Programmazione ad Oggetti

Relazione per progetto

GuessWho

Di Tullio Dalila e Masi Elisabetta

1 Marzo 2015

Capitolo 1 - Analisi

1.1 Requisiti

Il software GuessWho mira alla ricreazione del classico gioco da tavolo, “Indovina chi?”. Il programma dove essere in grado di implementare una semplice partita al famoso gioco, dove due utenti si sfideranno nell’indovinare per primo il personaggio dell’avversario.
In ogni turno i due giocatori si alternano, avendo così la possibilità di porre una specifica domanda e ottenere la relativa risposta.

1.2 Problema

L’applicazione deve essere in grado di dirigere i due giocatori e l’insieme di tutte le loro attività eseguite ad ogni turno. La difficoltà principale è nel controllare lo scambio altalenante di informazioni tra questi, assicurando una corretta gestione delle medesime.

Le diverse situazioni in cui il programma può trovarsi vengono gestite attraverso un insieme di stati (come ad esempio: MenuState, LevelState1, ecc ..).

Creato ogni livello, l’elaborazione dell’applicazione si divide tra i due giocatori (user e server) che eseguono in modo parallelo le proprie attività e comunicano tra loro attraverso un’ unica classe che ne gestisce l’alternanza .

Capitolo 2 - Design

2.1 Architetture

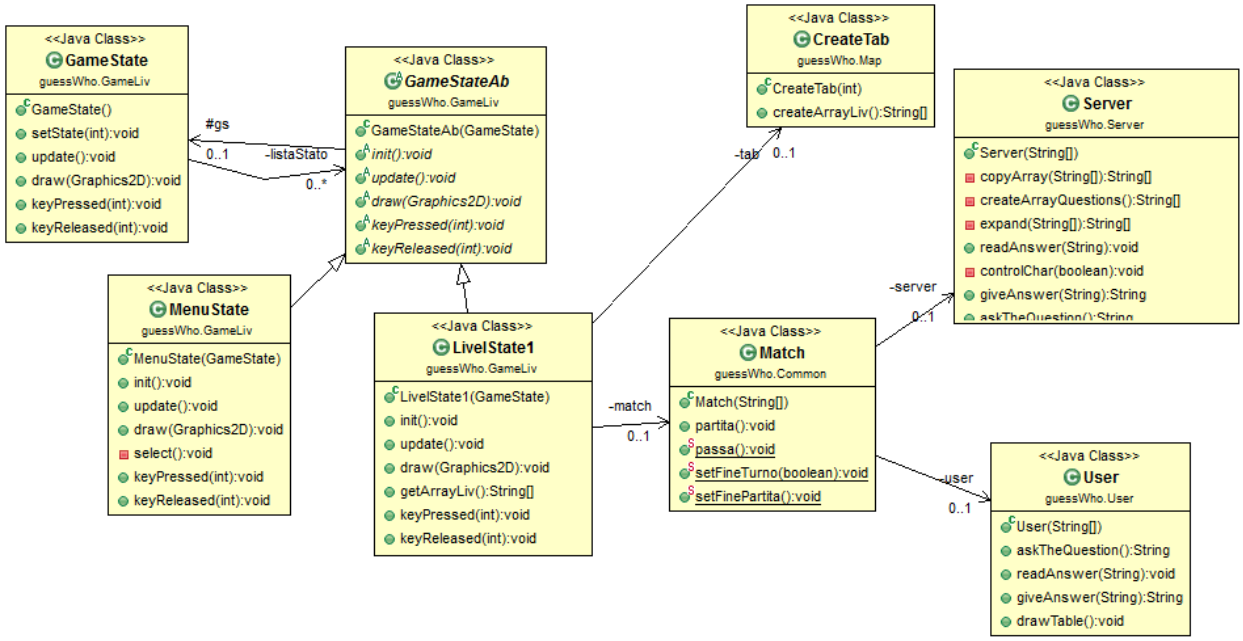
Come parte architetturale abbiamo implementato il pattern MVC.
In visualizzazione (View), abbiamo creato una classe che di volta in volta implementa lo stato di un nuovo livello e quindi disegna e crea il suo

contenuto. Come Model utilizziamo una classe astratta che costituisce la struttura di un livello generico. In particolare questa classe, a sua volta, dà vita alla costruzione della singola partita, avviando l'esecuzione e la gestione dell'attività dei due giocatori. Con tecniche similari viene gestito User. L'implementazione della gestione della sua attività è, infatti, tenuta separata dalla visualizzazione a schermo dei suoi risultati.

Ogni livello poi, implementerà questa classe e quella del Server specificandone nel singolo le particolarità.

Scendendo nei dettagli di queste classi abbiamo:

- View è costituita principalmente da una classe GameState che, attraverso un array di oggetti, i quali estendo la classe astratta GameStateAb, carica i vari stati. Se lo stato della classe è diverso da uno dei livelli, l'applicazione richiamerà MenuState implementando la pagina iniziale con il menu di scelta.
- Server è composto da un'unica classe omonima che contiene una serie di metodi che controllano e modificano le componenti del gioco.
- User è composto dalle classi che controllano e modificano il tabellone grafico del gioco: User e TableGame, . La differenza essenziale tra server e user sta nell'implementare, nella prima, una strategia di gioco da seguire, nella seconda , una visione grafica interattiva del gioco stesso.



