Relazione PMO

A.A. 2020/2021

Nicola Casagrande, 299165

Specifica del software:

Il progetto consiste nella simulazione di una coda di matchmaking, ovvero un tipo di coda che si ha in alcuni videogiochi online che distribuisce i giocatori in partite. All'avvio del programma verrà chiesto all'utente di selezionare il numero di giocatori che dovranno partecipare, a quel punto si hanno tre scelte: La prima è far scegliere i ranghi dei giocatori al programma in modo randomico; La seconda è utilizzare la console per l'inserimento manuale dei ranghi; Mentre la terza importa i ranghi da un file csv. Una volta fatto il programma comincerà a dividere i giocatori in base al loro rango in partite che verrano infine stampate, insieme agli eventuali giocatori rimasti in coda, sia su console che su file csv.

Studio del problema:

Punti critici:

1. Ricezione e utilizzo dei ranghi da file csv
2. Collocazione di giocatori in partite
3. Scrittura dei risultati su file csv

Scelte di progetto:

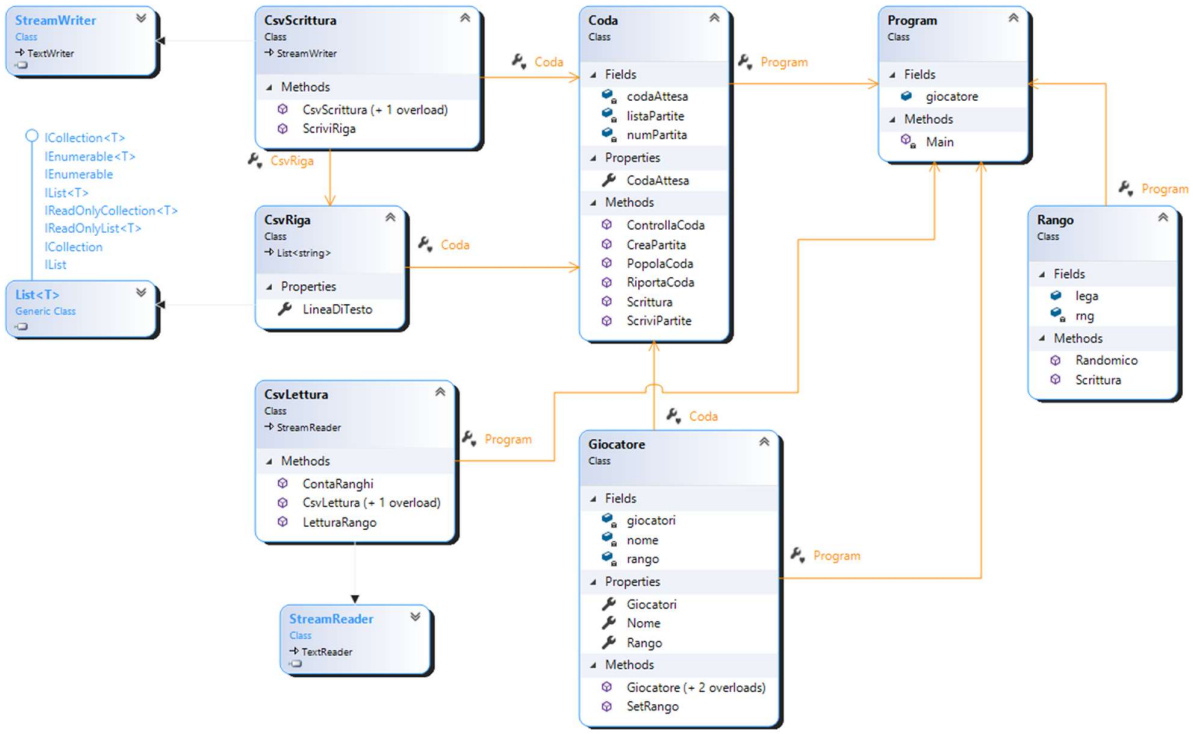
1.Per la risoluzione di questo problema ho deciso di utilizzare uno stream reader, questi andrà a prendere in input ogni riga del file csv che gli sarà dato e la analizzarà per vedere se il contenuto di essa è effettivamente un valore intero che può essere utilizzato come rango.
Questo avviene attraverso due metodi, la prima conterà quanti sono effettivamente i ranghi validi contenuti all'interno del file, questo perchè il programma deve sapere se il numero dei giocatori selezionato dall'utente e il numero dei ranghi nel file corrisponde per poter procedere. Il secondo metodo invece servirà ad assegnare i ranghi ai vari giocatori.

