Prova Finale di Ingegneria Del Software

Anno Accademico 2019-2020

Requisiti

Contents

1 Introduzione

Il progetto consiste nello sviluppo di una versione software del gioco da tavolo *Santorini*.
Il progetto finale dovrà includere:

- diagramma UML iniziale dell'applicazione (ad alto livello);
- diagrammi UML finali che mostrino come è stato progettato il software, i diagrammi potranno essere generati a partire dal codice sorgente del progetto utilizzando tool automatici;
- implementazione funzionante del gioco conforme alle regole del gioco e alle specifiche presenti in questo documento;
- codice sorgente dell'implementazione;
- codice sorgente dei test di unità.

Data di consegna: Venerdì 03 Luglio 2020 23:59:59 CEST
Data di valutazione: Da fissare (a partire da Lunedì 6 Luglio 2020)

Modalità di valutazione, aule e orari, verranno comunicate successivamente.

2 Requisiti di Progetto

I requisiti del progetto si dividono in due gruppi:

I **Requisiti game-specific** che riguardano le regole e le meccaniche del gioco.

II **Requisiti game-agnostic** che riguardano aspetti di design, tecnologici o implementativi.

2.1 Requisiti Game-specific

Le regole del gioco sono descritte nei file `santorini-rules.pdf`, caricati su BeeP.
I nomi di classi, interfacce, le variabili, ed in generale tutti gli identificativi nel codice dovranno essere in lingua inglese. Sempre in inglese dovranno essere anche i commenti nel codice e la documentazione tecnica (JavaDoc).
Per la valutazione (vedi Tabella 1) si fa riferimento a due possibili set di regole: le regole semplificate e le regole complete.

- **Regole Semplificate**: deve essere possibile completare una partita con esattamente due giocatori; devono essere supportate le carte dalla 1 alla 5 del manuale di gioco.
- **Regole Complete**: deve essere possibile completare una partita con due oppure tre giocatori; devono essere supportate tutte le carte "divinità semplici" (simple gods) del manuale di gioco, ad esclusione di Hermes.

Ogni giocatore è identificato da un *nickname* che viene impostato lato client e deve essere univoco in ogni partita. L'univocità del *nickname* deve essere garantita dal server in fase di accettazione del giocatore.

2.2 Requisiti Game-agnostic

In questa sezione vengono presentati i requisiti tecnici dell'applicazione.
Il progetto consiste nell'implementazione di un **sistema distribuito** composto da un *singolo* **server** in grado di gestire una partita alla volta e *multipli* **client** (uno per giocatore) che possono partecipare ad una sola partita alla volta. Si richiede l'utilizzo del pattern **MVC** (Model-View-Controller) per progettare l'intero sistema.

2.2.1 Server

Di seguito la lista dei requisiti tecnici per il lato server.

- Deve implementare le regole del gioco utilizzando *JavaSE*.
- Deve essere istanziato una sola volta al fine di gestire una singola partita (tranne nel caso in cui venga implementata la funzionalità avanzata "partite multiple").

2.2.2 Client

Di seguito la lista dei requisiti tecnici per il lato client.

- Deve essere implementato con *JavaSE* ed essere instanziabile più volte (una per giocatore).
- L'interfaccia grafica deve essere implementata mediante Swing o JavaFX.
- Nel caso in cui venga implementata sia un'interfaccia testuale (CLI) che un'interfaccia grafica (GUI), all'avvio, deve permettere al giocatore di selezionare il tipo di interfaccia da utilizzare.

Si assume che ogni giocatore che voglia partecipare ad una partita conosca l'indirizzo IP o lo URL del server. Quando un giocatore si connette:

- Se non ci sono partite in fase di avvio, viene creata una nuova partita, altrimenti l'utente entra automaticamente a far parte della partita in fase di avvio.
- Il giocatore che crea la partita sceglie il numero di giocatori che ne fanno parte (2 o 3).
- Se c'è una partita in fase di avvio, il giocatore viene automaticamente aggiunto alla partita.
- La partita inizia non appena si raggiunge il numero di giocatori atteso (2 oppure 3 in base alla scelta effettuata dal primo giocatore in fase di creazione della partita).

Il server consente ai vari giocatori di svolgere i propri turni secondo le regole del gioco. È necessario gestire sia il caso in cui i giocatori escano dalla partita, sia il caso in cui cada la connessione di rete. In entrambi i casi la partita dovrà terminare e tutti i giocatori verranno notificati.

2.3 Funzionalità Avanzate

Le funzionalità avanzate sono requisiti **facoltativi** da implementare al fine di incrementare il punteggio in fase di valutazione. Queste funzionalità vengono valutate solo se i requisiti game-specific e game-agonistic presentati nelle sezioni precedenti sono stati implementati in maniera sufficiente. Le funzionalità avanzate implementabili sono:

- **Partite Multiple**: Realizzare il *server* in modo che possa gestire più partite **contemporaneamente**, dopo la procedura di creazione della prima partita, i giocatori che accederanno al server verranno gestiti in una *sala d'attesa* per creare una seconda partita e così via.
- **Persistenza**: Lo stato di una partita deve essere salvato su disco, in modo che la partita possa riprendere anche a seguito dell'interruzione dell'esecuzione del server. Per riprendere una partita, i giocatori si devono ricollegare al server utilizzando gli stessi *nickname* una volta che questo sia tornato attivo. Si assume che il server non interrompa la propria esecuzione durante il salvataggio su disco e che il disco costituisca una memoria totalmente affidabile.
- **Divinità Avanzate (Advanced Gods)**: Implementare 5 divinità avanzate (advanced gods) a propria scelta tra quelle presenti nel manuale di gioco (numeri 11–30).
- **Undo**: Implementare la funzionalità di *undo*, che permette a un giocatore di annullare la propria mossa entro un periodo di 5 secondi da quando l'ha effettuata, prima di passare il turno.

3 Valutazione

In Tabella 1 sono riportati i punteggi **massimi** ottenibili in base ai requisiti implementati.

Requisiti Soddisfatti	**Voto Massimo**
Regole Semplificate + CLI + Socket	18
Regole Complete + CLI + Socket	21
Regole Complete + GUI + Socket	24
Regole Complete + GUI + Socket + 1 FA	27
Regole Complete + CLI + GUI + Socket + 1 FA	30
Regole Complete + CLI + GUI + Socket + 2 FA	30L

Table 1: Tabella di valutazione (FA=Funzionalità avanzata)

Di seguito si riportano gli aspetti che vengono valutati e contribuiscono alla definizione del puntegio finale:

- La qualità della *progettazione*, con particolare riferimento ad un uso appropriato di interfacce, ereditarietà, composizione tra classi, uso dei design pattern (statici, di comunicazione e architetturali) e divisione delle responsabilità.
- La *stabilità* dell'implementazione e la *conformità alle specifiche*.
- La *leggibilità del codice* scritto, con particolare riferimento a nomi di variabili/metodi/classi/package, all'inserimento di commenti in inglese e documentazione JavaDoc in inglese, la mancanza di codice ripetuto e metodi di eccessiva lunghezza.
- L'*efficacia* e la *copertura* dei casi di test, il nome e i commenti di ogni test dovranno chiaramente specificare le funzionalità testate e i componenti coinvolti.
- L'utilizzo degli strumenti (IntelliJ IDEA, Git, Maven, ...).
- L'*autonomia*, l'*impegno* e la *comunicazione* (con i responsabili e all'interno del gruppo) durante tutte le fasi del progetto.