strutturare la lettura su due metodi, appoggiandosi a un metodo privato ausiliario ***readOne*** per la lettura di una singola ***Ghigliottina***, così da separare meglio la fase di lettura e verifica della singola ghigliottina da quella, più esterna, di lettura della sequenza di ghigliottine. A questo fine, è opportuno che il metodo privato ausiliario ***readOne*** restituisca *null* quando arriva a fine file, così da permettere un'articolazione standard del ciclo di lettura esterno in ***readAll.***



Il metodo ***readAll*** riceve un ***Reader*** già aperto: restituisce la lista, *eventualmente vuota ma mai nulla*, delle ghigliottine lette dal file. In caso di problemi di I/O dev'essere lasciata uscire ***IOException***, mentre in caso di problemi nel formato delle righe si deve lanciare ***BadFileFormatException*** (fornita) con preciso e specifico messaggio d'errore.

FORMATO DEL FILE: il file è strutturato a blocchi, ognuno dei quali descrive una ghigliottina. Ogni blocco consiste innanzitutto di un certo numero di righe che descrivono terne: a seguire vi è la riga con la risposta esatta, indi per ultima una riga composta di lineette (almeno 6), che chiude il blocco.

Le righe che descrivono terne hanno la forma *PAROLA1***/***PAROLA2***=***QUALEDELLEDUE*, dove *PAROLA1* e *PAROLA2* sono sequenze di caratteri qualsiasi diversi da '/' e '=', mentre *QUALEDELLEDUE* è "**FIRST**" o "**SECOND**". Intorno alle parole, o prima o dopo di esse, possono esservi spazi o tabulazioni in numero qualunque.

ESEMPI DI FILE LECITI (il rombo, *solo per motivi di visibilità in questa sede*, indica la tabulazione):

```
LEGGE/DECRETO=FIRST
PATATA/CAROTA=FIRST
SOPRA/SOTTO=SECOND
FATICA/SACRIFICIO=SECOND
FANTASMA/ASMA=FIRST
Risposta esatta=SPIRITO
-----------------------
CANTATO/INCANTATO=SECOND
...
```

```
LEGGE/DECRETO = FIRST
PATATA /  CAROTA=FIRST
SOPRA◆/SOTTO=SECOND
FATICA/SACRIFICIO◆=◆SECOND◆
  FANTASMA/ASMA=FIRST
Risposta esatta=SPIRITO
-----------------------
CANTATO/INCANTATO=SECOND
...
```

Parte 2 (punti: 12)

Controller (namespace ghigliottina.ui) *(punti: 3)*

a) l'interfaccia ***Controller*** (fornita – UML mostrato insieme a quello della UI) dichiara i due metodi ***listaGhigliottine*** e ***sorteggiaGhigliottina***: il primo restituisce la lista di ***Ghigliottina*** (che si suppone ricevuta dal costruttore della classe che la implementa), il secondo sorteggia e restituisce una singola ***Ghigliottina*** a caso fra quelle disponibili

b) la classe **MyController** (da realizzare) implementa tale interfaccia.

GUI (namespace ghigliottina.ui) *(punti: 9)*

L'interfaccia grafica è costituita da due pannelli annidati: all'interno, un ***GhigliottinaPanel*** (fornito) incorpora e gestisce tutta la grafica del gioco; all'esterno, un ***OuterGhigliottinaPanel*** (da completare) gestisce i campi di testo in cui il concorrente scrive la risposta e quello, adiacente, in cui – alla pressione del pulsante SVELA – comparirà la risposta corretta. L'applicazione nel suo complesso è attivata da ***GhigliottinaApp*** (fornita), che provvede anche a fornire al controller una ghigliottina prestabilita in caso di malfunzionamento del reader, fungendo anche da mock.

La GUI dev'essere simile (non necessariamente identica) a quella più oltre illustrata (Fig.1): si tratta di un ***BorderPane***, in cui in alto sono presenti due campi di testo, sormontati da apposite label: quello destinato a mostrare la risposta esatta non è editabile. Alla loro destra il pulsante SVELA, quando premuto:

- scatena la verifica della risposta (che dev'essere insensibile alla grafia e quindi a maiuscole/minuscole)
- fa comparire i dialoghi corrispondenti (Figg. 6,7,8)
- infine resetta l'app allo stato iniziale, con una nuova ghigliottina (Fig. 9).

Sotto, invece, è presente un ***GhigliottinaPanel*** (fornito), il cui costruttore si aspetta di ricevere l'ammontare iniziale del montepremi: appositi accessor (***getMontepremi***/***getMontepremiAsString***) consentono di recuperare in qualsiasi momento, e in particolare quindi al termine della ghigliottina, l'ammontare residuo del montepremi, rispettivamente sotto forma di intero e di stringa già opportunamente formattata.

