**“Ingegneria del Software”**

**2021-2022**

**Docente: Prof. Angelo Furfaro**

**KENKEN**

|  |  |
| --- | --- |
| **Data** | <12/06/2022> |
| **Documento** | Documento Finale – D3 |

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **Nome e Cognome** | **Matricola** | **E-mail address** |
| **Francesco Plutino** | **189582** | **Pltfnc98p24f112f@studenti.unical.it** |

Sommario

**Non è stata trovata alcuna voce d'indice.**

## List of Challenging/Risky Requirements or Tasks

|  |  |  |  |
| --- | --- | --- | --- |
| **Challenging Task** | **Date the task is identified** | **Date the challenge is resolved** | **Explanation on how the challenge has been managed** |
| **Operazioni nei blocchi della griglia di gioco** | **5/5/2022** | **12/5/2022** | **La griglia di gioco combina tutti i numeri dentro le celle di ogni blocco per verificare il soddisfacimento del risultato atteso** |
| **Creazione della GUI per renderla user-friendly** | **1/6/2022** | **5/6/2022** | **L’intera applicazione è stata strutturata in tre schermate differenti per permetterne l’uso ad ogni tipo di utente. L’utente è guidato passo passo attraverso l’inizio della partita e l’impostazione della grilia.** |

A. Stato dell’Arte

Il KenKen è un gioco di logica, ispirato al Sudoku che consiste in una griglia di dimensioni variabili (i più comuni vanno dalle griglie 3x3 fino alle più difficili 6x6), nelle quali bisogna disporre le cifre da 1 a n senza che ci siano ripetizioni né nelle righe né nelle colonne (come per il Sudoku). La griglia inizialmente è totalmente vuota, e divisa in blocchi di diverse forme da linee più spesse; in ogni blocco viene riportato un numero, seguito da un operatore aritmetico (+, -, x o ÷), che indica l'operazione da effettuare tra le varie cifre del blocco.

La griglia va completata in modo che, effettuando l'operazione riportata in ciascun blocco tra le sue cifre, si ottenga esattamente il risultato richiesto (sempre un numero intero positivo). Le cifre si possono ripetere all'interno dei blocchi, a condizione però che non si trovino sulla stessa riga o colonna.

B. Raffinamento dei Requisiti

***A.1 Servizi (con prioritizzazione)***

Le voci che seguono, rappresentano i requisiti funzionali volti a garantire il corretto funzionamento dell’applicazione. Per ogni requisito, vengono specificate l’importanza e la complessità.

* A1. Impostazioni di gioco (Alta, Alta)

Questo requisito funzionale, permette all’utente di determinare le impostazioni generali della griglia, ovvero, specificarne le dimensioni e configurandone i blocchi (caselle, operazione e valore del vincolo).

* A2. Inizio partita (Alta, Media)

L’applicazione permette di iniziare la partita immettendo i numeri nella griglia di gioco, rappresentata all’utente tramite una GUI con i dati specificati al punto precedente.

* + - A2.1. Verifica in tempo reale.

L’utente può abilitare la verifica in tempo reale che, ad ogni inserimento, ne verifica la legalità.

* A3. Visualizzazioni delle soluzioni (Media, Media)

Questo requisito permette (come richiesto nelle specifiche di progetto) di visualizzare le soluzioni, navigando tra di esse tramite appositi pulsanti.

* A4. Salvataggio e caricamento della partita (Alta, Media)

Questo requisito permette all’utente di salvare la propria partita per poi ricaricarla in seguito. Il salvataggio, ricorda i dati della griglia (dimensioni e vincoli) e i numeri inseriti dal giocatore.

***A.2 Requisiti non Funzionali***

Nessun requisito non funzionale da specificare

***A.3 Scenari d’uso dettagliati***

**- Creazione nuova partita:**

1. L’applicazione viene avviata. Questa mostrerà all’utente il suo menu principale, nel quale sono presenti i tasti: “Nuova partita “e “Carica partita”.
2. L’utente, per iniziare una nuova partita, cliccherà sul pulsante “Nuova partita”.
3. L’applicazione chiuderà il menu principale e mostrerà la schermata di impostazione della griglia di gioco. In questa schermata saranno presenti i pulsanti: “3x3”, “4x4”, “5x5” e “6x6”, che hanno lo scopo di far selezionare all’utente la dimensione della griglia che si desidera.
4. Dopo aver scelto la dimensione, andrà premuto il pulsante “CONFERMA” che consentirà all’utente di accedere alla schermata di gioco, chiudendo quella attuale.

