**RougueKill**

****

***Rougue Kill*** è un 2D top-down shooter.

***Plot***

Il giocatore riveste, in questo gioco, i panni di un Generale che combatte elementi controllati dalla I.A. (prevalentemente soldati traditori) e uccide il loro capo a bordo di un ‘ aeronave.

**Gameplay**

In ogni piano (o livello che dir si voglia) bisogna attraversare e combattere le ondate di nemici per poi trovarsi di fronte al boss, e una volta sconfitto questo, Il livello finisce. Il giocatore inizialmente ha 100 di vita e 50 Proiettili. Questi potranno essere recuperati tramite la raccolta di consumabili all'interno della mappa. I soldati nemici hanno invece 30 di vita e possono sia sparare come il protagonista sia fargli del male nell'attacco corpo a corpo. Il mondo in cui si svolge il video gioco può essere sia generato proceduralmente sia creato tramite l'apposito editor.

**Comandi**

Il protagonista si muove usando le frecce direzionali e spara tramite il tasto B.

Per andare al menù di pausa e tramite questo Salvare/Caricare e/o tornare al menù principale basta premere il tasto ESC mentre si è in partita.

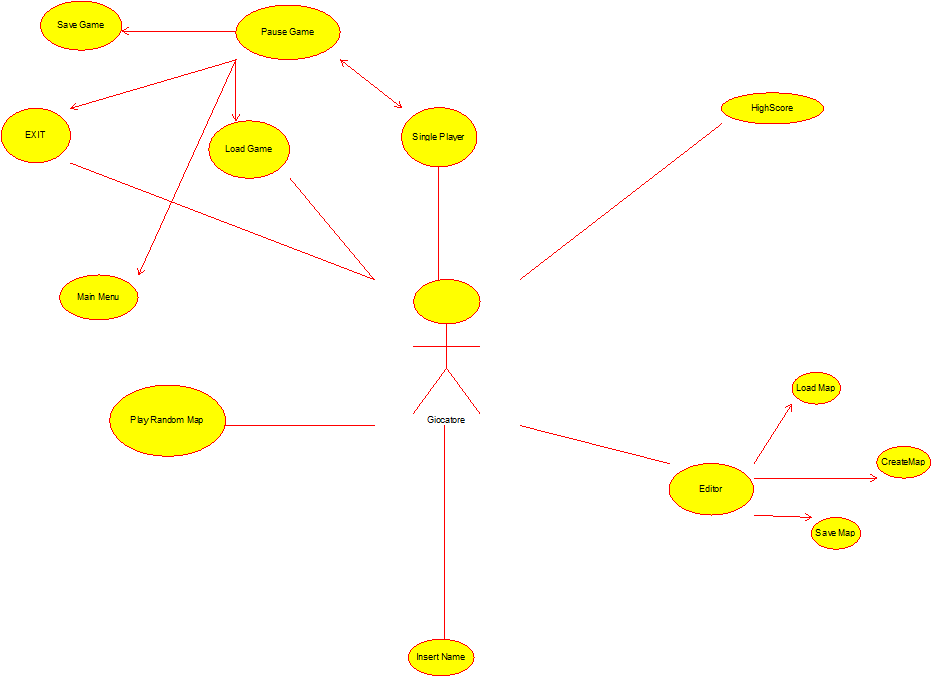
***Stats Online***

Una volta completato un livello il giocatore riceve un punteggio, che verrà salvato sul cloud di APP42.

***Hall of Fame***

Nella schermata principale di RougueTeam il Giocatore potrà selezionare la voce "HIGHSCORES", così facendo potrà visualizzare la TOP 10 dei punteggi di RougueKill.

**USE CASE DIAGRAM**

****

**Single Player**

Permette al giocatore di avviare istantaneamente una partita con una mappa creata proceduralmente tramite il MapCreator di RougueKill.

**Load Game**

Permette di caricare un Salvataggio di RougueKill. Estensione: file.save

**Save Game ( All' interno del gioco)**

Permette di salvare la partita in corso per poi caricarla tramite il Load Game. Estensione: file.save

**Editor**

Permette di creare una mappa di gioco di RougueKill, è possibile posizionare muri, terreno, nemici, medikit e munizioni usando il drag&drop, oppure inserendo la percentuale di muri, RougueKill provvederà a creare una mappa random per l'utente. Salva un file .map

**PlayEditorMap**

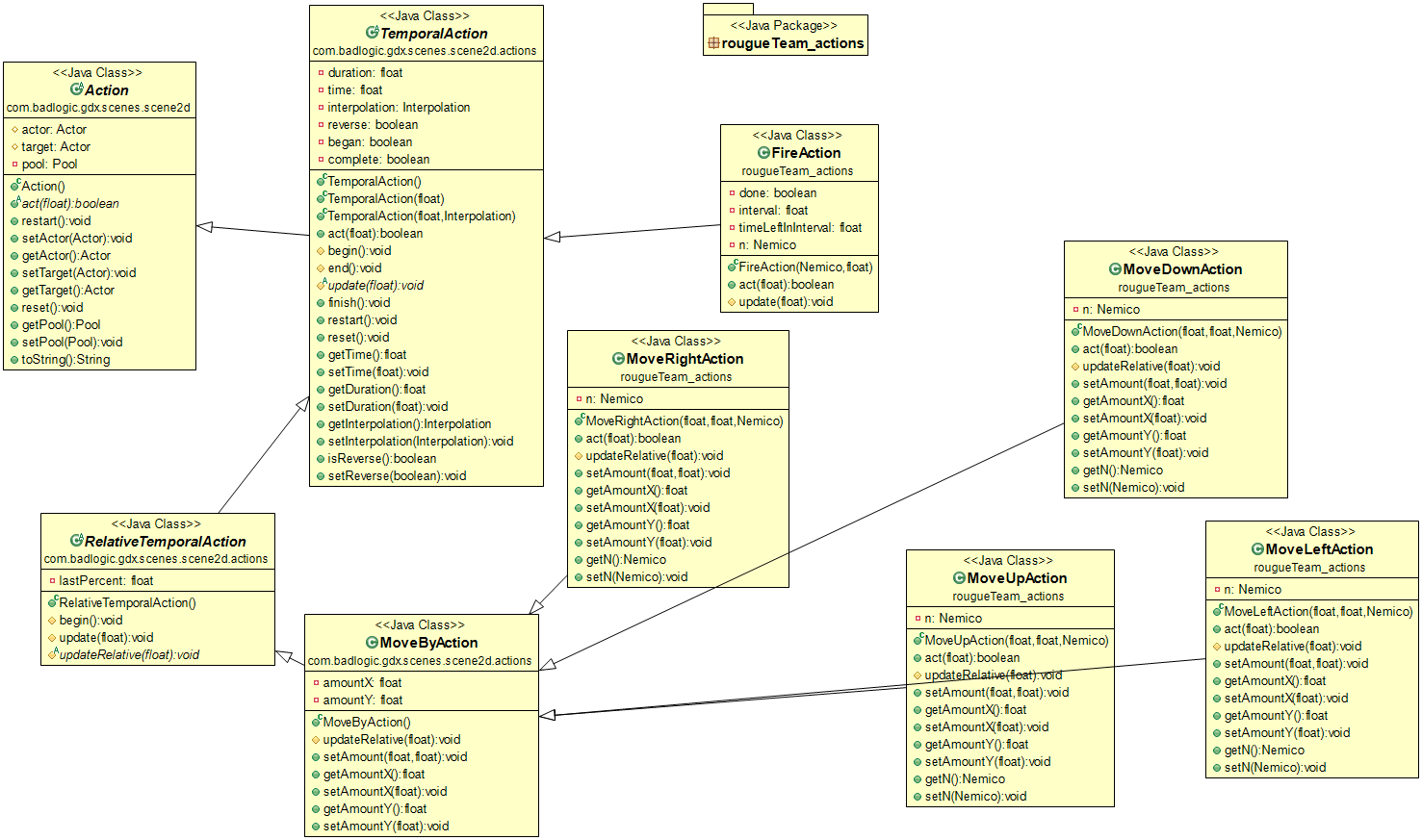
Permette di caricare e giocare una mappa creata precedentemente tramite l'editor. Carica un file .map

**Highscores**

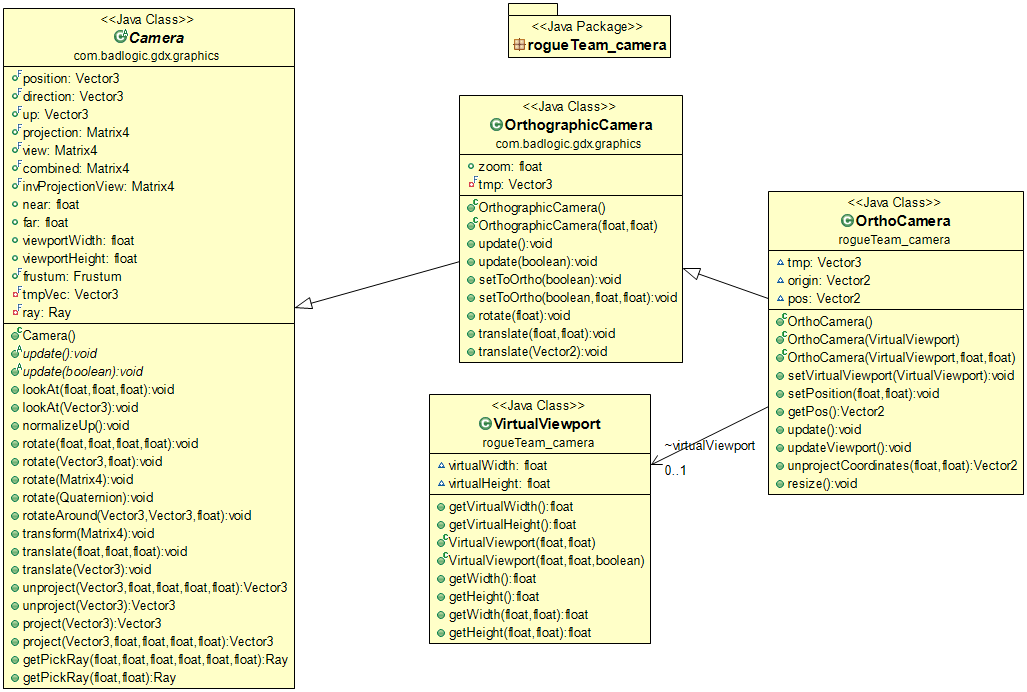
Tramite APP42, è presente online un database con i punteggi di tutti gli utenti. Selenzionando HighScores, l'utente ha la possibilità di visionare la top 10.

