

**Prova Finale di Ingegneria Del Software**

**Anno Accademico 2023-2024**

**Requisiti**

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Requisiti di Progetto</b>	<b>3</b>
2.1	Requisiti game-specific . . . . .	3
2.2	Requisiti game-agnostic . . . . .	4
2.2.1	Server . . . . .	4
2.2.2	Client . . . . .	4
2.3	Funzionalità Avanzate . . . . .	5
<b>3</b>	<b>Valutazione</b>	<b>5</b>

## 1 Introduzione

Il progetto consiste nello sviluppo di una versione software del gioco da tavolo *Codex Naturalis*.

Il progetto finale dovrà includere:

- diagrammi UML di alto livello dell'applicazione, che mostrino, con tutti e soli i dettagli utili, il progetto generale dell'applicazione;
- diagrammi UML di dettaglio che mostrino tutti gli aspetti dell'applicazione. Questi 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;
- documentazione del protocollo di comunicazione tra client e server;
- documenti di peer review (uno relativo alla prima peer review e uno relativo alla seconda);
- codice sorgente dell'implementazione;
- documentazione Javadoc dell'implementazione (generata dal codice);
- codice sorgente dei test di unità.

**Data di consegna: venerdì 28 giugno 2024 13:00:00 CEST**

**Data di valutazione: da fissare (a partire da lunedì 1 luglio 2024)**

Modalità di valutazione, aule e orari, verranno comunicati successivamente dal docente responsabile e potranno variare da docente a docente.

## 2 Requisiti di Progetto

I requisiti del progetto si dividono in due gruppi:

- **Requisiti game-specific** che riguardano le regole e le meccaniche del gioco.
- **Requisiti game-agnostic** che riguardano aspetti di design, tecnologici o implementativi.

### 2.1 Requisiti game-specific

Le regole del gioco sono descritte nei due file `CODEX_Rulebook_[ITA-ENG].pdf`, caricati sul sito WeBeeP del corso. In caso di discordanza tra tali file si faccia riferimento alla versione italiana.

I nomi di classi, interfacce, metodi, 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). La lingua dell'interfaccia utente dell'applicazione può essere scelta liberamente tra inglese e italiano.

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 consideri una versione semplificata del gioco che non comprenda le carte obiettivo personale (obiettivo segreto) e i relativi punteggi. Le carte obiettivo comune sono invece incluse.
- **Regole Complete:** si considerino tutte le regole per lo svolgimento di normali partite, come indicato nel manuale del gioco.

Ogni giocatore è identificato da un *nickname* che viene impostato lato client. Tale nickname deve essere univoco, nel senso che non possono esserci due utenti collegati contemporaneamente con lo stesso *nickname*. 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** con tecnologia *client-server*. Per ogni partita verrà lanciata una istanza del **server** in grado di gestire una partita alla volta e due o più **client** (uno per giocatore) che possono partecipare ad una sola partita alla volta. Si richiede l'utilizzo del pattern *Model-View-Controller* - *MVC* 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 (o più partite nel caso in cui venga implementata la funzionalità avanzata "partite multiple").
- Consente ai vari giocatori di svolgere i propri turni tramite le istanze del client, connesse al server attraverso un opportuno protocollo di rete (socket TCP-IP e/o RMI).
- Nel caso in cui venga implementata la comunicazione client-server sia via socket che via RMI, deve poter supportare partite in cui i diversi 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 istanziabile più volte (una per giocatore), **anche sulla stessa macchina**.
- L'interfaccia grafica deve essere implementata mediante Swing o JavaFX.
- Nel caso in cui venga implementata sia un'interfaccia testuale (TUI) che un'interfaccia grafica (GUI), all'avvio deve permettere al giocatore di selezionare il tipo di interfaccia da utilizzare.
- Nel caso in cui venga implementata la comunicazione client-server sia via Socket che via RMI, all'avvio deve permettere al giocatore di selezionare la tecnologia 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 faranno parte.
- 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 (in base alla scelta effettuata dal primo giocatore in fase di creazione della partita).

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 (anche se in fase di avvio) e tutti i giocatori verranno notificati di questo evento.

## 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-agnostic 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**. Ai fini dell'implementazione di questa funzionalità aggiuntiva, le regole precedentemente specificate in merito alla creazione delle partite possono essere modificate in base alle esigenze implementative o di interfaccia utente. Ad esempio per consentire ai giocatori in fase di ingresso di scegliere a quale partita aperta e non ancora iniziata collegarsi o per creare una nuova partita.
- **Persistenza:** Fare in modo che il server salvi periodicamente lo stato della partita su disco, in modo che l'esecuzione possa riprendere da dove si è interrotta anche a seguito del crash del server stesso. Per riprendere una partita, i giocatori si dovranno ricollegare al server utilizzando gli stessi *nickname* una volta che questo sia tornato attivo. Si assume che il disco della macchina su cui gira il server costituisca una memoria totalmente affidabile.
- **Resilienza alle disconnessioni:** I giocatori disconnessi a seguito della caduta della rete o del crash del client, possono ricollegarsi e continuare la partita. Mentre un giocatore non è collegato, il gioco continua saltando i turni di quel giocatore. Se rimane attivo un solo giocatore, il gioco viene sospeso fino a che non si ricollega almeno un altro giocatore oppure scade un timeout che decreta la vittoria dell'unico giocatore rimasto connesso.
- **Chat:** Client e server devono offrire la possibilità ai giocatori coinvolti in una partita di chattare tra di loro, inviando messaggi (testuali) indirizzati a tutti i giocatori della partita o a un singolo giocatore.

## 3 Valutazione

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

Requisiti Soddisfatti	Voto Massimo
Regole Semplificate + TUI + RMI o Socket	18
Regole Complete + TUI + RMI o Socket	20
Regole Complete + TUI + RMI o Socket + 1 FA	22
Regole Complete + TUI + GUI + RMI o Socket + 1 FA	24
Regole Complete + TUI + GUI + RMI + Socket + 1 FA	27
Regole Complete + TUI + GUI + RMI + Socket + 2 FA	30
Regole Complete + TUI + GUI + RMI + Socket + 3 FA	30L

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

Di seguito si riportano gli aspetti che vengono valutati e contribuiscono alla definizione del punteggio 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 *correttezza e stabilità* dell'implementazione in conformità alle specifiche.

- La *leggibilità del codice* scritto, con particolare riferimento a nomi di variabili/metodi/classi/package, all'inserimento di commenti in inglese, nonché la mancanza di codice ripetuto e metodi di eccessiva lunghezza.
- La *qualità della documentazione*, con riferimento ai commenti JavaDoc in inglese presenti nel codice, alla documentazione aggiuntiva riguardante il protocollo di comunicazione, e ai documenti delle peer review.
- 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.