Programmazione ad Oggetti

Progetto

Be a Manager

Luca Miciletto

Mattia Linguerri

29 maggio 2015

Indice

Capitolo 1 – Analisi

1.1 Requisiti

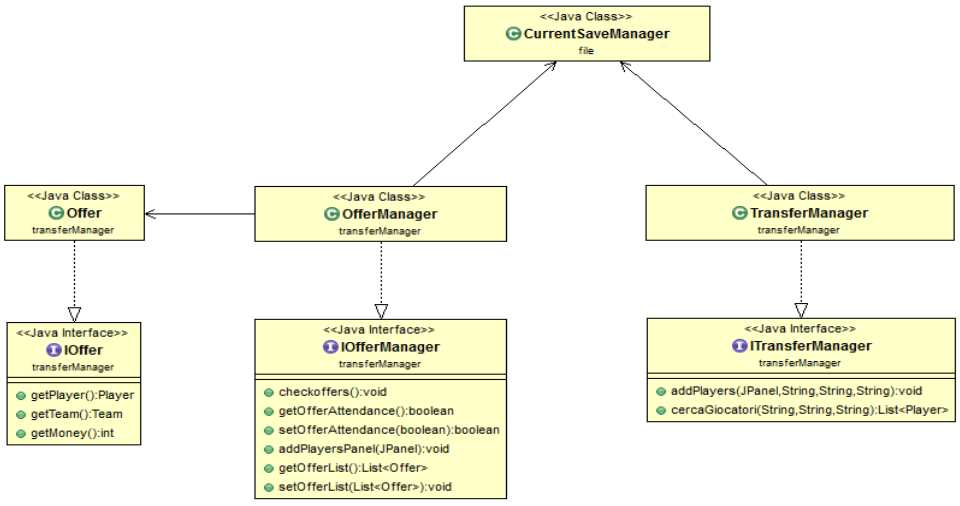
Il software mira alla realizzazione di un videogioco manageriale di calcio che simuli una stagione di Serie A Tim.
L'utente avrà la possibilità di inserire la formazione titolare, avanzare di settimana simulando la partita e vendere, comprare e scambiare giocatori con altre squadre gestite dalla AI (Intelligenza Artificiale).

Le funzionalità che il software deve gestire sono:

- Caricamento dei salvataggi effettuati in precedenza
- Inizializzazione di una nuova stagione (scelta della squadra)
- Gestione della formazione titolare
- Simulazione di una partita
- Acquisto o scambio di giocatori di altre squadre
- Gestione delle offerte per i giocatori ricevute
- Salvataggio della stagione corrente
- Visualizzazione dei vari obiettivi e di quello raggiunto

1.2 Problema

Il software dovrà gestire lo stato corrente della stagione anche dopo la sua chiusura, quindi si rendono necessari file di salvataggio (la cartella predefinita è situata nella cartella utente).
L'applicazione dovrà gestire, inoltre, le offerte provenienti dalle altre squadre e l'accettazione delle proposte di acquisto o scambio generate dall'utente.

