Programmazione 13/06/2017

Note sullo svolgimento della prova

- Non è consentito l'uso di vecchi testi d'esame, libri, o appunti.
- Utilizzare NetBeans o Notepad++ come editor, e finestre DOS per compilare ed eseguire i programmi.
- La cartella C:\prg contiene il JDK da utilizzare e la documentazione delle Java API. Eventuali file ausiliari preparati dal docente devono essere scaricati da elearn.ing.unipi.it
- I file sorgenti relativi all'esercizio 1 devono essere posti nella cartella C:\prg\myapps\esercizio1 mentre quelli relativi all'esercizio 2 devono essere posti nella cartella C:\prg\myapps\esercizio2.
- Al termine della prova, dopo la sottomissione degli elaborati su elearn.ing.unipi.it, il docente mostrerà una possibile soluzione. Dopo aver visto la soluzione, gli studenti avranno la possibilità di riprendere il proprio elaborato.

Esercizio 1

Vogliamo realizzare un sistema per l'esecuzione di task in cui il numero di thread esecutori varia a seconda del carico. I task sono contenuti in una coda in cui vengono inseriti da uno o più produttori. La coda possiede un numero di esecutori attivi che varia da *min* a *max*. Ogni esecutore attivo, ciclicamente, preleva un task dalla coda (secondo disciplina FIFO) e lo esegue. Quando la coda viene creata il numero di esecutori attivi è pari a *min*. Il numero di esecutori attivi viene incrementato ogni volta che si tenta di inserire un nuovo task e la coda è piena. Questo avviene fino a raggiungere il numero massimo (*max*) di esecutori attivi. Quando la coda diventa vuota il numero di esecutori attivi viene invece via via decrementato (ma non può scendere sotto *min*).
Realizzare una classe *Coda* dotata almeno dei seguenti costruttori/metodi:

- *Coda(int n, int min, int max)*: crea una coda in grado di contenere al più *n* task; il numero di esecutori attivi varia tra *min* e *max* a seconda del carico.
- *void inserisci(Task t)*: inserisce il task *t* in coda; se la coda è piena incrementa, se possibile, il numero di esecutori attivi.

Realizzare anche la classe *Task*.

Esercizio 2

Un sistema di *crowd answering* ("risposta della folla") consente di fornire una risposta adeguata ed economica a una domanda, sulla base della somma di esperienze particolari, più di quanto si ottenga da un singolo esperto. Si consideri un numero di *Opinionisti* non noto a priori, in comunicazione secondo una configurazione ad anello. Un *Opinionista* (interrogante) formula una *Domanda*, esprime la propria opinione, la visualizza e la invia al successivo *Opinionista*. Si supponga per semplicità che la opinione sia un numero reale. Una domanda è composta da: *testo*, *somma delle opinioni*, *numero di opinioni, numero sufficiente di opinioni*, e *porta di ascolto dell'opinionista interrogante*. Ogni *Opinionista* riceve la domanda, dà una opinione, la visualizza e la invia all'*opinionista* successivo, e cosi via. Durante il primo giro, non appena si giunge al numero sufficiente di opinioni (anche se questo dovesse avvenire con una sola opinione) la domanda viene inviata all'opinionista interrogante, il quale visualizza la domanda e la risposta (calcolata come valor medio delle opinioni), la invia all'opinionista successivo, e cosi via per il secondo giro. Il secondo giro viene effettuato solo se c'è una risposta, e serve a ciascun opinionista a ricevere, visualizzare la domanda e la risposta, inviare la domanda, fino a fermarsi all'opinionista che per primo ha precedentemente verificato il numero sufficiente di opinioni, il quale rimanderà la domanda all'opinionista interrogante per concludere il giro.

