ESAME DI FONDAMENTI DI INFORMATICA T-2 del 12/6/2024

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

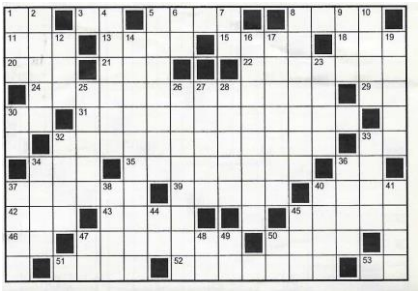
Tempo a disposizione: 3h30

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".

L'Associazione Enigmisti di Dentinia ha richiesto un'applicazione che *numeri le caselle di uno schema* appena creato dai propri Esperti Pensatori.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Appena creato dall'Esperto Pensatore, un *cruciverba* è costituito semplicemente dallo schema con le parole e le caselle nere, senza alcuna numerazione delle caselle (v. immagine a lato).

Come operazione preliminare alla scrittura delle definizioni orizzontali e verticali, si rende quindi necessario numerare le caselle: il risultato è illustrato a lato.

La numerazione delle caselle segue un apposito algoritmo:

- le caselle sono numerate da sinistra a destra, dall'alto in basso
- non tutte le caselle devono essere numerate, ma soltanto quelle che si trovano all'inizio di una parola di lunghezza almeno pari a 2, orizzontale o verticale.

Una casella (non nera) deve quindi essere numerata se:
- si trova nella prima riga e la casella sotto di lei non è nera
- si trova in righe successiva e la casella sotto di lei non è nera, mentre quella superiore lo è
- si trova nella prima colonna e la casella alla sua destra non è nera
- si trova in colonne successiva e la casella alla sua destra non è nera, mentre quella alla sua sinistra lo è

ESEMPIO: come si vede in figura, nella prima riga vengono numerate tutte le caselle tranne la "L" di "NOLI" e la "L" di "VLAD", che hanno sotto di sé una casella nera; nelle righe successive non è numerata, ad esempio, la "S" di "EST" poiché non si trova a inizio di parola né in verticale né in orizzontale, mentre lo è la "T" della stessa parola in quanto si trova all'inizio della parola verticale "TIC"; nella quarta colonna sono numerate la "C" e la "E" di "CETTO", poiché si trovano rispettivamente all'inizio di una verticale e di una orizzontale; e così via.

Lo schema creato dal Pensatore è descritto nel file di testo schema.txt, nel formato esplicitato più oltre.

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

PARTE 1 – Modello dei dati: Punti 15 [TEMPO STIMATO: 80-100 minuti]

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

PARTE 3 – Grafica: Punti 7 [TEMPO STIMATO: 25-35 minuti]

NUMERO MINIMO DI TEST CON SUCCESSO PERCHÉ IL COMPITO SIA CORRETTO

Considerando solo Numeratore e MySchemaReader 4 su 7

Considerandoli tutti: 8 su 11

JAVAFX – Parametri run configuration nei LAB

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

Cose da ricordare

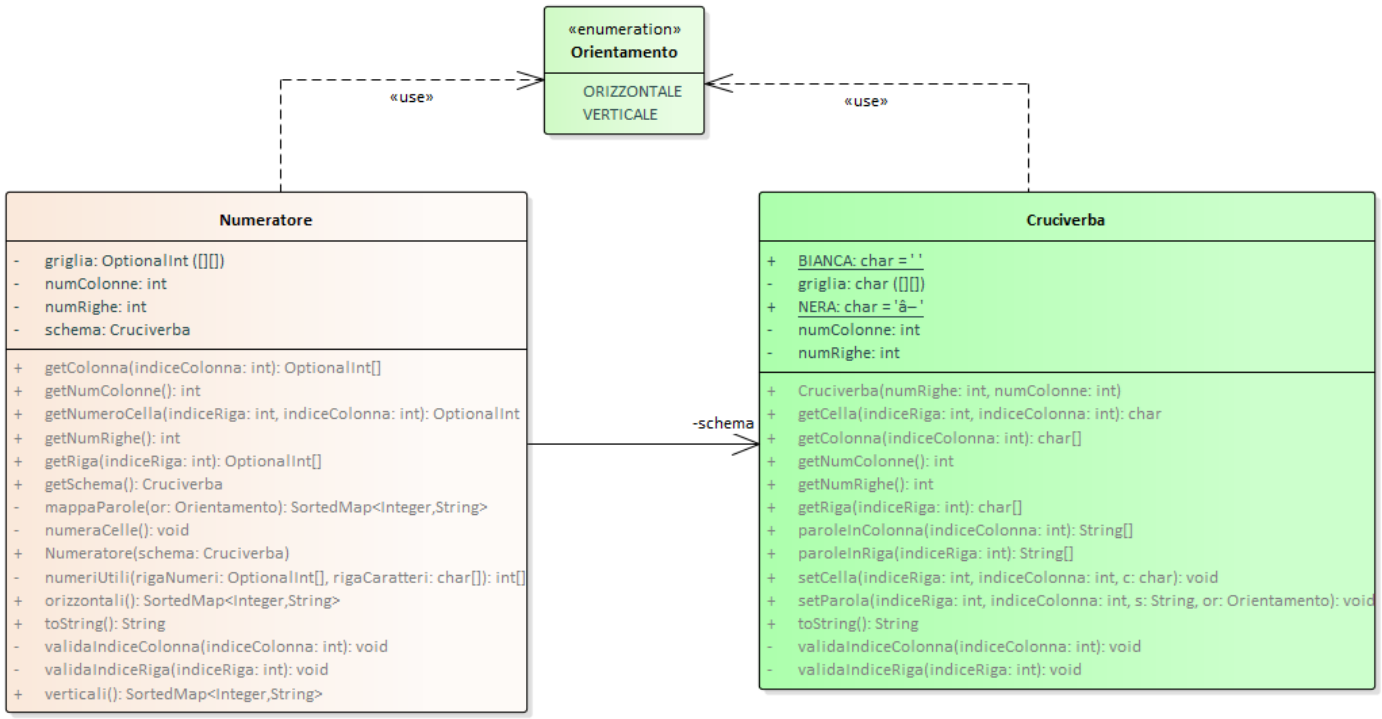
- salva costantemente il tuo lavoro: l'informatica a volte può essere "subdolamente ostile"..
- in particolare: se ora compila e stai per fare modifiche, salva la versione attuale (non si sa mai)