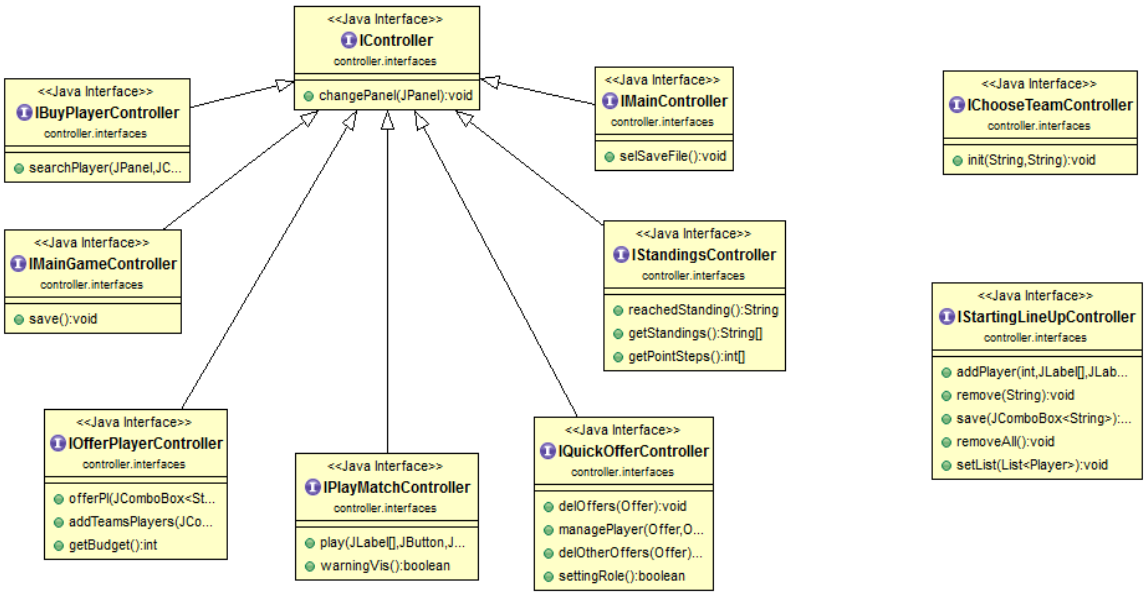


Capitolo 2 – Design

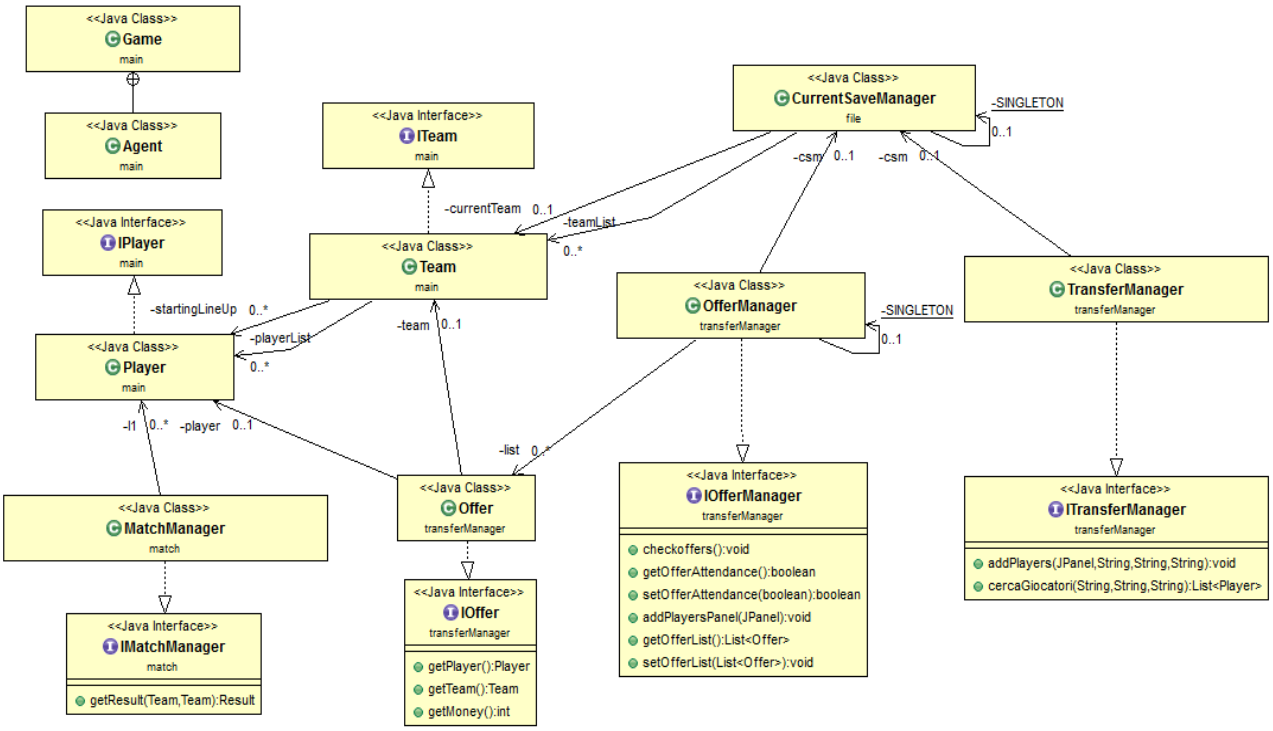
2.1 Architettura

Il software è stato sviluppato seguendo il pattern architetturale MVC (Model – View - Controller) dove:

- **View**: è rappresentata dal frame principale nel quale vengono cambiati i panel in base alle necessità e da altri due frame più piccoli che vengono costruiti solo in determinate situazioni.
- **Controller**: ogni View ha un Controller che gestisce le azioni che l'utente può eseguire in essa.
- **Model**: è rappresentato dalle altre classi che gestiscono il corretto funzionamento del software.