**Exit**

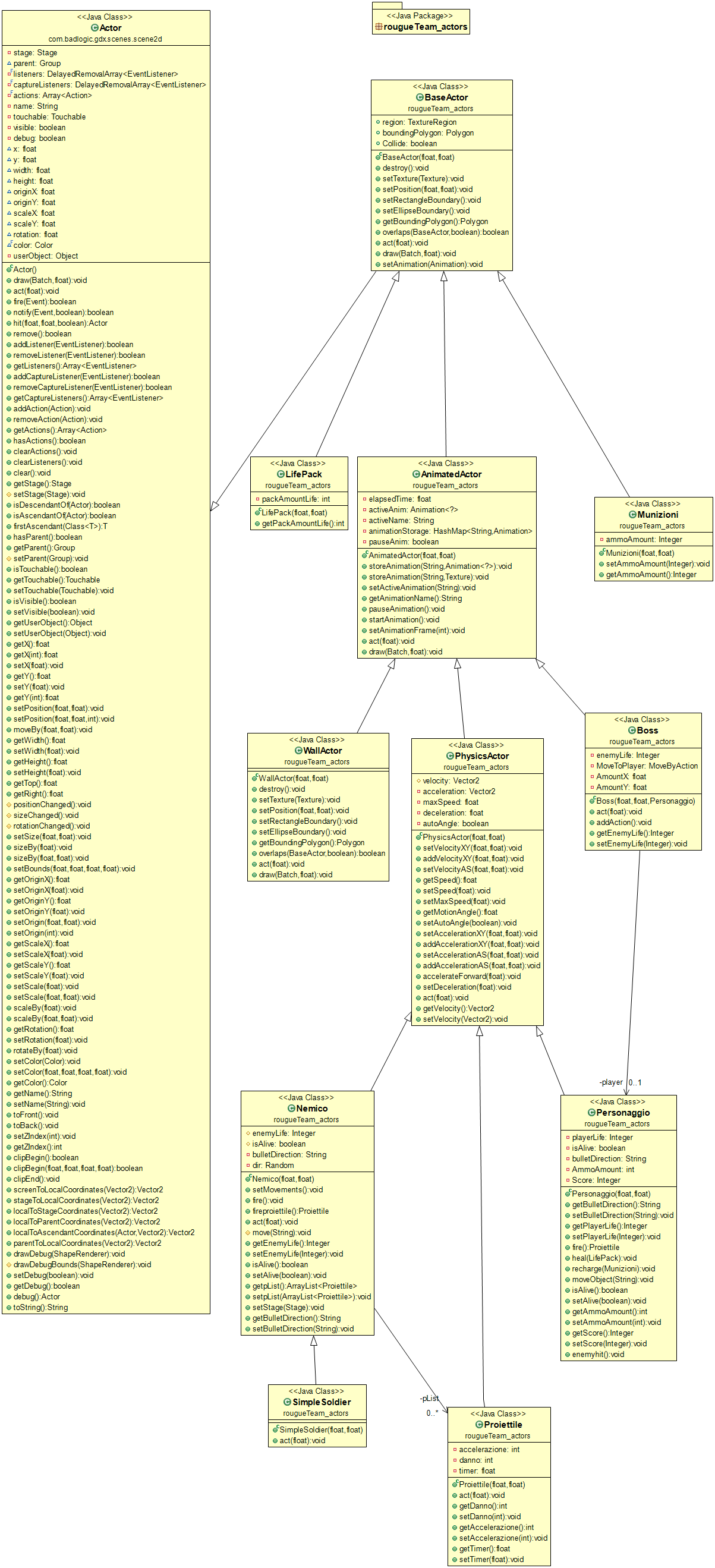
Chiude il Gioco.

