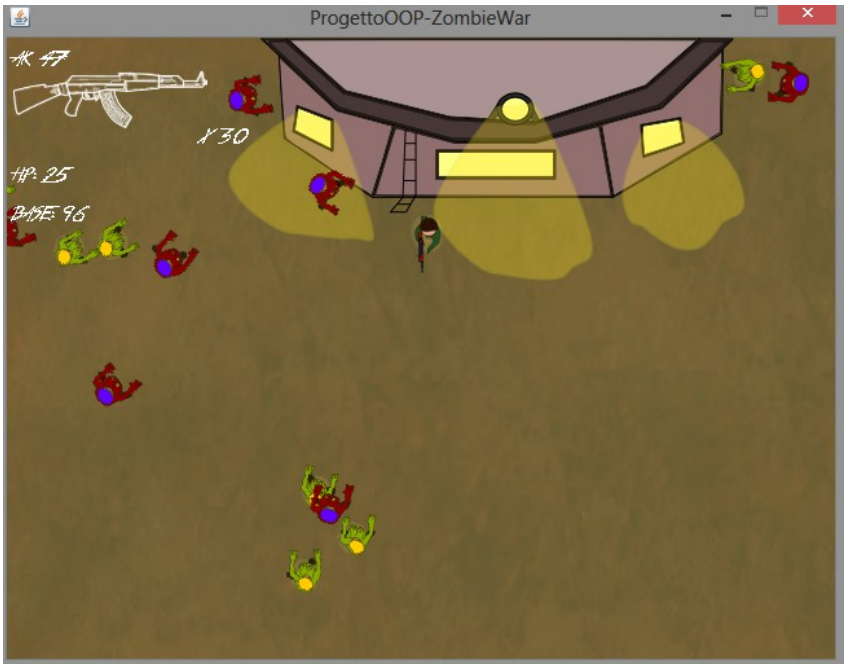
Corso di Programmazione ad Oggetti – a.a. 2013-2014

Relazione relativa a ZombieWar2014 – un gioco sviluppato in java sulla base del famoso flash game Endless Zombie Rampage

Componenti del gruppo: Giovanni Romio

1 DESCRIZIONE DEL GIOCO

Il gioco realizzato è uno sparatutto 2D nel quale il giocatore impersonati i panni di un civile, deve cercare di sopravvivere ad un'ordata di zombie cercando di rimanere in vita e di salvare la sua base.
Il gioco dal quale ho preso spunto è Endless Zombie Rampage prodotto dalla CrazyMonkeyGame.
Il giocatore avrà a disposizione 3 armi differenti: una pistola, un fucile ed un mitragliatore.
Ciascuna arma ha differenti caratteristiche come capacità, danno e numeri di proiettili in grado di sparare alla volta.
Glock21: 15 colpi, spara un colpo alla volta di danno 5.
AK47: 30 colpi, spara un colpo alla volta di danno 8.
Minigun: 120 colpi, spara 3 colpi ciascuno di danno 6.
Gli Zombie, dotati di una demenziale I.A. cercheranno di attaccare la base o il giocatore qualora si trovi nel campo visivo di ciascuno di essi.
Gli zombie rossi, a differenza di quelli versi quando avranno un livello di salute basso aumenteranno la loro velocità complicando così la difficoltà del gioco.