2.2 Design dettagliato



La classe Game è quella principale che contiene il frame di gioco e quello dei salvataggi, richiamando una sua funzione statica è possibile modificare la View corrente.
Il model è composto da:

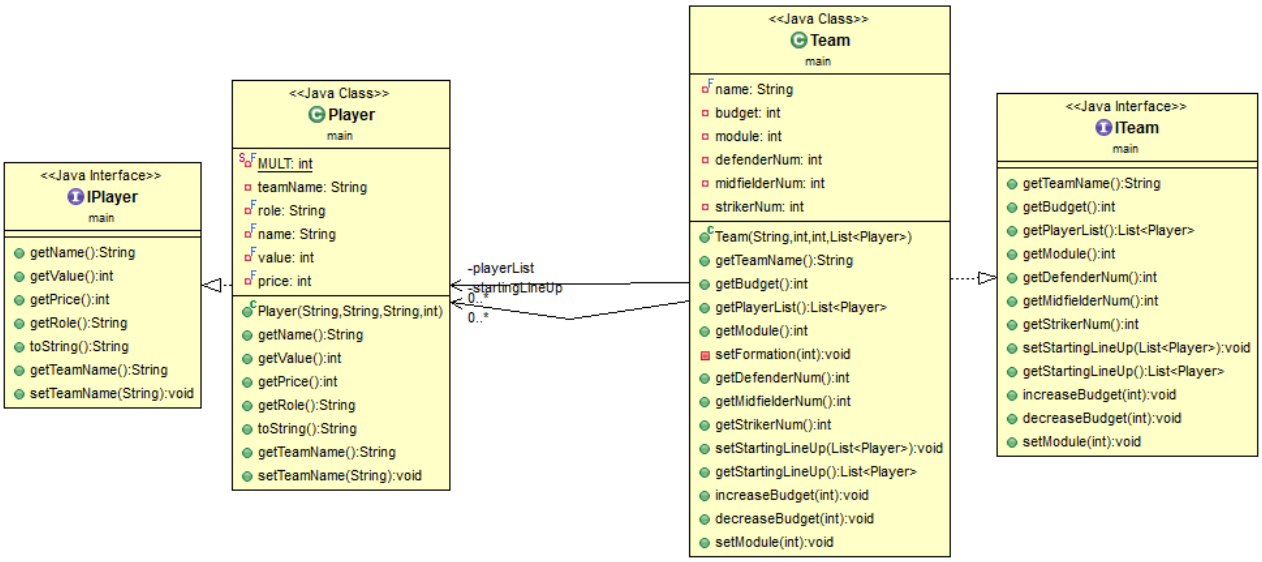
- **CurrentSaveManager** : è il model principale, permette la scrittura e la lettura dai file di salvataggio, tiene conto dello stato corrente della stagione (settimana, punti e lista delle squadre).
- **TransferManager** : è il model relativo alla sezione di acquisto e scambio di giocatori .
- **OfferManager** : è il model relativo alla sezione delle offerte, tiene conto delle offerte ricevute dalle altre squadre

In CurrentSaveManager e OfferManager si è utilizzato il pattern Singleton per avere un'unica istanza delle due classi e per essere richiamate più facilmente.
Le View e i Controller sono così suddivisi:

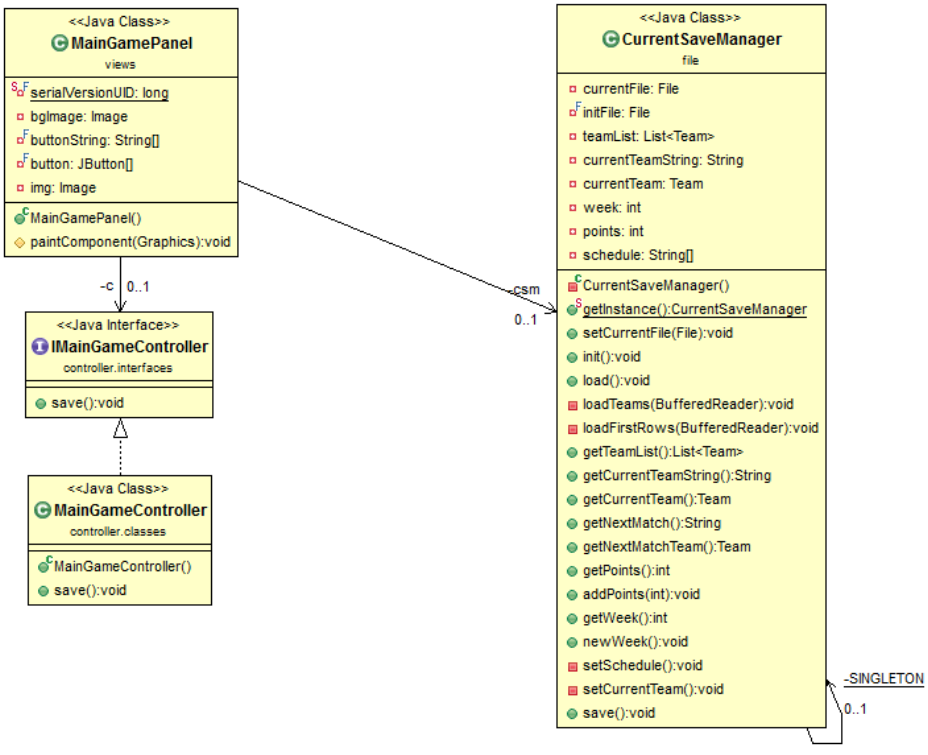
- **BuyPlayer** : si occupa della ricerca del giocatore che si vuole acquistare.
- **ChooseTeam** : si occupa della scelta della squadra con la quale si vuole iniziare una nuova stagione.
- **Credits** : visualizza i crediti (autori e immagini).
- **EndSeason** : visualizza l'obiettivo raggiunto nel corso della stagione in base ai punti totalizzati.
- **MainGame** : è la View principale della stagione, permette di passare alle View secondarie e di salvare la stagione corrente.
- **Main** : è la View principale del gioco.
- **OfferPlayer** : permette di avanzare un'offerta per un giocatore.
- **Offers** : visualizza le offerte ricevute e permette di accettarle o rifiutarle.
- **PlayMatch** : si occupa della preparazione alla simulazione di una partita.
- **QuickOffer** : visualizza l'offerta accettata.

- **Saves** : permette di inizializzare o caricare un file di salvataggio.
- **Standings** : permette di visualizzare gli obiettivi del gioco e quello già raggiunto.
- **StartingLineUp** : si occupa di gestire la formazione titolare, costruisce un SelPlayerFrame ogni volta che si aggiunge un nuovo giocatore.
- **Transfer** : la View principale per i trasferimenti.

La figura sottostante rappresenta la relazione tra Team e Player.



La figura sottostante è un esempio di MVC, quello applicato alla sezione MainGame.



Capitolo 3 – Sviluppo

3.1 Testing automatizzato

Sono stati svolti due tipi di Testing, uno automatizzato che verifica il corretto funzionamento di alcune piccole parti del software (Player, Team, Result, la sezione riguardante l'avanzamento di settimana in CurrentSaveManager, Offer e OfferManager), il secondo manuale riguardante View e Controller.
Il testing automatizzato è stato eseguito con la suite di JUnit.

3.2 Divisione dei compiti e metodologia di lavoro

Il lavoro è stato diviso nel seguente modo:

- Mattia Linguerri : gestione dei file di salvataggio, GUI e Controller principali.
- Luca Miciletto : gestione dei match e trasferimenti.
- Insieme : sviluppo del Model e Testing

La maggior parte del lavoro è stata effettuata singolarmente.
Il DVCS utilizzato è Mercurial e il progetto è stato condiviso tra gli studenti attraverso un repository creato su Bitbucket sul quale i membri hanno lavorato tramite il plug-in di Mercurial per Eclipse.

Capitolo 4 – Commenti Finali

4.1 Conclusioni e lavori futuri

Il progetto copre diversi argomenti del corso (file, pattern, GUI).

Un punto debole del progetto è la scrittura dei salvataggi su file, sarebbe stato meglio utilizzare un database. In più alcune informazioni si perdono nella chiusura del gioco, ad esempio la formazione titolare e le offerte in stato di attesa.
Per motivi di comodità si è deciso di non utilizzare una classifica, ma obiettivi da raggiungere.

Nel caso volessimo portare avanti questo progetto si potrebbe inserire una parte visuale del match durante la simulazione della partita che coinvolgerebbe ulteriormente l'utente, inoltre, inserendo dati relativi ai marcatori delle relative squadre, si potrebbe fornire a fine stagione una classifica dei marcatori.
Implementando un server si potrebbe anche creare un videogioco online manageriale nel quale gli utenti potrebbero sfidarsi tra di loro, eliminando l'AI.

4.2 Difficoltà incontrate

Abbiamo riscontrato problemi nel caricare il file di inizializzazione eseguendo il file jar.
Abbiamo risolto seguendo la guida: http://www.mkyong.com/java/how-to-convert-inputstream-to-file-in-java/