**CLASS DIAGRAM**

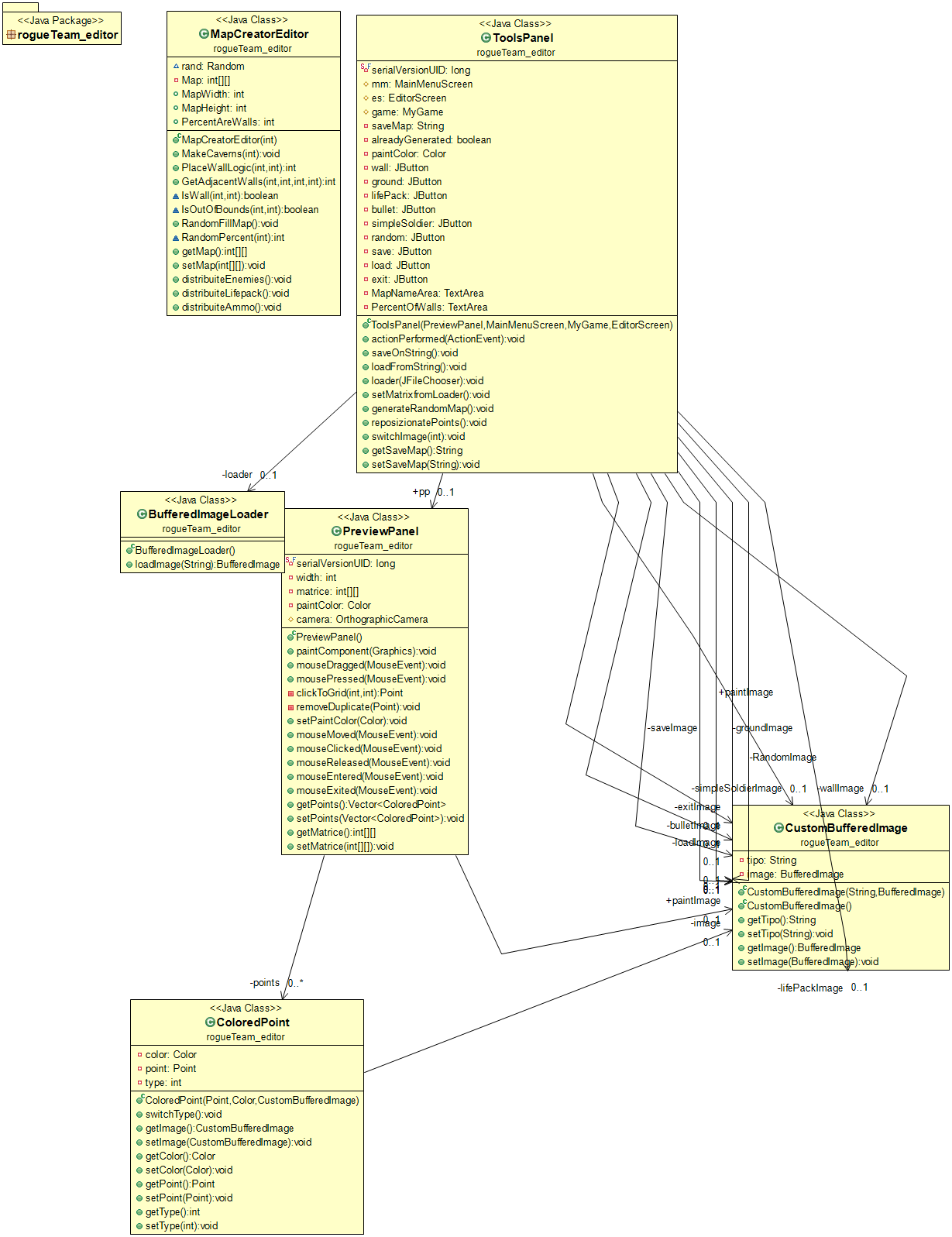
*Package rougue Team\_actions*

**

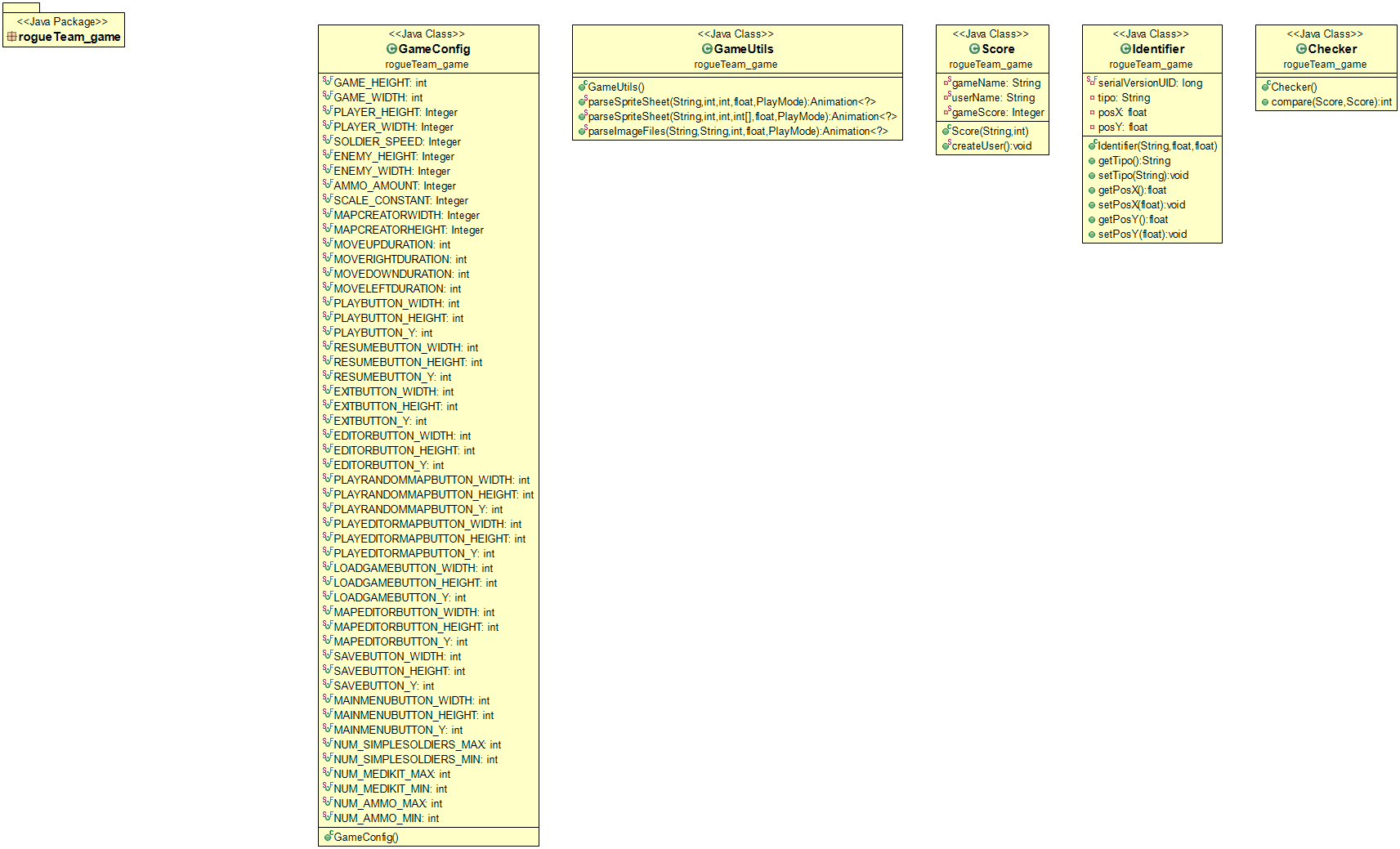
*Package Rougue Team\_camera*

****

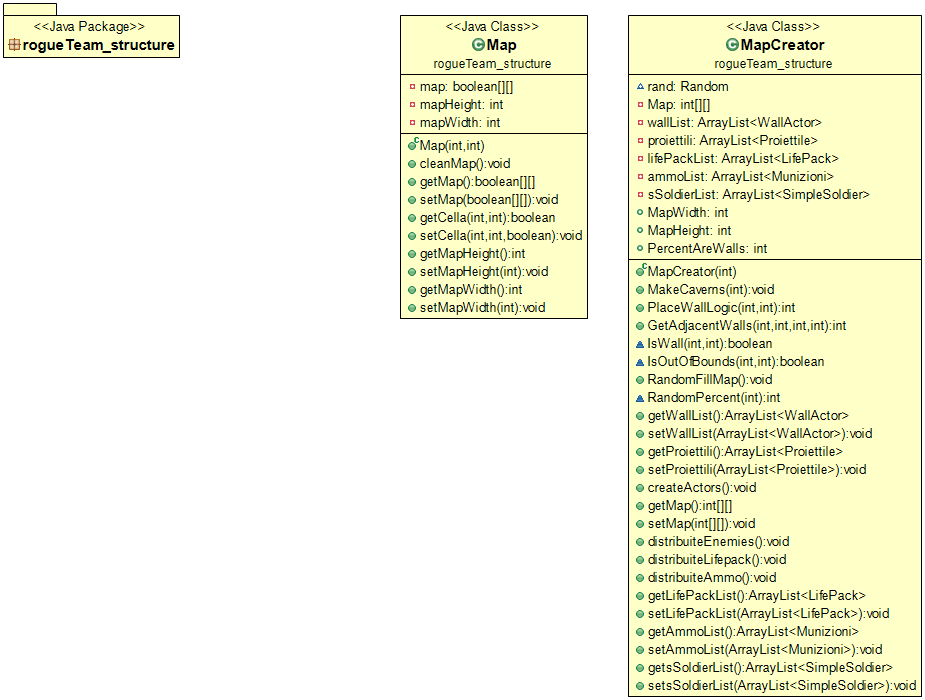
*Package Rougue Team\_actors*

****

*Package Rougue Team\_editor*

****

*Package Rougue Team\_game*

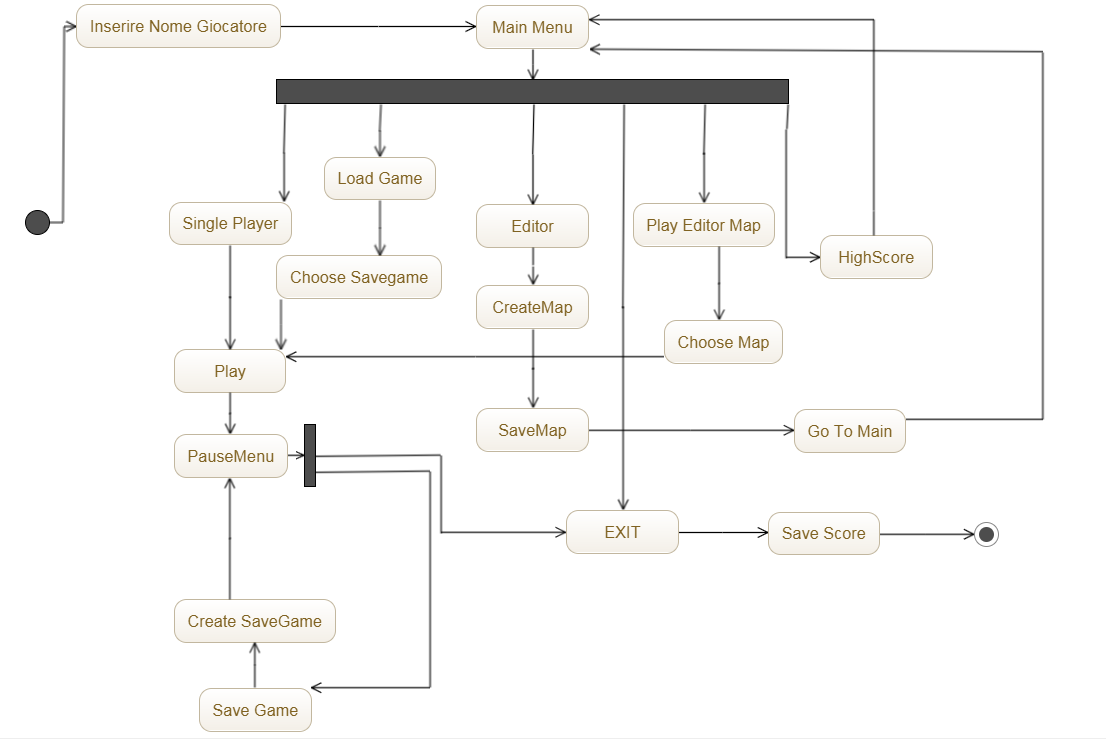
**

*Package Rougue Team\_Structure*

**

*Package Rougue Team\_Screen*

**ACTIVITY DIAGRAM**

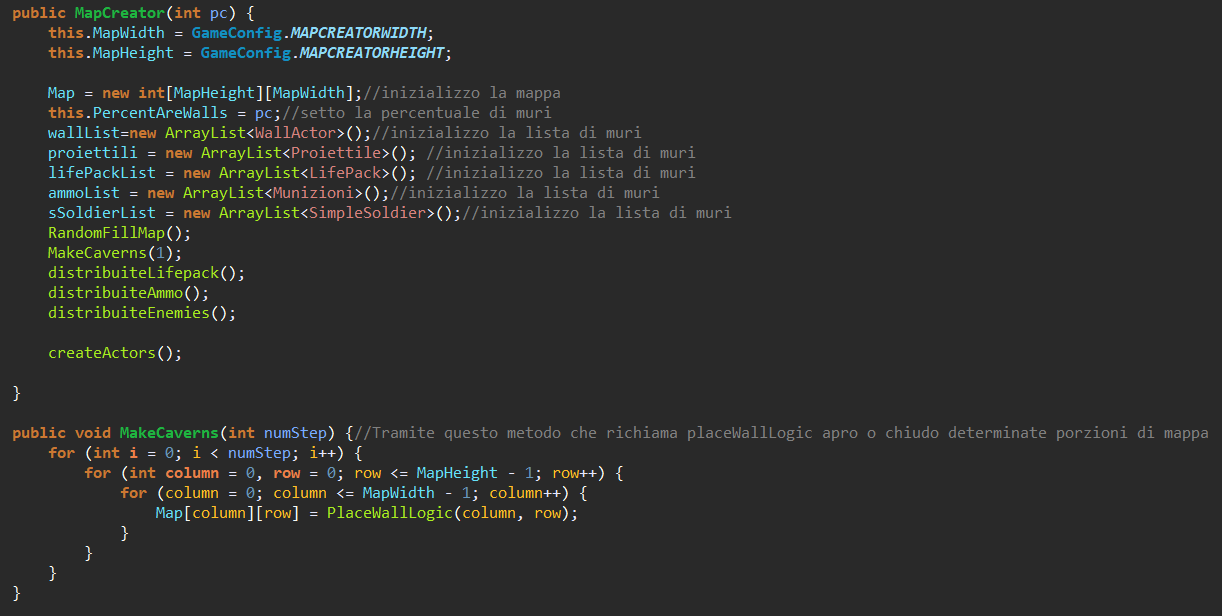
****

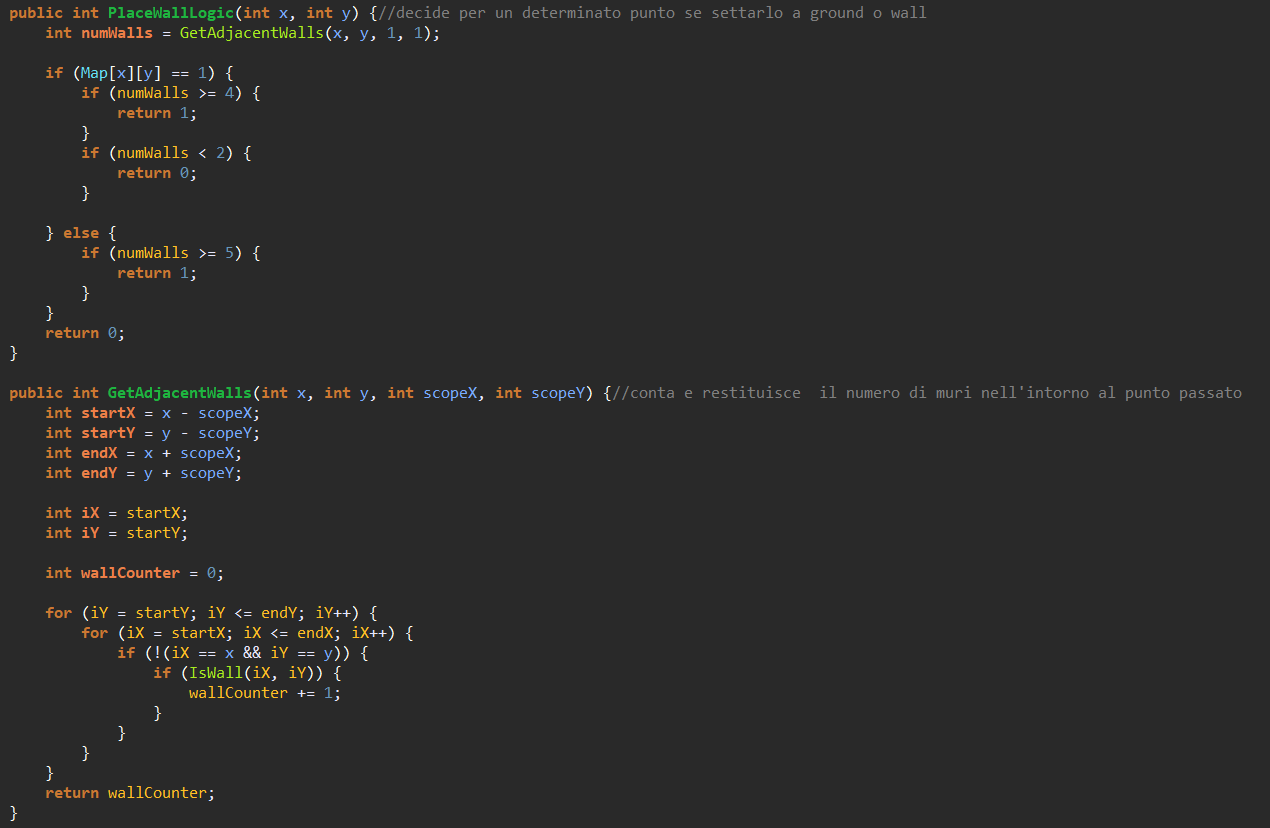
**Porzioni Significative di Codice**

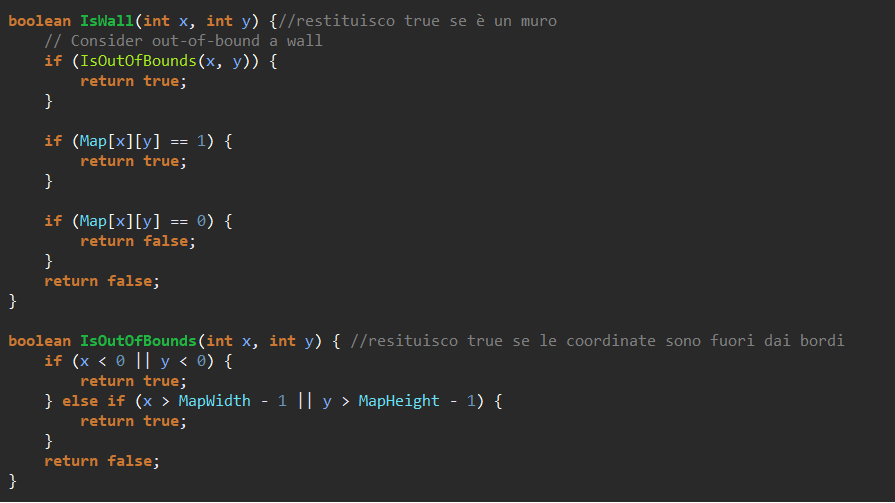
***MapCreator***

La classe MapCreator.java può essere considerata la più importante di tutto il videogioco. Essa permette la creazione procedurale della mappa in modo random e la distribuzione dei vari Nemici e Consumabili.