**- Creazione della griglia di gioco:**

1. L’applicazione, creerà una griglia delle dimensioni precedentemente specificate e la mostrerà all’utente.
2. A questo punto, l’utente, può iniziare a creare a piacimento i blocchi, assegnando ad ognuno, il risultato e l’operazione.
3. L’utente selezionerà un numero di blocchi (da 1 a 3), per confermare la volontà di aggiungere un blocco, cliccherà sul pulsante “Aggiungi Blocco”.
4. Nel caso in cui l’utente provi ad aggiungere un blocco senza aver selezionato alcuna casella, verrà generato un messaggio di errore.
5. L’utente per confermare l’aggiunta di un blocco dovrà inserire nelle finestre pop-up che compariranno, prima il risultato da conseguire e successivamente l’operazione. Nel caso in cui uno dei due valori (risultato o operazione) non fosse valido, l’applicazione mostrerà il messaggio di errore associato e chiederà di ripetere l’inserimento del dato errato.

**- Partita**

1. Cliccando su “Gioca” l’utente potrà iniziare la propria partita.
2. L’utente gioca inserendo nella griglia di gioco i numeri che ritiene opportuni.
   1. L’utente può, cliccando sul tasto “Verifica integrità”, verificare passo-passo che i numeri da lui inseriti siano corretti.
3. L’utente può visualizzare le soluzioni mediante un pulsante “Visualizza soluzioni”, dopo di che su “Soluzione precedente” o “Soluzione successiva” per spostarsi tra le possibili soluzioni.

**-Salva partita**

1. Cliccando sul tasto “Salva Partita” l’utente potrà salvare la propria partita.
2. L’utente dovrà specificare il nome del file di salvataggio e la directory nella quale allocare il file.

**-Carica partita**

1. Nel menu principale sarà presente un tasto “Carica partita”, questo permetterà all’utente di caricare una partita precedente salvata sul device.
2. L’utente dovrà cercare il file nella directory di salvataggio e selezionarlo.
3. La partita verrà caricata.

***A.4 Excluded Requirements***

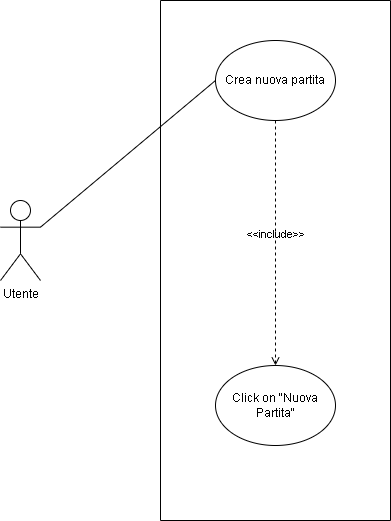
***A.5 Assunzioni***

Si assume che l’utente, per correggere nell’immediato i numeri nella griglia, li possa cancellare da tastiera.

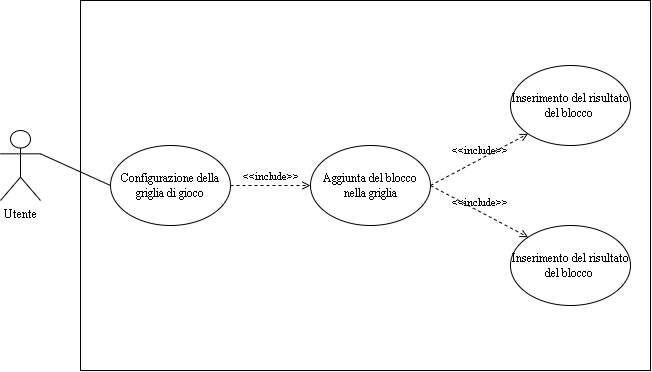
Si assume che l’utente non modifichi le partite salvate sul database, dal momento che, potrebbero essere facilmente manomesse.

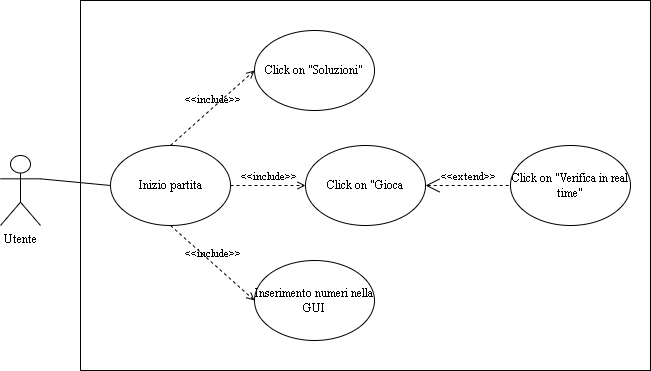
Si assume che, per creare una nuova partita o cambiare la dimensione della griglia una volta superata la schermata “Impostazioni partita” vada riavviata l’applicazione.