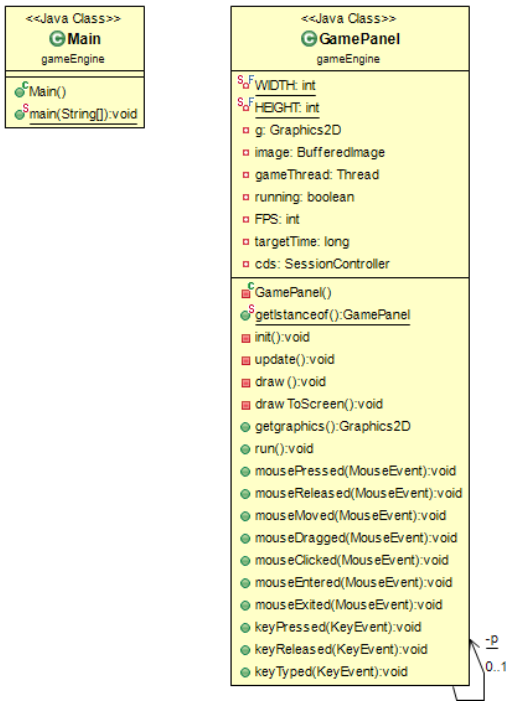
2 PROGETTAZIONE ARCHITETTURALE

METODO MAIN

Il main() del nostro progetto istanzia un JFrame ed un JPanel al suo interno.
Il Jpanel che andiamo ad associargli è una classe che estende da Jpanel ed implementa MouseLister, KeyLister, MouseMotionLister ed Runnable.

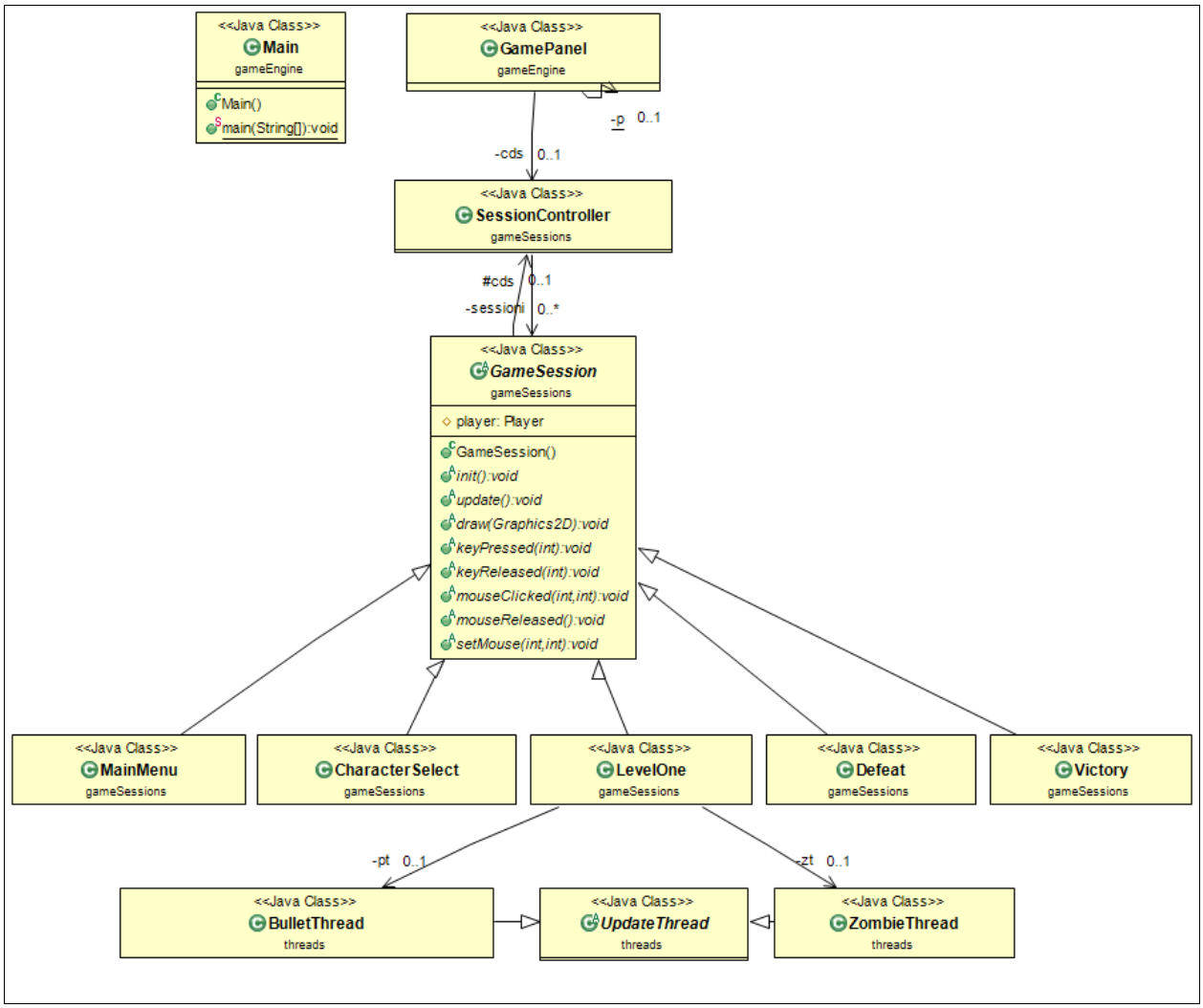
All'interno del Pannello Di Gioco vi è il metodo run, che nella sua routine richiama i metodi init(), update(), draw(), drawToScreen().
Dunque Pannello di gioco contiene il Thread principale, questo Thread è responsabile di richiamare il metodo update e draw del ControllerDiSessione, il quale a sua volta richiama il metodo relativo alla SessioneCorrente.

