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

Prova pratica del giorno 24/05/2021
Durata della prova: 120 minuti

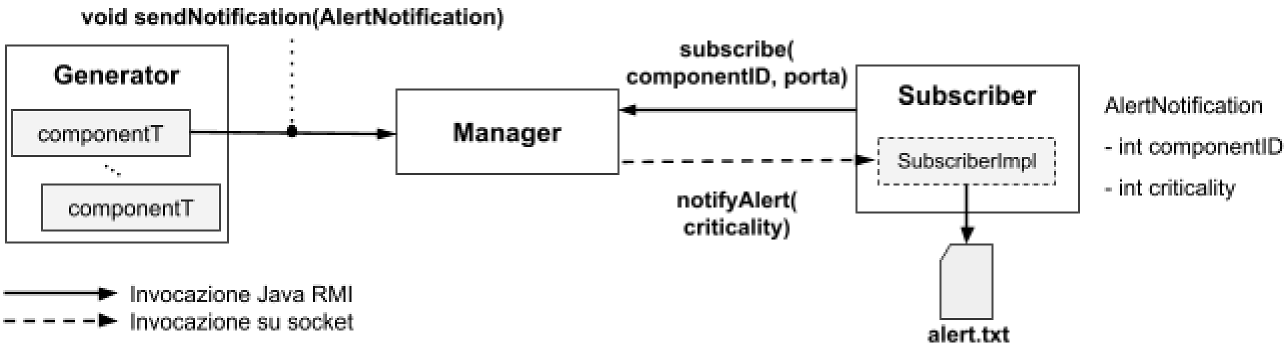
°°

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 consegnare un file compresso contenente la cartella del progetto creato in Eclipse per lo svolgimento della traccia. Per la consegna:

- Rinominare il file compresso in `Cognome_Nome_Matricola.zip`
- Collegarsi al link Dropbox fornito
- Nel form, cliccare su **"aggiungi file"**, selezionare il file compresso contenente il proprio svolgimento, e indicare il proprio nome, cognome ed email @studenti.unina.it
- Attendere una notifica del docente, e quindi scollegarsi dalla piattaforma di Virtual Classroom

°°

Testo della prova



Il candidato realizzi un sistema per la distribuzione di *notifiche di alert* (rappresentate da istanze di **AlertNotification**), generate da componenti remoti. Il sistema è composto dalle seguenti entità.

Generator. Avvia 3 thread; ciascun thread invoca **`sendNotification`** su Manager (invocazione <u>Java RMI</u>) passando un'istanza di **`AlertNotification`**. Il valore del campo **`componentID`** (intero) dell'istanza di **`AlertNotification`** è scelto a caso tra 1 e 5; **`criticality`** (intero) è scelto a caso tra 1 e 3.

Subscriber. Attende da Manager la produzione di alert associati ad uno specifico **`componentID`**. A tale scopo, Subscriber *–al proprio avvio–* si sottoscrive al Manager invocando il metodo **`subscribe`** (invocazione <u>Java RMI</u>) e passando: 1) **`componentID`** (intero, ossia l'ID del componente di cui il Subscriber intende ricevere gli alert) e 2) **numero di porta**, utilizzata dal Manager per contattare successivamente il Subscriber <u>tramite socket</u>, come specificato nel seguito.

Manager. Espone i metodi Java RMI **`subscribe`** e **`sendNotification`**. Per ogni **`subscribe`**, il Manager memorizza i dati di sottoscrizione ricevuti da un Subscriber (lo studente utilizzi i meccanismi ritenuti più opportuni). Ad ogni **`sendNotification`**, il Manager verifica il valore di **`componentID`** contenuto nell'istanza di **`AlertNotification`** ricevuta: nel caso in cui esistano uno (o più) Subscriber sottoscritti per quel **`componentID`**, essi saranno notificati dal Manager tramite l'invocazione di **`notifyAlert`**.

L'invocazione di **`notifyAlert`** avviene tramite **proxy** e **socket**: il metodo **`notifyAlert`** richiede in ingresso il solo valore di *criticality* (intero). Tale valore è stampato a video e salvato su file lato Subscriber. Il metodo **`sendNotification`** di Manager è eseguito in mutua esclusione.

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

```
java subscriber.Subscriber 3 8000 alert.txt
```

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