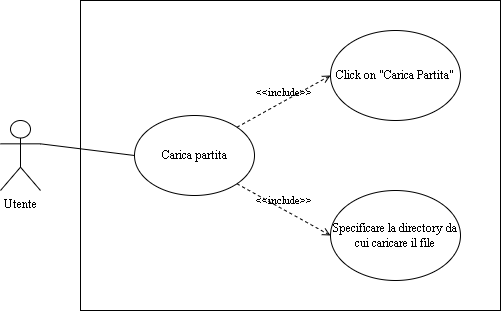
***A.6 Use Case Diagrams***

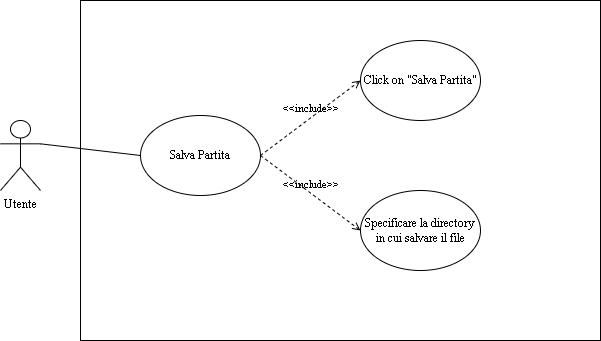
******







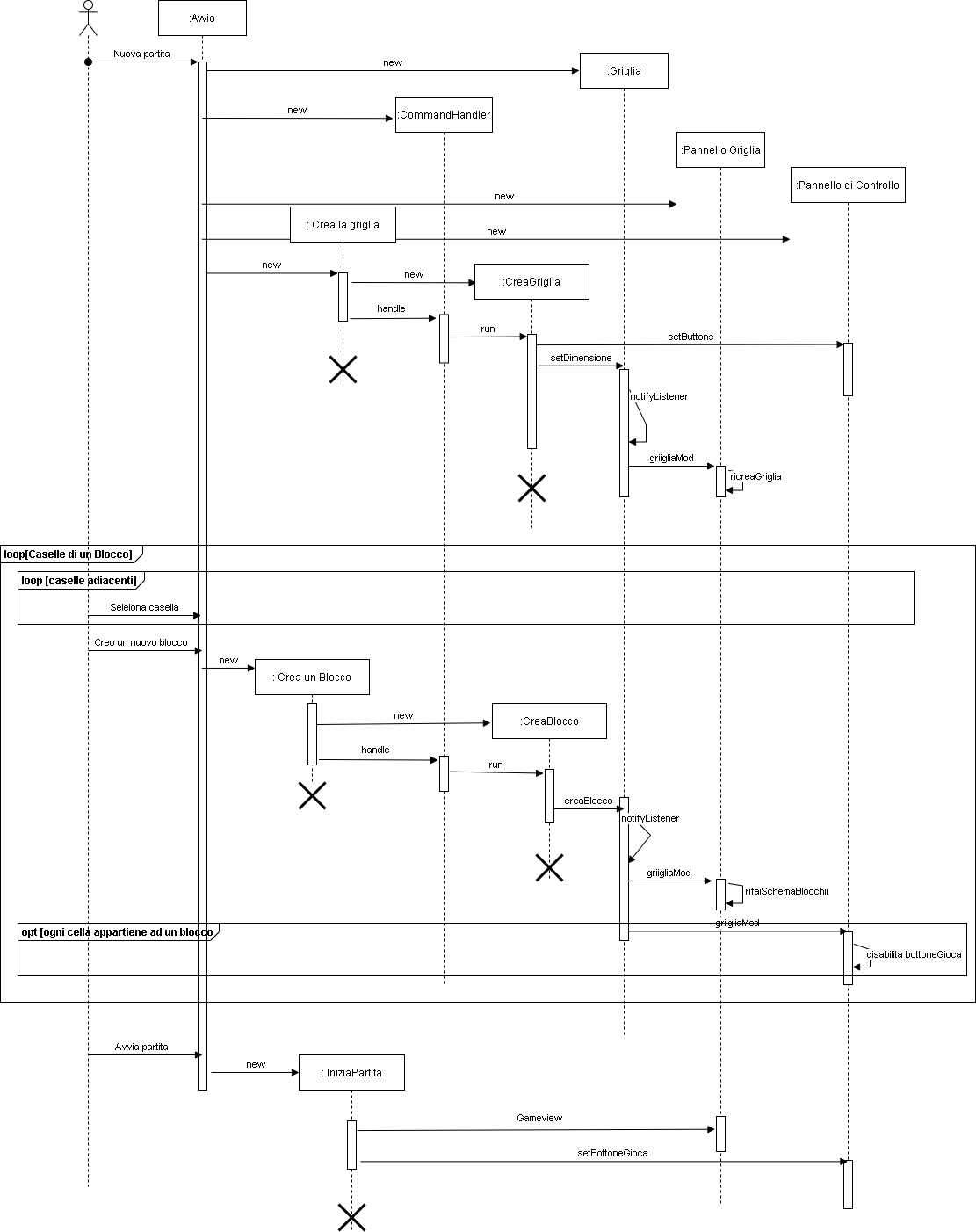