2.2 Design dettagliato

Come costruzione dettagliata, abbiamo costruito i packege:

- guessWho.Main: da cui parte l'intero gioco,viene gestito l'avvio del thread e quindi la creazione del frame iniziale e del pannello su cui si visualizzeranno le diverse pagine di gioco.
- guessWho.GameLiv: gestiste il caricamento dei diversi stati in cui si può trovare il programma.
- guessWho.Map: crea di volta in volta l'array contenente ogni singolo personaggio che comporrà il livello.
- guessWho.Background: gestisce le immagini di sfondo del frame iniziale.
- guessWho.Common: contiene tutte le classi che vengono implementate da entrambi i giocatori. Esse permettono di gestire l'attività comune ad essi collegandoli così tra loro.
- guessWho.Server: è formato dalla classe omonima che gestiosce l'intera strategia di gioco del secondo giocatore, ovvero il computer.
- guessWho.User: ha al suo interno due classi: User che, essendo il vero e proprio giocatore, effettua i controlli necessari e crea, attraverso TableGame, il tabellone grafico di gioco, ovvero il nuovo frame.

Gestione dei giocatori:

- User:
  Quando l'Utente sceglierà nel menù principale di iniziare la partita si creerà un nuovo frame con il tabellone di gioco e gli si aprirà un popup che gli permetterà di inserire la domanda desiderata. A ogni turno gli verrà anche data la possibilità di comunicare il personaggio da indovinare. Ciò determinerà, oltre alla sua vittoria o sconfitta, la fine della partita.
  Ottenuta la risposta da parte dal server, egli procederà bloccando, secondo la sua totale discrezione, i personaggi che corrispondono, o meno, alla caratteristica richiesta. Comunicherà al server la terminazione del turno attraverso il pulsante "FineTurno".
- server:
  Quando gli è posta la domanda egli eseguirà una serie di metodi che effettuano controlli sulla lettura e sulla presenza dalla domanda

richiesta all'interno delle caratteristiche del proprio personaggio. Verrà, poi, che restituita la risposta, positiva o negativa, a seconda del risultato. Una volta ottenuto il turno, i ruoli sono invertiti: il server effettua la domanda, attende la risposta e gestisce l'array dei propri personaggi a seconda della risposta data. Questo metodo di gestione si basa su una ricerca della parola data all'interno delle caratteristiche di ogni personaggio rimasto in campo. La formulazione della domanda, invece, viene implementata attraverso l'estrazione a sorte di un componente contenuto in un array di caratteristiche.

Tabellone:

Creato il livello di gioco viene generato un array basato su un' enumeration che contiene tutti i personaggi e le loro caratteriste. L'array di livello è generato nella classe CreateTable e viene passato all'utente, il quale creerà sulla sua base tutta la parte grafica e effettuerà l'estrazione della sua carta da gioco. Quest'ultima viene estratta dalla classe ExtractionCard, la quale implementa l'interfaccia IExtractionCard. La stessa cosa verrà fatta anche nel server, il quale, però, non creerà la parte grafica, bensì una strategia di gioco. Attraverso metodi di controllo e ricerca basati sugli array, creati partendo da quello di livello, ricreerà un comportamento simile a quello di un secondo utente esterno.

Gestione tabellone grafica:

Dalla classe User viene creato un pannello raffigurante l'intero tavolo da gioco appartenente al giocatore stesso. Verranno quindi visualizzati una seri di bottoni con cui l'utente potrà svolgere e impostare la propria tecnica di gioco. Durante poi ogni turno verranno visualizzati una serie di pop-up con cui l'utente può interagire con il server, inviando e ricevendo informazioni.

Capitolo 3 - Sviluppo

3.1 Testing

L'applicazione è stata testata su PC:

- Acer ASPIRE V3 – 771G, Intel Core i5, Windows 7
- Asus X53S, Intel Core i7, Windows 7

Si è inoltre utilizzato il test JUtil per verificare se le collisioni venivano effettuate correttamente.

3.2 Divisione dei compiti e metodologia di lavoro

I compiti sono stati suddivisi nel seguante modo:

Elisabetta Masi:

- Creazione e gestione user
- Creazione tabellone grafico

Dalila Di Tullio

- Creazione e gestione server

In comune:

- Struttura principale, ovvero lo scheletro del gioco, e modellazione dei principali stati che dovevano essere gestiti

Capitolo 4 - Commenti finali

Essendo stata la prima esperienza di programmazione ad oggetti e di sviluppo di giochi, abbiamo ritenuto utile strutturare le basi di quest'ultimo insieme e di sviluppare la parte centrale del software individualmente.

Per svariati imprevisti personali non siamo riuscite a completare l'idea originale del progetto ma abbiamo considerato opportuno rispettare i termini stabiliti dalla deadline sviluppando una versione più semplice ma funzionante

del programma. Se eventualmente lo riteniate non completamente idoneo all’idea iniziale possiamo apportarvi le giuste modifiche.