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: 15)

Package: cruciverba.model **[TEMPO STIMATO: 80-100 minuti]**



SEMANTICA:

a) L'enumerativo ***Orientamento*** (fornito) definisce semplicemente le due costanti ***ORIZZONTALE*** e ***VERTICALE***

b) La classe ***Cruciverba*** (fornita) rappresenta il cruciverba così come creato dall'Esperto Pensatore, inteso come matrice rettangolare di caratteri: le due costanti pubbliche ***BIANCA*** e ***NERA*** costituiscono i particolari caratteri usati per rappresentare rispettivamente una casella bianca (vuota) o nera. Più in dettaglio:

- Il costruttore riceve le dimensioni (numero di righe, numero di colonne) dello schema e alloca la matrice interna (inizialmente riempita per comodità tutta con caselle nere)
- i due accessor *getNumRighe*, *getNumColonne* restituiscono i valori passati al costruttore
- i due metodi *getRiga(int i)*, *getColonna(int i)* estraggono rispettivamente la riga / la colonna i-esima, lanciando ***IllegalArgumentException*** nel caso l'indice ricevuto non sia nel range ammesso (0..N-1)
- i due metodi *getCella(int r, int c)*, *setCella(int r, int c, char ch)* rispettivamente recuperano il carattere / impostano il carattere della casella alle coordinate indicate: come sopra, lanciano ***IllegalArgument-Exception*** nel caso l'indice ricevuto non sia nel range ammesso (0..N-1)
- il metodo di utilità *setParola(int r, int c, String parola, Orientamento or)* inserisce nello schema la parola data a partire dalle coordinate indicate, nell'orientamento (orizzontale/verticale) indicato; il metodo verifica preventivamente che gli indici siano nel range ammesso (0..N-1) e che la parola non fuoriesca dallo schema, lanciando ***IllegalArgumentException*** in caso contrario
- i due metodi *paroleInRiga(int i)*, *paroleInColonna(int i)* estraggono tutte le parole, indipendentemente dalla loro lunghezza (anche quelle di lunghezza 1, ossia i caratteri singoli), contenute rispettivamente nella riga / nella colonna specificata: ovviamente anch'essi lanciano ***IllegalArgumentException*** nel caso l'indice ricevuto non sia nel range ammesso (0..N-1).
- un'apposita *toString* emette una rappresentazione stampabile del cruciverba.

c) La classe ***Numeratore*** (da completare nella parte algoritmica) incorpora la logica di numerazione delle caselle descritta nel Dominio del Problema. *Internamente è organizzato come matrice di* ***OptionalInt****,* le cui celle

vengono istanziate all'opportuno intero quando la casella viene numerata, mentre restano *empty* per le caselle che non devono essere numerate. Sono forniti già implementati:

- il costruttore, che riceve il ***Cruciverba*** su cui operare
- gli accessor *getNumRighe*, *getNumColonne, getSchema* che restituiscono rispettivamente il numero di righe/colonne del ***Cruciverba*** passato al costruttore, e il ***Cruciverba*** stesso
- i due metodi privati *validaIndiceRiga(int i)*, *validaIndiceColonna(int i)* , identici a quelli di ***Schema***
- il metodo *getNumeroCella (int r, int c)* che restituisce l'***OptionalInt*** relativo alla cella data; come sempre, viene lanciata ***IllegalArgumentException*** nel caso gli indici non siano nel range (0..N-1)
- i due metodi *getRiga(int i)*, *getColonna(int i)*, che recuperano rispettivamente la numerazione di una data riga/ di una data colonna sotto forma di array di ***OptionalInt***: ovviamente, lanciano anch'essi ***IllegalArgumentException*** nel caso l'indice ricevuto non sia nel range (0..N-1)
- un'apposita *toString,* che emette una rappresentazione stampabile del cruciverba numerato
- il metodo privato di utilità *numeriUtili* che, date una riga (o una colonna) del cruciverba e la corrispondente riga (o colonna) dello schema numerato, restituisce un array con i soli numeri corrispondenti a iniziali di parole di lunghezza almeno pari a 2, ossia quelli che serviranno per elencare le parole orizzontali o verticali.

