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

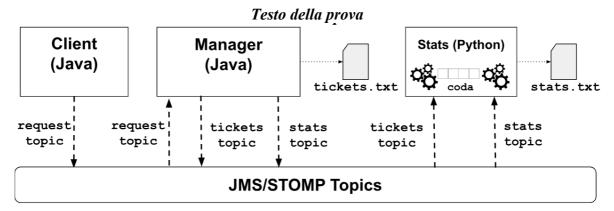
Proff. De Simone, Della Corte

Prova pratica del giorno 26/06/2024 a.a. 2022/2023

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.



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 <u>request</u>. Alla ricezione di ciascun MapMessage, il <u>listener</u> JMS estrae i due campi (<u>type e value</u>). Nel caso il campo <u>type sia stats</u>, Manager prepara un TextMessage nel quale inserisce il valore presente nel campo <u>value</u>, e lo invia sul topic <u>stats</u>. Nel caso il campo <u>type sia buy</u>, Manager scrive il contenuto del campo <u>artist</u> su un file (<u>tickets.txt</u>), e prepara un TextMessage nel quale inserisce il valore presente nel campo <u>value</u>, e lo invia sul topic <u>tickets</u>.
- Stats. Componente <u>Python</u> che implementa la <u>ricezione asincrona</u> STOMP sul topic <u>tickets</u> e sul topic <u>stats</u>. Alla ricezione di un TextMessage sul topic <u>tickets</u>, 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 <u>stats</u>, il componente verifica che la stringa nel messaggio sia pari a <u>Sold</u>. 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 <u>coda</u>. Dopo aver costruito il dizionario, il processo scrive su un file (<u>stats.txt</u>) 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 <u>Client</u>, <u>Manager</u> e <u>Stats</u>. Il sistema sarà testato da terminale con 2 Client (uno per <u>buy</u> e uno per <u>stats</u>), 1 Extractor, e 1 Stats.</u>