Quando la GUI compare, il gioco è già attivo e viene proposta al concorrente la prima coppia di parole (Fig. 1): il concorrente sceglie quella desiderata utilizzando i radiobutton, poi conferma premendo il pulsante "Ghigliottina". Successivamente il gioco evolve, eventualmente dimezzando via via il montepremi a ogni scelta errata (Figg. 1-3): una volta mostrate tutte le terne (solitamente cinque), il pulsante "Ghigliottina" viene disabilitato e lo sfondo diventa grigio (Fig. 4). A quel punto il concorrente scrive la propria soluzione nel campo di testo apposito (a destra in Fig. 5), indi preme il pulsante SVELA: il dialogo che compare mostra l'esito del gioco (Fig. 6 o, in altra situazione, Fig. 7). Se si preme il pulsante SVELA prima di aver scritto una soluzione (Fig. 8), compare un dialogo di errore.

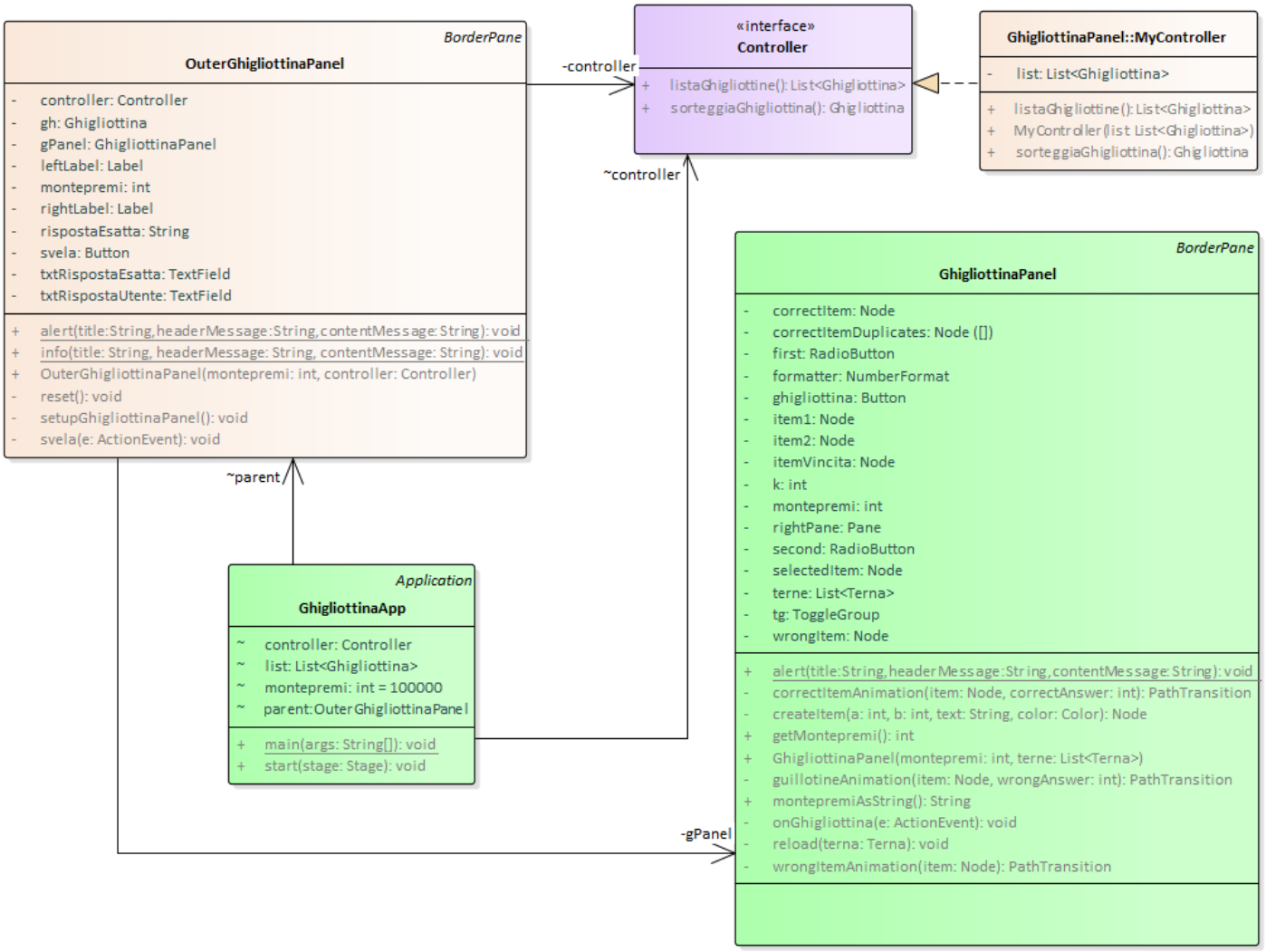
Per far comparire questi dialoghi sono disponibili nella classe ***OuterGhigliottinaPanel*** due metodi statici:

- ***alert***, che mostra consente di una finestra di dialogo utile a segnalare errori all'utente si riconosce per l'icona "X" (Figg. 8).
- ***info***, che mostra consente di una finestra di dialogo utile a mostrare informazioni all'utente (per le risposte): si riconosce per l'icona "i" (Figg. 6,7).