Per cui THREAD_UPDATE >> CONTROLLERDISESSIONE_UPDATE >> SESSIONECORRENTE_UPDATE
Per cui THREAD_DRAW >> CONTROLLERDISESSIONE_DRAW >> SESSIONECORRENTE_DRAW



IL GIOCO E' SUDDIVISO IN DIVERSE SESSIONI DI GIOCO

- MENU PRINCIPALE
- OPZIONI
- CREDITI
- SELEZIONE PERSONAGGIO
- LIVELLO UNO
- PAUSA
- VITTORIA
- SCONFITTA



Ciascuna delle quali contiene i metodi update(), draw().
Il Thread principale si occupa di richiamare nella sua routine tali metodi, relativi alla sessione corrente.
Le sessioni sono gestite da un Controller di Sessione che contiene anche esso i metodi Update e draw.
Ciascuna delle sessioni descrive una parte del gioco.
Le sessioni possono essere create, eliminate oppure mantenute in memoria.
Questo compito è svolto dal Controller di Sessione.

LIVELLO UNO
Il livello uno contiene tutto ciò che permetterà al giocatore di affrontare gli zombie.
Il gioco deve constantemente elaborare le seguenti entità:

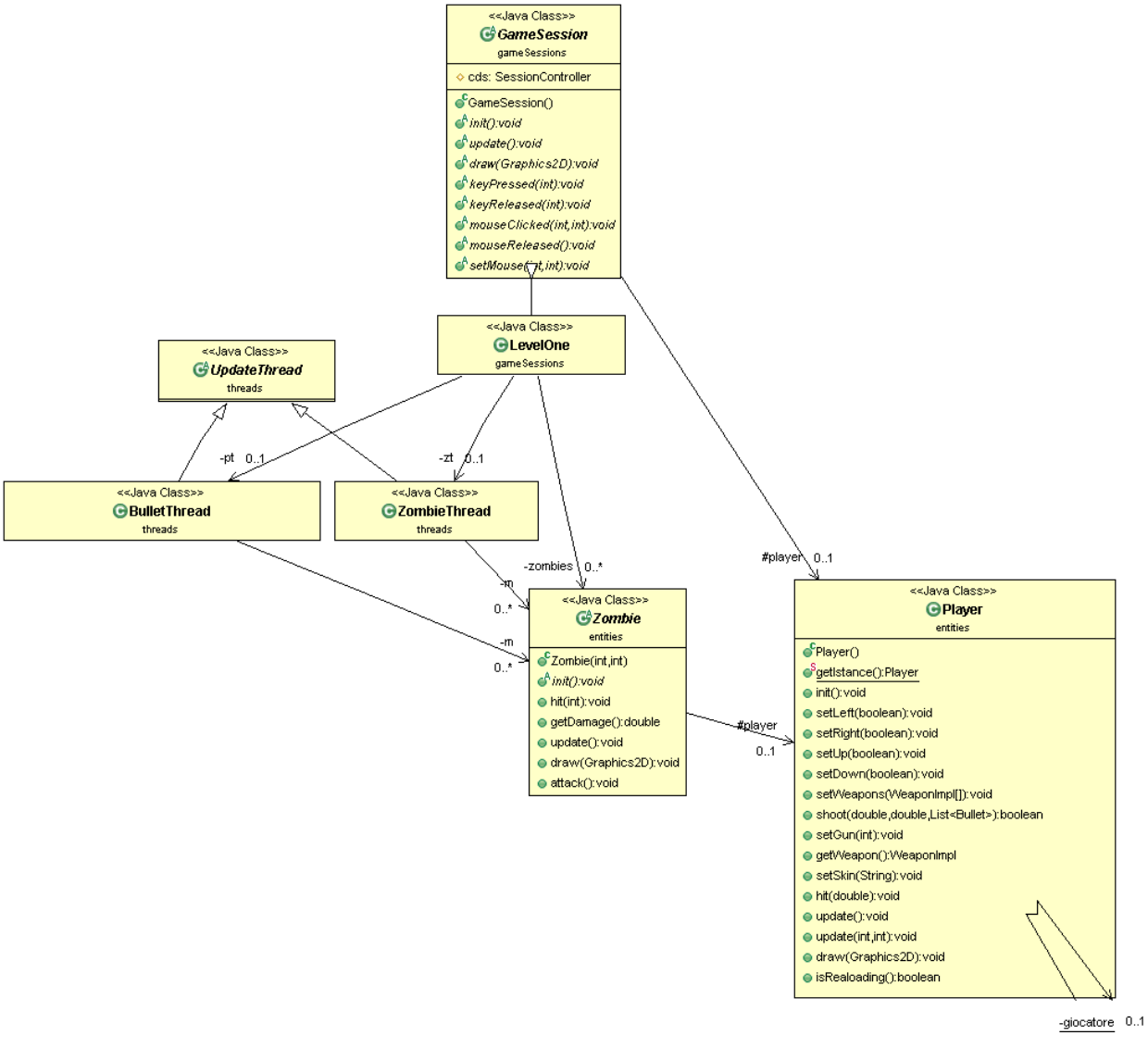
- Giocatore
- Proiettili
- Zombies
- Mappa di Gioco

