

INGEGNERIA DELLA CONOSCENZA

*AA 19/20*



DOCUMENTAZIONE DEL PROGETTO

Michele Messina [676717]

Francesco Zingariello [683004]

[m.messina11@studenti.uniba.it](mailto:m.messina11@studenti.uniba.it)

[f.zingariello1@studenti.uniba.it](mailto:f.zingariello1@studenti.uniba.it)

<https://github.com/mikkmessi/ICON_Project>



# Indice

1. Analisi………………….…………………………………………………………………….. 3

3. Progettazione……….…………………………………………………………………. 4-5

2.Implementazione….………………………………………………………………..….5-6



# Analisi

L’obiettivo del progetto è quello di realizzare un’applicazione basata su tecniche di AI in grado di predire la performance dei giocatori di una squadra di Fantacalcio e consigliarci i giocatori migliori da schierare la prossima giornata.

L’applicazione è sviluppata completamente in linguaggio Python con l’ausilio delle librerie Scikit-learn, Pandas, Seaborn, Matplotlib e dei software Microsoft Excel e Access.

I dataset con le statistiche dei calciatori e delle squadre sono stati recuperati da “[Fbref.com](https://fbref.com/it/)”, mentre le medie FantaVoto da “[Fantacalcio.it](https://www.fantacalcio.it/statistiche-serie-a)”

Per il versionamento è stato utilizzato il repository Github <https://github.com/mikkmessi/ICON_Project> dove è possibile reperire i dataset, gli script e la documentazione.



# Premesse

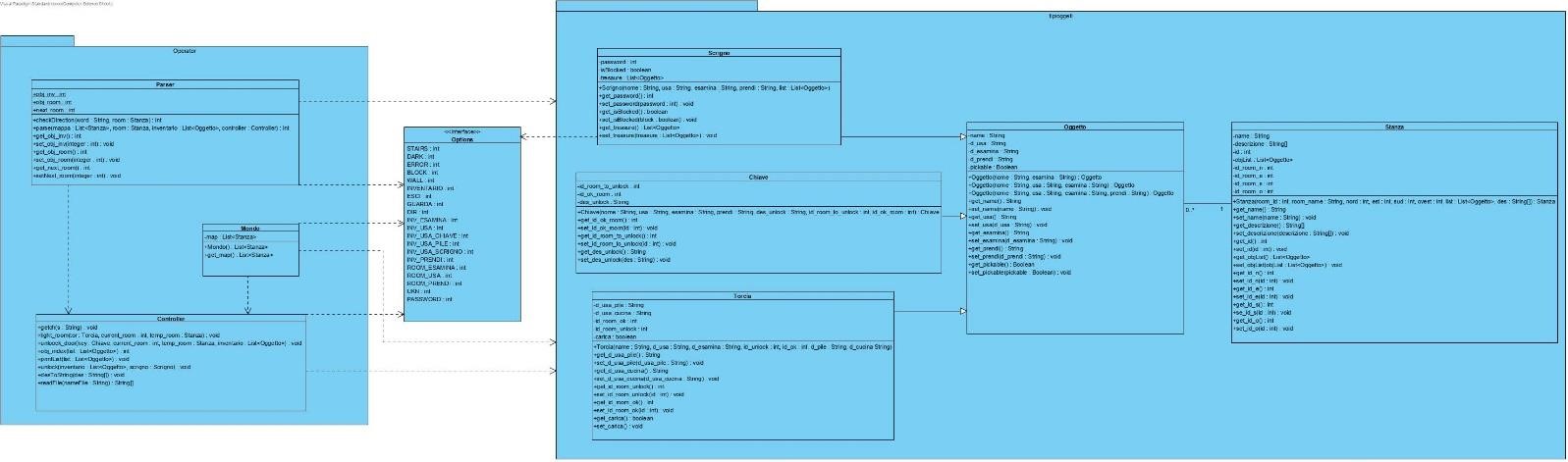
I dataset sono stati modellati in modo importante per adattarli alle esigenze:

* Per avere risultati ottimali, sarebbe stato più opportuno avere a disposizione i dati per ogni singola giornata di campionato. Dopo varie ricerche infruttuose si è preferito il sito scelto poiché presentava dati precisi e completi anche se cumulativi, non specifici per ogni giornata;
* Il sito forniva più tabelle distinte per ambito calcistico (es. Azioni Difensive, Tiri, Passaggi...). Si è optato per assemblarle in un unico grande dataset dopo un lavoro di scrematura dei parametri. A questo è stato aggiunto il parametro della media fantavoto;



# Progettazione

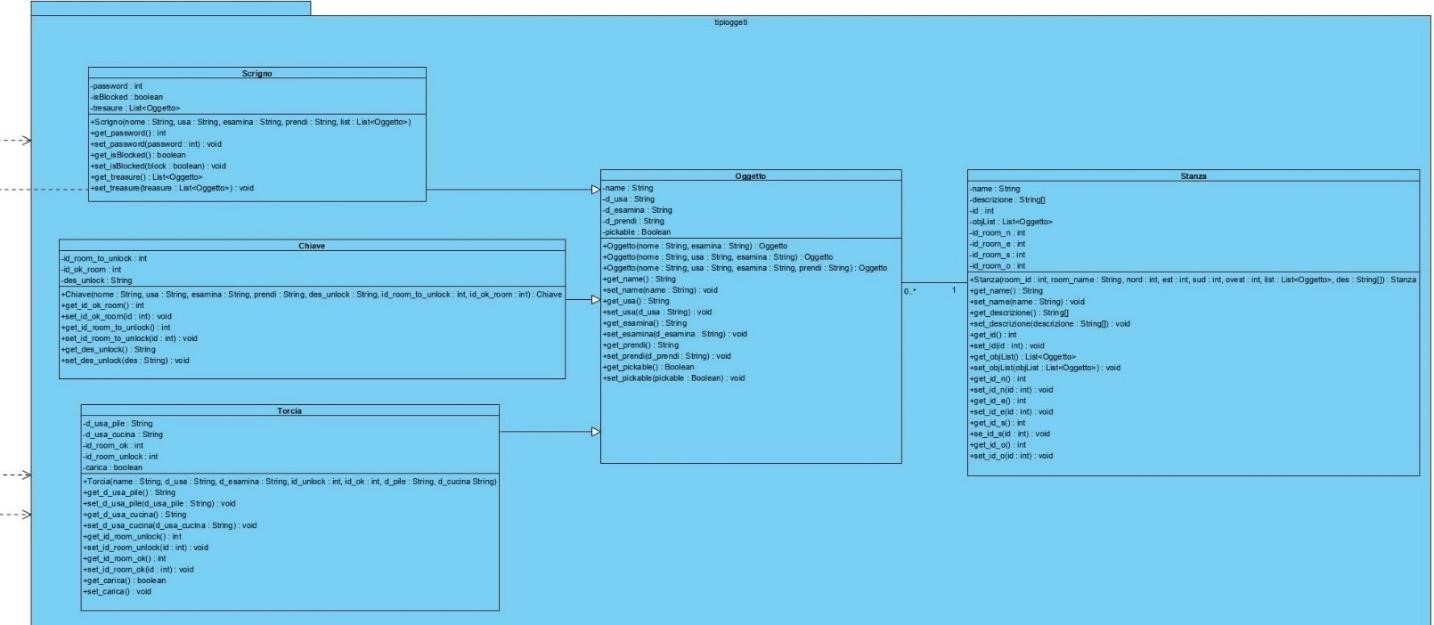
Dopo la fase preliminare di brainstorming e quindi scelta la trama e le ambientazioni, si è passati quindi alla valutazione e successiva definizione delle scelte progettuali. E’ seguita la realizzazione del diagramma delle classi con i relativi collegamenti.