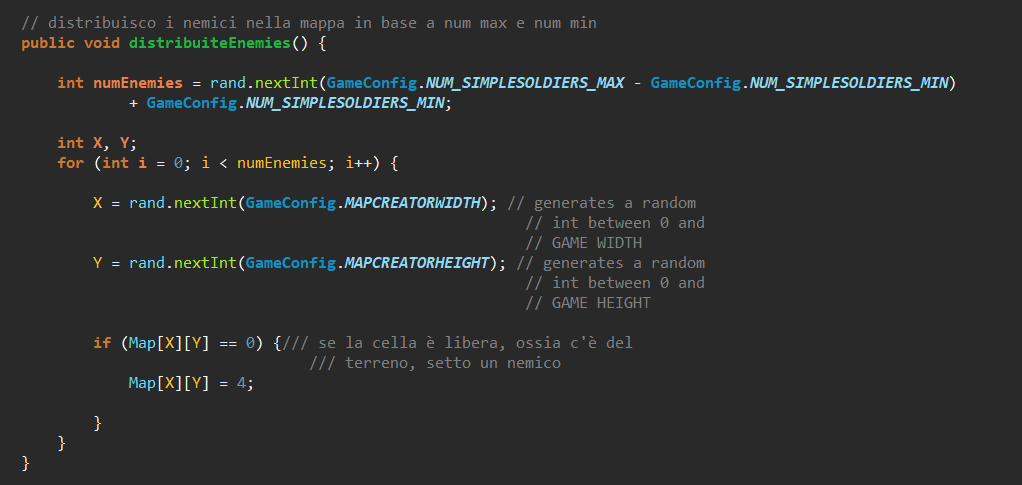
Di seguito il codice :

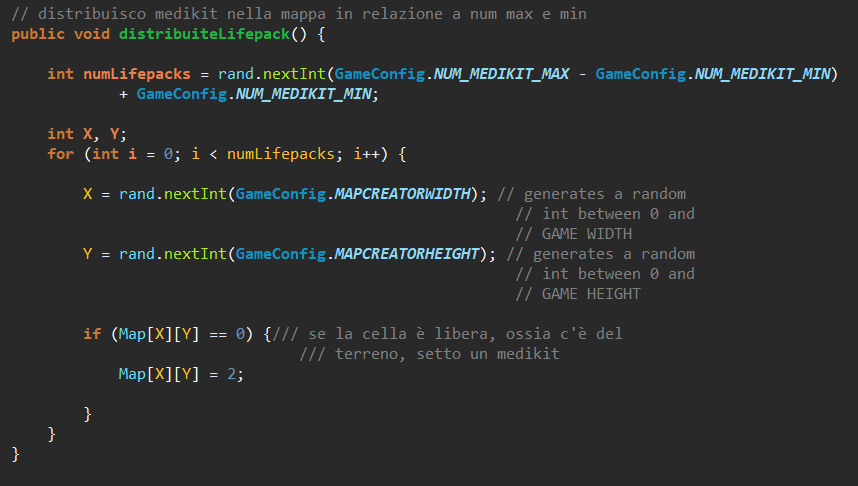


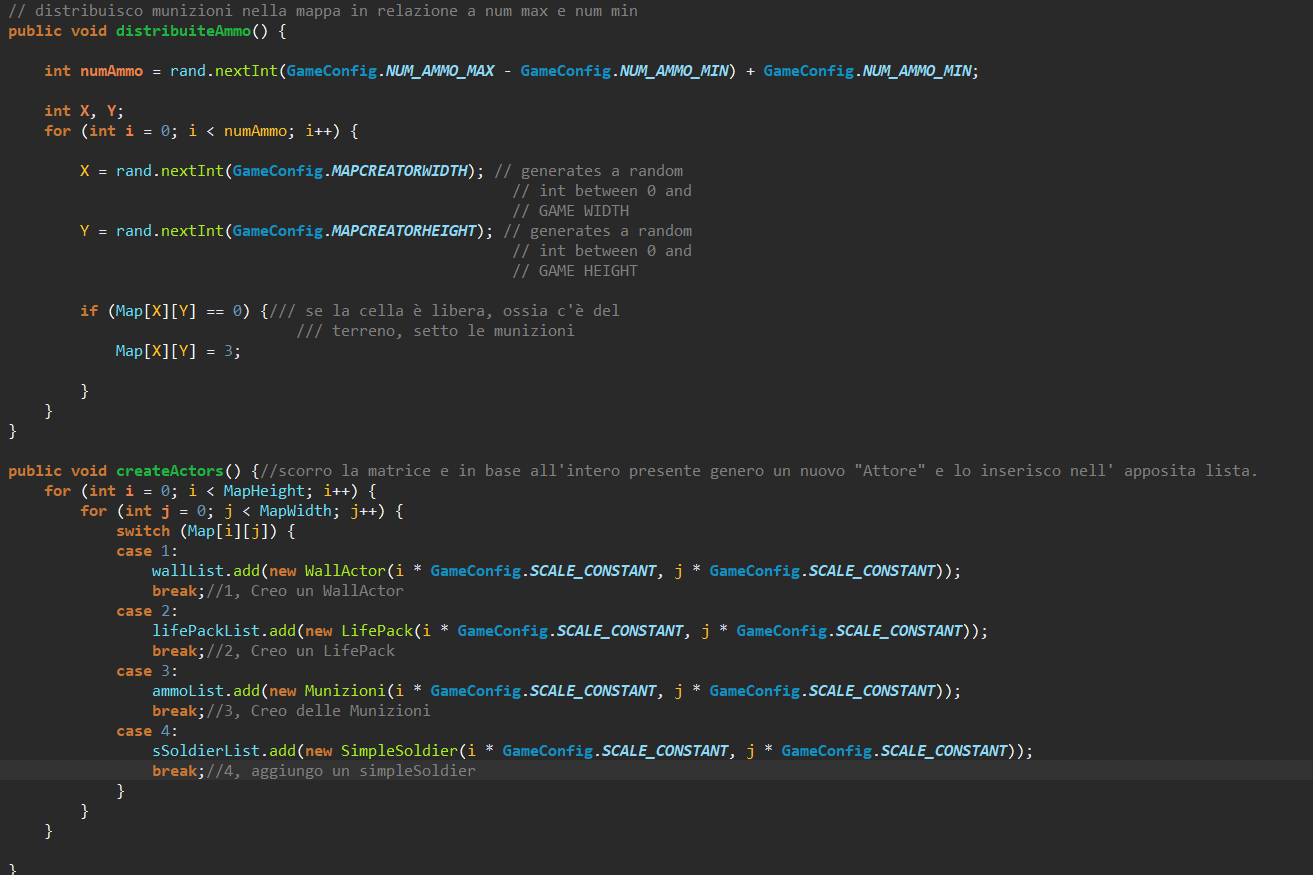












**Gestore delle Collisioni, la funzione “overlaps” di Actor.class**

“Overlaps” è un metodo che fa parte della classe BaseActor che eredita dalla classe Actor.class della libreria libGdx. Questo metodo gestisce le collisioni fra due Actor presenti sullo Stage e in input riceve, oltre all’attore (su cui controlla la collisione), un booleano. Questa variabile booleana, se è true, sposta il secondo attore al primo punto libero del vettore delle traslazioni in modo tale da non farlo collidere col primo attore.