Devono invece essere implementati *(VEDERE LE FIGURE SEGUENTI PER OPPORTUNI ESEMPI):*

- il metodo privato ***numeraCelle***, che incorpora la logica di numerazione delle celle: invocato solo dal costruttore, esso deve popolare la matrice di ***OptionalInt*** secondo l'algoritmo descritto nel Dominio del Problema
- i due metodi ***orizzontali*** e ***verticali,*** che restituiscono una **SortedMap<Integer,String>** con l'elenco delle parole di lunghezza almeno pari a 2 (rispettivamente, solo orizzontali o solo verticali) del cruciverba, ordinate in senso crescente sui rispettivi numeri: tale elenco servirà poi per produrre l'elenco della parole da mostrare nell'applicazione (vedere figure).
  IMPORTANTE: mentre i metodi *paroleInRiga(int i)*, *paroleInColonna(int i)* di **Schema** estraggono tutte le parole, quindi anche quelle di lunghezza 1, qui sono richieste invece solo le parole di lunghezza almeno pari a 2, in quanto i caratteri singoli non vengono mai definiti nei cruciverba.
  - **SUGGERIMENTO**: è opportuno confrontare le parole di ogni riga (o colonna), opportunamente filtrate per escludere i caratteri singoli, con i corrispondenti *"numeri utili"* calcolati del metodo omonimo *numeriUtili*, che fornisce appunto i soli numeri della riga (o colonna) corrispondenti a *inizi di parole* di quella riga (o colonna).
  - NB: volendo, poiché l'algoritmo è sostanzialmente identico / speculare per orizzontali e verticali, si può anche pensare (ma NON è assolutamente obbligatorio!) di fattorizzare il codice comune in un metodo ausiliario *mappaParole*, a cui passare soltanto l'***Orientamento*** desiderato.

Parte 2 – Persistenza (punti: 8)

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

Il cruciverba ideato dal Pensatore è descritto nel file di testo schema.txt: per convenzione, le caselle nere sono rappresentate nel file dal trattino "-" (vedere a lato).

ESEMPIO

```
LI-AV-NOLI--VLAD-
EST-ALEA-RETE-SIM
MAI-JET---VERBANO
-ACCONTENTARSI-OR
EC-ENTUSIASMARE-S
A-ATTANAGLIATO-ME
-INT-MOLECOLA-LA-
CATONE-TRONI-SEGA
OSE-INIA--E-SUGAR
CI-ALTERCO-ANSA-G
O-ALOE-ENOTECA-AO
```

SEMANTICA:

a) Il record di utilità ***ElementoRiga*** (fornito) rappresenta la coppia (stringa, posizione)

b) L'interfaccia ***SchemaReader*** (fornita) dichiara:

- il metodo *leggiSchema,* che carica i dati necessari e restitisce un ***Cruciverba*** perfettamente configurato; lancia ***BadFileFormatException*** con opportuno