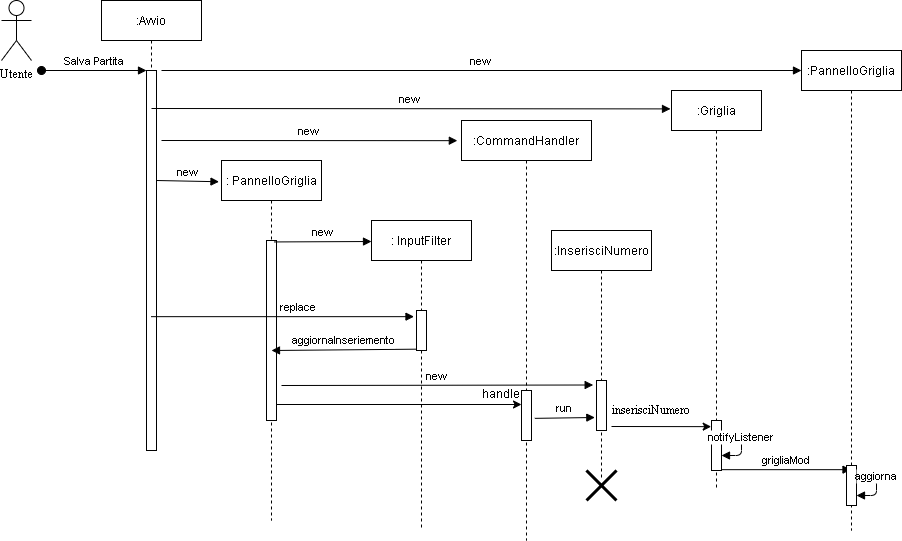


C. Architettura Software

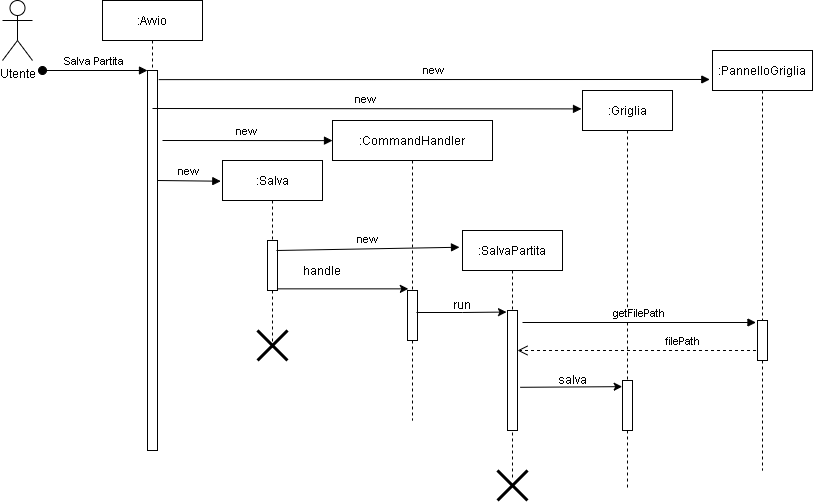
***C.1 The static view of the system: Component Diagram***