2.Qui ho optato per l'utilizzo di una "chiave", la *codaAttesa* conterrà i vari giocatori creati in ordine sparso, decido quindi come prima cosa, di ordinarli in ordine crescente in base al loro rango, così da avere i ranghi più bassi all'inizio e i più alti alla fine. Dopodichè preleverò il primo giocatore della *codaAttesa* e utilizzerò il suo rango come chiave, quindi basandomi sul questo il programma cercherà, controllando la *codaAttesa* altri nove giocatori con un rango di ±1. In caso vengano trovati verrà di fatto creata una partita, in caso contrario i giocatori che erano stati raccolti rivengono messi in coda e si cambia la chiave passando al rango del giocatore in coda successivo. Una volta che il programma avrà provato a creare una partita con tutte le chiavi terminerà, e stamperà sia a schermo che a file le partite create e gli eventuali giocatori rimasti in coda.

3.Qui invece ho utilizzato uno stream writer, il programma non farà altro che leggere la lista di stringhe con all'interno i dati dei giocatori, trasformarle in righe leggibili da un file csv e stamparle su di esso.

Scelte architetturali:

Diagramma delle classi:



Descrizione dell'architettura:

L'architettura del programma è basata su classi che lavorano insieme per raggiungere la terminazione del programma. Nella fattispecie abbiamo la classe ***Giocatore*** che altri non è che il giocatore stesso, che avrà parametri come il nome e il rango. La classe ***Rango*** invece servirà a generare i ranghi, al suo interno abbiamo due metodi: quello per la selezione automatica e randomica, e quello per la selezione manuale da console.

La classe ***Coda*** servirà alla popolazione, creazione e gestione della stampa di se stessa.