Il giocatore siccome è comandato da input, è direttamente collegato al Thread principale, quello che contiene i Listener per gli input da mouse e da tastiera.

Livello uno Update() accede a Player,HUD,MyMap.
Livello uno Draw() accede a Player,HUD,MyMap,Bullets,Zombies.

Per quanto riguarda Zombies e Proiettili ho deciso di assegnare il compito di richiamare gli update per ciascuno di essi("quelli vivi") a due Thread separati.

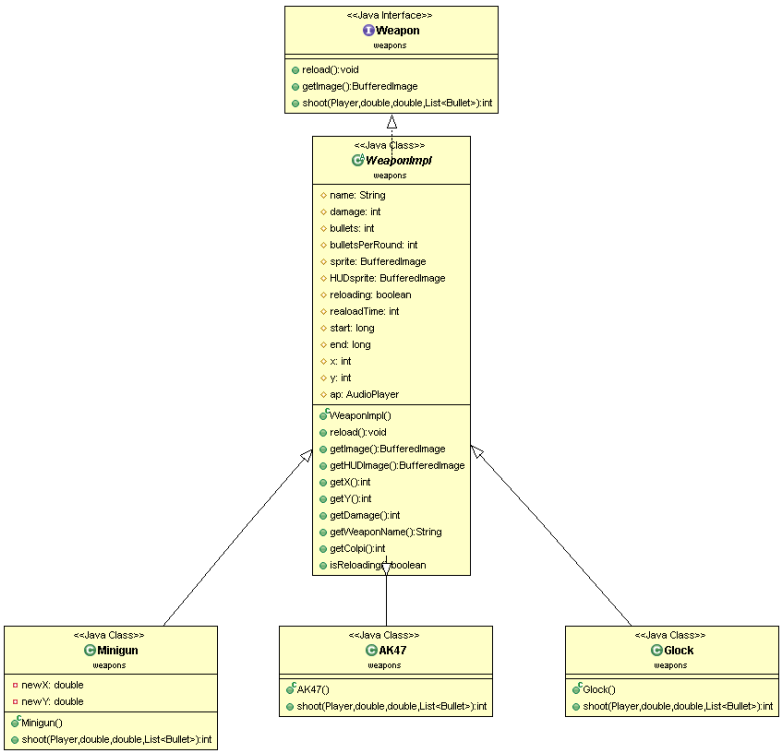
- Zombie Thread: richiama l'update per gli zombie vivi.
- Proiettile Thread: richiama l'update per i proiettili che sono all'interno della finestra.