***C.2 The dynamic view of the software architecture: Sequence Diagram***

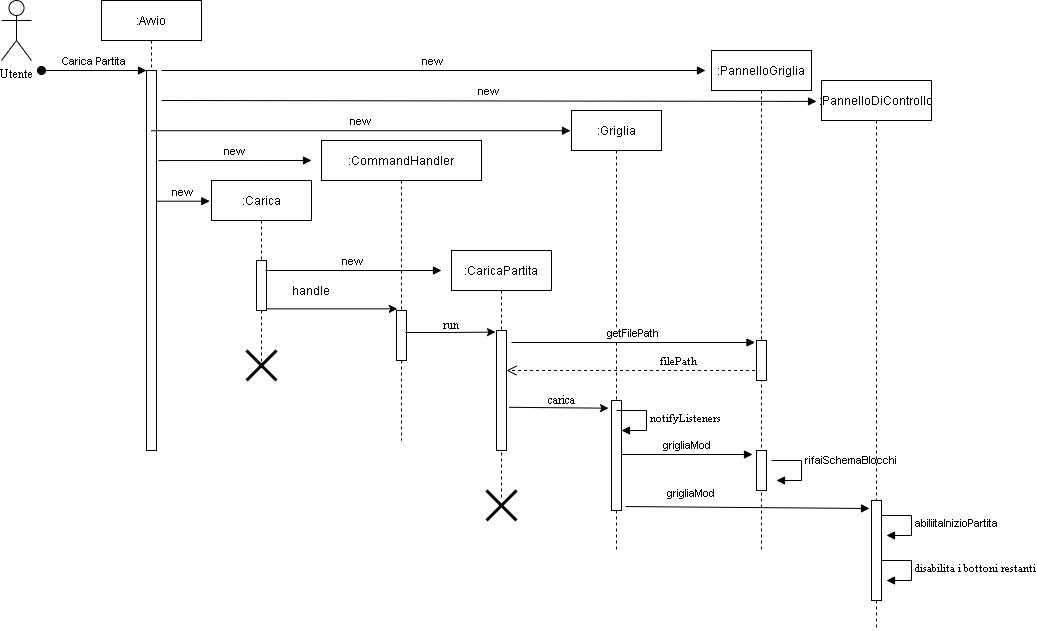
Configurazione e inizio di una partita

Inserimento di un numero nella griglia di gioco

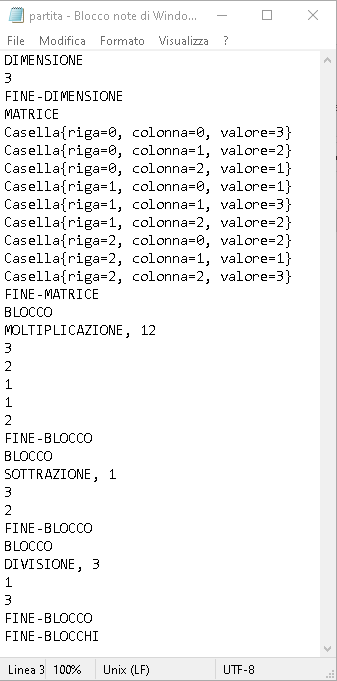
Salvataggio di una partita

******

Caricamento di una partita

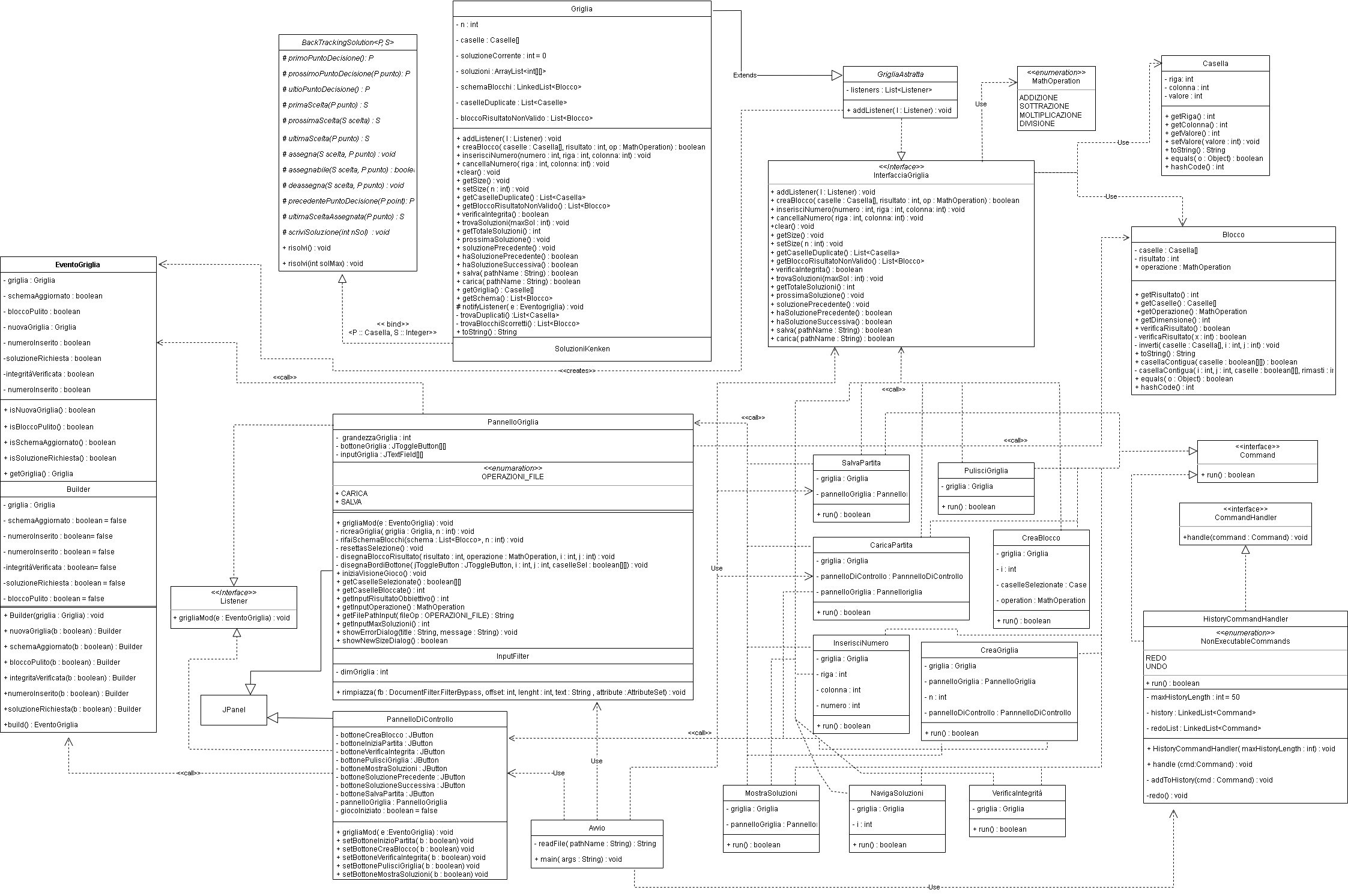


D. Dati e loro modellazione (se il sistema si interfaccia con un DBMS)



L’applicazione memorizza i dati in file.ken. Nella classe *Griglia* sono stati definiti i metodi salva(String pathName) e carica(String patName), che strutturano i file di salvataggio. I file possono essere aperti tramite editor di testo quindi possono essere facilmente corrotti, la loro forma è pressappoco la seguente:

E. Scelte Progettuali (Design Decisions)   
<Document here the **5** most important design decisions you had to take. You can use both a textual or a diagrammatic specification.>

Di seguito verrà mostrato il Class Diagram dell’applicazione

L’applicazione è supportata da alcuni Design Pattern presentati durante le lezioni, ovvero, *Observer, Template Method, Command e Builder*. Lo scopo di questi DP verrà illustrato al punto successivo.

F. Progettazione di Basso Livello

Design pattern *Template method* per la risoluzione delle partite di Kenken tramite la tecnica Backtracking.

La classe astratta BacktrackingSolution<P, C> implementa una versione iterativa della tecnica backtracking nel metodo template risolvi (). Tutti i metodi concreti necessari alla risoluzione del gioco KenKen, sono implementati nella inner class concreta SoluzioniKenken, situata dentro la classe Griglia.

Tramite il design pattern Template Method, la classe astratta BacktrackingSolution<P, C> imbastirà una struttura preliminare volta alla risoluzione della griglia di gioco. Le soluzioni trovate, verranno memorizzate e rese disponibili.

