ESAME DI FONDAMENTI DI INFORMATICA T-2 dell'11/9/2024

Proff. E. Denti – R. Calegari – A. Molesini

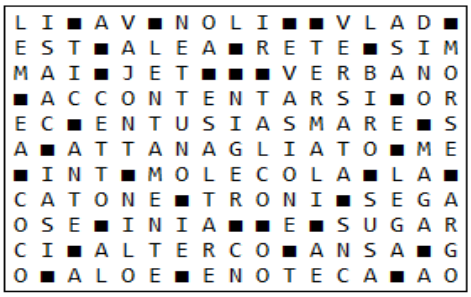
Tempo a disposizione: 3h30

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*

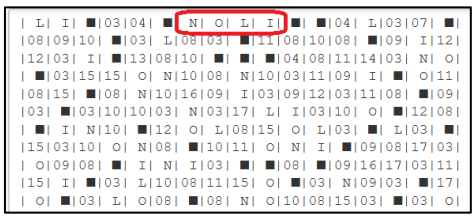
Si ricorda che compiti *non compilabili*, o che *non passino almeno 2/3 dei test* o siano *palesemente lontani da 18/30* NON SARANNO CORRETTI e causeranno la verbalizzazione del giudizio "RESPINTO".

È stato richiesto lo sviluppo di un'applicazione che, a partire da un classico cruciverba già risolto, produca un crittocruciverba (o cruciverba cifrato), ossia quel tipo di cruciverba in cui al solutore non vengono date definizioni per le parole, ma vige il principio secondo cui "a lettera uguale corrisponde numero uguale": il solutore dovrà ricostruire lo schema in base al proprio intuito, partendo dalle poche lettere inizialmente disposte sullo schema.



L'applicazione dovrà generare il crittocruciverba, popolarlo con le lettere iniziali previste e poi mostrarlo a video così da consentire al giocatore-utente di risolverlo interattivamente. In base alla bravura del giocatore, si potrà scegliere fra due modalità, AGEVOLATA ed ESPERTA:

- in modalità agevolata, le lettere inizialmente fornite saranno 4 e l'interfaccia grafica non consentirà al solutore di inserire associazioni lettera/numero errate;
- in modalità esperta, invece, le lettere fornite saranno solo 3 e non vi sarà alcun impedimento a inserire associazioni errate.



DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Nel crittocruciverba le caselle sono numerate secondo il principio "a lettera uguale corrisponde numero uguale". Lo schema si presenta inizialmente vuoto, con le sole caselle nere: le celle vuote sono pre-numerate secondo tale principio. Il solutore deve ricostruire lo schema attribuendo una lettera diversa a ogni numero fino a formare tutte parole di senso compiuto.

Per generare il crittocruciverba si parte da un cruciverba classico già risolto (immagine in alto) e se ne numerano le celle partendo da quella in alto a sinistra e procedendo poi verso destra, dall'alto in basso, secondo il predetto principio: la prima lettera sarà quindi messa in corrispondenza col numero 1, la seconda (se diversa dalla precedente) col numero 2, e così via. Naturalmente, una volta che si stabilisce una corrispondenza lettera-numero, tutte le caselle che contengono la stessa lettera saranno etichettate con lo stesso numero. Per ipotesi, le caselle nere non sono numerate. Nel caso dell'esempio sopra, ciò produce la situazione illustrata nell'immagine intermedia.

Per proporre lo schema al giocatore, tuttavia, è necessario disporre preliminarmente alcune lettere sullo schema: a tal fine viene scelta casualmente una parola corta, composta rispettivamente di 3 (modalità esperta) o 4 (modalità agevolata) lettere distinte, che vengono poi riprodotte ovunque sullo schema (nell'esempio sopra, in modalità agevolata, è stata sorteggiata la parola "NOLI", quindi l'intero schema è stato pre-popolato con le associazioni previste per le lettere 'L' = 1, 'I' = 2, 'N' = 5, 'O' = 6).

Ovviamente, non è detto che uno schema usi tutte le 26 lettere dell'alfabeto: i numeri utilizzati saranno quindi una sequenza di valori continui da 1 fino al numero di lettere distinte presenti nello schema (nell'esempio sopra 17, non essendo presenti le nove lettere F,H,K,P,Q,W,X,Y,Z).

Il file di testo schema.txt contiene, nel formato descritto più oltre, il cruciverba di partenza risolto. L'applicazione dovrà:

a) leggere tale file e caricare il cruciverba risolto
b) utilizzarlo per generare lo schema numerato del crittocruciverba
c) permettere al giocatore-solutore, tramite la GUI, di risolvere per tentativi lo schema inserendo via via le associazioni lettera/numero e mostrando l'evoluzione del gioco. In modalità AGEVOLATA, la GUI deve altresì prevenire l'inserimento di associazioni errate.

TEMPO STIMATO PER SVOLGERE L'INTERO COMPITO: 2h15 – 3h

PARTE 1 – Modello dei dati: Punti 7 [TEMPO STIMATO: 30-40 minuti]

PARTE 2 – Persistenza: Punti 8 [TEMPO STIMATO: 30-45 minuti]

PARTE 3 – Grafica: Punti 15 [TEMPO STIMATO: 75-95 minuti]

NUMERO MINIMO DI TEST CON SUCCESSO PERCHÉ IL COMPITO SIA CORRETTO

Considerando solo SchemaNumerato , MyReader e MyController 18 su 27

Considerandoli tutti: 26 su 38

JAVAFX – Parametri run configuration nei LAB

```
--module-path "C:\applicativi\moduli\javafx-sdk-21.0.2\lib"
--add-modules javafx.controls
```