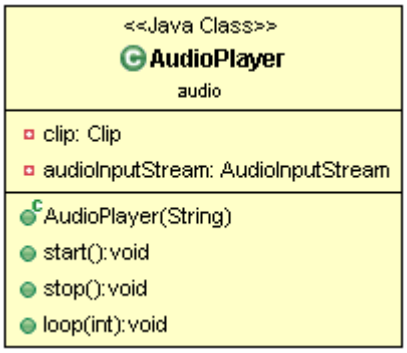
3 ORGANIZZAZIONE IN PACKAGE

Viene ora effettuata una analisi dell'organizzazione in package dell'applicazione.

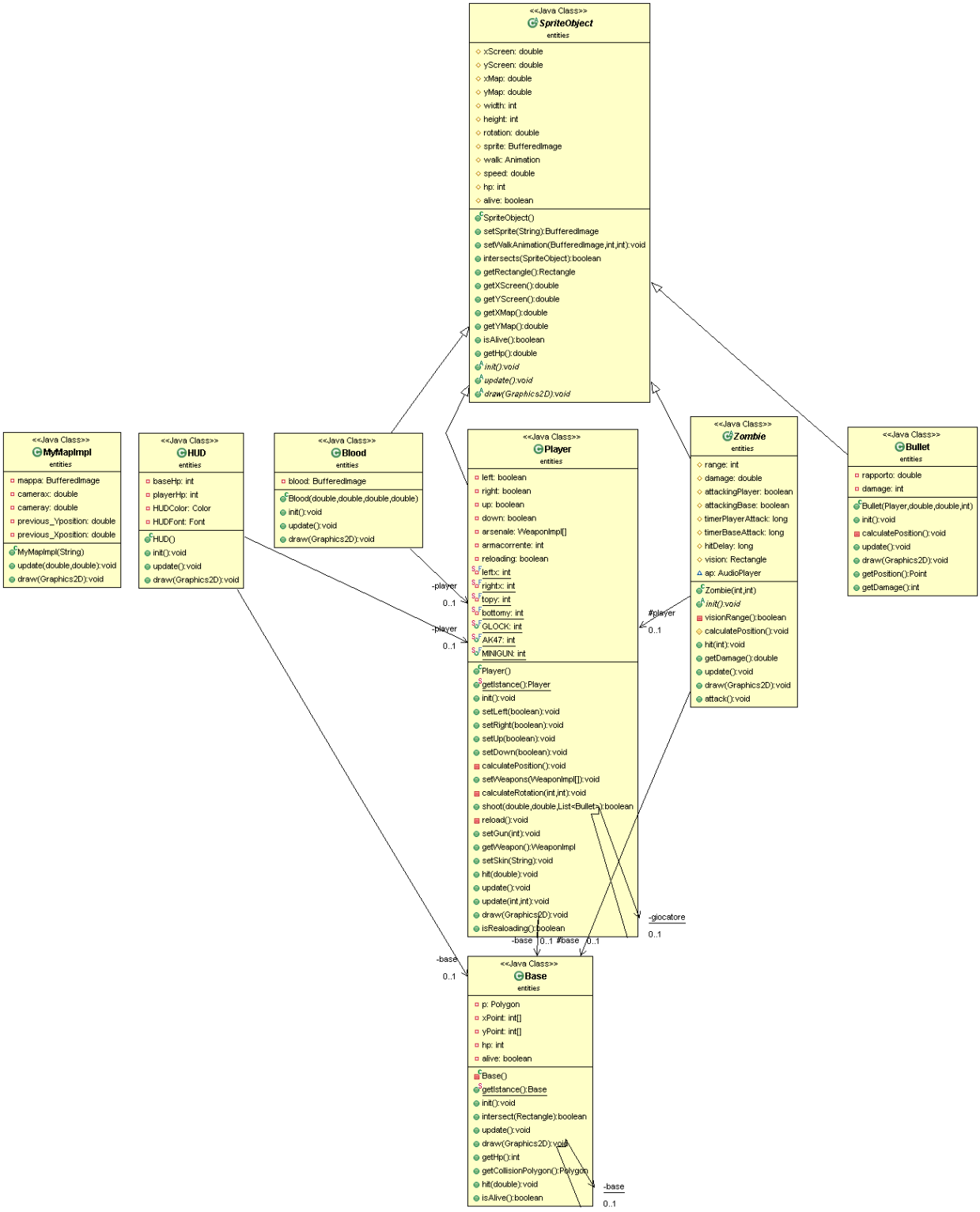
- **ARMI**: contiene l'insieme delle classi che definiscono la struttura e le caratteristiche delle varie armi nel gioco.
  L'interfaccia Weapon definisce i metodi che devono implementare le sue armi. Weapon Impl invece è una classe astratta che contiene tutti i parametri e le operazioni comuni alle 3 armi.