Design pattern *Command* come interfaccia per i diversi comandi della griglia.

Le modifiche del modello, ovvero del gioco, non sono eseguite direttamente dai Controller che compongono la vista ma sono incapsulate ed inoltrate ad un CommandHandler sotto forma di Command. Gli oggetti di tipo Command contengono le operazioni necessarie all’esecuzione di un particolare comando e sono gli unici oggetti tramite cui la vista può apportare modifiche al modello. Il ciclo di vita di un comando, però, non è lineare; i controller inviano i comandi ad un HistoryCommandHandler che memorizza uno storico dei comandi e volge la funzione di Invoker.

Design pattern *Builder* serve come costruttore della classe EventoGriglia.

La classe EventoGriglia è dotata di diverse variabili locali che andranno fornite alla classe Griglia. Queste variabili forniscono dati sulle operazioni svolte sulla griglia, ma non saranno sempre necessarie alla classe Griglia, perciò, per scongiurare il problema dei costruttori telescopici, la classe EventoGriglia verrà creata con l’ausilio di un Builder, fornendo solo i dati necessari.

Design pattern *Observer* per l’interazione con la griglia di gioco.

Ogni volta che l’utente, tramite i pulsanti forniti, interagisce con la GUI, questa notificherà il cambiamento e aggiornerà la GUI. Il vantaggio principale tratto dall’uso del pattern Observer è, appunto, il meccanismo di notifica. È ampiamente sfruttata l’interazione publish-subscribe: ogni qual volta il gioco cambia stato, viene notificato agli osservatori che seguendo un push model riceveranno tutte le informazioni necessarie dal soggetto.

G. Spiegare come il progetto soddisfa i requisiti funzionali (FRs) e quelli non funzionali (NFRs)

**A1. Impostazioni di gioco**

Per impostare la propria partita a Kenken, l’utente dovrà attraversare tutte le finestre di cui è composta l’applicazione.