Durante il Render dei vari screen con dei cicli di for testo overlaps fra tutti gli attori dello Stage, in modo da verificarne le collisioni.

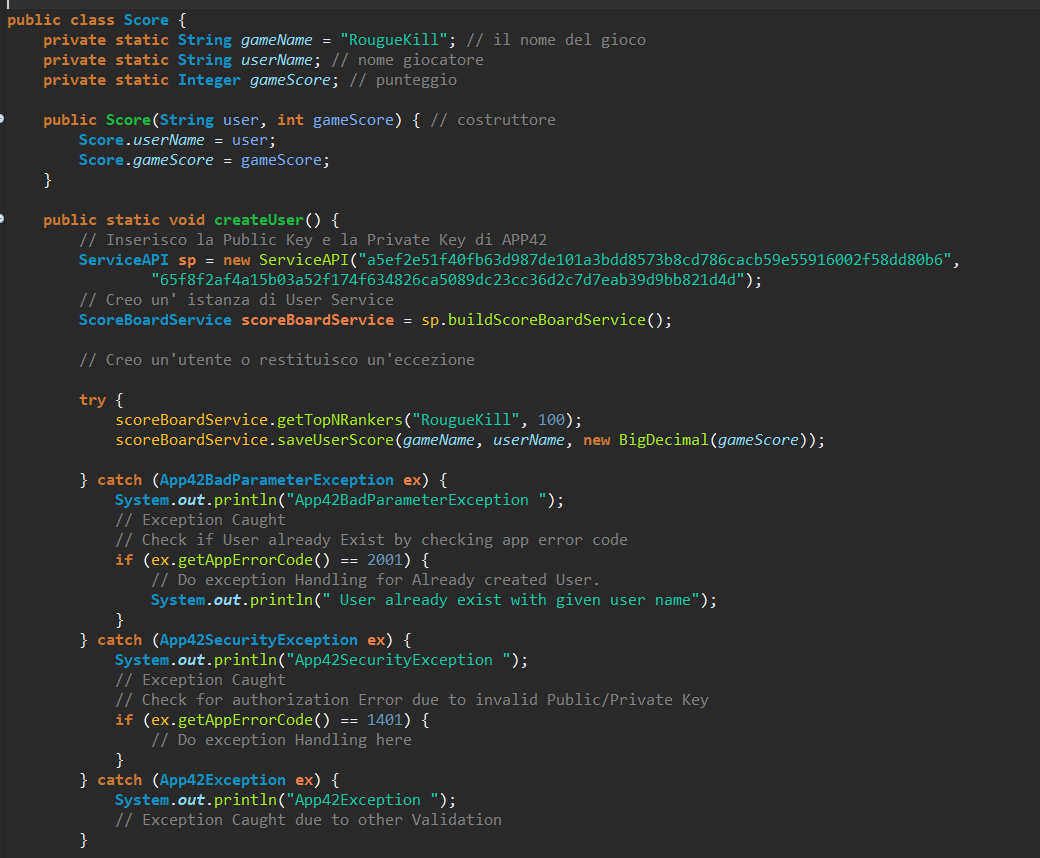
Di seguito il codice:



**Gestore dei punteggi, creazione di Score tramite l’api APP42**

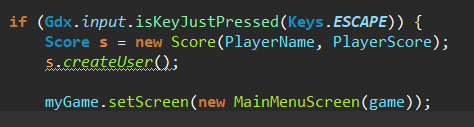
Gestisco i punteggi del gioco collegandomi al cloud di App42

Di seguito il codice:



I punteggi vengono salvati sul cloud quando :

1. Il giocatore muore.
2. La partita viene conclusa sconfiggendo il boss.
3. Quando il giocatore esce dal gioco.



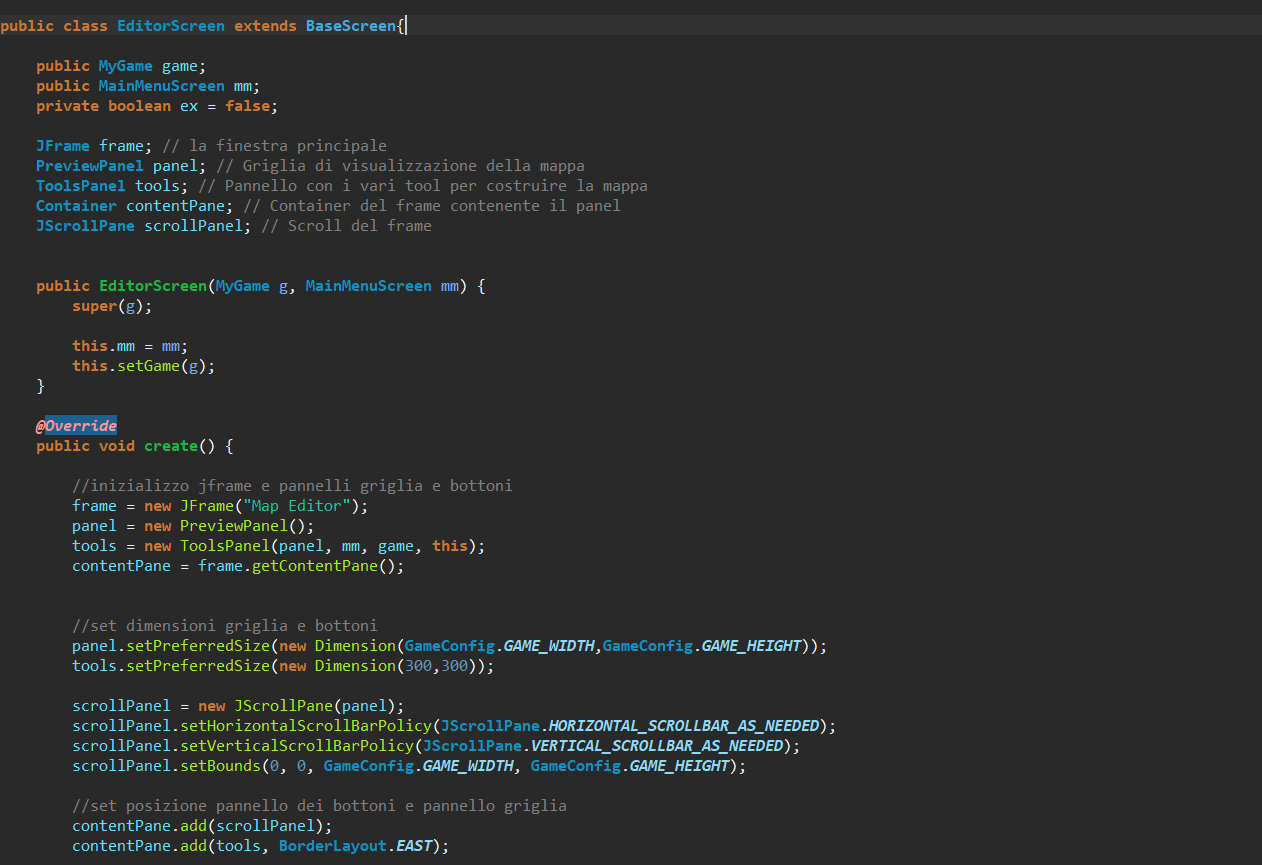
**Gestore dell’editor: EditorScreen & Panels**

Tramite l’ editorScreen vado a creare una mappa che poi potrà essere utilizzata per giocare e caricata dal MainMenù tramite la voce : “Play Editor Map”.

Il lavoro per “costruire” una mappa è svolto principalmente dall’EditorScreen , dal PrewievPanel e dal ToolsPanel.

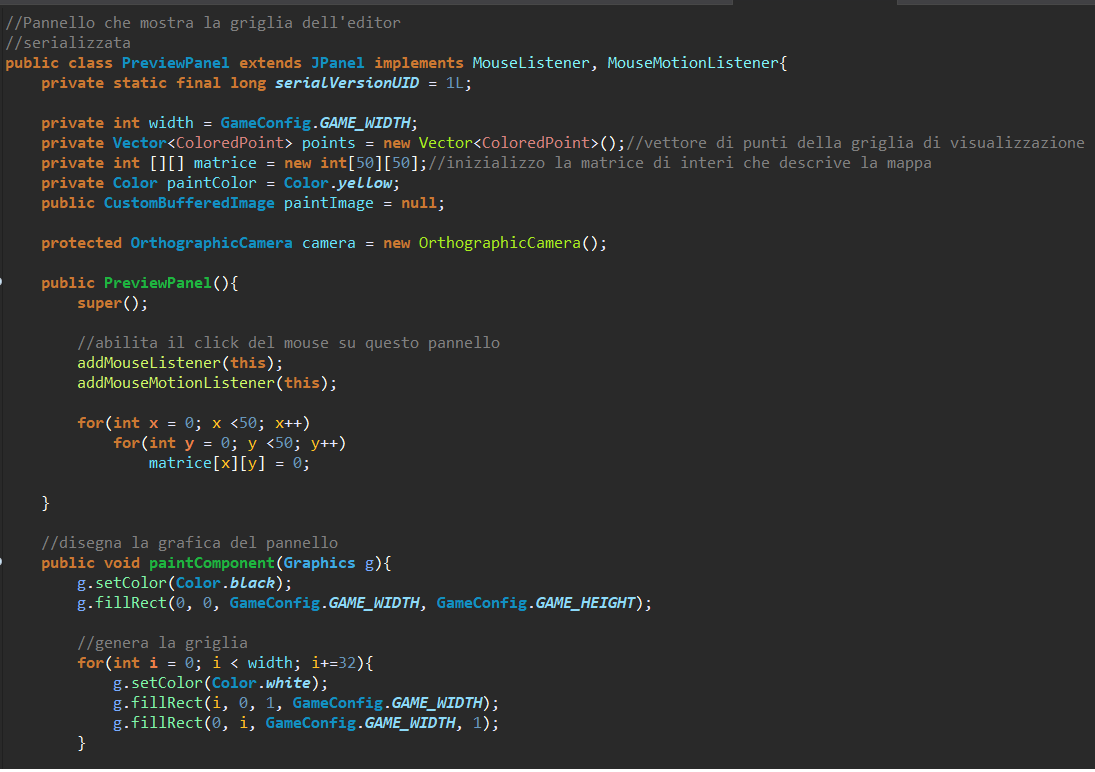
Di seguito il codice :