Cose da ricordare

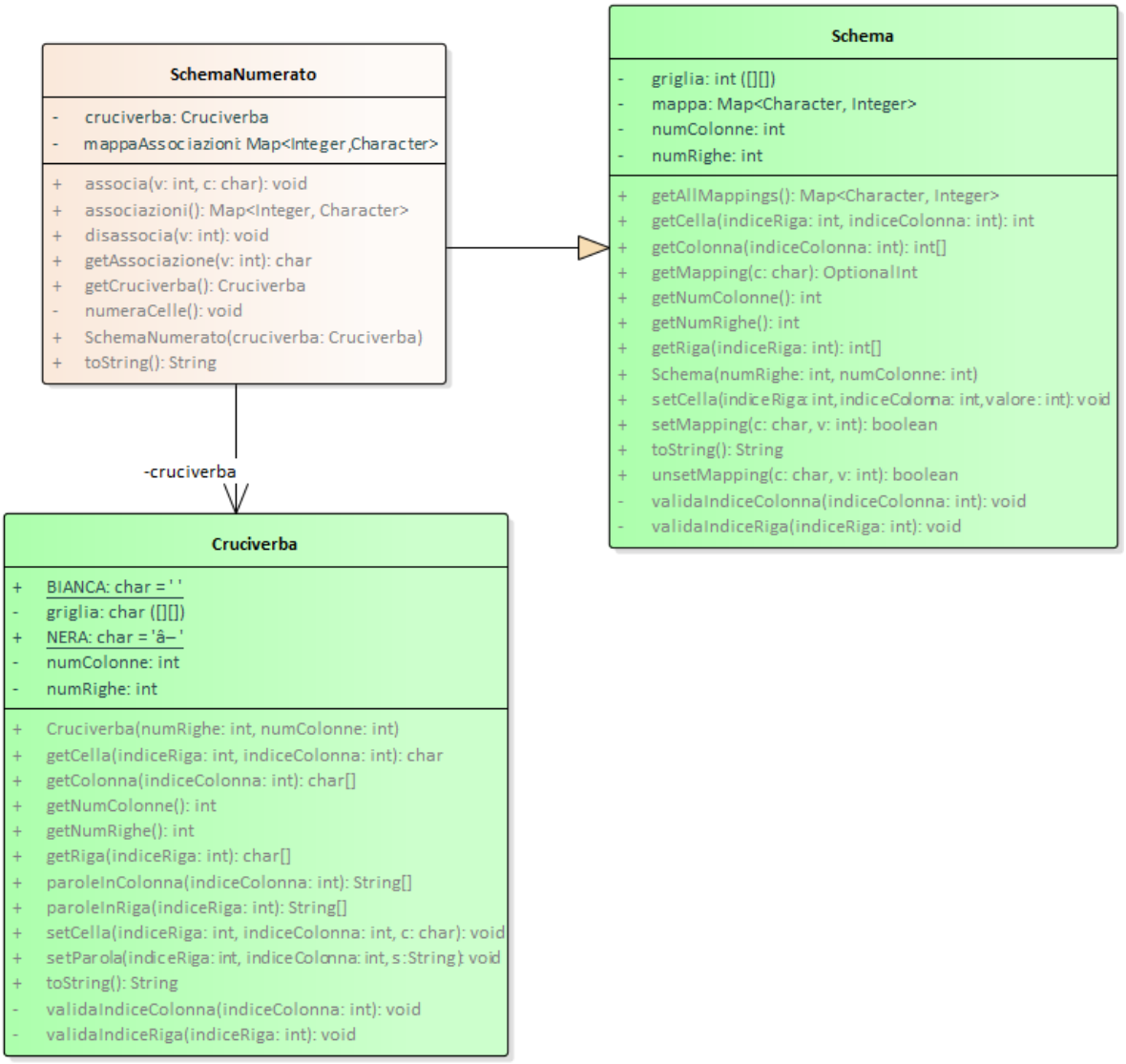
- salva costantemente il tuo lavoro: l'informatica a volte può essere "subdolamente ostile"..
- in particolare: se ora compila e stai per fare modifiche, salva la versione attuale (non si sa mai)

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?

Parte 1 – Modello dei dati (punti: 7)

Package: cruciverba.model **[TEMPO STIMATO: 30-40 minuti]**



SEMANTICA:

- La classe ***Cruciverba*** (fornita) rappresenta un cruciverba classico:
  - Sono definite le due costanti **BIANCA** e **NERA**, atte a rappresentare rispettivamente la casella vuota (non ancora numerata) o nera (da non numerare)
  - Il costruttore alloca la griglia di caratteri e la popola inizialmente con tutte caselle nere, verificando preventivamente che il numero di righe e colonne fornito sia positivo (altrimenti viene lanciata un'apposita ***IllegalArgumentException*** con idoneo messaggio d'errore); il numero di righe e colonne è recuperabile tramite l'apposita coppia di accessor *getNumRighe/getNumColonne*
  - La coppia di metodi *getCella/setCella* permette di recuperare/impostare il carattere nella cella di coordinate (riga,colonna) specificate: entrambi lanciano ***IllegalArgumentException*** con idoneo messaggio d'errore nel caso di indici fuori range
  - Il metodo di utilità *setParola* permette di impostare rapidamente una singola parola orizzontale in una serie di celle adiacenti, posto che naturalmente non si ecceda la dimensione prevista; in tutti i casi problematici viene lanciata ***IllegalArgumentException*** con idoneo messaggio d'errore

- La coppia di metodi *paroleInRiga/paroleInColonna* restituisce l'elenco, sotto forma di array di stringhe, di parole presenti nella riga o colonna specificata; anche in questo caso naturalmente viene lanciata ***IllegalArgumentException*** con idoneo messaggio d'errore nel caso di indici fuori range
- un'apposita *toString* emette una rappresentazione stampabile del cruciverba.

- La classe ***Schema*** (fornita) rappresenta invece lo schema base di un crittocruciverba: mantiene al suo interno una griglia di interi e una tabella in cui registra, per ogni carattere, il numero ad esso attribuito. *Nel seguito ci si riferirà a questa corrispondenza come "mapping".* Tale tabella riflette semplicemente il cruciverba iniziale: non ha alcuna relazione con le azioni del solutore, che saranno gestite altrove.
  - Il costruttore alloca la griglia di numeri di dimensioni date (due valori strettamente positivi), recuperabili tramite l'apposita coppia di accessor *getNumRighe/getNumColonne*; predispone inoltre la tabella di mapping caratteri/numeri, popolandola inizialmente con l'unica entry della casella **NERA**, a cui viene fatto corrispondere convenzionalmente il numero 0 (le altre caselle sono numerate a partire da 1)
  - La coppia di metodi *getCella/setCella* permette di recuperare/impostare il numero nella cella di coordinate (riga,colonna) specificate: entrambi lanciano ***IllegalArgumentException*** con idoneo messaggio d'errore nel caso di indici fuori range
  - La coppia di metodi *getRiga/getColonna* recupera l'intera riga (o colonna) di indice specificato: ovviamente lanciano ***IllegalArgumentException*** nel caso di indici fuori range
  - La terna di metodi *setMapping/unsetMapping/getMapping* gestisce i mapping carattere/numero, rispettivamente impostando (*setMapping*), rimuovendo (*unsetMapping*) o recuperando (*getMapping*) un mapping carattere/numero; sono accettabili solo caratteri maiuscoli e valori numerici compresi fra 1 e 26: diversamente verrà lanciata lanciano ***IllegalArgumentException*** con idoneo messaggio d'errore
  - un'apposita *toString* emette una rappresentazione stampabile dello schema.

