

# Progetto JLife di Alessandro Pallotta - 102627

Per consentire l'estendibilità del programma ho diviso il più possibile le varie classi in base alle loro responsabilità, così da garantire il funzionamento anche dopo aver apportato piccole modifiche o aver aggiunto oggetti che estendono quelli già presenti, senza dover necessariamente modificare tutta la struttura. Il programma è stato diviso in diversi packages per raggruppare elementi legati tra loro.

## Funzionalità:

Il mio programma principalmente permette di creare e visualizzare l'evoluzione della tabella del 'game of life' avanzando manualmente con il comando next o semplicemente cliccando Enter. L'utente potrà scegliere se utilizzare la versione classica del 'game of life' oppure una versione 'smart' con cui è possibile personalizzare le regole di vita che seguono le cellule. Inoltre si possono utilizzare i comandi 'save' e 'load' per salvare o caricare una certa tabella, ( con l'utilizzo di Gson ) semplicemente inserendo il percorso della cartella di salvataggio.

## Descrizione architettura:

Il programma cerca il più possibile di rispettare il pattern MVC, la parte principale del Model è la classe Board, che ha la responsabilità di gestire la griglia di gioco; si occupa della sua generazione, dell'evoluzione nel tempo. La seconda parte del Model è la classe Cell, che ha la semplice responsabilità di conoscere il suo stato e le regole per modificarlo. Nella mia implementazione del "game of life" ho aggiunto 2 classi "gemelle" a quelle appena descritte, SmartCell e SmartBoard. Queste mantengono lo scopo delle precedenti, la differenza si trova nella gestione delle regole di vita, che sono contenute in un HashMap, ho fatto questo per permettere di modificare e personalizzare le loro "leggi di vita". Per creare e gestire queste regole ho utilizzato la classe RulesManager, in cui sono presenti le regole Default e una versione inventata da me ( chiamata zombie ). Il Controller è rappresentato dalla classe Controller che ha la responsabilità di ricevere comandi dalla vista ed applicarli alla Board. La View del modello MVC è rappresentata dalla classe ViewCli, questa si occupa di gestire input e output dell'applicazione; riceve i comandi dell'utente e in base ad essi, sceglie l'azione da eseguire e comunica al Controller eventuali operazioni. La classe App ha la responsabilità di inizializzare il programma, definisce ed esegue i comandi che vengono ricevuti dalla View.

I test che ho utilizzato sono ridotti perché non avvengono grandi operazioni tra gli oggetti, infatti i test che sono stati sviluppati servono per lo più a verificare il corretto funzionamento del metodo `getNeighbors`.