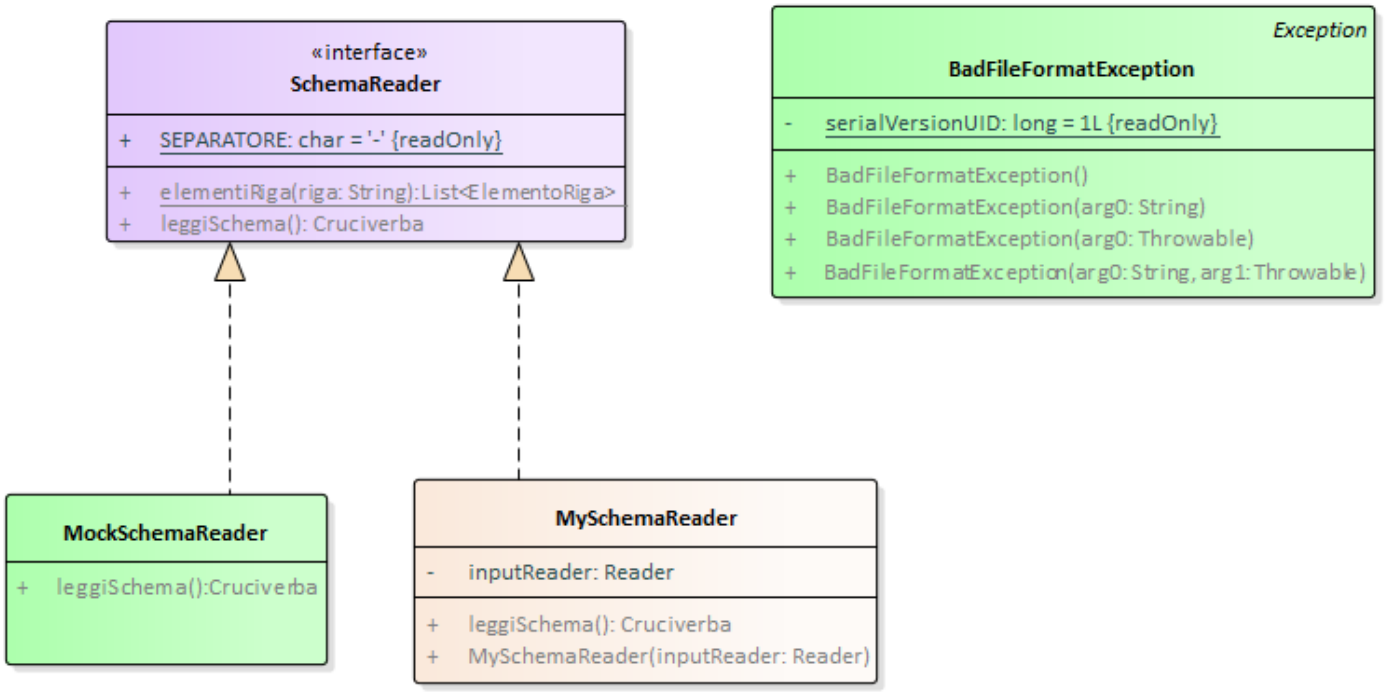
messaggio d'errore in caso di problemi nel formato del file, ossia *righe di lunghezza diversa fra loro*, o ***IOException*** in caso di problemi di I/O

- la costante carattere **SEPARATORE** (attualmente, il trattino), utile per parametrizzare la rappresentazione su file delle caselle nere.
- Il metodo statico di utilità *elementiRiga*, che restituisce la lista degli ***ElementoRiga*** presenti nella riga ricevuta come argomento.

c) La classe ***MySchemaReader*** (da completare) implementa ***SchemaReader*** secondo le specifiche sopra descritte.

- Il costruttore riceve e memorizza il ***Reader*** (già aperto) da cui leggere
- Il metodo *leggiSchema* dapprima acquisisce tutte le righe, poi istanzia il ***Cruciverba*** della giusta dimensione e lo popola, tramite il metodo *setParola* di ***Cruciverba***, scrivendo in esso tutte e sole le parole orizzontali (le verticali risulteranno presenti implicitamente), anche quelle di lunghezza 1, ottenute elaborando ogni riga tramite il metodo *elementiRiga* di ***SchemaReader.***

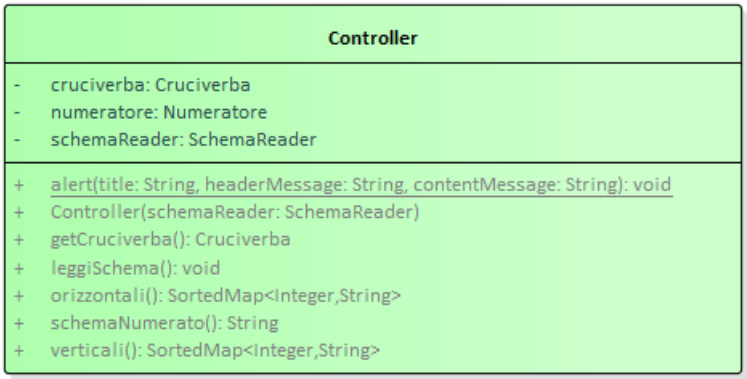
NB: la classe ***MockSchemaReader*** (fornita) serve unicamente a supporto della ***CruciverbAppMock*** descritta più oltre e non è quindi di interesse per il candidato.



Parte 3 (punti: 7)

Package: cruciverba.controller *(punti 0)*

Il Controller funge semplicemente da front-end verso l'analizzatore, ed è quindi è organizzato come segue:



SEMANTICA:

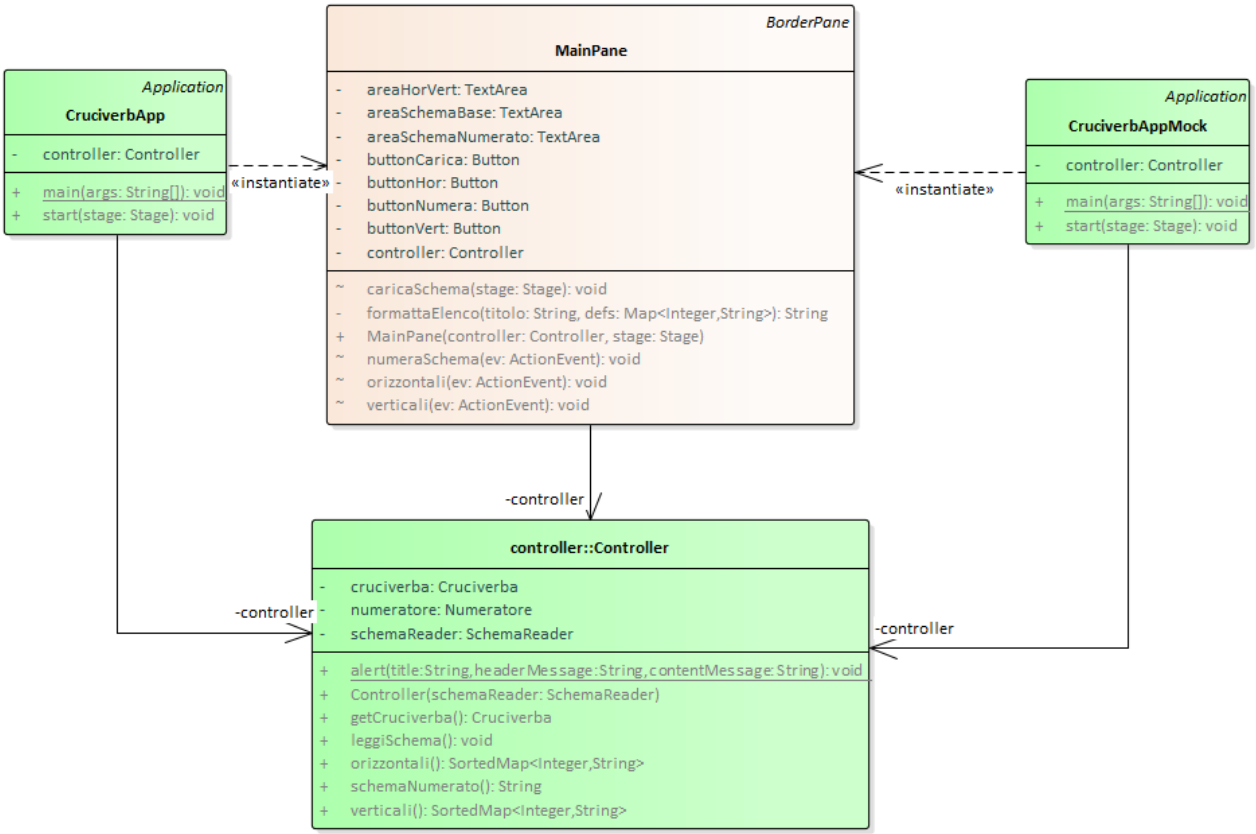
La classe ***Controller*** (fornita) riceve in fase di costruzione lo ***SchemaReader*** da cui leggere il ***Cruciverba***, di cui poi curerà anche la numerazione celle (tramite opportuno ***Numeratore*** creato e gestito al proprio interno); espone vari metodi utili a rispondere alle interrogazioni dell'utente. In particolare:

- Il costruttore si limita a memorizzare il ***Reader*** da cui leggere, ma non effettua operazioni
- *leggiSchema* legge effettivamente il ***Cruciverba*** dal reader e lo memorizza nel proprio stato interno, creando contestualmente il corrispondente ***Numeratore*** (istanziato per l'occasione), che provvede a generare immediatamente la numerazione
- *schemaNumerato* recupera e restituisce lo schema numerato sotto forma di stringa, previa verifica che il cruciverba sia stato letto e il numeratore istanziato: altrimenti, lancia un'apposita ***UnsupportedOperation-Exception*** con adeguato messaggio d'errore
- *orizzontali, verticali* recuperano la ***SortedMap<Integer,String>*** contenente le parole associate ai rispettivi numeri, appoggiandosi agli omonimi metodi del ***Numeratore*** interno; anche in questo caso si verifica preventivamente che il cruciverba sia stato letto e il numeratore istanziato, altrimenti viene lanciata un'apposita ***UnsupportedOperationException*** con adeguato messaggio d'errore.

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

Package: cruciverba.ui **[TEMPO STIMATO: 25-35 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 reader simulato ausiliario ***MockSchemaReader***).



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

- a sinistra, quattro pulsanti consentono di controllare il funzionamento dell'applicazione (sopra), mentre una piccola area di testo (sotto) mostra, quando richiesto, le parole orizzontali o verticali

- a destra, due aree di testo di grandi dimensioni mostrano rispettivamente il cruciverba così come ideato dal Pensatore (sopra, ancora da numerare) e la versione numerata prodotta dall'applicazione (sotto).