- La classe ***SchemaNumerato*** (da completare) specializza **Schema** incorporando la logica di numerazione delle caselle e gestendo le associazioni intero/carattere inserite dall'utente durante il gioco. Da notare che, mentre i "mapping" lettera/numero della classe precedente sono pre-calcolati all'atto della generazione del crittocruciverba perché riflettono semplicemente la numerazione attribuita ai caratteri, queste "associazioni" numero/lettera riflettono invece le "mosse" del giocatore che cerca di risolvere lo schema e verranno quindi aggiunte/tolte dinamicamente durante il gioco. Più precisamente:
  - il costruttore riceve un **Cruciverba**, che usa per istanziare uno **Schema** di uguali dimensioni, e predispone la mappa in cui saranno poi registrate le associazioni numero/lettera durante il gioco; infine numera le caselle, tramite il metodo privato ausiliario *numeraCelle* (fornito), che incorpora la logica di numerazione descritta nel Dominio del Problema.
  - La quaterna di metodi *associa/disassocia/getAssociazione/associazioni* **(da realizzare)** gestisce le associazioni numero/lettera, rispettivamente impostando (*associa*), rimuovendo (*disassocia*) o recuperando (*getAssociazione*) una specifica associazione numero/lettera; se l'associazione non è presente, per convenzione *getAssociazione* deve restituire la costante BIANCA. Il metodo *associazioni* restituisce l'intera mappa con tutte le associazioni attualmente in essere: pertanto, essa *non deve contenere* le associazioni ancora mancanti (neppure a livello di chiavi). Come già per **Schema**, sono accettabili solo valori numerici compresi fra 1 e 26 e, nel caso di *associa*, il carattere deve essere maiuscolo: altrimenti verrà lanciata ***IllegalArgumentException*** con idoneo messaggio d'errore
  - un'apposita *toString* emette una rappresentazione stampabile dello schema numerato.

Parte 2 – Persistenza (punti: 8)

Package: cruciverba.persistence **[TEMPO STIMATO: 30-45 minuti]**

Il file di testo schema.txt descrive il cruciverba classico, nel formato sotto descritto e illustrato nell'esempio in calce:

- la prima riga contiene l'intestazione "Cruciverba ***n*** x ***m***", dove ***n*** ed ***m*** sono interi positivi (***n*** = numero di righe, ***m*** = numero di colonne = lunghezza della riga) e le parole "Cruciverba" e "x" sono scritte esattamente così; i vari termini sono separati fra loro da *uno o più spazi*
- tutte le altre ***n*** righe hanno *eguale struttura* e devono contenere lo *stesso numero **m** di elementi:* ogni riga contiene una sequenza di caratteri singoli (per le caselle vuote, il carattere spazio) separati dal carattere speciale '|', usato anche come marca di inizio e fine riga.

```
Cruciverba 11 x 17
|L|I| |A|V| |N|O|L|I| | |V|L|A|D| |
|E|S|T| |A|L|E|A| |R|E|T|E| |S|I|M|
|M|A|I| |J|E|T| | | |V|E|R|B|A|N|O|
| |A|C|C|O|N|T|E|N|T|A|R|S|I| |O|R|
|E|C| |E|N|T|U|S|I|A|S|M|A|R|E| |S|
|A| |A|T|T|A|N|A|G|L|I|A|T|O| |M|E|
| |I|N|T| |M|O|L|E|C|O|L|A| |L|A| |
|C|A|T|O|N|E| |T|R|O|N|I| |S|E|G|A|
|O|S|E| |I|N|I|A| | |E| |S|U|G|A|R|
|C|I| |A|L|T|E|R|C|O| |A|N|S|A| |G|
|O| |A|L|O|E| |E|N|O|T|E|C|A| |A|O|
```

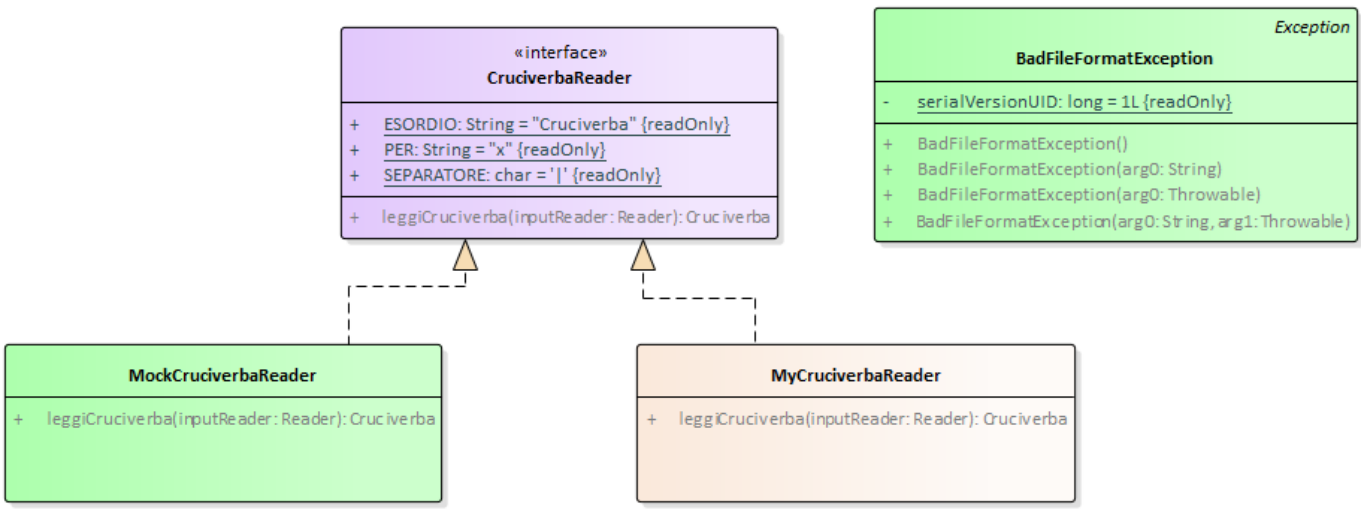
SEMANTICA:

a) L'interfaccia ***CruciverbaReader*** (fornita) dichiara:

- le tre costanti SEPARATORE, ESORDIO e PER che fattorizzano rispettivamente le costanti '|' (un carattere singolo), "Cruciverba" e "x" (stringhe)
- il metodo *leggiCruciverba,* che legge dal ***Reader*** (già aperto) fornito i dati necessari e restituisce un ***Cruciverba*** perfettamente configurato; esso lancia ***BadFileFormatException*** con opportuno messaggio d'errore in caso di problemi nel formato del file, ivi inclusi i casi in cui il numero di colonne non sia ***m*** e/o il numero di righe non sia ***n***, o ***IOException*** in caso di altri problemi di I/O

b) La classe ***MyCruciverbaReader*** (da completare) implementa ***CruciverbaReader***:

