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

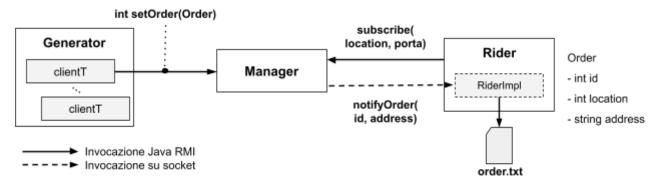
Proff. De Simone, Della Corte

Prova pratica del giorno 20/07/2023 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.

Testo della prova



Il candidato realizzi un sistema per la distribuzione di *notifiche di ordini* (rappresentate da istanze di **Order**) verso dei rider, generate da clienti remoti. Il sistema è composto dalle seguenti entità.

Generator. Avvia 3 thread; ciascun thread invoca **setOrder** su Manager (invocazione <u>Java RMI</u>) passando un'istanza di **Order**. Il valore del campo id (intero) dell'istanza di **Order** è scelto a caso tra 1 e 100, **location** (intero) è scelto a caso tra 1 e 5; **address** (stringa) è generata concatenando un valore numerico (generato a caso tra 4 e 10) alla stringa "th avenue", ad esempio 5th avenue.

Rider. Attende da Manager gli ordini per una specifica **location**. A tale scopo, Rider *-al proprio avvio-* si sottoscrive al Manager invocando il metodo **subscribe** (invocazione <u>Java RMI</u>) passando: 1) **location** (intero, ossia la location di cui il Rider intende ricevere gli ordini) e 2) **numero di porta**, utilizzata dal Manager per contattare il Rider <u>tramite socket</u>, come specificato nel seguito.

Manager. Espone i metodi Java RMI subscribe e setOrder. Per ogni subscribe, il Manager memorizza i dati di sottoscrizione ricevuti da un Rider (lo studente utilizzi i meccanismi ritenuti più opportuni). Ad ogni setOrder, il Manager verifica il valore di location contenuto nell'istanza di Order ricevuta: nel caso in cui esista un Rider sottoscritto per quella location, esso sarà notificato dal Manager tramite l'invocazione di notifyOrder e restituirà 1 come valore di ritorno alla setOrder; in caso contrario il Manager restituirà -1 come valore di ritorno della setOrder.

L'invocazione di **notifyOrder** avviene tramite **proxy** e **socket**: il metodo **notifyOrder** richiede in ingresso l'*id* (intero) dell'ordine e l'*address* (stringa). Tali valori sono stampati a video e salvati su file lato Rider. Il metodo **setOrder** di Manager è eseguito in <u>mutua esclusione</u>.

Il sistema sarà testato da prompt con 1 Generator, 1 Manager, ed un numero arbitrario di **Rider** (un solo Rider per una location). Location, porta e nome del file (necessari lato Rider) sono specificati da prompt, per es.:

java rider.Rider 5 8000 order.txt

Il candidato utilizzi Java RMI per i metodi di Manager e proxy-skeleton con socket TCP per il Rider.

A tal fine, il candidato predisponga le opportune interfacce e le classi Proxy-Skeleton. Si utilizzino inoltre synchronized per la mutua esclusione e skeleton per delega per il Rider.