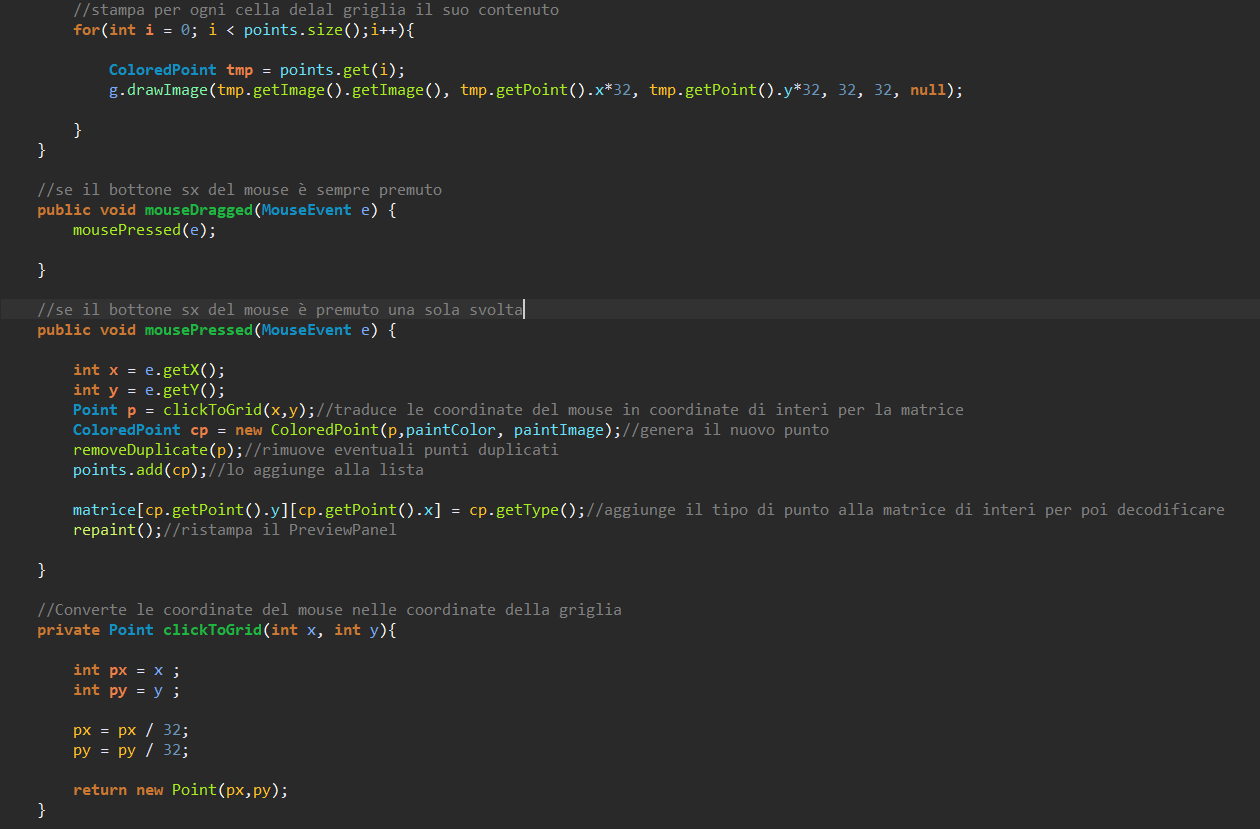
EDITOR SCREEN

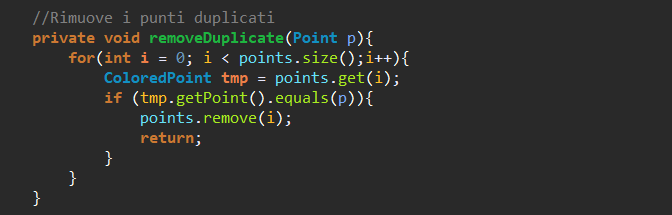




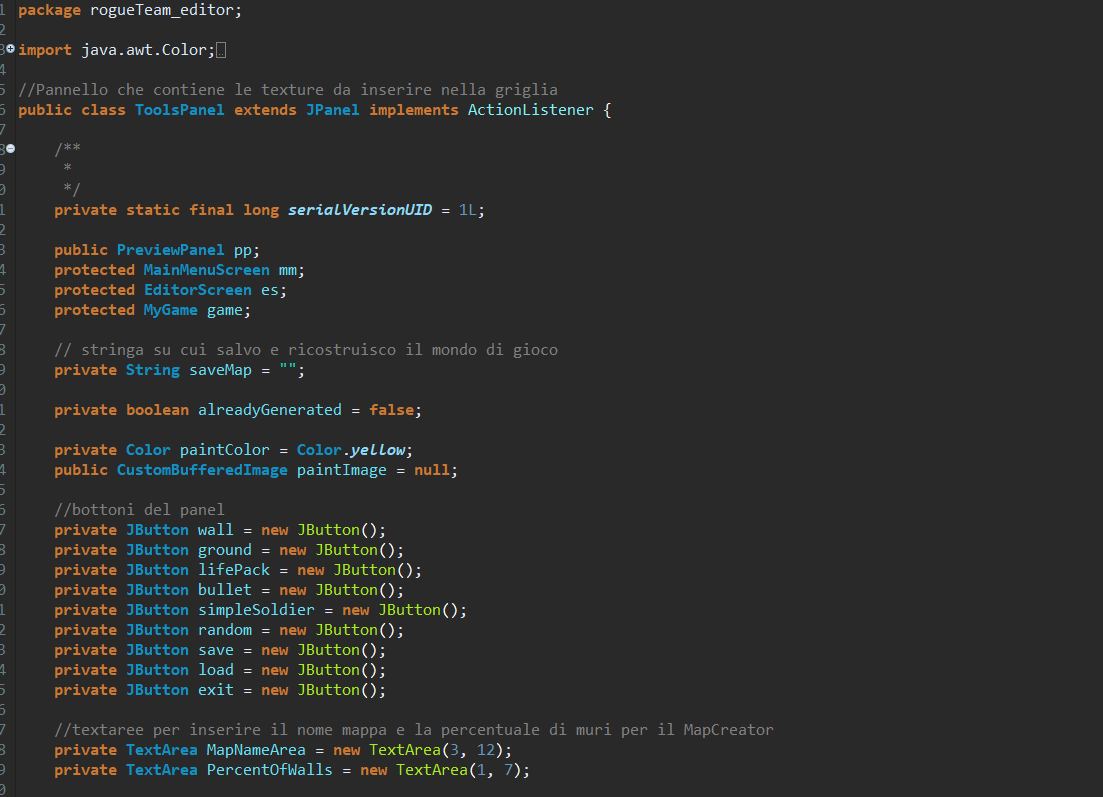
PREVIEW PANEL

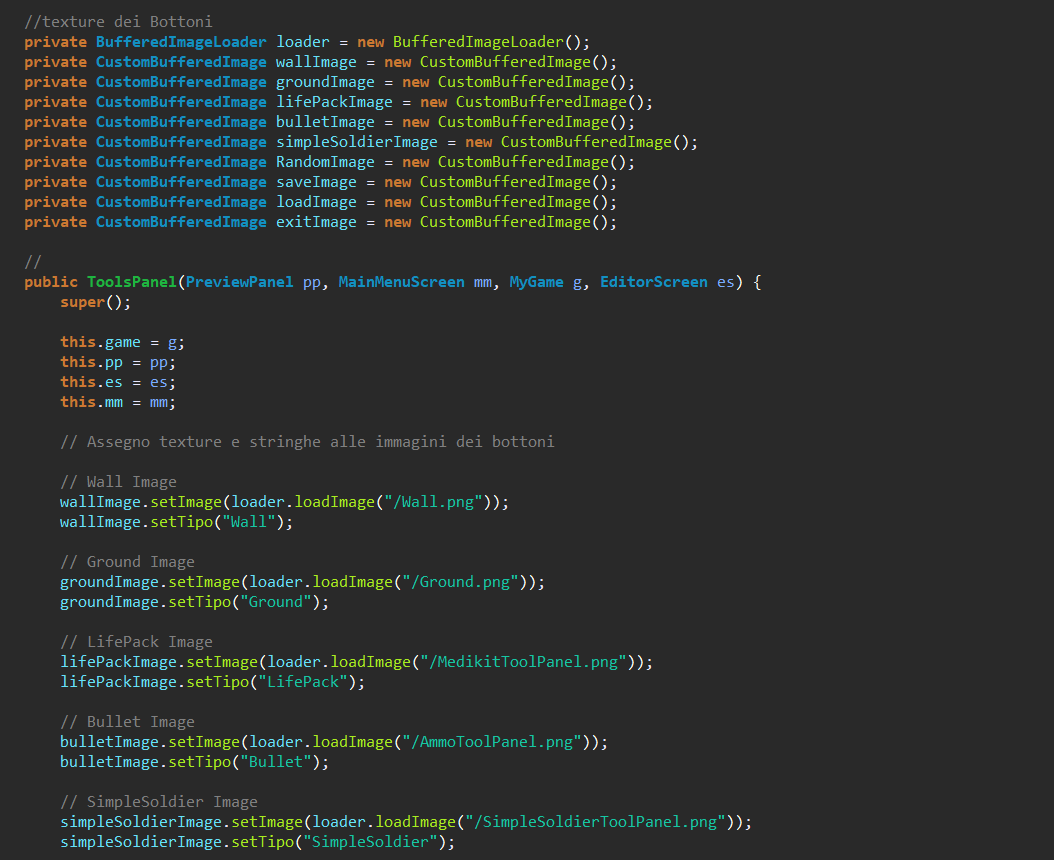


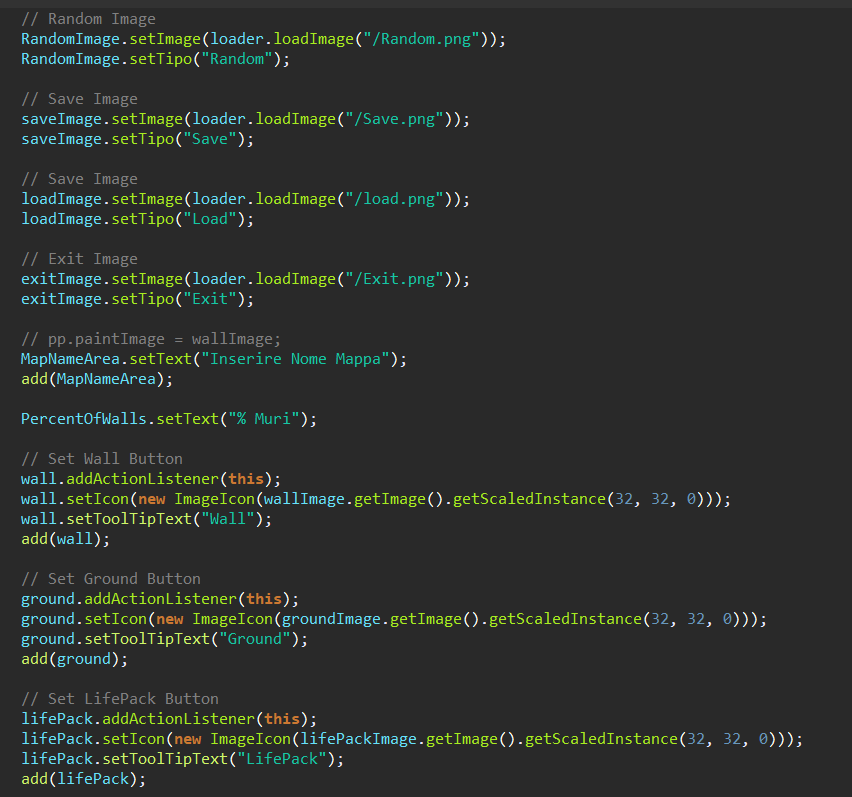




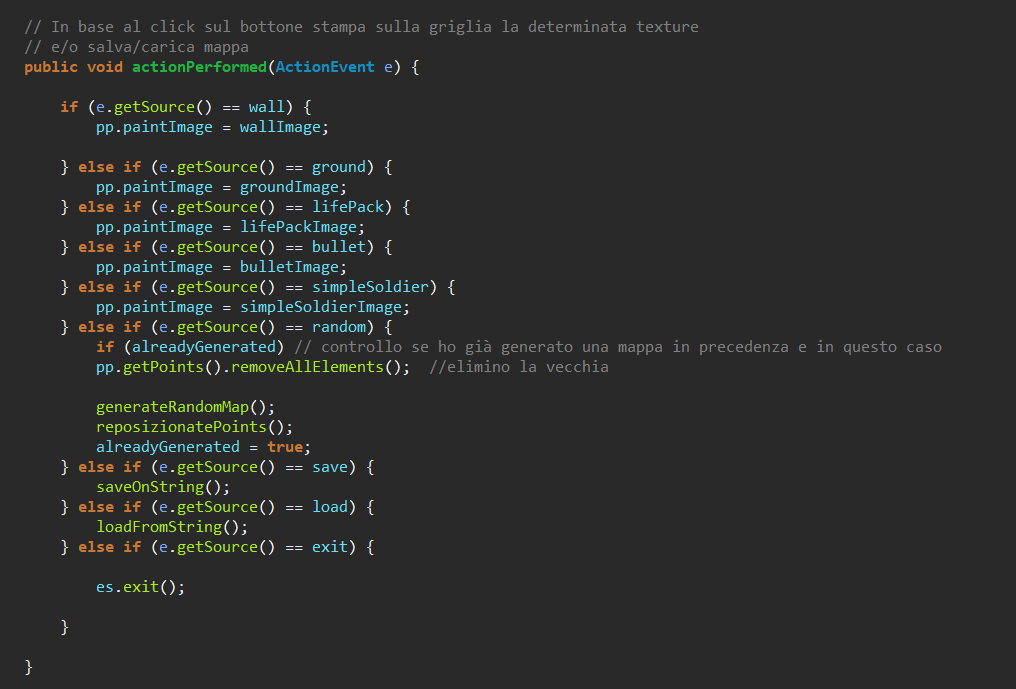
TOOLS PANEL





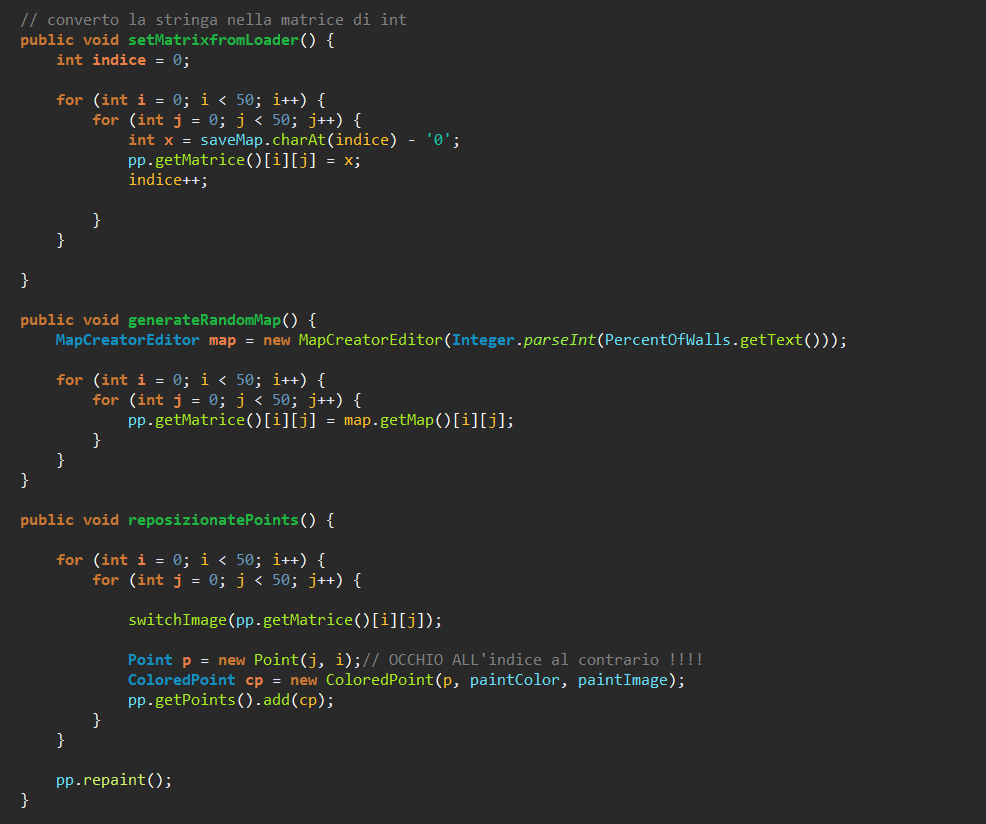


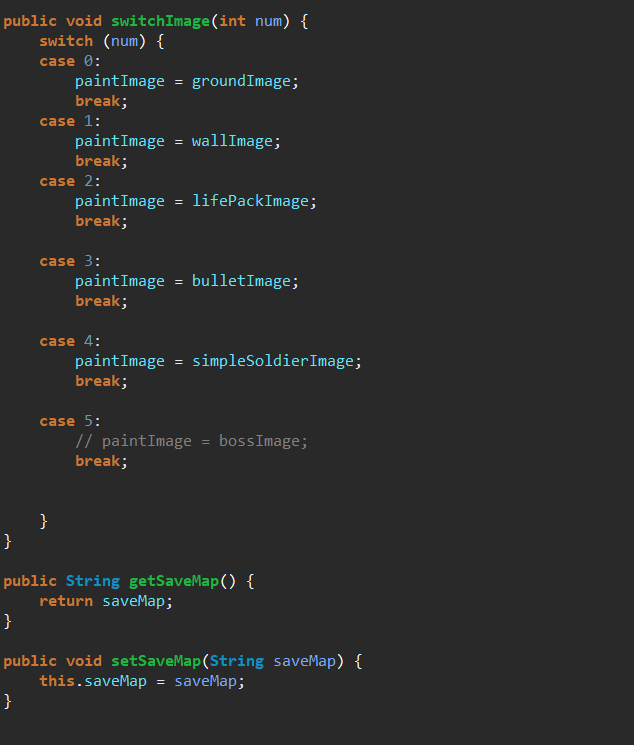








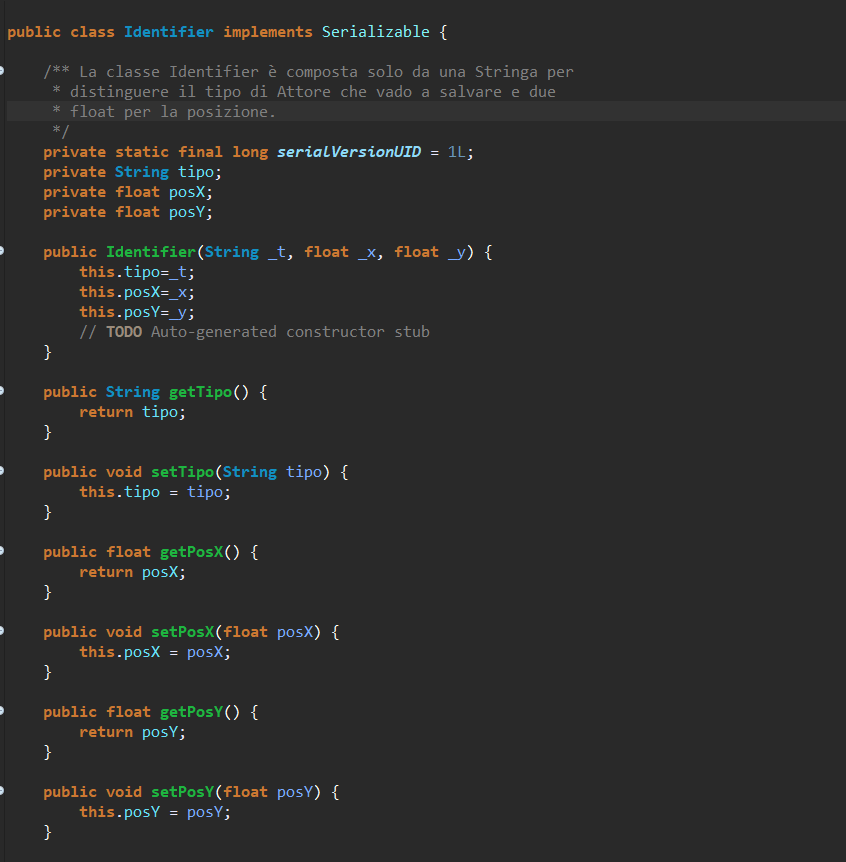




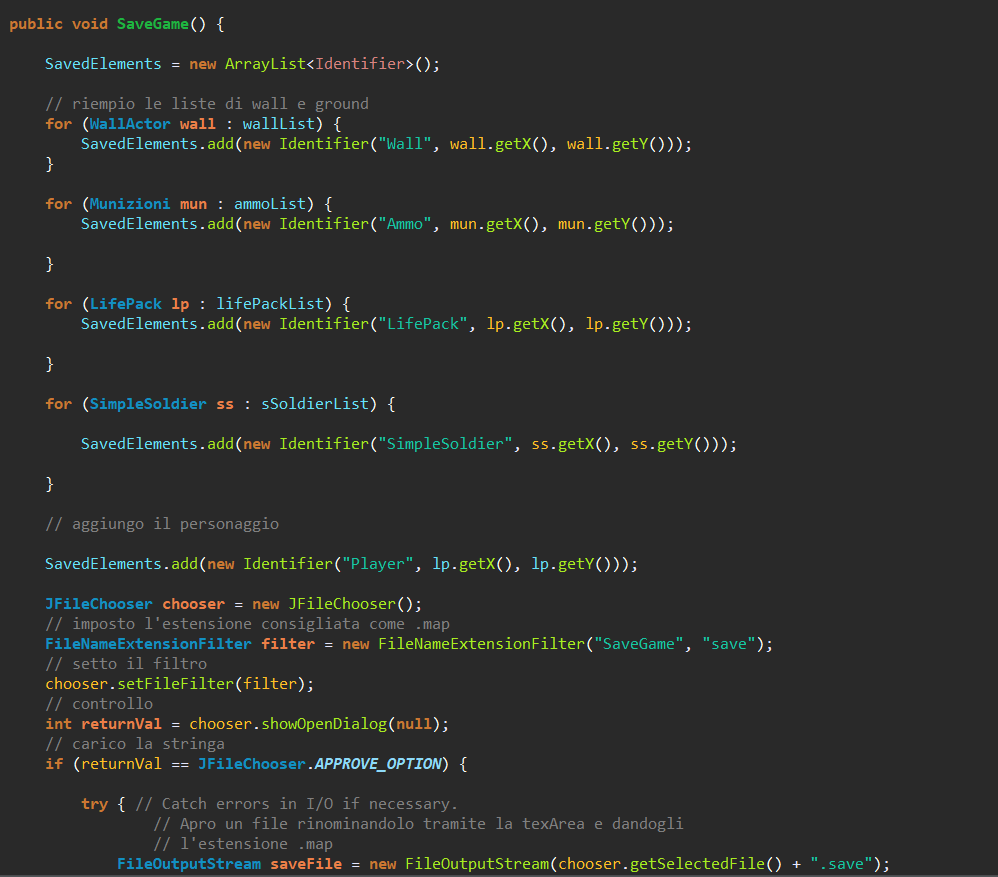
**Gestore della partita: Carica & Salva**

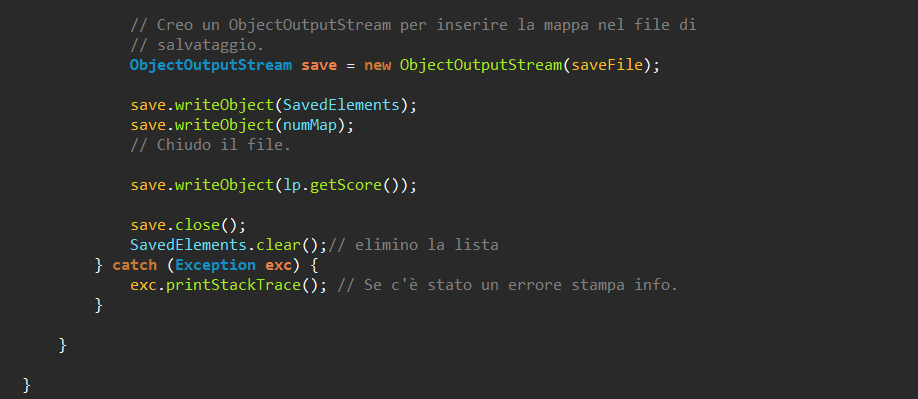