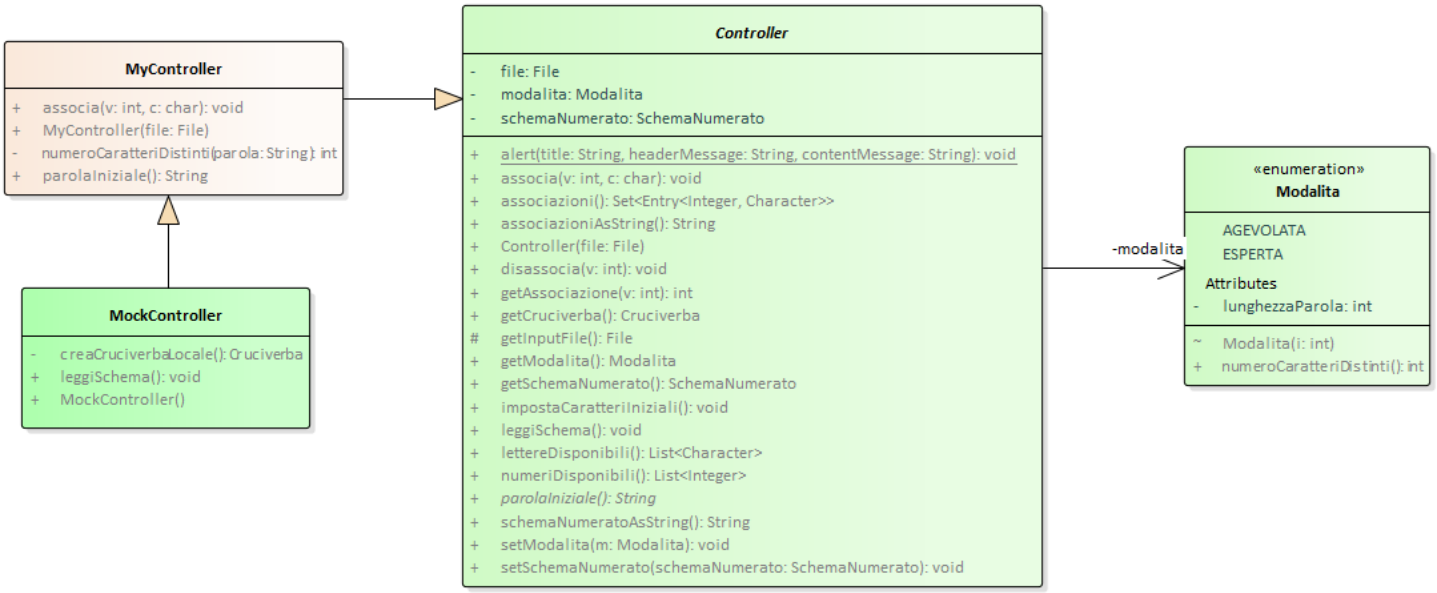
- non c'è alcun costruttore
- Il metodo *leggiCruciverba* legge il file, lo interpreta secondo le specifiche sopra fornite e, se tutto è ok, crea e popola il ***Cruciverba*** cella per cella; ogni deviazione dal formato previsto, incluso il caso in cui il numero di colonne non sia ***m*** e/o il numero di righe non sia ***n***, deve causare una ***BadFileFormatException*** con dettagliato e specifico messaggio d'errore (vedere test)



Parte 3 (punti: 15)

Package: cruciverba.controller **[TEMPO STIMATO: 35-45 minuti]** ***(punti 8)***

Il Controller costituisce il nucleo centrale di questa applicazione: crea e mantiene le strutture dati, gestisce la logica di gioco e le modalità dello stesso. A tal fine è organizzato come segue:



SEMANTICA:

a) La classe astratta ***Controller*** (fornita) implementa già il grosso delle funzionalità: rimane astratto solo il metodo *parolainiziale*, che dovrà essere implementato in una sottoclasse concreta.

- Il costruttore riceve il **File** da cui dovranno essere letti i dati (cioè il cruciverba classico di partenza), recuperabile dall'omonimo accessor *getInputFile*
- Il metodo *leggiSchema* legge, tramite un apposito ***CruciverbaReader*** di sua proprietà, il ***File*** fornito e usa il ***Cruciverba*** così ottenuto (eventualmente recuperabile tramite l'accessor *getCruciverba*) per creare lo ***SchemaNumerato*** corrispondente, che viene quindi memorizzato nel proprio stato interno; imposta altresì, di default, la modalità di funzionamento ESPERTA.
- Il metodo *impostaCaratteriIniziali* si procura, tramite il metodo *parolainiziale* (da realizzare nella sottoclasse concreta), la parola iniziale sorteggiata, ne estrae i caratteri distinti e li usa per impostare sia il mapping lettera/numero (tramite il metodo *setMapping* di ***SchemaNumerato***) sia l'associazione duale numero/lettera (tramite il metodo *associa* di ***SchemaNumerato***), così da tenere tutto sincronizzato
- I due metodi *lettereDisponibili* e *numeriDisponibili* restituiscono rispettivamente la lista ordinata delle sole lettere / dei soli numeri presenti nel crittocruciverba ma non ancora usati dal giocatore/solutore (ossia, presenti nei mapping ma non nelle associazioni): quindi, eventuali lettere dell'alfabeto non presenti in un dato schema (come 'W' nell'esempio) *non* saranno mai restituite.
- Le due coppie di accessor *getModalita/setModalita, setSchemaNumerato/schemaNumeratoAsString* accedono / impostano rispettivamente la modalità e lo schema numerato interni al Controller.

Il metodo statico *alert*, utilizzabile dal ***MainPane*** quando è attiva la grafica, consente di far comparire all'utente una finestra di dialogo che mostri il messaggio d'errore specificato.

La sottoclasse ***ControllerMock*** (fornita) è utilizzata esclusivamente da ***CruciverbAppMock*** (fornita, vedere sotto) per simulare la presenza dei dati letti da file: pertanto, non deve essere considerata in questo compito.

