SERVER MULTITHREADED SOCKET

Per completare l'architettura del nostro sistema di gestione di prenotazione di eventi ci serve ancora il middleware di comunicazione con potenziali clienti collegati in rete.

Nella versione distribuita i client usano comunicazione TCP per inviare le richieste (prenotazione e lista eventi) al server.

Il server deve avere architettura **multithreaded** basata su **ServerSocket** su TCP (vedi lezioni su networking in Java) e deve poter gestire le richieste di prenotazione dei client confinando i socket generati da una richiesta in un **task** separato.

Per quanto riguarda la struttura dati per memorizzare gli eventi: raffinate la soluzione del foglio 2 usando non più metodi sincronizzati ma invece **strutture dati concorrenti** cercando di parallelizzare se possibile operazioni su eventi indipendenti tra loro.

Associate al programma client una **GUI** minimale per visualizzare la lista degli eventi e inviare le richieste di prenotazione come nell'esercitazione su Swing.

Preparate un breve video di presentazione della vostra soluzione (con una parte per ogni membro del gruppo) e inserite il link insieme al codice.
In particolare spiegate in maniera chiara come avete gestito l'accesso concorrente alla struttura dati per la gestione degli eventi.

Opzionale:

Versione di client e server con le librerie Sparkjava che supportano comunicazione RESTful (https://sparkjava.com/).