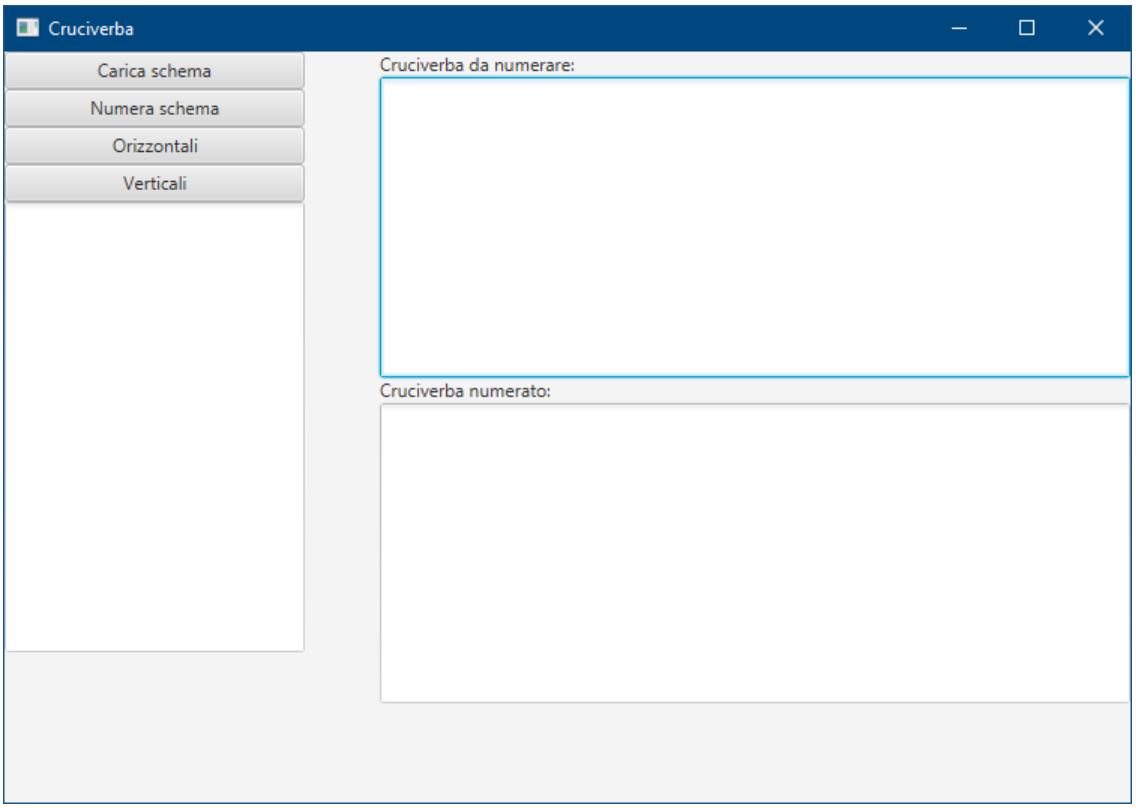


Fig. 1: vista generale della GUI: a sinistra l'area pulsanti, a destra le textarea che mostrano le view del cruciverba.

