Progetto di PCAD 2016/17:
Il Buono, il Brutto e il Cattivo
(Spaghetti Western Distributed Multiplayer Game)

Costruire un'architettura distribuita per implementare un gioco multiplayer. Il gioco è basato su una board condivisa rappresentata in forma di **grafo** (che collega città del vecchio West) con un nodo speciale di partenza (ingresso dei giocatori nel gioco).

I giocatori chiedono di entrare nel gioco registrandosi con un nickname (possibilmente spaghetti-western) scegliendo inoltre la propria squadra - berretti bianchi (buoni) o neri (cattivi).
Il server ogni 30 minuti fa partire un round del gioco con i giocatori presenti in quel momento.
Il server sceglie inoltre il grafo iniziale (ad esempio proporzionale al numero di giocatori) e la distribuzione di un certo numero di munizioni sui nodi.

I giocatori possono vedere il grafo di gioco, la posizione degli altri giocatori (quindi la composizione e posizione di buoni e cattivi) ma non le munizioni e neppure la posizione del rinnegato (il brutto). Ogni round dura 10 minuti. Quando inizia un round ogni giocatore è libero di muoversi da nodo in nodo (non ci sono turni ognuno muove in maniera concorrente) seguendo gli archi del grafo in cerca di munizioni. Il primo giocatore che arriva ad un nodo acquisisce le munizioni al suo interno. Il rinnegato si muove casualmente da nodo in nodo seguendo gli archi e toglie munizioni (es una pallottola) ad un giocatore che sta sul suo stesso nodo.

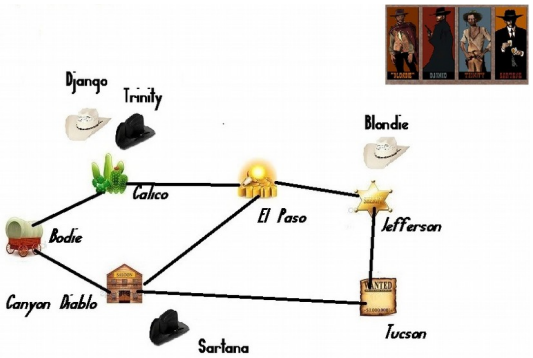
Assumiamo che solo due giocatori possano entrare in un nodo contemporaneamente (restringere di conseguenza l'insieme di nodi successori disponibili ad un giocatore). Se due giocatori sono sullo stesso nodo e sono di squadre diverse allora si sfidano. La sfida è a vostra scelta. Una possibilità è quella di usare la regola del risiko: ogni giocatore acquista da 1 a 3 dadi, tira i dadi e poi confronta i valori ordinandoli in prima ìn senso decrescente (es. 6,4,3 batte 5,4,2). Al termine del round il server ferma i giocatori e annuncia la squadra vincitrice (quella con più munizioni).

Requisiti utente

Il sistema deve essere composto da un server Java e da client Java e Android.
Il server deve permettere l'accesso da parte degli utenti al gioco (un giocatore può partecipare a diversi round).
L'applicazione client, tramite opportuni metodi remoti forniti dal server, deve poter effettuare login, visualizzare lo stato corrente del gioco (grafo di gioco, posizione dei giocatori, squadre e punteggi) e dare all'utente la possibilità di scegliere il prossimo nodo su cui spostarsi).

Fasi di sviluppo e suggerimenti

Si consiglia di lavorare in gruppi al massimo di 3 persone. Partire da un prototipo con solo server e client Java. Usare i pattern visti a lezione (task asincroni, strutture dati concorrenti e sincronizzazione per dati condivisi). Aggiungere quindi i client Android e la gestione della comunicazione con il server.
Come tecnologia di comunicazione usate Socket Java. Provate poi a valutare l'utilizzo degli Stream di Java8 visti a LPOO per interrogare i dati ricevuti dal server e/o fare una versione RMI di server e client (per i soli client Java).
L'ultima parte è opzionale (per aficionados della programmazione Java).



Consegna

Caricare su aulaweb un documento (slide) che spieghi l'architettura del sistema e le scelte fatte.
Il progetto verrà poi discusso di persona (se riusciamo con presentazione finale a cui invitiamo gli studenti del primo anno!!).

Rawhide!!!!

Gioco ideato da Giorgio, Greta ed Alice Delzanno

Documentazione e manualistica

Concurrent Java (monitor, pool, task asincroni)
https://docs.oracle.com/javase/tutorial/essential/concurrency/

Networking in Java (Socket, ServerSocket, connessioni HTTP)
https://docs.oracle.com/javase/tutorial/networking/overview/networking.html

Stream di Java 8 (pipelining su stream di dati)
http://www.oracle.com/technetwork/articles/java/ma14-java-se-8-streams-2177646.html

GUI Swing in Java
https://docs.oracle.com/javase/tutorial/uiswing/

RMI (remote method invocation in Java)
https://docs.oracle.com/javase/tutorial/rmi/
http://docs.oracle.com/javase/7/docs/technotes/guides/rmi/enhancements-7.html

Java per Android
http://developer.android.com/index.html
Android Hub: https://androidhub.intel.com/en/

Diversi modi di condividere dati (es. socket) tra activity:

Singleton pattern https://gist.github.com/Akayh/5566992
Variabili statiche o application context http://stackoverflow.com/questions/23249163/transfersocket-from-one-activity-to-another

Service: http://stackoverflow.com/questions/14985678/how-to-keep-the-android-client-connected-to-the-server-even-on-activity-changes

Indirizzi virtuali: Le informazioni sugli indirizzi virtuali di rete usati dall'emulatore (ad es per collegarsi al server sulla stessa macchina):

https://developer.android.com/studio/run/emulator-commandline.html#emulatornetworking
L'indirizzo 10.0.2.2 dovrebbe corrispondere al localhost della macchina
(host loopback interface, 127.0.0.1 on your development machine)

Librerie per grafi in Java:

- JGraphT (libreria per manipolare grafi in Java) http://jgrapht.org/

- Grph http://www.i3s.unice.fr/~hogie/software/index.php?name=grph&page=../../Documentation

- JgraphX https://github.com/jgraph/jgraphx