

Conference Registration

📖 Course	🏠 <u>Distributed Programming</u>
📅 Date	@November 3, 2023
📌 Type	Assignment
🌟 Status	Completed

Descrizione

L'applicazione **Conference Registration** presenta un'architettura di tipo Client-Server che utilizza le RMI per consentire la registrazione ad una conferenza.

Requisiti del server

Il *server* deve:

[FS1] consentire ai partecipanti di registrarsi alla conferenza

[FS2] visualizzare le informazioni circa i partecipanti registrati,

[FS3] cancellare le registrazioni,

[FS4] mantenere i programmi dei tre giorni del congresso sul proprio residence node,

[FS5] rappresentare ogni programma in una struttura dati in cui ogni riga corrisponde a una sessione (12 sessioni al giorno) e per ogni sessione vengono mantenuti i nomi degli speaker che si sono registrati (massimo 5),

[FS6] avere un meccanismo di gestione di disconnessione brusca.

Requisiti del client

Il *client* deve:

[FC1] poter vedere i dettagli della conferenza,

[FC2] potersi registrare per partecipare a una sessione,

[FC3] inoltrare le richieste al Server in maniera appropriata e, per ogni possibile operazione, fornire anche un meccanismo di gestione di condizioni anomale (come la richiesta di

registrazione per un giorno che non esiste o una sessione per cui gli spazi di intervento sono pieni),

[FC4] essere implementato come un processo ciclico che continua ad effettuare richieste sincrone fino a che tutte le esigenze dell'utente non saranno esaurite.

[FC5] poter chiudere la connessione in maniera appropriata.

Struttura

Interfaccia remota

Sia il client, che il server possiedono l'interfaccia remota all'interno del modulo `common`, tale interfaccia è denominata `ConferenceRegistration`.

L'interfaccia remota possiede i metodi:

- `boolean register(String name, int day, int session) throws RemoteException`
che serve per consentire all'utente di registrarsi ad una sessione di un certo giorno del congresso, restituisce true se l'operazione è andata a buon fine, false altrimenti,
- `String [][][] getInformation() throws RemoteException`
che serve per consentire all'utente di ottenere informazioni riguardanti ciascuno dei giorni del congresso,
- `String [][] getInformationByDay(int day) throws RemoteException`
che serve per consentire all'utente di ottenere informazioni riguardanti un preciso giorno del congresso,
- `int getDays() throws RemoteException`
che serve per consentire all'utente di conoscere il numero di giorni per i quali il congresso avrà luogo,
- `int getSessions() throws RemoteException`
che serve per consentire all'utente di conoscere il numero di sessioni che saranno disponibili per ciascuno dei giorni del congresso,
- `int getMaxSpeakers() throws RemoteException`
che serve per consentire all'utente di conoscere il numero massimo di speaker che potranno esserci durante una sessione del congresso,
- `boolean cancelRegistration(int day, int session, int intervention) throws RemoteException`

che serve per consentire all'utente di cancellare la propria registrazione ad un intervento di una sessione per uno dei giorni in cui si terrà il congresso, restituisce true se l'operazione è andata a buon fine, false altrimenti.

Server

ConferenceRegistrationImpl

La classe `ConferenceRegistrationImpl` contiene l'implementazione dei metodi dell'interfaccia remota e mantiene un riferimento a un `ResidenceNode`.

ResidenceNode

Il residence node del server, mantiene una matrice tridimensionale di stringhe `programs`, in cui:

- la prima dimensione rappresenta il *numero di giorni* del congresso,
- la seconda dimensione rappresenta il *numero di sessioni* per ogni giorno in cui si tiene il congresso,
- la terza dimensione rappresenta il *numero di interventi massimo* che può esserci in una sessione.