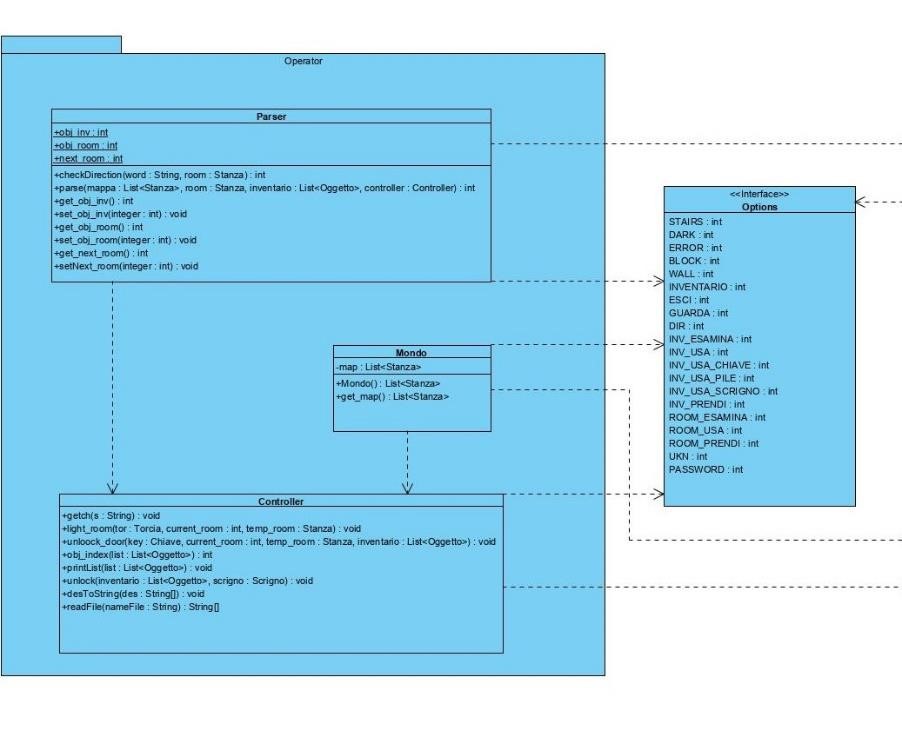


Il diagramma riflette la divisione in pacchetti dell’implementazione:

* il pacchetto “tipioggetti” include le classi degli oggetti con cui il giocatore può interagire; Troviamo la classe “Oggetto” e la classe “Stanza” sono collegate tramite una relazione di associazione, mentre “Scrigno”, “Chiave” e “Torcia”, che dipendono dalla classe padre “Oggetto”. Questo perché la classe “Stanza” contiene una lista di istanze di tipo “Oggetto” e le 3 classi “Scrigno”, “Chiave” e “Torcia”

non sono altro che specializzazioni della classe “Oggetto” con metodi e attributi specifici.





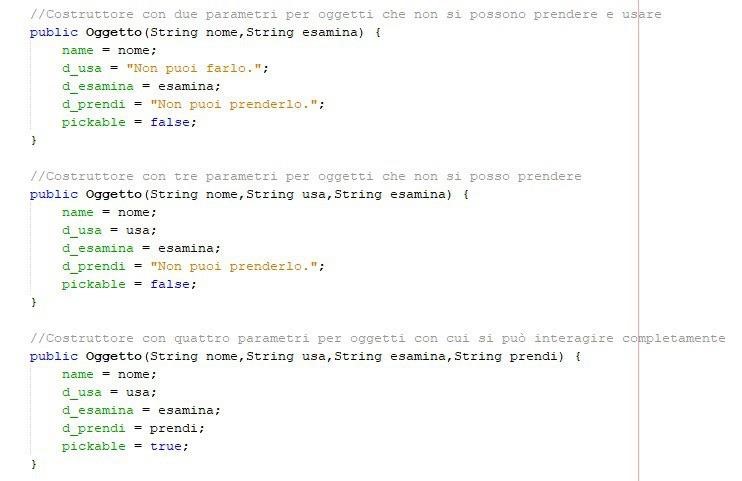
* Il pacchetto “operator” contiene le classi utili al funzionamento e controllo dell’applicazione mediante le classi “Mondo” e “Parser” che dipendono dalla classe “Controller”; “Mondo” si occupa di creare la mappa di gioco, con la lista delle stanze necessarie mentre “Parser” ha il compito di interpretare i comandi del giocatore. “Controller” invece contiene una serie di metodi di supporto a tutte le altre classi.

