BURRACO TOURNAMENT

Progetto per la materia “Sistemi di Elaborazione” a cura di

Giorgio Romano

Il progetto "Burraco Tournament" ha come scopo la realizzazione di un programma che permetta all'utente di essere facilitato nella creazione e gestione di un torneo del gioco di carte chiamato "Burraco".

Il Progetto si avvale delle classi ***SWING*** per creare un interfaccia grafica utile per semplificare ulteriormente l'utilizzo.
Sono implementate nel progetto differenti tecniche di trattamento delle informazioni, quali interfacciamento con database, parsing e writing di documenti xml, e utilizzo di socket per un sistema di storage remoto di informazioni.

Sono stati sviluppati due programmi. Un programma gestisce le fasi client , ed è il programma più complesso e indipendente. Successivamente è stato creato un programma lato server che si occupa solamente di accettare richieste del client per poter salvare o caricare informazioni remote.

Funzionalità Base

Il programma è strutturato dando la possibilità all'utente di poter gestire in toto un torneo di burraco.
Alcune funzionalità:

- Creazione di una anagrafica base con nome e cognome
- Creazione di tornei con accoppiamenti già effettuati o con una metodologia random la cui gestione è a cura del programma
- Inserimento e visualizzazione di punteggi e classifiche
- Salvataggio dello stato del torneo attraverso salvataggio dati in xml
- Memorizzazione in database di informazioni riguardanti ai precedenti tornei di ogni singolo giocatore

Client

Classi base

I Fondamenti del programma lato client sono le due classi **SINGLE** e **COPPIA**, grazie alle quali sono organizzati i concetti di giocatore,e quello di coppia di due giocatori, che prendono parte alla partita.

Nella classe **Single** sono presenti delle variabili di informazione che permettono salvataggio di **Anagrafica**,**Punteggi**, la situazione di accoppiamento o meno, 2 **ID**, uno interno al torneo in corso, e uno stante a indicare l'id generale di riscontro su un database MYSQL

Nella classe COPPIA vengono associati due istanze di classe SINGLE grazie agli id interni al torneo, oltre ai punteggi. Inoltre viene inserito un riferimento a un **ArrayLis**t contenente tutti i giocatori del torneo per aiutare nel **toString** della coppia.

Entrambe le classi implementano l'interfaccia **Comparable** che permette di poter ordinare correttamente tutte le coppie, o single, contenuti in una lista,invocando il metodo **Sort** presente in LIST

Tutti gli array contenti invocazioni di classe **Single** sono di tipo **SingleList** , una estensione della classe **ArrayList**, creata per poter implementare un metodo ***findSingle(int id)*** che ritorna un oggetto di tipo **Single** contenuto in quella lista con id specificato.

Un altra classe fondamentale per il funzionamento del programma è la classe **Tavolo**
In un tavolo durante una partita di burraco sono sedute due coppie, oppure quattro giocatori singoli. Il comportamento di questa classe rispecchia questa situazione.
Infatti la classe Tavolo è una classe base dal quale vengono estese due classi ulteriori, **TavoloCoppie** e **TavoloSingoli**,a seconda di quale è la tipologia del torneo, se a coppie o ognuno per se.

Di base la classe Tavolo presenta **4 INT** di tipo protetto, che identificano l'id del tavolo,il numero di "smazzate" da effetture,e i punteggi per ogni coppia.
Inoltre viene implememenata l'interfaccia **Tables**, ove sono presenti due metodi, ***assegnapunteggi***, e ***annullapunteggi***. La classe Tavolo è astratta infatti non può essere istanziata ,inoltre vengono inseriti due metodi *toString* astratti riferiti alla coppia1 e alla coppia2.

La classe ***TavoloCoppie*** presenta a parte i costruttori anche un metodo ***findCoppia*** utile per trovare una data coppia inserita nelle liste contenti le coppie presenti in questa classe, inoltre vengono implementati tutte le classi astratte.
Ugualmente viene implementata la classe ***TavoloSingoli***,dove però avviene un override del metodo ***equals*** per verificare se due tavoli contengono gli stessi giocatori.