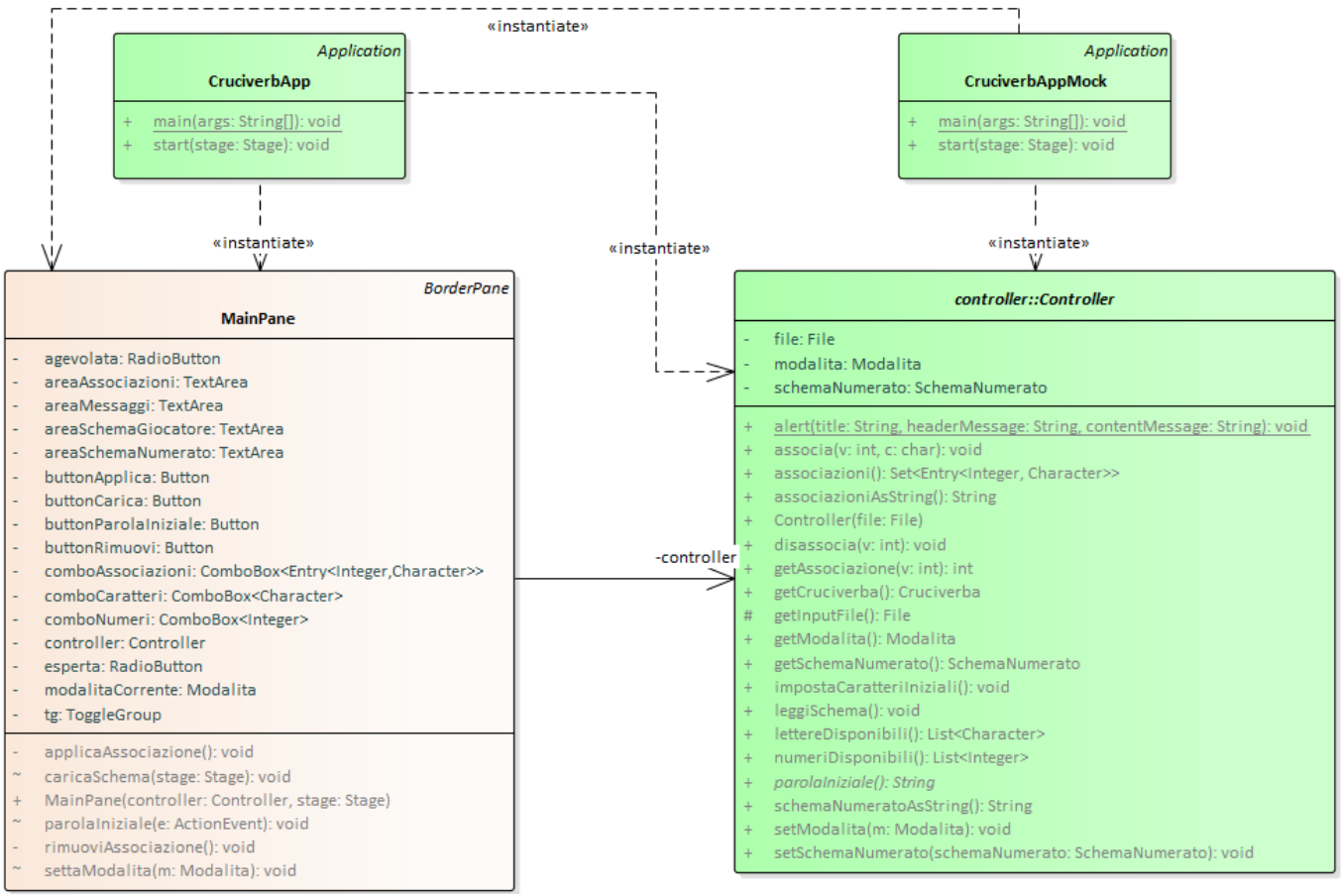
b) La classe concreta ***MyController*** (da realizzare) implementa il metodo *parolainiziale* astratto nel ***Controller*** e ridefinisce conservativamente il metodo *associa* in modo che tenga conto anche della modalità di funzionamento (AGEVOLATA o ESPERTA) prescelta. Più precisamente:

- Il costruttore riceve il **File** da cui dovranno essere letti i dati (cioè il cruciverba classico di partenza), e rimpalla l'operazione sulla classe base
- Il metodo *associa (da realizzare)*, già definito in ***Controller***, deve essere ridefinito in modo che, nel caso di modalità AGEVOLATA, ogni tentativo di impostare una associazione lettera/numero errata venga respinto lanciando ***UnsupportedOperationException*** con idoneo messaggio; diversamente, se tutto è ok, deve rimpallare l'operazione sulla classe base.
- Il metodo *parolainiziale* (da realizzare) contiene l'algoritmica fondamentale: esso deve dapprima estrarre dal ***Cruciverba*** tutte le parole esistenti in riga, poi filtrare solo quelle in cui il numero di lettere distinte sia coerente con la modalità selezionata (ossia, 3 per la modalità ESPERTA e 4 per la modalità AGEVOLATA), infine sorteggiare a caso una parola fra queste e restituirla. Ai fini di questo compito, si può fare l'ipotesi che una tale parola esista sempre.
  NB: il numero di caratteri distinti è ottenibile dall'omonimo metodo di ausilio fornito a corredo.

Package: cruciverba.ui **[TEMPO STIMATO: 40-50 minuti]** *(punti 7)*

La classe ***CruciverbApp*** (fornita) costituisce l'applicazione JavaFX che si occupa di aprire i file, creare il controller e incorporare il ***MainPane***. Per consentire di collaudare la GUI anche in assenza / in caso di malfunzionamento della parte di persistenza, è possibile avviare l'applicazione mediante la classe ***CruciverbAppMock*** (che si avvale del controller ausiliario ***MockController*** e del ***MockCruciverbaReader***).

L'interfaccia utente è illustrata nelle figure seguenti e segue il modello sotto illustrato*:*



- a sinistra vi sono tutti i pulsanti e i controlli (radiobutton, combobox, aree messaggi)
- a destra vi sono invece le due aree di output, che mostrano rispettivamente lo schema iniziale (sopra) e quello di gioco (sotto) secondo l'evoluzione corrente.

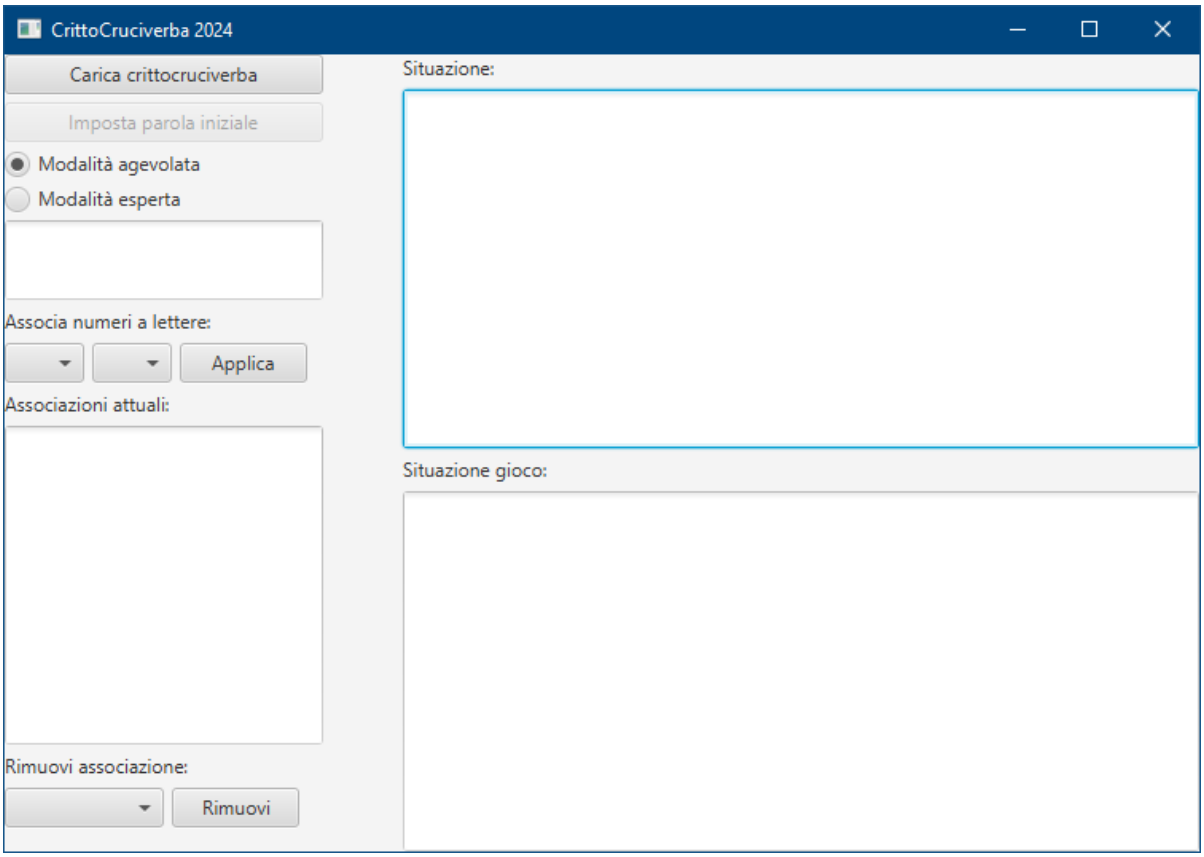


Fig. 1: situazione iniziale della GUI.