È richiesto di completare opportunamente ***OuterGhigliottinaPanel,*** in particolare:

- predisponendo la struttura del pannello interno superiore, con i campi di testo e il pulsante SVELA;
- gestendo l'evento corrispondente alla pressione di tale pulsante, ovvero:
    - scatenando la verifica della risposta (insensibile alla grafia e quindi a maiuscole/minuscole)
    - facendo comparire i dialoghi corrispondenti (Figg. 6,7,8)
    - infine resettando l'app allo stato iniziale, con una nuova ghigliottina (Fig. 9).

*****SEGUE ALLA PAGINA SUCCESSIVA*****



Cose da ricordare

- salva costantemente il tuo lavoro: l'informatica a volte può essere "subdolamente ostile"..
- in particolare: se ora compila e stai per fare modifiche, salva la versione attuale (non si sa mai)

Checklist di consegna

- Hai fatto un **JAR eseguibile**, che contenga cioè l'indicazione del main?
- Hai controllato che **si compili** e **ci sia tutto**? [NB: non includere il PDF del testo]
- Hai **rinominato** IL PROGETTO, lo ZIP e il JAR esattamente come richiesto?
- Hai **chiamato** la cartella del progetto esattamente come richiesto?
- Hai fatto un **unico file ZIP (NON .7z, rar o altri formati)** contenente l'intero progetto? In particolare, ti sei assicurato di aver incluso tutti i file .java (e non solo i .class)?
- **Hai consegnato DUE file distinti, ossia lo ZIP col progetto e il JAR eseguibile?**
- Su EOL, hai **premuto** il tasto "CONFERMA" per inviare il tuo elaborato?

Figura 1

Figura 2

Figura 3

Figura 4

Figura 5

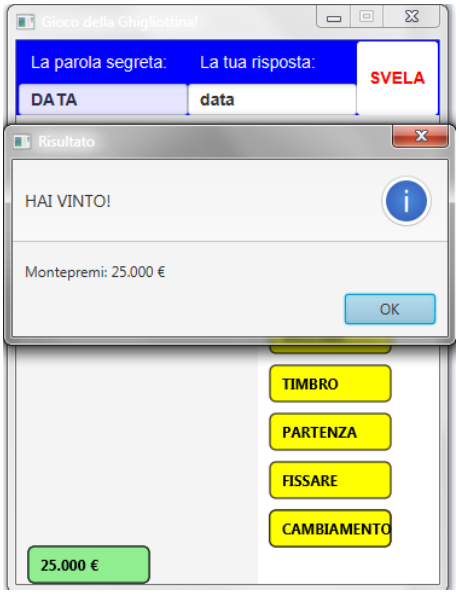


Figura 6

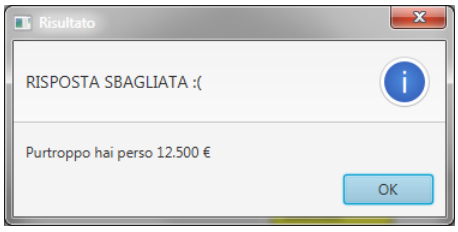


Figura 7

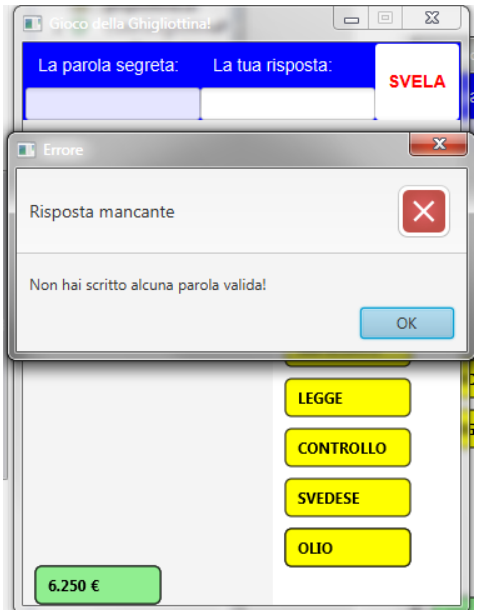


Figura 8

Figura 9