Classi di organizzazione

Il torneo è diviso in un numero n di turni, a scelta dell'utente
Questa organizzazione viene implementato attraverso la classe **Turno**

Nella classe Turno è presente un Array all'interno del quale vengono inseriti un numero n di istanze della classe Tavolo , ovviamente ognuna relativa allo specifico tavolo; inoltre sono presenti diversi ArrayList con l'elenco delle coppie e dei singoli giocatori (classi single e coppie).

Ulteriormente sono presenti nel codice della classe Turno istanze della classe **Urna**, che servono per "estrarre" in modo random dei numeri attraverso uno specifico algoritmo; essi serviranno per poter comporre i vari tavoli di ogni turno. Infine sono presenti alcune variabili che servono a memorizzare alcune informazioni.

Alcuni costruttori della classe Turno permettono la sola memorizzazione di valori, altri utilizzano Urne e algoritmi per poter comporre da zero tutte le associazioni tra coppie , turni e tavoli

Tutte queste precedenti classe permettono la realizzazione di un unica classe finale , una sorta di raccoglitore di tutti i metodi e dati del torneo, la classe **Torneo**; essa serve a operare in modo totale su ogni aspetto del torneo. Avviarlo, concluderlo, modificare variabili, ecc...

Classi di Collegamento

La classe **MainClass** serve a fare da collegamento tra gli algoritmi contenuti in una istanza **Torneo**, e l'interfaccia grafica; infatti nel metodo ***main*** della gui verrà istanziato uno oggetto di tipo MainClass.

Particolare è la classe **CreationTables**, che è utile alla creazione dei tavoli per ogni turno. Similmente alla classe TAVOLO, questa classe è di tipo astratto, e viene estesa da **CreationTablesCoppie** e **CreationTablesSingles**a seconda di quale sia il tipo di torneo in corso.

Infine la classe **ConnectDatabase** permette il dump e l'update del database remoto, dove vengono inseriti i risultati per ogni torneo per ogni giocatore, e dal quale durante l'atto di creazione del torneo l'utente può estrapolare i dati di un giocatore che abbia già partecipato a un torneo precedente.

Le classi **XmlWriter e XmlParser** servono a salvare e caricare un file xml all'interno del quale vengono inseriti tutti i dati di un torneo, per poter essere recuperati in un momento successivo. Allegato si trova il DTD.

Classi GUI

La classe principale della della GUI è **MainGui**, che essendo la classe base estende **JFrame**.
Al suo interno troviamo diversi oggetti che estendono le classi **JDialog e JPanel**, un istanza di tipo **MainCLasse**, e gli oggetti utili alla creazione del **JMenu**.

Le classi che estendon JPanel, ovvero **StartedPanel e NotStartedPanel**, contengono dei JButton e dei JLabel, e inoltre presentano dei listener per alcune operazioni effettuate in dei JDialog istanziati dalla pressione di alcuni bottoni presenti in questi pannelli

Questi JDialog sono:
DialogAdd
DialogEdit
CoupleDialogList

DialogAdd permette l'interazione con l'aggiunta di giocatori, o coppie al torneo, infatti da esso vengono estese le due classi **CoupleDialogAdd e SingleDialogADdd**.
In DialogAdd vengono istanziati tutti i bottoni e label , e vengono inseriti nel layout. Nelle due estensioni vengono inseriti i testi dei bottoni e delle lables, e solo in Single Dialog Add, vengono rimossi quelli non necessari.
Ognuna delle due classi ha un ovverride per quanto riguarda il comportamento dei bottoni, poichè a seconda se il torneo sia di singoli, o a coppie devono essere chiamati metodi e classi differenti.

In questi due JDialog sono presenti due bottoni per poter istanziare un JDialog di tipo **DialogFromDatabase**, che utilizzando la classe ConnectDatabase, genera una tabella dove vengono elencati tutti i giocatori presenti nel database remoto, con una media punteggio.
Cliccando due volte su un nome viene chiuso questo Dialogo e nei campi dell'aggiunta giocatore sottostanti vengono inseriti i dati di quel giocatore remoto,inserendo pure l'id di riferimento sul database.

DialogEdit si comporta in modo simile, ma serve per editare le coppie.

CoupleDialogList serve a mostrare le coppie e i singoli già inseriti nel torneo.