Vengono poi mantenute le informazioni sul numero di giorni del congresso `numDays`, sul numero di sessioni `numSessions` e sul numero massimo di interventi `numMaxSpeakers` che vengono forniti in fase di creazione. Inoltre, viene mantenuta un'ulteriore struttura di utilità, che è una matrice bidimensionale, che mantiene per ogni giorno e sessione, il numero di interventi prenotati, in modo tale da poter effettuare il controllo sulla disponibilità degli slot in maniera più efficiente.

La matrice degli interventi viene compilata con il carattere `-` in fase di creazione del residence node e, man mano che vengono effettuate registrazioni, tale carattere viene sostituito con il nome della persona prenotata.

I metodi pubblici che vengono forniti, oltre i metodi getter per gli attributi della classe sono:

- `boolean insertSpeaker(String name, int day, int session)`

che consente di prenotare uno speaker per un intervento, dopo aver verificato la validità del giorno e della sessione e la disponibilità dello slot, restituisce true solo se l'operazione è andata a buon fine, altrimenti false,

- `boolean removeSpeaker(int day, int session, int intervention)`

che consente di cancellare la prenotazione di uno speaker per un intervento, dopo aver verificato la validità del giorno, della sessione e dell'intervento, restituisce true solo se l'operazione è andata a buon fine, altrimenti false.

Gli altri metodi privati presenti nella classe, sono di utilità e servono per il controllo della validità dei parametri forniti.

Main

Il server istanzia un `Registry`, crea un'istanza di `ConferenceRegistrationImpl` con i parametri richiesti dalle specifiche (3 giorni, 12 sessioni e 2 interventi) ed effettua il `rebind` al registry, specificando il path dell'interfaccia remota e l'oggetto remoto.

Client

ConferenceRegistrationClient

La classe `ConferenceRegistrationClient` contiene invece la logica del client.

In fase di creazione dell'oggetto viene creata un'istanza di `Scanner` che consente di leggere l'input dell'utente e si ottiene un riferimento a `ConferenceRegistration`, su cui verranno poi chiamati i metodi dell'interfaccia remota.

Tale classe mette a disposizione il metodo pubblico `chooseOperation`, che permette 4 possibili operazioni, a seconda dell'input fornito dall'utente:

- `sign`, che, una volta ottenuti i parametri necessari e confermata la loro validità, consente all'utente di provare a registrarsi per un determinato intervento,
- `info`, che mostra all'utente il programma del congresso per un determinato giorno,
- `unsign`, che, una volta ottenuti i parametri necessari e confermata la loro validità, consente all'utente di provare a cancellare la registrazione per un determinato intervento,
- `exit`, che consente di terminare l'applicazione.

Ciascuna delle prime tre operazioni viene portata a termine nell'omonimo metodo privato, dove vengono anche effettuati i controlli di validità dei parametri forniti in input.

Main

Il client ottiene un riferimento al registry, dopo aver specificato indirizzo IP e numero di porta. Usando il metodo `lookup` ottiene poi il riferimento all'oggetto remoto e crea un'istanza di `ConferenceRegistrationClient` alla quale viene passato tale riferimento come parametro in fase di creazione.

Viene eseguito poi il metodo `chooseOperation`, che consente all'utente di scegliere l'operazione che vuole eseguire, fino a che non chiede di uscire dall'applicazione.

Funzionamento

Registrazione

Il client avvia l'applicazione e gli viene chiesta l'operazione che intende eseguire.



Se sceglie *sign*, può richiedere di effettuare la registrazione. Ed in sequenza gli viene richiesto di fornire il giorno (*Day?*), la sessione (*Session?*) ed il nome (*Speaker name?*). Se uno di questi non è valido, viene mostrato un messaggio di errore ed il processo di richiesta ricomincia.