L’unica classe esterna è l’interfaccia “Options” che contiene tutte le costanti.



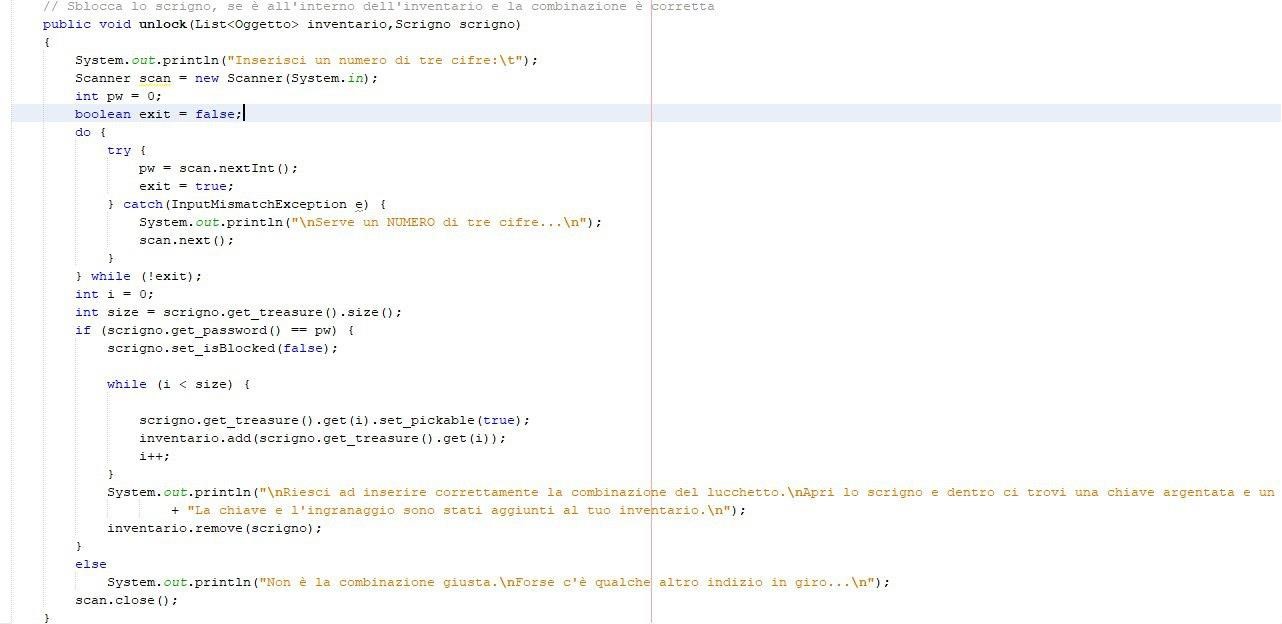
# Implementazione

L’implementazione che ne segue è di facile interpretazione; solo nella classe padre Oggetto possiamo notare un *overloading* del costruttore che differisce nell’implementazione in base ai parametri formali, mentre le altre contengono solo i propri metodi di accesso agli attributi.



Poiché è il controller che gestisce e opera sugli oggetti, essi contengono solo i propri metodi di accesso agli attributi.

Ad esempio, ecco un metodo della classe “controller” che gestisce l’apertura dello scrigno:



Altra caratteristica del controller è la lettura dei file di testo che contengono le descrizioni delle varie stanze. Tramite un BufferedReader, leggiamo le diverse righe dal file, salvandole in un array.

Il “Parser” invece legge e traduce le istruzioni dell’utente in una delle tante opzioni possibili, regolando il flusso di istruzioni nel “Main”.



# Manuale

Al percorso “LadroPerHobby/target” è possibile trovare il file LadroPerHobby.jar.

E’ eseguibile solamente tramite il comando “java -jar LadroPerHobby-1.0.jar” nel prompt dei comandi.