Università degli Studi di Napoli Federico II
Esame di Advanced Computer Programming

Proff. De Simone, Della Corte

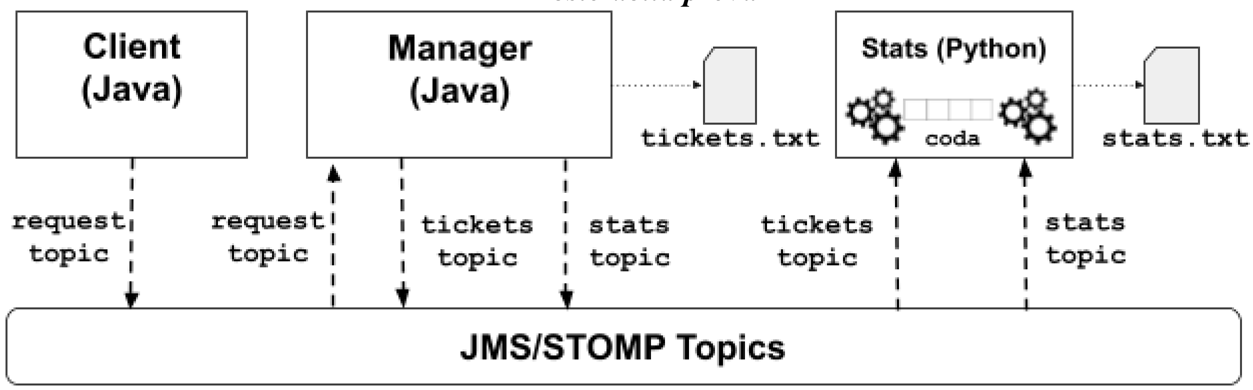
Prova pratica del giorno 19/03/2024
Durata della prova: 120 minuti

°°

Lo studente legga attentamente il testo e produca il programma ed i casi di test necessari per dimostrarne il funzionamento. La mancata compilazione dell'elaborato, la compilazione con errori o l'esecuzione errata daranno luogo alla valutazione come ***prova non superata****.*
Al termine della prova lo studente dovrà far verificare il funzionamento del programma ad un membro della Commissione.

°°

Testo della prova



Il candidato realizzi un sistema di acquisto biglietti per concerti su **JMS** e **STOMP**. Il sistema include i seguenti componenti:

- **Client**. Componente **<u>Java</u>** che genera richieste di tipo *buy* o *stats*, sotto forma di messaggi JMS. Client attende 2 secondi tra le richieste, e genera un totale di 20 richieste. Per ogni richiesta, Client genera un MapMessage composto da due campi: ***type,*** che conterrà il tipo di richiesta (cioè *buy* o *stats*), e ***value***, che conterrà una *Stringa* scelta a caso tra *Jovanotti*, *Ligabue*, *Negramaro*. nel caso di type pari a *buy*, e la Stringa *Sold* nel caso di type pari a *stats*. Il MapMessage è poi inviato sul topic *request*. Il tipo di richiesta è fornito da terminale. Ad esempio: **`java nomepackage.Client buy`**
- **Manager.** Componente **<u>Java</u>** che implementa la <u>ricezione asincrona</u> JMS sul topic `request`. Alla ricezione di ciascun MapMessage, il *listener* JMS estrae i due campi (*type* e *value*). Nel caso il campo *type* sia *stats*, Manager prepara un TextMessage nel quale inserisce il valore presente nel campo *value*, e lo invia sul topic `stats`. Nel caso il campo *type* sia *buy*, Manager scrive il contenuto del campo *artist* su un file (*tickets.txt*), e prepara un TextMessage nel quale inserisce il valore presente nel campo *value*, e lo invia sul topic `tickets`.
- **Stats**. Componente **<u>Python</u>** che implementa la <u>ricezione asincrona</u> STOMP sul topic `tickets` e sul topic `stats`. Alla ricezione di un TextMessage sul topic `tickets`, il componente avvia un nuovo processo che inserisce la stringa presente nel messaggio in una <u>coda</u> (process-safe e che implementi il problema del produttore/consumatore). Alla ricezione di un TextMessage sul topic `stats`, il componente verifica che la stringa nel messaggio sia pari a *Sold*. In caso affermativo, avvia un nuovo processo che svuota la coda, e costruisce un dizionario dove per ogni artista (chiave del dizionario) è memorizzato il numero di biglietti venduti (valore del dizionario). <u>N.B.: il numero di biglietti venduti può essere ottenuto contando quante volte il nome di un artista appare nella coda.</u> Dopo aver costruito il dizionario, il processo scrive su un file (*stats.txt*) il contenuto del dizionario (ogni riga deve contenere il nome dell'artista ed il numero di biglietti venduti per quell'artista).

Lo studente dovrà sviluppare le applicazioni **Client**, **Manager** e **Stats**. Il sistema sarà testato da terminale con 2 Client (uno per *buy* e uno per *stats*), 1 Extractor, e 1 Stats.