

# Introduzione

## SCOPO

Questo documento descrive i requisiti e le funzionalità del progetto “Dubito”, un gioco multiplayer client-server sviluppato in Java tramite RMI.

## OBIETTIVO

Il sistema consente a diversi giocatori di partecipare ad una partita collegandosi ad un server che gestirà in modo centralizzato turni, mosse e stato della partita. Ogni client ha quindi la possibilità di unirsi ad una partita e di inviare le giocate al server tramite servizio esposto tramite RMI.

## DESCRIZIONE DEL GIOCO

Il progetto consiste nella riproduzione del gioco di Dubito con i dati (anche detto Perudo). Il gioco consiste nel rimanere l'ultimo giocatore in partita con almeno 1 dado. Ad inizio partita ogni giocatore ha a disposizione un bicchiere con 5 dadi, un turno segue la seguente struttura:

1. i giocatori scuotono il bicchiere per mescolare i dadi e lo girano sul tavolo, ognuno può guardare i propri dadi
2. il primo giocatore dichiara il numero di dadi di una determinata faccia, che secondo lui sono presenti in gioco
3. a turno ogni giocatore potrà decidere di rilanciare o di dubitare
4. chi perde il turno perderà un dado, il giro ricomincia dal giocatore che perde il dado
5. l'ultimo giocatore rimasto con almeno un dado vince la partita

## DEFINIZIONI E ACRONIMI

RMI: Remote Method Invocation

DiceCup: classe che rappresenta il bicchiere di dadi, definito dal numero di dadi all'interno e dalle facce uscite ogni turno

GameClient: classe che rappresenta il giocatore, utilizzata dal server per comunicare con il Client e per gestire le identità

Move: classe che rappresenta una mossa, è definita da un GameClient e dal tipo di giocata effettuata (Raise per rilanciare l'ultima chiamata, e Doubt per dubitare la chiamata dell'ultimo giocatore)

Paco: faccia 1 del dado, nel gioco rappresenta il jolly e verrà considerato come qualunque faccia

# Funzionalità

## GESTIONE LOBBY

Ogni Client ha la possibilità di effettuare il join alla lobby, ovvero una stanza virtuale che rimane aperta finché non viene raggiunto il numero massimo di giocatori o finché non inizia la partita. Ogni giocatore deve presentare il proprio nickname e verrà rifiutato in caso di partita iniziata o lobby piena, appena il numero di giocatori è sufficiente per iniziare la partita, partirà un conto alla rovescia alla fine del quale inizierà il gioco anche se il limite di giocatori non viene raggiunto, il counter verrà fermato in caso si raggiunga il massimo numero di giocatori per iniziare subito.

## GESTIONE STATO

Il server si occupa di mantenere lo stato della partita per poter validare ogni mossa di ogni giocatore, definire l'ordine dei turni e garantire il corretto svolgimento del gioco fino alla fine. Ad ogni giocatore verrà associato una DiceCup, inizializzata a 5 dadi all'ingresso nella lobby.

## GESTIONE MOSSA

In un turno il server chiederà ai giocatori in ordine prestabilito di effettuare una mossa (Move) che può essere un rilancio (Raise) o un dubito (Doubt). Il primo giocatore dovrà giocare forzatamente un rilancio proponendo un qualunque numero di dadi con faccia diversa dal Paco, il giocatore successivo potrà decidere di rilanciare nei seguenti modi:

1. aumentando il numero di dadi o il numero della faccia ( o entrambi)
2. dimezzando per eccesso il numero di dadi chiamando il Paco come faccia
3. se l'ultima chiamata è di Paco, è possibile o aumentare il numero di dadi, o chiamare almeno il doppio +1 di dadi con qualunque faccia

Dubitare è possibile solo dal secondo giocatore in poi concludendo il turno

## **GESTIONE TURNO**

Ogni turno è definito dal momento in cui i giocatori girano i bicchieri, quando ogni DiceCup viene “riempita” di 5 valori casuali da 1 a 6 che rappresentano le facce dei dadi, fino al momento in cui un giocatore perde un dado. Il server definisce la sequenza in cui i giocatori potranno effettuare una Move, che verrà validata squalificando chiunque provi ad imbrogliare. Il turno finisce nel momento in cui un giocatore dubita (mossa di tipo Doubt), a questo punto il server verifica la quantità di facce chiamate dall’ultimo giocatore che ha rilanciato (contando sempre anche i Paco): se il numero di facce chiamate dall’ultimo rilancio è inferiore o uguale al numero di facce totali, il giocatore che ha dubitato perde un dado, altrimenti perde il dado chi ha fatto l’ultima chiamata. A questo punto il server toglie un dado al perdente e ricomincia un nuovo turno a partire da esso.