Inizialmente è usabile solo il pulsante “Carica Crittocruciverba”: ogni tentativo di cliccare su “Applica” o “Rimuovi” in questa fase causerà l’apparizione di un dialogo con msg d’errore (Fig. 2); parimenti, le combo sono tutte vuote.

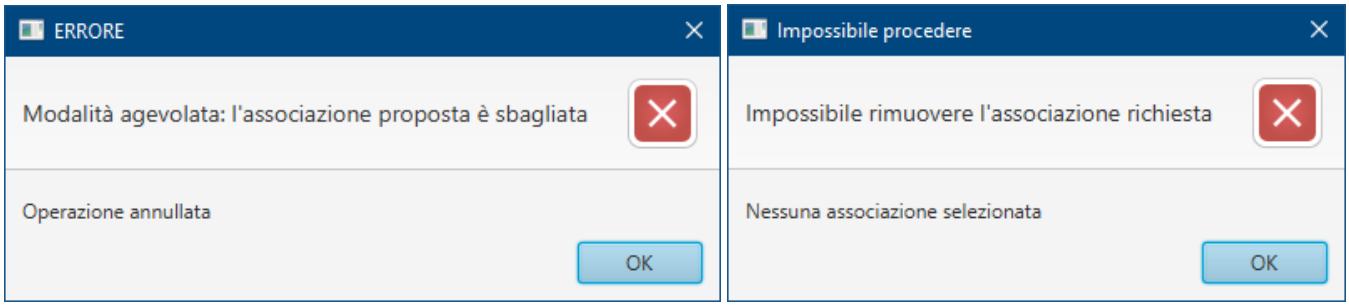


Fig. 2: messaggi d’errore premendo “Applica” o “Rimuovi” in questa fase iniziale.

Premendo il pulsante “Carica Crittocruciverba” viene caricato il cruciverba classico da file e viene generato e mostrato lo schema numerato: il pulsante di caricamento si disabilita, mentre si abilita il successivo che consente l’impostazione della parola iniziale, previa scelta della modalità (di default l’AGEVOLATA) (Fig. 3). L’area messaggi mostra le operazioni via via eseguite (caricamento cruciverba, selezione modalità).

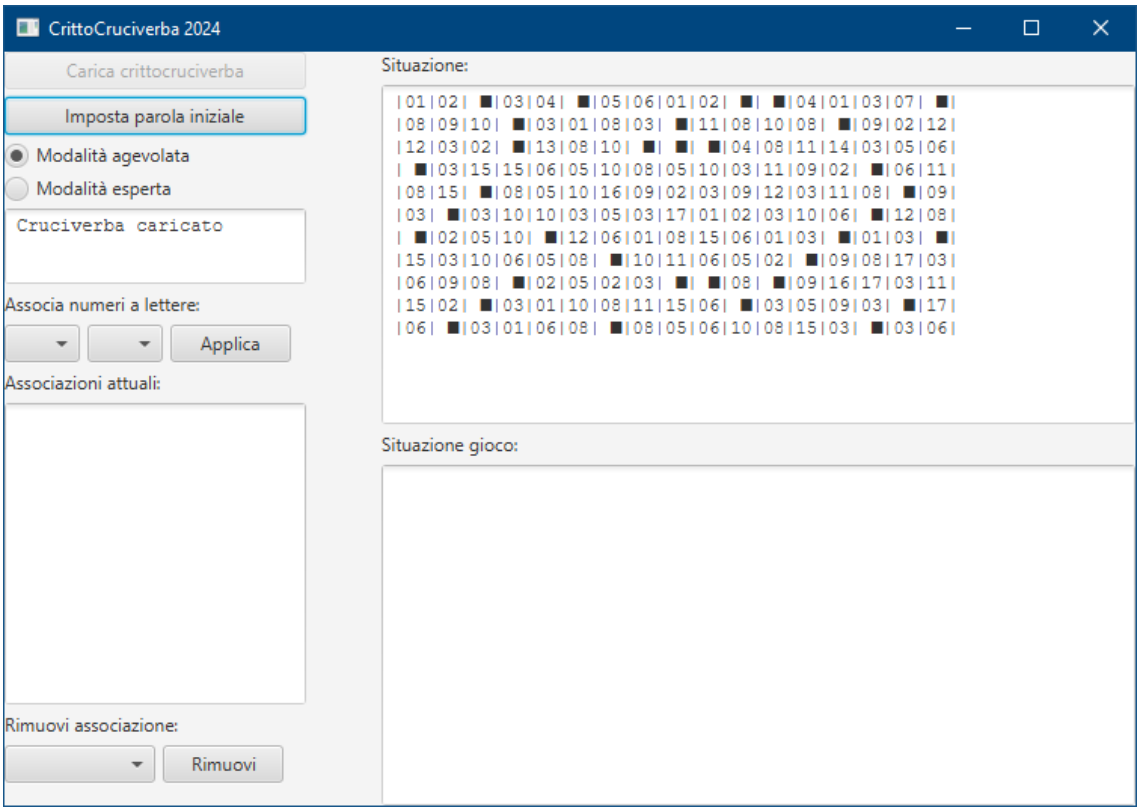


Fig. 3: la GUI dopo aver premuto il pulsante “Carica crittocruciverba”.

Premendo ora il pulsante “Imposta parola iniziale”, il sistema sorteggerà una parola orizzontale adatta e ne propagherà le lettere su tutto lo schema, popolando altresì le combo di numeri e lettere con quelli attualmente disponibili. In Fig. 4 si vede la situazione risultante: in questo caso è stata sorteggiata la parola “VLAD” (prima riga).

