Prova Finale di Ingegneria del Software: Requisiti

Gian Enrico Conti, Niccolò Izzo

Contents

1 Introduzione

Il progetto consiste nello sviluppo di una versione software del gioco da tavolo *Adrenalina*.
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 non dovranno 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: Domenica 07 Luglio 2019 23:59:59 CEST
Data di valutazione: Da fissare (a partire da 8 Luglio 2019)

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 `adrenalina-rules-it.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**: Si considera solo la modalità o effetto base di tutte le armi; non sono presenti la "frenesia finale" e le "azioni adrenaliniche".
- **Regole Complete**: Si considerano tutte le modalità delle armi; sono presenti la "frenesia finale" e le "azioni adrenaliniche".

Sia nelle regole semplificate che nelle regole complete non sono incluse le modalità "dominazione", "torrette" e l'aggiunta di un "terminator".
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").
- Nel caso in cui venga implementata la comunicazione client-server *sia* via socket *sia* via RMI, deve poter supportare partite in cui i giocatori utilizzano tecnologie diverse.

2.2.2 Client

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

- Deve essere implementato con *JavaSE* ed essere instanziabile piú volte (una per giocatore).
- L'interfaccia grafica deve essere implementata mediante Swing o JavaFX.
- Nel caso in cui venga implementata la comunicazione client-server *sia* via Socket *sia* via RMI, all'avvio deve permettere al giocatore di selezionare la tecnologia da utilizzare.
- 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.

2.2.3 Avvio della partita

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.
- Se c'è una partita in fase di avvio, il giocatore viene automaticamente aggiunto alla partita.
- La partita inizia non appena si raggiungono i 5 giocatori. Quando 3 giocatori si connettono a una partita viene inizializzato un timer di N secondi, caricato da un file di configurazione presente lato server o specificato tramite parametri *command line*. Se non si raggiungono 5 giocatori entro la scadenza del timer la partita inizia comunque con il numero di giocatori raggiunto, a patto che questo sia $>= 3$. Se prima della scadenza del timer il numero di giocatori in attesa scende sotto i 3, il timer viene resettato.

2.2.4 Corso 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.

- Ogni giocatore ha un tempo predefinito per eseguire le mosse (caricato da file di configurazione o indicato come parametro sulla *command line*). Il server attende per il periodo di cui sopra, dopo il quale sospende il giocatore, i.e. il giocatore non esegue mosse ma viene comunque considerato nel conteggio dei punti.
- Tutti i giocatori vengono infiormati quando un giocatore esce dal gioco o viene escluso secondo il punto preceednte.
- Il gioco continua, saltando i giri del giocatore sospeso.
- Il giocatore può riconnettersi e continuare il gioco.

Se in ogni momento dovessero rimanere meno di 3 giocatori in una partita in corso, il giocatore che ha il punteggio più alto in quel momento risulta vincitore. Si faccia riferimento al regolamento del gioco per il calcolo dei punteggi.

2.2.5 Funzionalità Avanzate

Le funzionalità avanzate sono requisiti **facoltativi** da implementare al fine di incrementare il punteggio in fase di valutazione. Si noti che queste funzionalità vengono valutate solo se i requisiti game-specific e game-agonistic presentati in Sezione 2 sono stati implementati in maniera sufficiente (vedi Sezione 3). I requisiti avanzati 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.
- **Modalità dominazione o torrette**: Questa funzionalità avanzata consiste nell'implementare una delle due modalità "dominazione" o "torrette", come specificate nel regolamento. L'implementazione di entrambe le modalità viene comunque considerata come un'unica funzionalità avanzata.
- **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 nick name 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.
- **Terminator**: Questa funzionalità avanzata consiste nell'implementare la possibilità di aggiungere un "terminator", come specificato nel regolamento del gioco.

3 Valutazione

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

Requisiti Soddisfatti	**Voto Massimo**
Regole Semplificate + CLI + (Socket o RMI)	18
Regole Complete + CLI + (Socket o RMI)	21
Regole Complete + CLI + (Socket o RMI) + 1 FA	22
Regole Complete + GUI + (Socket o RMI) + 1 FA	24
Regole Complete + GUI + Socket + RMI + 1 FA	27
Regole Complete + CLI + GUI + Socket + RMI + 1 FA	30
Regole Complete + CLI + GUI + Socket + RMI + 2 FA	30L

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

- 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.