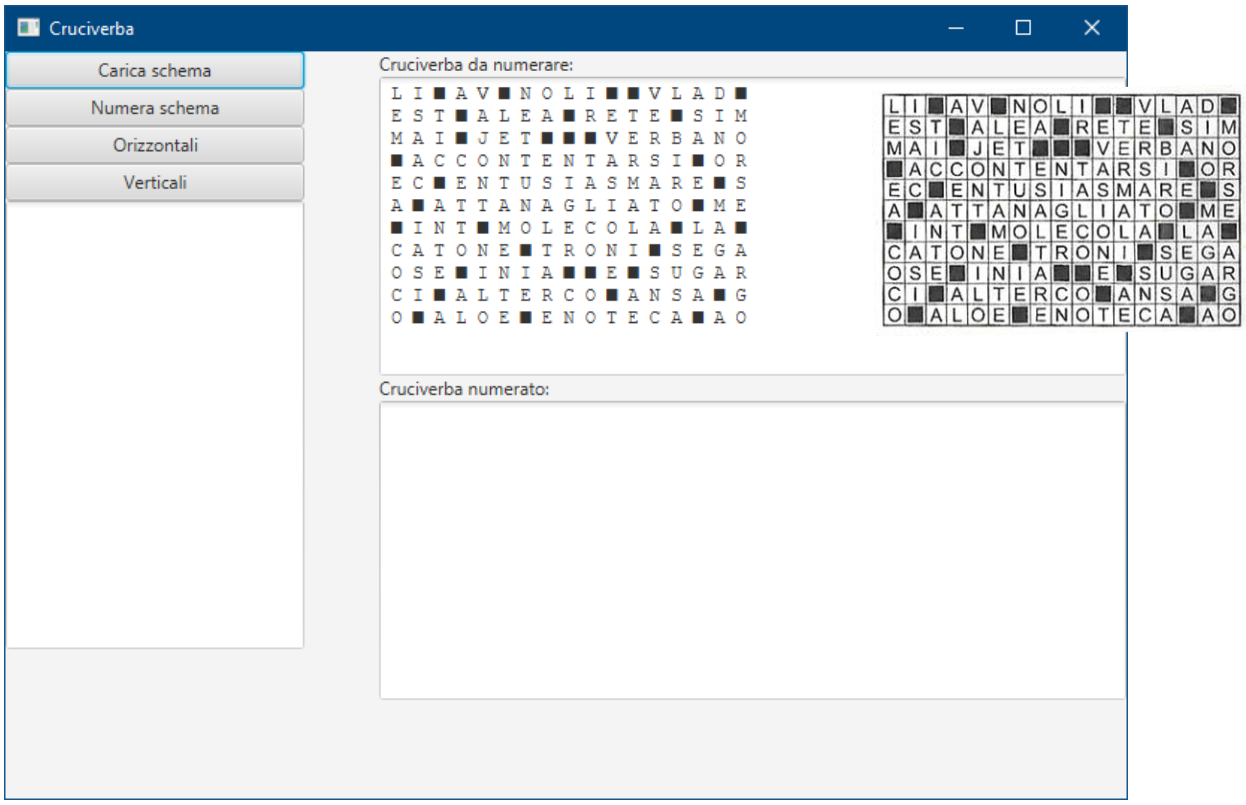


Fig. 2: la GUI dopo aver premuto il pulsante "Carica schema".

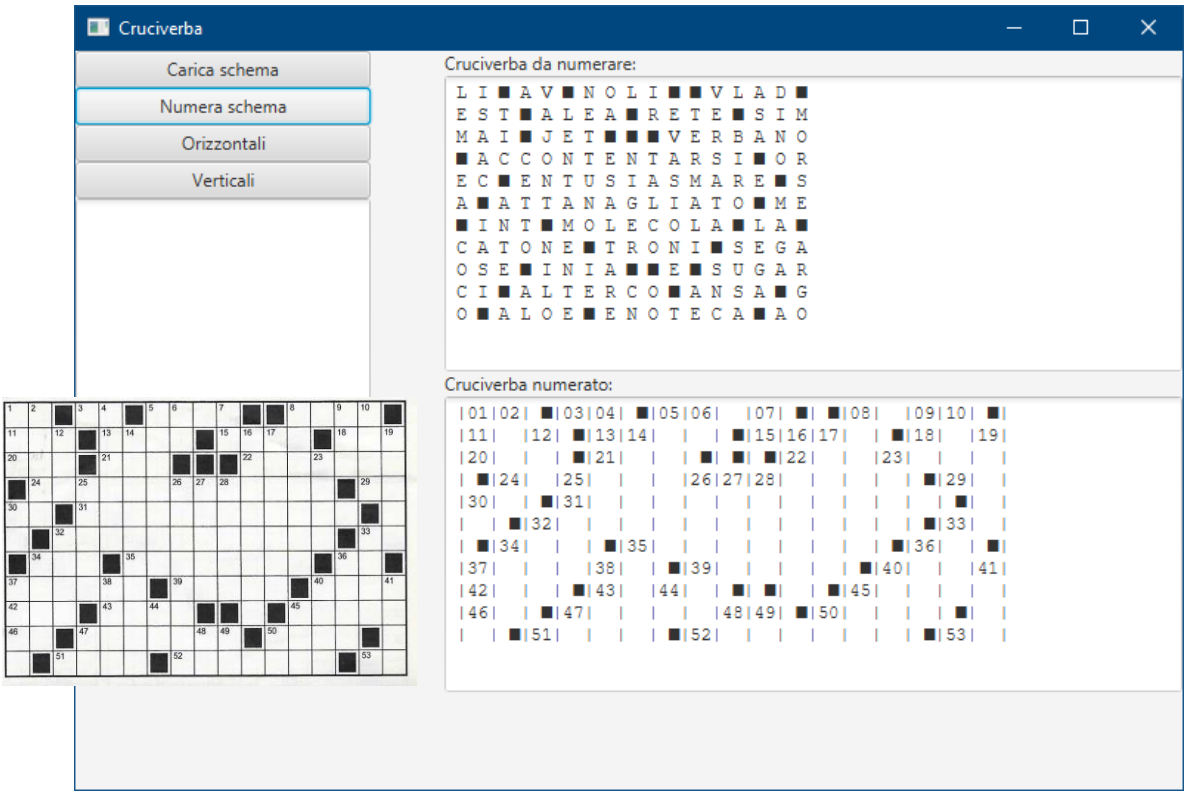


Fig. 3: la GUI dopo aver premuto il pulsante “Numera schema”.