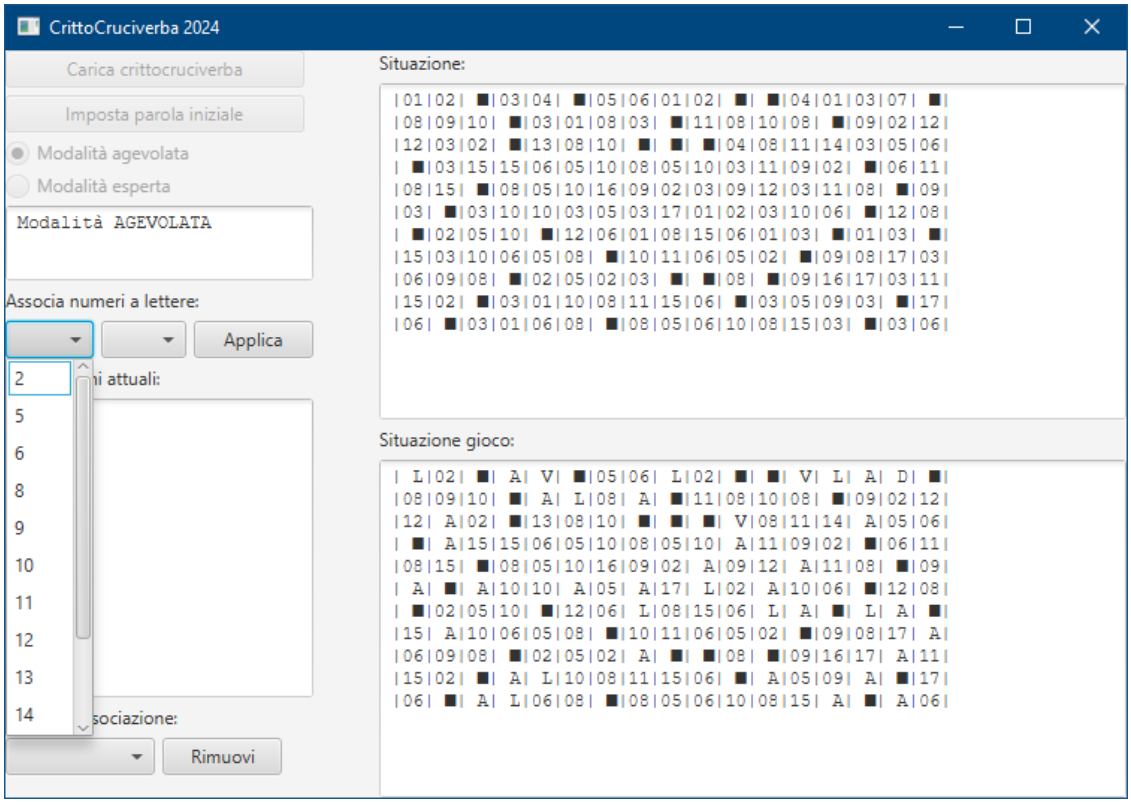


Fig. 4: la GUI dopo aver premuto il pulsante “Imposta parola iniziale” , che subito dopo si disabilita. Notare le combo numeri/lettere, popolate coi numeri/lettere ancora non svelati nello schema (“VLAD” usava i numeri 4,1,3,7, che infatti mancano nell’elenco della combo; lo stesso vale per le lettere A,D,L,V per l’elenco adiacente).

A questo punto si può iniziare a giocare: il solutore sceglie possibili associazioni e le imposta premendo il pulsante "Applica" (Fig. 5); in modalità agevolata, un'associazione errata verrà respinta (Fig. 6) mentre in modalità esperta sarà comunque accettata. Dopo ogni mossa, l'area sottostante mostra le associazioni in vigore, che sono anche selezionabili nella combo in basso per l'eventuale rimozione nel caso in cui ci accorga di aver sbagliato e si voglia tornare indietro a uno stadio precedente del gioco (cosa ovviamente sensata solo in modalità ESPERTA, dato che in agevolata è materialmente impedito inserire associazioni errate).

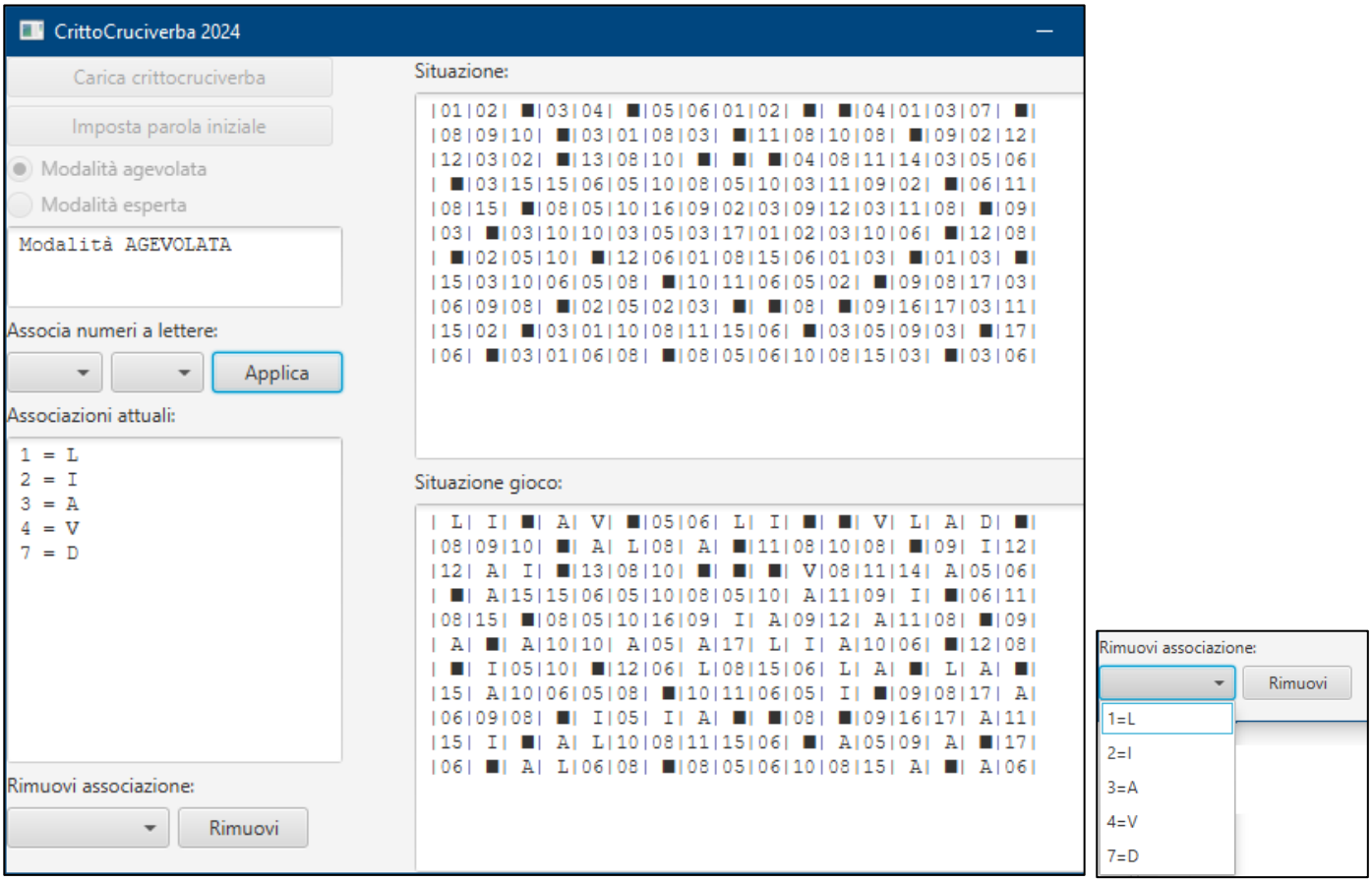


Fig. 5: la GUI dopo aver inserito l'associazione 2 = 'I': notare che questi due valori vengono rimossi dai possibili item selezionabili nelle due combo singole, mentre compare l'associazione 2=I nell'elenco associazioni in basso.