Nella schermata iniziale “Kenken Menu” l’utente cliccando sul pulsante “Nuova Partita”, verrà orientato sulla seconda schermata “Impostazioni Partita”, qui, dovrà scegliere le dimensioni della propria griglia e successivamente cliccare sul tasto “Conferma” per creare, in una schermata successiva, una griglia vuota delle dimensioni desiderate.

A questo punto ci troveremo nell’ultima schermata denominata “Kenken” nella quale sarà presente la griglia. Per configurare i blocchi, la classe *Griglia* mette a disposizione il metodo *createBlocco(),* tramite il quale è possibile specificare una lista di caselle, basterà selezionare un numero congruo di caselle (almeno una) con il mouse, un risultato da ottenere e l’operazione aritmetica con cui combinare i valori all’interno del blocco e premere il tasto “Aggiungi Blocco”.

**A2. Inizio partita**

Dopo aver configurato correttamente la griglia, premendo il tasto “Gioca” inizierà la partita. L’utente potrà inserire in griglia i numeri da tastiera, ciò è possibile poiché ogni singola casella è un JTextField. Per rimuovere un numero basterà banalmente cancellarlo da tastiera, invece, per la rimozione di tutti i numeri dovrà essere cliccato il tasto “Pulisci Griglia”.

**A3. Visualizzazione delle soluzioni della partita**

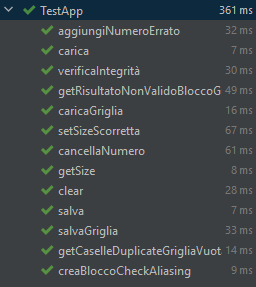
Durante la partita l'utente può scegliere di visualizzare sulla griglia corrente le soluzioni. Cliccando su visualizza soluzioni il software ne chiederà all'utente il numero desiderato. In seguito verranno abilitati i pulsanti di navigazione attraverso cui l'utente può navigare nello spazio delle soluzioni mostrandole a schermo. Tale azione comporta la creazione di un oggetto della classe astratta BackTrackingSolution con conseguente invocazione del metodo risolvi sullo stesso. Quest’ultimo segue la tecnica Back Tracking iterativa, assegnando e deassegnando opportunamente i valori qualora questi non rispettino le condizioni dei blocchi e la non ripetizione sulla stessa riga e/o colonna.

N.B Una griglia del gioco Kenken, se composta correttamente, presenta un’unica soluzione, dunque è probabile che venga mostrata solo una soluzione

**A4 Salvataggio e caricamento della partita**

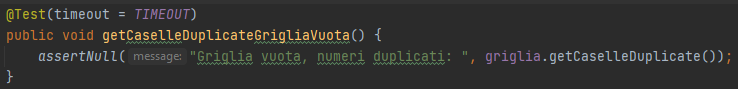
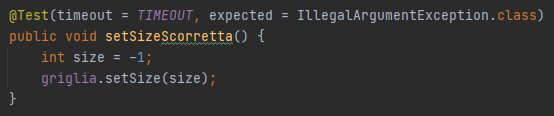
Tale requisito è stato ricoperto finendo rispettivamente i metodi salva (accessibile tramite il tasto “Salva” nella schermata “Kenken”) e carica (accessibile tramite il tasto “Carica Partita” nella schermata “Kenken Menu”) all'interno dell'applicazione. Questi metodi utilizzando le classi ObjectOutputStream e ObjectInputStream della libreria java.io per scrivere e leggere i file.Per la scelta della directory di salvataggio o caricamento viene istanziato un j filechooser che consente tramite un Explorer grafico di navigare all'interno del file system.

Appendix. Prototype   

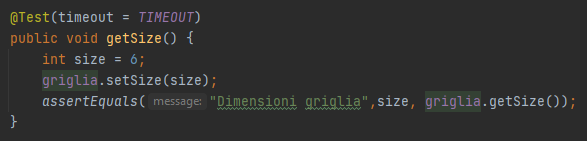
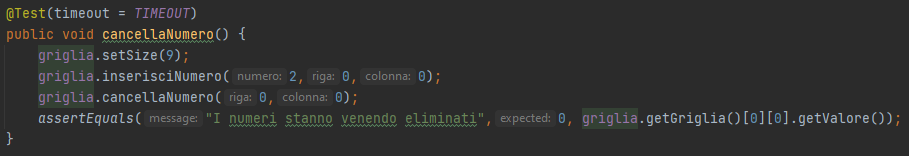

**Per verificare la robustezza dell’applicazione, è necessario testarla sottoponendola ad alcuni test strutturati appositamente utilizzando il framework JUnit.