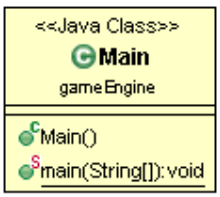
- **AUDIO:** contiene audioplayer che permette di istanziare un clip sul quale invocare start(), stop(), loop().

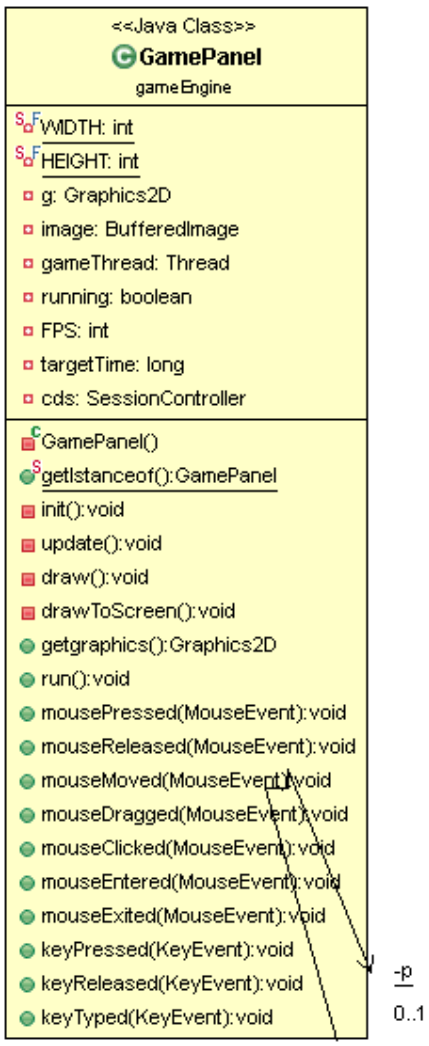


- **ENTITA**: contiene tutte le classi associate alle entità del gioco come Giocatore, MammaZombie, Base, HUD, Sangue e la classe che gestisce le animazione.