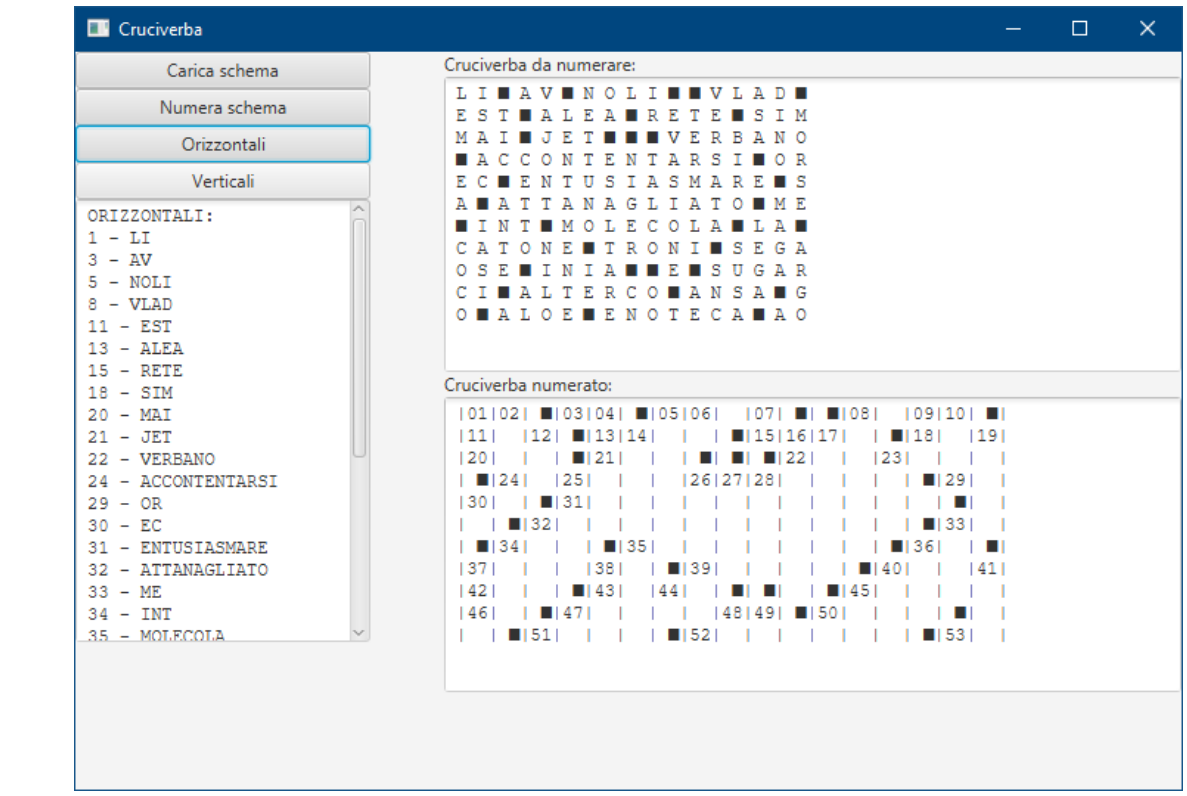


Fig. 4: la GUI dopo aver premuto il pulsante “Orizzontali” (analogo output si ha premendo il pulsante “Verticali”)

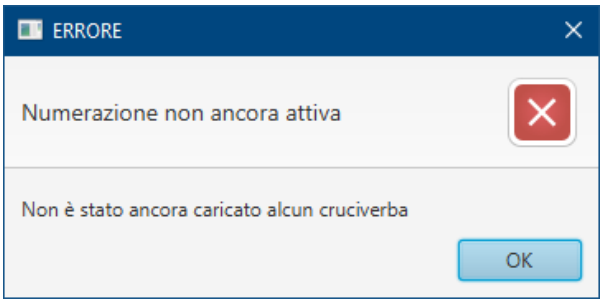


Fig. 5: messaggio d'errore nel caso si cerchi di attivare la numerazione, o di ottenere l'elenco delle parole orizzontali o verticali, senza aver prima caricato lo schema del cruciverba.

Il MainPane è fornito *parzialmente realizzato*: è presente tutta la parte strutturale, mentre rimangono da realizzare tutti i gestori degli eventi (metodi privati *caricaSchema, numeraSchema, orizzontali, verticali*)

In particolare:

- *caricaSchema* deve leggere il ***Cruciverba*** tramite il ***Controller*** e mostrane il contenuto nell'apposita area di testo: eventuali ***IOException*** o ***BadFileFormatException*** riportate dal ***Controller*** devono essere intercettate e gestite mostrando all'utente, tramite il metodo *alert* del ***Controller***, un apposito messaggio d'errore, ritornando poi senza fare altro.
- *numeraSchema* deve numerare lo schema tramite il ***Controller*** e mostrane il risultato nell'altra area di testo: eventuali ***UnsupportedOperationException*** riportate dal ***Controller*** devono essere intercettate e gestite mostrando all'utente, tramite il metodo *alert* del ***Controller***, un apposito messaggio d'errore, ritornando poi senza fare altro.
- *orizzontali, verticali* devono recuperare dal ***Controller*** l'elenco delle parole e mostrarle nella piccola area di testo in basso a sinistra, sfruttando il metodo *formattaElenco* fornito: anche in questo caso, eventuali ***UnsupportedOperationException*** riportate dal ***Controller*** devono essere intercettate e gestite mostrando all'utente, tramite il metodo *alert* del ***Controller***, un apposito messaggio d'errore, ritornando poi senza fare altro.

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?