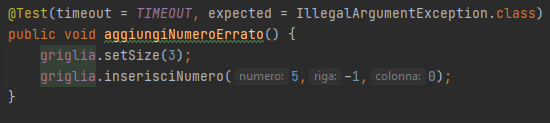
Per far ciò abbiamo creato una classe test denominata TestApp e abbiamo imposto un tempo di esecuzione per ciascun test di massimo 3 secondi, questa misura è stata considerata per verificare la presenza di potenziali deadlock e loop infiniti che comprometterebbero l’utilizzo dell’applicazione.

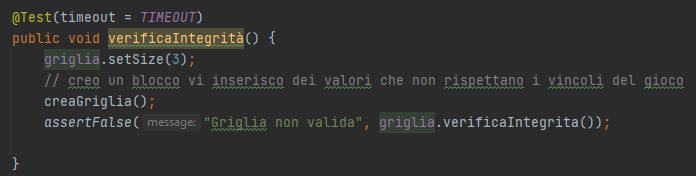
Come si può vedere nell’immagine qui a fianco, tutti i test sono andati a buon fine.



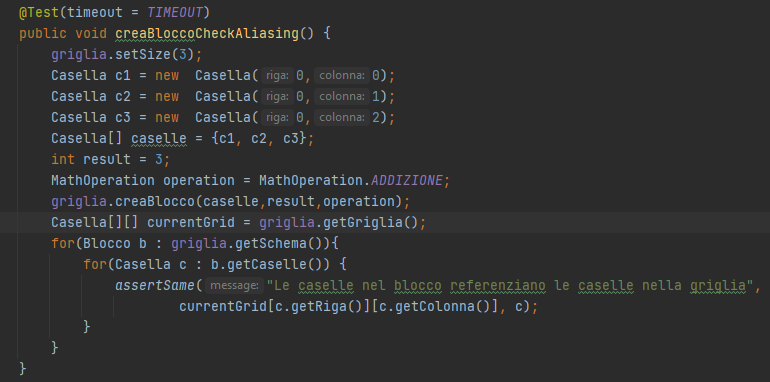
Il test setSizeScorretta() verifica che l’applicazione sollevi l’eccezione IllegalArgumetnException a seguito dell’inserimento di una dimensione negativa.

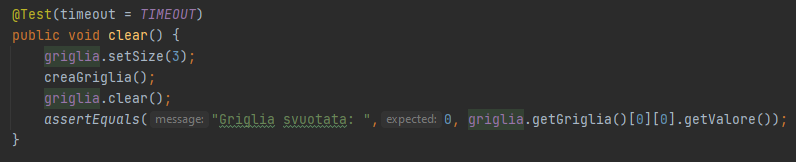


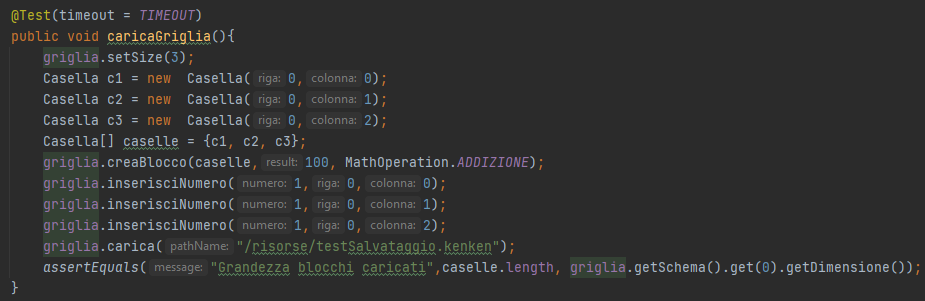
aggiungiNumeroErrato() verifica che, a seguito di un inserimento di un numero scorretto, l’applicazione sollevi un IllegalArgumetException (con una dimensione di 3, i numeri che possono essere inseriti in griglia sono 1,2,3)

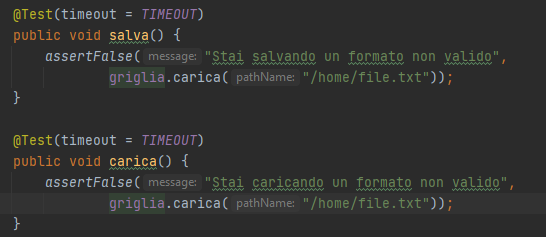


verificaIntegrita() verifica la correttezza degli inserimenti nella griglia durante la partita

Tramite il test creaBloccoCheckAliasing() verifico l'aliasing tra le celle della griglia di gioco e le celle del blocco partendo da un’istanzazione all’interno del test stesso





 **

I test: salvaGriglia(), caricaGriglia(), salva() e carica() servono a verificare la correttezza del processo di caricamento e salvataggio di una griglia, ciò avviene accertandosi che la griglia al momento del caricamento sia nello stesso stato della griglia subito dopo essere stata salvata.