Utente 1



Campi validi.

```
Problems Javadoc Declaration Console Coverage Git Staging
ConferenceRegistrationClient [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.macosx.aarch64_17.0.5.v20221102-0933/jre/bin/java (Nov 1, 2023, 6:57:24 PM) [pid: 98570]
What operation do you want to do? (sign, info, unsign, exit) sign
Day?
1
Session?
1
Speaker name?
Grazia
What operation do you want to do? (sign, info, unsign, exit) sign
Day?
10
Invalid day, try again!
Day?
```

Giorno non valido.

```
Problems Javadoc Declaration Console Coverage Git Staging
ConferenceRegistrationClient [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.macosx.aarch64_17.0.5.v20221102-0933/jre/bin/java (Nov 1, 2023, 6:57:24 PM) [pid: 98570]
What operation do you want to do? (sign, info, unsign, exit) sign
Day?
1
Session?
1
Speaker name?
Grazia
What operation do you want to do? (sign, info, unsign, exit) sign
Day?
10
Invalid day, try again!
Day?
1
Session?
13
Invalid session, try again!
Day?
```

Sessione non valida.

Utente 2

È presente anche un altro client ad effettuare richieste al server in contemporanea. Il client ripete il procedimento come nell'esempio precedente e si registra per la sessione 2 nel medesimo giorno, con il nome Laura.

Visualizzazione del programma

L'utente 2 ora chiede di visualizzare il programma del giorno 1 del congresso e ottiene:

```
Problems Javadoc Declaration Console X Coverage Git Staging
ConferenceRegistrationClient [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.macosx.aarch64_17.0.5.v20221102-0933/jre/bin/java (Nov 1, 2023, 6:57:38 PM) [pid: 98573]
What operation do you want to do? (sign, info, unsign, exit) info
Day?
1
Program for the day 1 of the congress:
S1 session:
Intervention 1: Grazia
Intervention 2: -
Intervention 3: -
Intervention 4: -
Intervention 5: -
S2 session:
Intervention 1: Laura
Intervention 2: -
Intervention 3: -
Intervention 4: -
Intervention 5: -
S3 session:
Intervention 1: -
Intervention 2: -
Intervention 3: -
Intervention 4: -
Intervention 5: -
S4 session:
Intervention 1: -
Intervention 2: -
Intervention 3: -
Intervention 4: -
```

Visualizzazione.

Come possiamo notare entrambe le registrazioni sono andate a buon fine.

Cancellazione della registrazione

Immaginiamo che ora l'utente 1 voglia richiedere la cancellazione.

Dopo aver visualizzato il programma per vedere quale intervento gli è stato assegnato, può richiedere di cancellarne la registrazione come segue:

```
What operation do you want to do? (sign, info, unsign, exit) info
Day?
1
Program for the day 1 of the congress:
S1 session:
Intervention 1: -
Intervention 2: -
Intervention 3: -
Intervention 4: -
Intervention 5: -
S2 session:
Intervention 1: Laura
Intervention 2: -
Intervention 3: -
Intervention 4: -
Intervention 5: -
S3 session:
Intervention 1: -
Intervention 2: -
Intervention 3: -
Intervention 4: -
Intervention 5: -
S4 session:
Intervention 1: -
Intervention 2: -
Intervention 3: -
```

Cancellazione valida.

Visualizzando nuovamente il programma, possiamo vedere che la cancellazione è andata a buon fine.