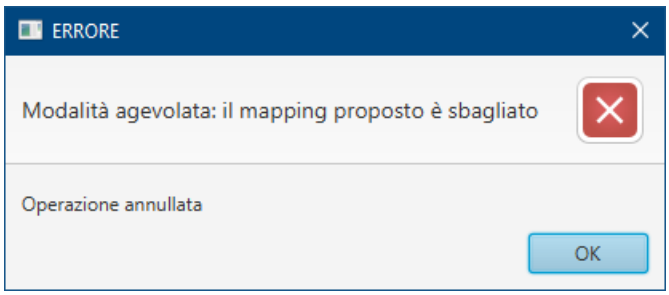


Fig. 6: messaggio d'errore nel caso si imposti una associazione errata in modalità AGEVOLATA.

Ad esempio, se si volesse rimuovere l'associazione 1=L, basterebbe selezionarla nella combo e premere il pulsante "Rimuovi": dallo schema numerato sparirebbero le L, sostituite da "1" come a uno stadio precedente.

Proseguendo, si giungerà infine ad aver utilizzato tutte le lettere e tutti i numeri, completando lo schema (Fig. 7)

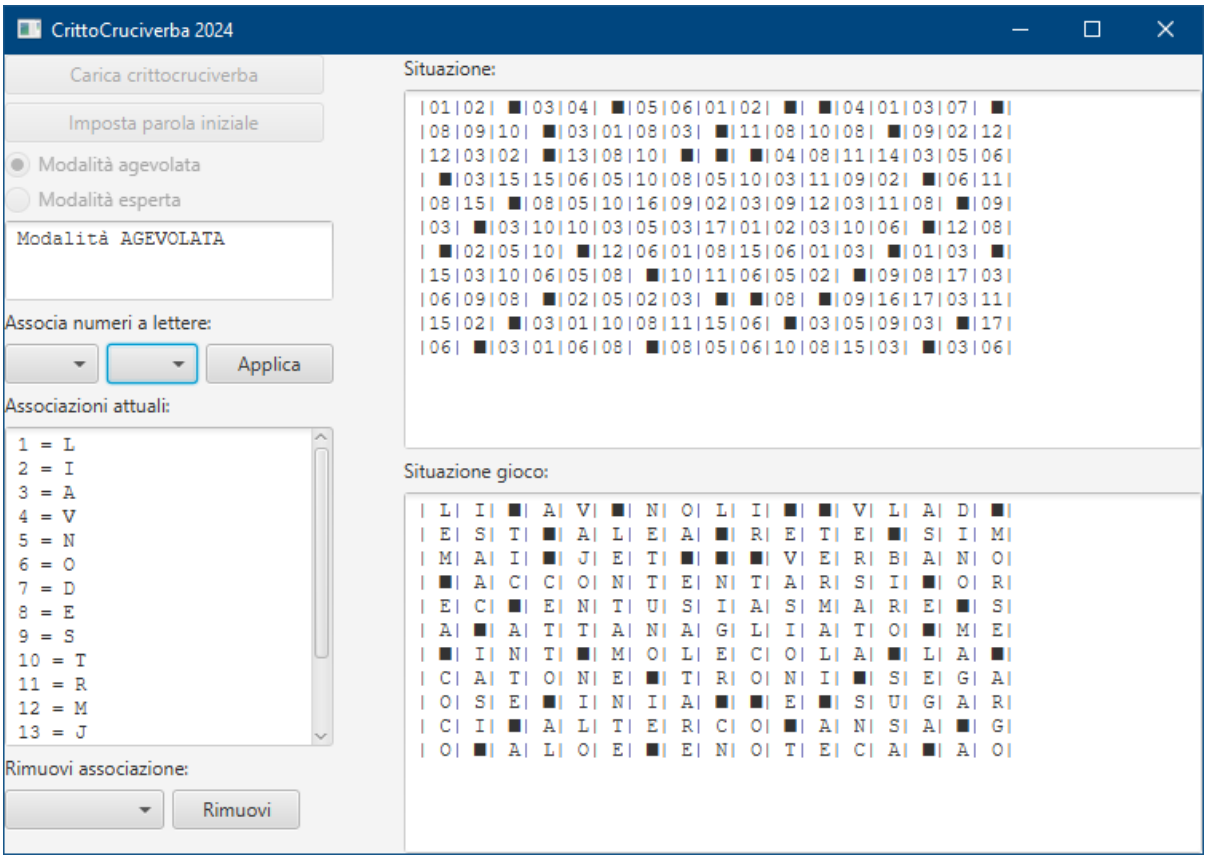


Fig. 7: lo schema completato.

Il MainPane è fornito *quasi completamente realizzato*: è presente tutta la parte strutturale, mentre rimangono da realizzare due gestori degli eventi (metodi privati *applicaAssociazione*, *rimuoviAssociazione*) e da garantire la mutua esclusione fra i due radiobutton.

In particolare:

- *parolaIniziale* imposta sul ***Controller*** la modalità prescelta, indi imposta le lettere iniziali, mostra lo schema numerato nell'apposita area di uscita, disabilita i pulsanti già utilizzati e popola le combo con le lettere e i numeri ancora disponibili
- *settaModalita* imposta sul ***Controller*** la modalità attualmente selezionata sui radiobutton
- *caricaSchema* attiva sul ***Controller*** la lettura da file e la successiva generazione dello schema numerato, abilitando/disabilitando i pulsanti come opportuno; in caso di errore di I/O o di formato, emette un'alert differenziato, con opportuna messaggistica
- *applicaAssociazione* recupera gli item selezionati nelle due combo (se uno o entrambi non sono selezionati, viene emesso opportuno msg d'errore e non si fa altro) e li usa per impostare un'associazione sul controller. Poi, ripopola le due combo col nuovo elenco aggiornato (che non comprende più i valori ora usati) ottenibile dai metodi del controller, aggiorna la visualizzazione dello schema numerato e infine aggiorna l'elenco delle associazioni correnti dell'apposita combo in basso; tale elenco viene anche stampato nell'area messaggi.
- *rimuoviAssociazione* agisce simmetricamente, recuperando dalla combo associazioni l'elemento selezionato (se esiste, altrimenti errore da mostrare tramite opportuna *alert*) usandolo per disassociare quei due elementi: di conseguenza deve essere aggiornato il contenuto delle due combo numeri e lettere, che devono ora includere nuovamente gli elementi dell'associazione cancellata, e della combo associazioni stessa, che avrà un elemento in meno.

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?