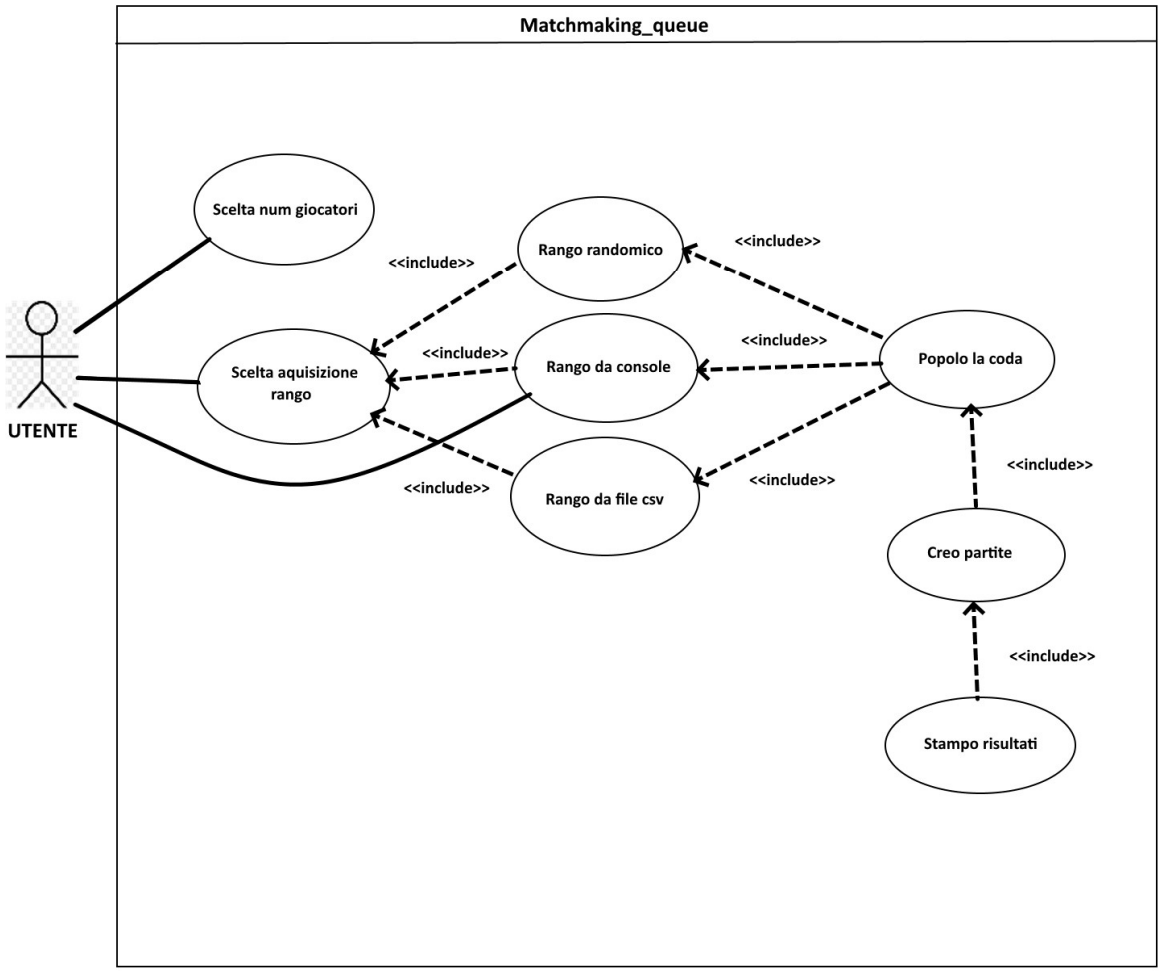
Al suo interno ho sei metodi, *PopolaCoda()* serve, come da nome, a popolare la coda di giocatori oltre che successivamente ordinarli in ordine di rango crescente. *CreaPartita()* servirà anche qui come da nome, a creare delle partite utilizzando il rango di un giocatore come chiave. *ControllaCoda()* controlla se ho giocatori in coda una volta che ho creato tutte le possibili partite. *RiportaCoda()* serve a mettere l'eventuale coda di giocatori rimanente nella lista di stringhe che andrà poi stampata. *ScriviPartite()* serve per riportare la

lista finale dei giocatori all'interno di *Scrittura()* che non farà altro che mandare in stampa, riga per riga, la lista.
La classe ***CsvLettura*** invece è composta da due metodi, entrambi utilizzati per leggere il file csv. Il primo metodo *ContaRanghi()* servirà al programma per contare effettivamente quanti ranghi sono presenti all'interno del file csv, questo servirà per valutare se il numero di giocatori scelto dall'utente corrisponde al numero di ranghi presente sul file. Dopodichè *LetturaRango()* non farà altro che assegnare ogni valore presente sul file al rango di un giocatore.
CsvScrittura è la classe invece dedita a riportare le partite (e eventuali giocatori rimasti in coda) su file, al suo interno avrà un solo metodo, *ScriviRiga()* che prenderà in pasto una riga e la scriverà su file.
CsvRiga infine non è altro che una lista di stringhe che rapresentano le righe.

Documentazione sull'utilizzo:

L'intero progetto viene fornito sull'apposita repository di GitHub, se si ha intenzione di utilizzare come input il file csv si consiglia di prestare attenzione a quanti ranghi vengono inseriti su file, in quanto se il numero di essi non corrisponderà al numero dei giocatori al lancio del programma quest'ultimo riporterà un errore.

Use Cases con relativo schema UML:



Una volta selezionato il numero di giocatori, come decidere i ranghi e in caso i ranghi stessi il programma eseguirà il tutto in modo autonomo.