Prova Finale

Srđan Krstić, Claudio Menghi

Politecnico di Milano

srdan.krstic@polimi.it, claudio.menghi@polimi.it

April 5, 2016

Introduzione

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

- diagramma UML iniziale dell'applicazione (ad alto livello)
- diagrammi UML finali che mostrino come è stato progettato il software;
- 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

La data di consegna: 3 Luglio 2016, AoE
La data di valutazione: TBD.


Simone Luciani
Daniele Tascini
Council of four

Game-specific requirements

Regole base

- Regole ufficiale del gioco sono descritte nel file **GameRules.pdf** (Inglese) e **RegoleGioco.pdf** (Italiano), caricati su BeeP.
- Ci sono buone spiegazioni anche su Youtube:
  `https://www.youtube.com/v/8fhU9SiI4eE?start=87&end=491&vq=hd1080`

Regole Addizionali

- **Gioco configurabile**: il gioco deve essere progettato al fine di poter creare:
  1. mappa configurabile
     1. connessioni arbitrarie tra le cittá;
     2. bonus arbitrari;
     3. numero di bonus arbitrari;
  2. numero arbitrario di giocatori;
  3. file di configurazione con lista di mappe (con 3 punti sopra). Numero dei giocatori deve essere un parametro alla creazione del gioco;
- **Market**: dopo che tutti i giocatori hanno giocato il loro turno (dopo ogni giro) viene lanciata la fase di Market. Partendo dal primo giocatore fino all'ultimo, ogni giocatore puó scegliere le carte (di permesso e politiche) e gli assistenti che intende vendere e il prezzo richiesto per ognuno di essi. Quando TUTTI i giocatori hanno scelto quali elementi vendere, seguendo un ordine casuale i giocatori possono comprare gli elementi messi in vendita dagli altri giocatori.

Game-agnostic requirements

Specifiche implementative

In questa sezione vengono presentati i requisiti tecnici dell'applicazione. Deve essere un sistema distributivo composto da un lato gioco che puó gestire molti giochi simultanei e multipli lati giocatore che possono partecipare nel un solo gioco. Si raccomanda l'utilizzo del pattern **MVC** (Model-View-Controller) per progettare l'intero sistema.

Lato Giocatore

- Il lato giocatore deve poter essere istanziato più volte.
- Il lato giocatore deve essere sviluppato obbligatoriamente utilizzando JavaSE.
- L'interfaccia grafica deve essere mediante Swing o altre tecnologie (tuttavia in questo caso non sarete supportati dai responsabili),
- Il lato giocatore deve supportare RMI e Socket, in relazione al numero di studenti del gruppo, come specificato in tabella 1.
- Nel caso in cui sia richiesta sia l'implementazione RMI che quelle per mezzo di socket, l'applicazione, all'avvio, deve permettere all'utente di selezionare il metodo utilizzato per la comunicazione.
- Nel caso in cui sia richiesta sia l'implementazione CLI che quelle per mezzo di GUI, l'applicazione, all'avvio, deve permettere all'utente di selezionare il metodo utilizzato per la visualizzazione.

Lato Gioco

Questo componente deve gestire le partite e deve poter essere istanziato una sola volta. Deve permettere di:

- creare una nuova partita, inizializzarla, giocarla e concluderla secondo le regole del gioco.
- Deve essere in grado di gestire più partite contemporaneamente.
- Deve essere implementato secondo la logica JavaSE.
- Nel caso in cui sia l'implementazione via socket che quelle via RMI sia richiesta, quando un giocatore si connette al server, il server deve comunicare utilizzando la connessione selezionata.

Avvio della partita

L'assunzione base è che ogni giocatore che voglia partecipare a una partita conosca l'indirizzo (numerico o simbolico) del lato gioco. Quando un giocatore si connette,

- se cè una partita in fase di avvio, il giocatore viene automaticamente aggiunto alla partita
- **Regole base:**[1] la partita inizia non appena si raggiungono i 4 giocatori. Quando 2 giocatori si connettono a una partita viene inizializzato un timeout di 20 secondi. Non appena il timeout scatta la partita inizia anche se non sono raggiunti i 4 giocatori.
- **Regole complete:**[2] quando 2 giocatori si connettono a una partita viene inizializzato un timeout di 20 secondi. Non appena il timeout scatta la partita inizia. Il numero di giocatori puó essere arbitrario, anche maggiore di 4.
- se non ci sono partite in fase di avvio, viene creata una nuova partita.

Si precisa che una nuova partita viene creata solamente quando un utente si connette e non ci sono partite in attesa, altrimenti l'utente entra automaticamente a far parte della partita in fase di avvio.

Corso della partita

Il lato gioco consente ai vari giocatori di svolgere i propri turni secondo le regole di gioco. E' necessario gestire il caso in cui i lati giocatore si disconnettano.

- ogni giocatore ha un periodo di tempo fissato passato come parametro al momento della creazione del server per eseguire le mosse
- se un giocatore va offline il lato gioco attende per il periodo di cui sopra, dopo il quale sospende il giocatore (nota il giocatore non esegue mosse ma viene comunque considerato nel conteggio dei punti etc.)
- tutti i giocatori vengono notificati della mancanza di un giocatore
- il gioco continua, saltando i giri del giocatore sospeso
- il lato giocatore puó riconnettersi e continuare il gioco se si sceglie di implementare la funzionalitá “Ripristino sessione giocatore”.