InfoDialogEdit viene instanziato alla pressione del bottone utile alla creazione da zero di un torneo, ed è utile per inserire le configurazioni iniziali, quali tipologia di torneo, numero di turni ecc..

Server & Networking

Come illustrato precedentemente nel progetto sono implementi strumenti di trasmissione e storage di informazioni
Questo avviene grazie a scambi di file xml tramite Socket TCP , o tramite semplici dump e update di un database remoto:
Nel database remoto sono presenti tre tabelle:
TORNEI - ID, NOME , NAMEFILE
GIOCATORI - ID,NOME
PUNTEGGI - ID_torneo , ID_giocatore, PUNTEGGIO

Nelle query di ConnectDatabase :
<<select *,SUM(punti) as somma , COUNT(*) as numero_partite from punteggi left outer join giocatori on giocatori.id=punteggi.id_giocatore group by punteggi.id_giocatore;>>
Possiamo notare un left outer join che mette in relazione la tabella giocatori, ottenendo cosi il nome di un giocatore associato a un id nella tabella punteggi, e avviene una somma di tutti i punteggi e un conteggio del numero di record nella table punteggi per ogni giocatore.

La tabella tornei serve anche a un programma lato server per poter memorizzare salvataggi remoti dello stato di un torneo.
Infatti le classi **ClientMode e ClientToSendRequest** del client permettono la creazione e gestione di file xml e il loro invio. Le comunicazioni avvengono attraverso l'utilizzo di 3 porte.
Lato Server il programma mantiene in esecuzioni tre Thread che contengono delle socket in ascolto su tre porte diverse.
Viene altresi utilizzata un ulteriore classe **XmlConnectionToServer** per effettuare parsing e writing di file xml utili al client per questa comunicazione. Lato server ciò viene gestito internamente dalle singole classi thread.

Fondamentalmente le comunicazioni tra client e server avvengono attraverso questi passi:

1. Client si connette al server
2. Server effettua un dump dal database della tabella torneo
3. Server manda al Client un file contenente tre tre informazioni per ogni torneo, id,nome, path del file memorizzato sul server riguardante quel dato torneo
4. Client sceglie il torneo da caricare (attraverso interfaccia JDialog con tabella)
5. Client compone un file xml al cui interno viene inserito un elemento con la stringa del file richiesto
6. Client manda questo file al server
7. Server riceve file, effettua il parsing, seleziona file indicato e lo invia
8. Client riceve il file xml , che è tale e quale a un file xml di salvataggio generato localmente, quindi lo manda al metodo che gestisce un caricamento essendo una istanza di tipo **File** esattamente come quella richiesta dal metodo di caricamento interno al client
9. Il Torneo viene caricato

Per quanto riguarda il salvataggio esso viene gestito attraverso una semplice comunicazione del client che manda il file salvato localmente al server, che lo gestisce controllando il nome. Se gia presente nel database sovrascrive il file di salvataggio precedente, altrimenti crea un nuovo record.

Sviluppi futuri & Conclusioni

Nel corso della realizzazione del progetto sono stati utilizzati tutti gli strumenti studiati nel corso delle lezioni. Collezioni di dati (sono presenti arraylist e hashmap) ,Ereditarietà delle classi , Interfacce , Networking ecc..
Inoltre sono stati utilizzati strumenti addizionali come la creazione di una interfaccia grafica.
Sono state elaborate diverse versioni del prodotto , partendo da un alpha a febbraio 2014, per arrivare ad una versione finale 1.1.1 quale quella presentata in data odierna.
Ovviamente possono essere presenti alcuni bug seppur il programma presenti solo qualche migliaio di righe di codice e prego i fruitori del progetto di segnalarmeli a lavoro@giorgioromano.it

Bisogna altresi sottolineare che il programma è stato sviluppato per esigenze personali dell'autore e non solo per poter soddisfare le richieste del programma; per cui si conta di continuare lo sviluppo nel futuro attraverso l'implementazione di sistemi di stampa delle informazioni, una migliore gestione dell'anagrafica, e un algoritmo piu efficente nella compilazione dei tavoli.

Il codice sarà rilasciato sul sito www.giorgioromano.it con licenza CreativeCommons