In caso di parametri non validi, la cancellazione non va a buon fine e viene richiesto di re-inserirli, esattamente come sopra:

```

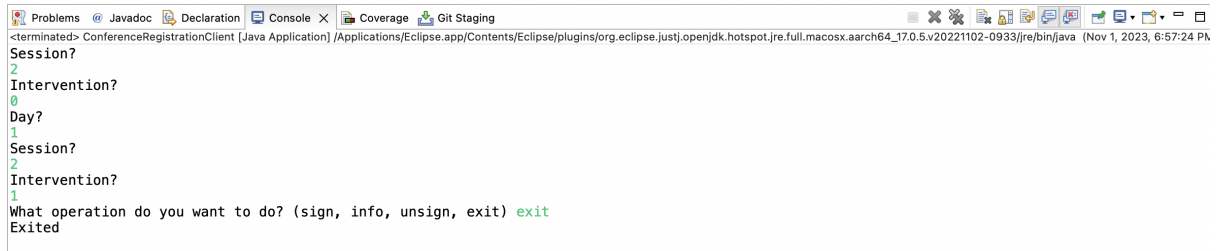
Intervention 2: -
Intervention 3: -
Intervention 4: -
Intervention 5: -
What operation do you want to do? (sign, info, unsign, exit) unsign
Day?
9
Invalid day, try again!
Day?
1
Session?
14
Invalid session, try again!
Day?
1
Session?
2
Intervention?
0
Day?
1
Session?
2
Intervention?
1
What operation do you want to do? (sign, info, unsign, exit)

```

Campi non validi.

Chiusura del programma

Si può richiedere poi di terminare l'applicazione.



```

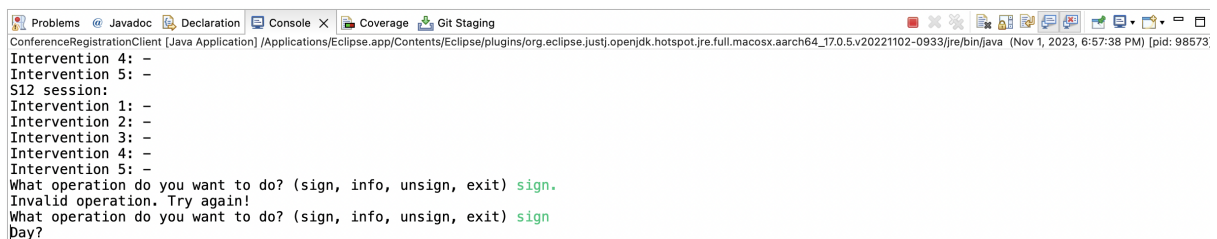
<terminated> ConferenceRegistrationClient [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.macosx.aarch64_17.0.5.v20221102-0933/jre/bin/java (Nov 1, 2023, 6:57:24 PM)
Session?
2
Intervention?
0
Day?
1
Session?
2
Intervention?
1
What operation do you want to do? (sign, info, unsign, exit) exit
Exited

```

Chiusura dell'applicazione.

Gestione di errori o campi non validi

Nel caso in cui l'operazione specificata non sia valida, viene mostrato un messaggio di errore e viene richiesto all'utente di riprovare.



```

ConferenceRegistrationClient [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.macosx.aarch64_17.0.5.v20221102-0933/jre/bin/java (Nov 1, 2023, 6:57:38 PM) [pid: 98573]
Intervention 4: -
Intervention 5: -
S12 session:
Intervention 1: -
Intervention 2: -
Intervention 3: -
Intervention 4: -
Intervention 5: -
What operation do you want to do? (sign, info, unsign, exit) sign.
Invalid operation. Try again!
What operation do you want to do? (sign, info, unsign, exit) sign
Day?

```

Operazione non valida.

Immaginiamo ora che per una data sessione di un dato giorno non ci siano più interventi disponibili. In tal caso viene mostrato il seguente messaggio di errore:


```

What operation do you want to do? (sign, info, unsign, exit) sign
Day?
valentina
Invalid fields, try again!
Day?
1
Session?
1
Speaker name?
valentina
Not available session, try again!
Day?
1
Session?
2
Speaker name?
valentina
What operation do you want to do? (sign, info, unsign, exit) info
Day?
1
Program for the day 1 of the congress:
S1 session:
Intervention 1: graziaa
Intervention 2: luca
Intervention 3: giovanni
Intervention 4: angela
Intervention 5: claudia
S2 session:
Intervention 1: valentina
Intervention 2: -
Intervention 3: -
Intervention 4: -
Intervention 5: -
S3 session:

```

Sessione piena.