Funzionalita Avanzate

Funzionalita Avanzate

Di seguito sono proposte alcune funzionalità avanzate che concorrono alla valutazione. Attenzione: il loro contributo non è necessariamente additivo. Design e codice verranno comunque valutati in quanto tali e contribuiranno al giudizio globale.

- **Gestione degli utenti**
- **Lato Gioco persistente**
- **Ripristino sessione giocatore**
- **Generazione automatica delle configurazioni di gioco**
- **Lobby**
- **Chat**
- **AI**

Valutazione

Valutazione

Saranno oggetto di valutazione

- la qualità della **progettazione** con particolare riferimento a un uso appropriato di interfacce, ereditarietà, composizione tra classi, uso dei design pattern;
- la qualità della progettazione dell'**architettura dell'applicazione**; divisione della responsabilitá; progettazione della comunicazione;
- la stabilità dell'implementazione e la **conformità alle specifiche** date;
- la **qualità** e la **leggibilità** del codice scritto, con particolare riferimento a nomi di variabili/metodi/classi/package, all'inserimento di commenti e documentazione JavaDoc (preferibilmente in inglese), la mancanza di codice ripetuto e metodi di eccessiva lunghezza;
- la **qualità** 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;
- il corretto utilizzo degli strumenti (Eclipse, Git, Maven, ...);
- il livello di autonomia e impegno nello svolgimento del progetto.

Valutazione - Acronimi

- **Command Line Interface (CLI)**: è implementato come un'interfaccia testuale e i vari giocatori si alternano nei turni utilizzando la tastiera.
- **GUI**: consiste in un interfaccia grafica swing
- **RMI:** la comunicazione avviene mediante "Remote method invocation"
- **Socket**: la comunicazione avviene mediante messaggi scambiati attraverso socket. Lo studente deve autonomamente definire e implementare un protocollo di comunicazione tra i componenti distribuiti.

Valutazione

Voto Max	Numero studenti 1	2	3
18	Regole base	Regole complete	Regole complete
22	CLI + GUI	RMI + CLI	Socket + CLI
24	RMI + CLI	Socket + CLI	RMI + Socket + CLI
26	Socket + CLI	RMI + Socket + CLI	RMI + Socket + GUI
28	RMI + Socket + CLI	RMI + Socket + GUI	All Tech
30L	RMI + Socket + GUI	All Tech	1 Funzionalità Avanzata

Table: Tabella di valutazione

La tabella di valutazione presenta le funzionalità da implementare nel caso di gruppi da 2 e 3 studenti. Gli studenti di telecomunicazioni, possono fare i gruppi insieme di un massimo di 4 persone ed i requisiti applicati corrisponderanno al gruppo con una persona in meno, con l'eccezione di un gruppo di 1 persona. **Non è possibile fare i gruppi di 1 persona, la tabella 1 contiene le informazioni solo come riferimento agli studenti di telecomunicazioni.**

Valutazione

I gruppi da 2 o 3 studenti devono implementare le specifiche descritte in tabella 1 in relazione al punteggio desiderato. Nota: in tabella sono rappresentati i **massimi** punteggi ottenibili in relazione alle features implementate. Per esempio, per un gruppo di 2 studenti e un punteggio desiderato di al massimo 24 punti è necessario implementare: comunicazione con Socket e il Command Line Interface (CLI). All Tech significa che il gruppo deve implementare i due modi di comunicazione (RMI e Socket) e i due tipi di Interface (CLI e GUI).
Le funzionalità avanzate possono essere implementate da tutti gruppi e comportano punteggi aggiuntivi. Ovviamente per implementare queste funzionalità è necessario che TUTTO il resto del progetto sia implementato in maniera COMPLETA e ADEGUATA (copertura con test, ben commentata etc).


Domande?

Organizzazione Laboratorio

Date	Format	Topics
12 Aprile	P	**Focus:** Groups, Eclipse, Maven, Git, Sonar **Checkpoint:** Installed Eclipse, Formed groups & repositories
19 Aprile	D	**Focus:** UML design (Model), Architexa **Checkpoint:** Created project, Maven, Git, Sonar
10 Maggio	D	**Focus:** Architecture design (MVC) **Checkpoint:** UML design (Model)
24 Maggio	P/D	**Focus:** Testing, JUnit, TDD **Checkpoint:** Architecture design (MVC), Implementation
31 Maggio	P/D	**Focus:** Communication, Multithreading and Protocol design **Checkpoint:** Test coverage and documentation
7 Giugno	D	**Focus:** General Discussion **Checkpoint:** Communication implementation
14 Giugno	P/D	**Focus:** Swing and GUI design **Checkpoint:** Implementation
21 Giugno	D	**Focus:** General Discussion and Deployment **Checkpoint:** GUI design and Implementation

Table: Lab schedule (P = Presentation, D = Discussion)

TODO (entro 12/04/2016)

1. Create un'account su BitBucket (`https://bitbucket.org/`)
2. Formate i gruppi
3. Un membro del gruppo deve creare un repository in BitBucket, aggungere i propri compagni come "admin" o "write"
4. E aggungere nostro account con username "krle" (Srdjan Krstic) come "read".
5. Completate il modulo (una persona per gruppo): `http://goo.gl/forms/UIutpuCoaa`
6. Installate gli strumenti (riferimento a "beep" per le guide su Eclipse, Sonar e Architexa)
7. Leggete le regole del gioco e le requisiti di progetto (riferimento a "beep"). Preparate domande. Iniziate le discussioni su BEEP.