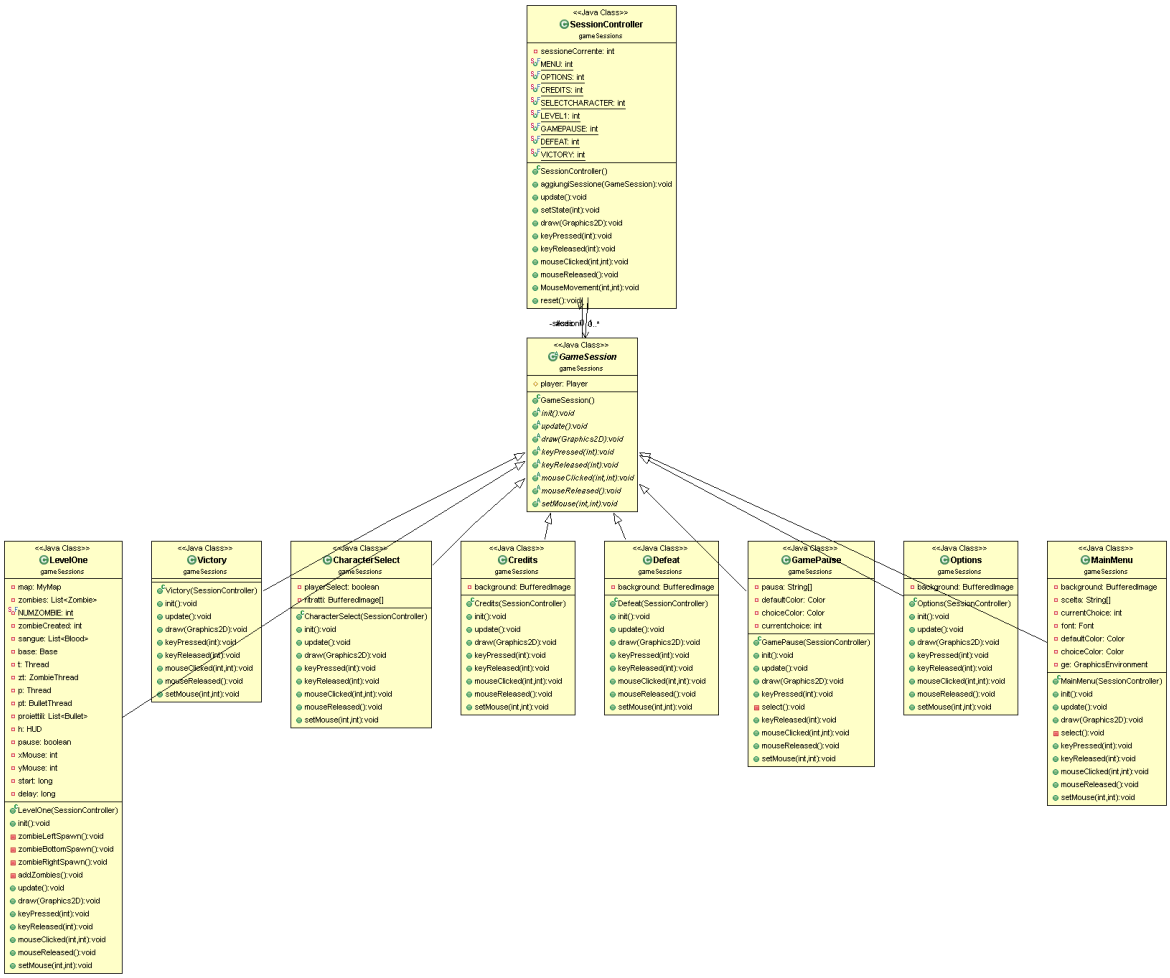


- **MOTORE DI GIOCO:** contiene il main() e il PannelloDiGioco a cui è associato il Thread principale.





- **SESSIONI DI GIOCO:** contiene tutti i sorgenti delle varie sessioni di gioco, così come le classi relative ai Thread relativi agli Zombie ed i Proiettili.



3 NOTE FINALI

Descrizione del flusso di lavoro:
Il flusso di lavoro è stato lineare, dapprima ho implementato il main e il PanellodiGioco, poi dopo aver deciso di adottare la strategia del Controller di Sessione ho sviluppo una sessione alla volta e creato quindi le entità di cui necessitavano i vari stati di gioco.

Dapprima il gioco era basato su un singolo Thread, ho deciso in seguito di suddividere le operazioni di aggiornamento degli zombie e dei proiettili in due Thread separati.
Questa modifica è stata forse la parte più complicata perchè i due Thread dovevano condividere una lista contenente gli elementi da aggiornare con il thread principale il quale eseguiva il draw su ciascuno di questi oggetti.
Grazie alla mutua esclusione ed ai blocchi synchronized() ho risolto questo problema.
Un altro inceppo c'è stato quando ho introdotto il gamePause() per cui i thread dovevano bloccarsi. Questa volta ho utilizzato i metodi wait() e notify() su una variabile flag di tipo boolean che interrompeva il thread qualora fosse false.
Con l'aggiunta di questi due thread è forse più difficile leggere il programma per un nuovo programmatore però basta far caso che il Player siccome è utilizzato da più thread contemporaneamente è stato realizzando secondo il Pattern Singleton per cui l'accesso simultaneo allo stesso oggetti è garantito in mutua esclusione dal metodo static getIstance() dello stesso Player.
Questo pattern mi ha permesso inoltre di memorizzare lo stato del giocatore nelle diversi sessioni.
La skin del Player viene scelta in CharachterSelect e qualora vengano aggiunte nuove sessioni come:"Seleziona arsenale", " Livello 2 " ecc.. che dovranno accedere a tutti i campi di Player quando andremo a richiamare il metodo getIstance() siamo sicuri che il Player mantiene il stato consistente.