La partita viene salvata tramite un’ArrayList di Identifier.

Di seguito il codice della suddetta classe:

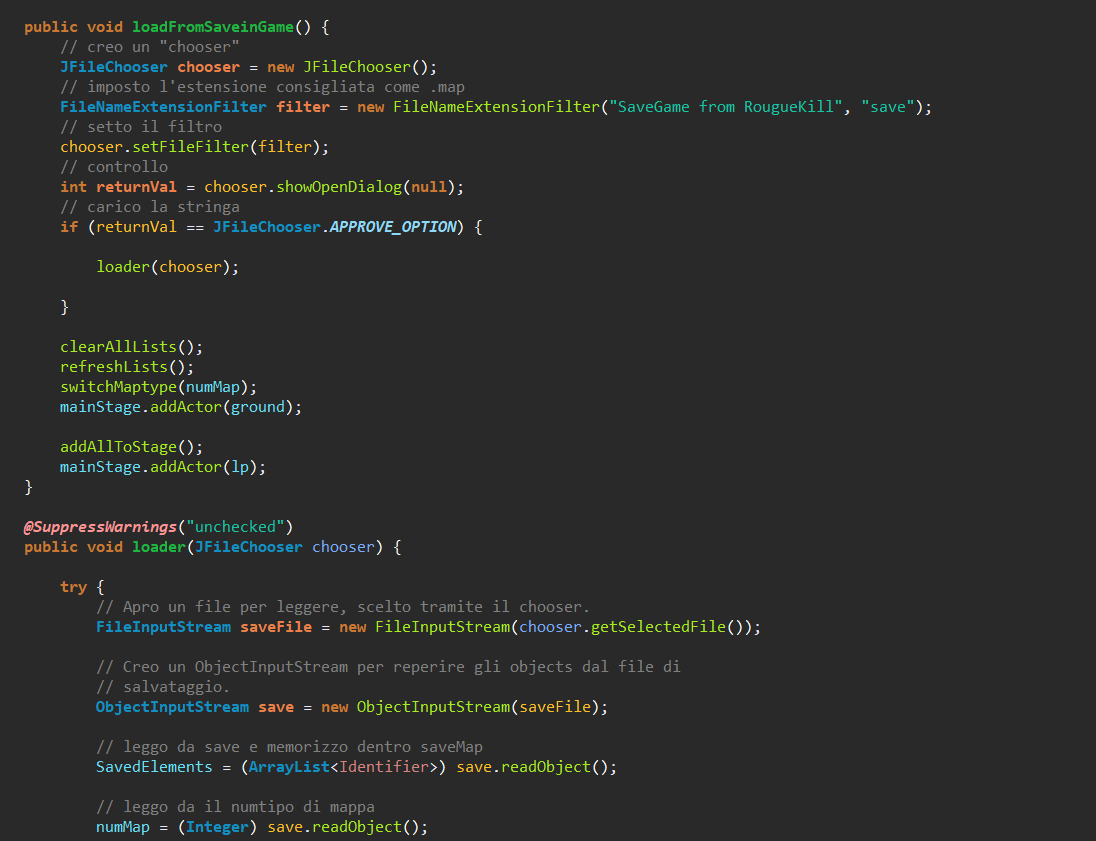


Dopodichè richiamo la funzione SaveGame per salvare lo stato della partita in un ArrayList di Identifier



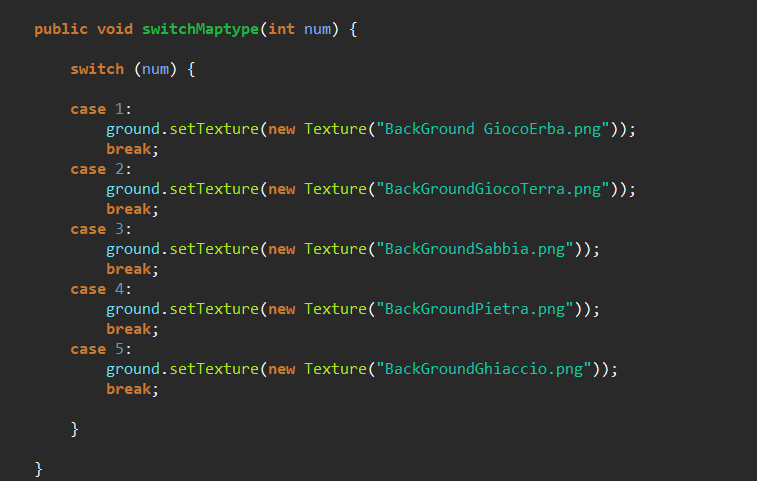


Uso il writeObject di Java per salvare prima gli identifier e poi il numMap per ricordarmi che tipo di mappa stavo giocando.

Carico La Mappa usando sempre un JFilechooser e usanto il readObject di Java e riposizono tutti i vari Attori sullo Stage.







**SCREENSHOTS**

****

Un esempio di partita. Da notare in alto a destra Punteggio, Vita del personaggio e Munizioni ( creati tramite dei Label).

**E’ ARRIVATO IL BOSS!** (Il boss compare quando rimangono 5 nemici da uccidere)****

**GAME OVER.**

****

**YOU WIN!**

****

**HIGHSCORES**

****

**Credits**

Programmatori:

Simone Crisafi

Francesco Marrara

Si ringrazia:

Consorzio Autolinee Cosenza

Tonno Callipo

Vivident Xylit

Trenitalia

Lirosi Autolinee

Mt5, Mt5 Bis.

**ARIA CONDIZIONATA**