## **CONCLUSIONE PARTITA**

Alla fine di ogni turno un giocatore perderà un dado, nel caso raggiunga lo 0 verrà eliminato dal giro, continuando ad assistere alla partita senza essere mai interrogato. La partita si conclude quando rimane un solo giocatore che il server dichiarerà vincitore concludendo la partita. A questo punto nuovi giocatori potranno contattare il server per unirsi alla lobby per ricominciare una nuova partita

# Struttura

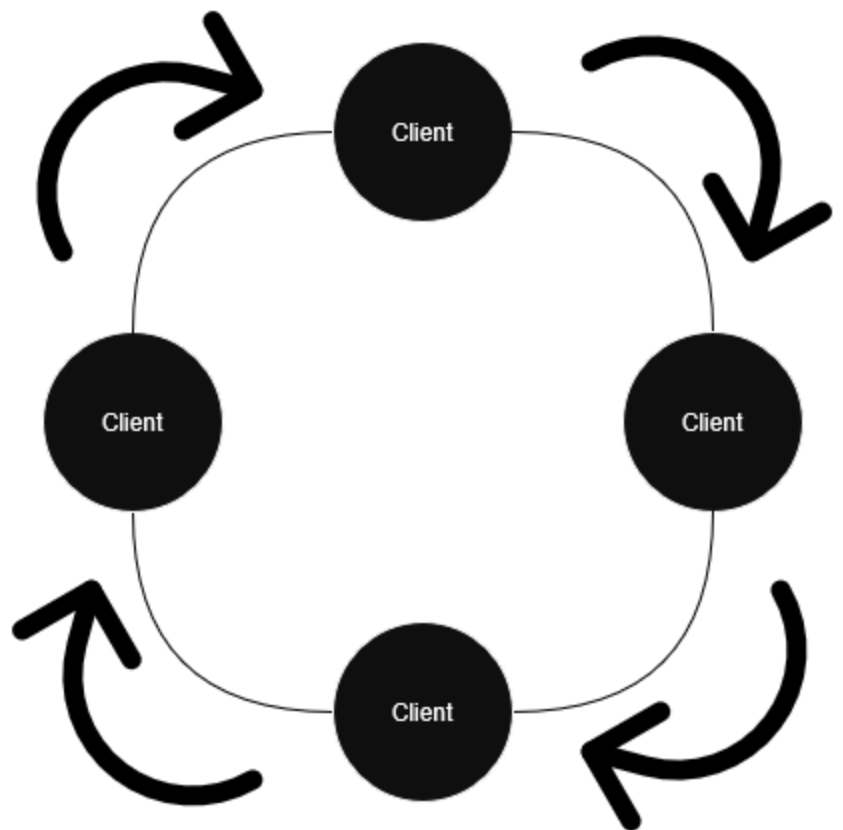
## PRIMA IPOTESI

Anello Circolare:

comunicazione peer-to-peer tra i giocatori, ogni client deve gestire l'intero stato della partita e controllarne l'integrità ad ogni interazione per rilevare errori o imbrogli.

Necessario algoritmo di leader election per decidere il primo giocatore, non c'è single point of failure in quanto la partita può continuare anche senza un giocatore (necessari algoritmi per gestire faults). Necessita inoltre di algoritmo per comunicare ogni mossa a tutti i giocatori (per mostrare la mossa di un giocatore a tutti gli users)

**Alta complessità lato codice**



## SOLUZIONE SCELTA

Stella con server centrale:

comunicazione client-server, centralizza la gestione della partita in un unico nodo, single point of failure (in caso di crash la partita non può continuare).

Client molto leggero, impossibile imbrogliare in quanto stato e controlli sono gestiti dal server, log sulla partita continui tramite callback. Il server decide il leader casualmente, in caso di crash del client la partita continua senza grossi problemi

**Complessità ridotta notevolmente**