Si realizzi un'applicazione Java distribuita composta dalle classi *Domanda* e *Opinionista*. La classe *Domanda* mantiene ed elabora i dati suddetti. La classe *Opinionista* svolge le operazioni di trasmissione di istanze di *Domanda* tra un opinionista e il successivo, in accordo al protocollo di cui sopra. Si assuma che le visualizzazioni, gli invii e le ricezioni avvengano in formato XML, e che le altre operazioni avvengano su istanze di Domanda. In particolare, le visualizzazioni di qualsiasi messaggio o contenuto non avvengono su console, ma tramite archiviazione su un database condiviso, visualizzato tramite MySQL Client

```
@echo off
c:\prg\jdk8\bin\javac -classpath c:\prg\libs\xstream-1.4.7.jar *.java
pause

set L=c:\prg\libs\
set
LIBS=%L%xstream-1.4.7.jar;%L%xmlpull-1.1.3.1.jar;%L%xpp3_min-1.1.4c.jar;%L%mysql-connector-java-5.1.34-bin.jar;

start cmd /k "color 2F && c:\prg\jdk8\bin\java -classpath %LIBS% Opinionista 8082 8080 50"
start cmd /k "color 2F && c:\prg\jdk8\bin\java -classpath %LIBS% Opinionista 8081 8082 60"
start cmd /k "color 2F && c:\prg\jdk8\bin\java -classpath %LIBS% Opinionista 8080 8081 55 "Altezza Torre di
Pisa" 2"
pause
taskkill /f /im "java.exe"
rem risultato 57.5  calcolato senza il contributo di 8082
pause

start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% Opinionista 8082 8080 50"
start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% Opinionista 8081 8082 60"
start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% Opinionista 8080 8081 55 "Altezza Torre di
Pisa" 3"
pause
taskkill /f /im "java.exe"
rem come risultato 55 calcolato con il contributo di tutti
pause

start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% Opinionista 8082 8080 50"
start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% Opinionista 8081 8082 60"
start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% Opinionista 8080 8081 55 "Altezza Torre di
Pisa" 4"
pause
taskkill /f /im "java.exe"
rem nessun risultato (numero minimo non raggiunto)
pause

start cmd /k "color 5F && c:\prg\jdk8\bin\java -classpath %LIBS% Opinionista 8082 8080 50"
start cmd /k "color 5F && c:\prg\jdk8\bin\java -classpath %LIBS% Opinionista 8081 8082 60"
start cmd /k "color 5F && c:\prg\jdk8\bin\java -classpath %LIBS% Opinionista 8080 8081 55 "Altezza Torre di
Pisa" 1"
pause
taskkill /f /im "java.exe"
rem risultato 55 calcolato con il solo contributo di 8080
pause
```

Fig.4 - File make.bat, da non modificare.

Attenersi esattamente a quanto espresso dalle seguenti figure. Fig.1 mostra una configurazione di test con soli tre opinionisti. Fig.2 mostra lo chema del database e i file dell'applicazione. Fig.3 presenta le classi da realizzare (in grigio), con i relativi campi e i metodi da creare, i package e le classi da importare e usare. Gestire la chiusura di flussi e connessioni

aperti. Catturare solo le eccezioni obbligatorie, e gestirle semplicemente stampando le relative informazioni. Fig.4 mostra il file di testing make.bat, con alcuni casi di test. Si noti che l'opinionista interrogante si distingue per avere in ingresso più di tre parametri. La logica di *Domanda* e Opinionista dovrà essere valida a prescindere dai numeri di porta e di opinionisti.

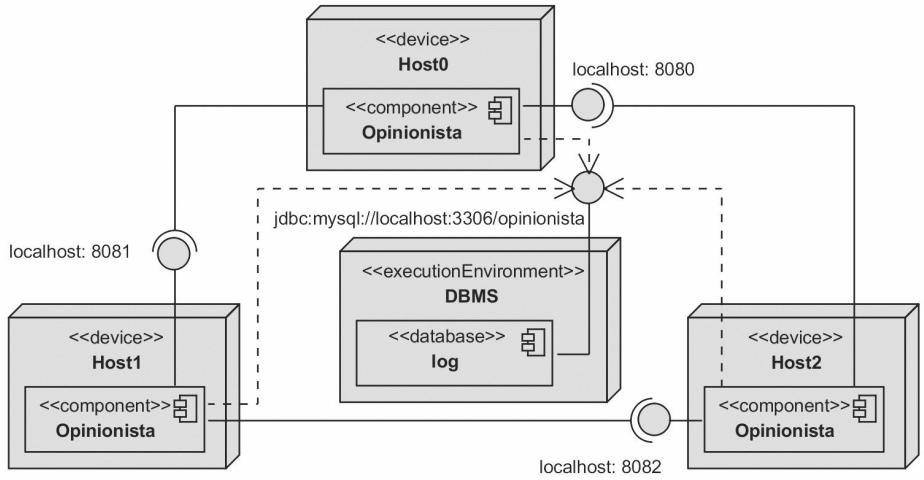

Fig.1 - Tre istanze di *Opinionista* e relative porte di ascolto, con database condiviso per i log

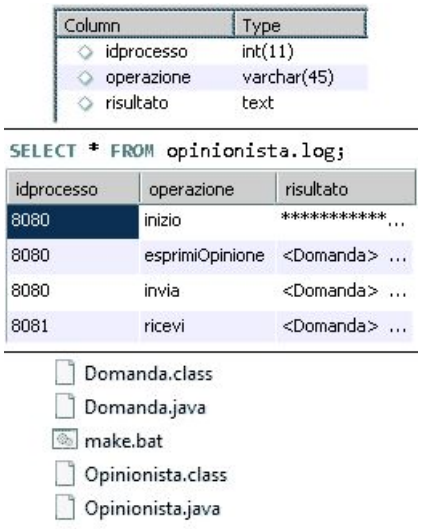

Fig.2 - schema del database e contenuto di c:\prg\esercizio2 e db

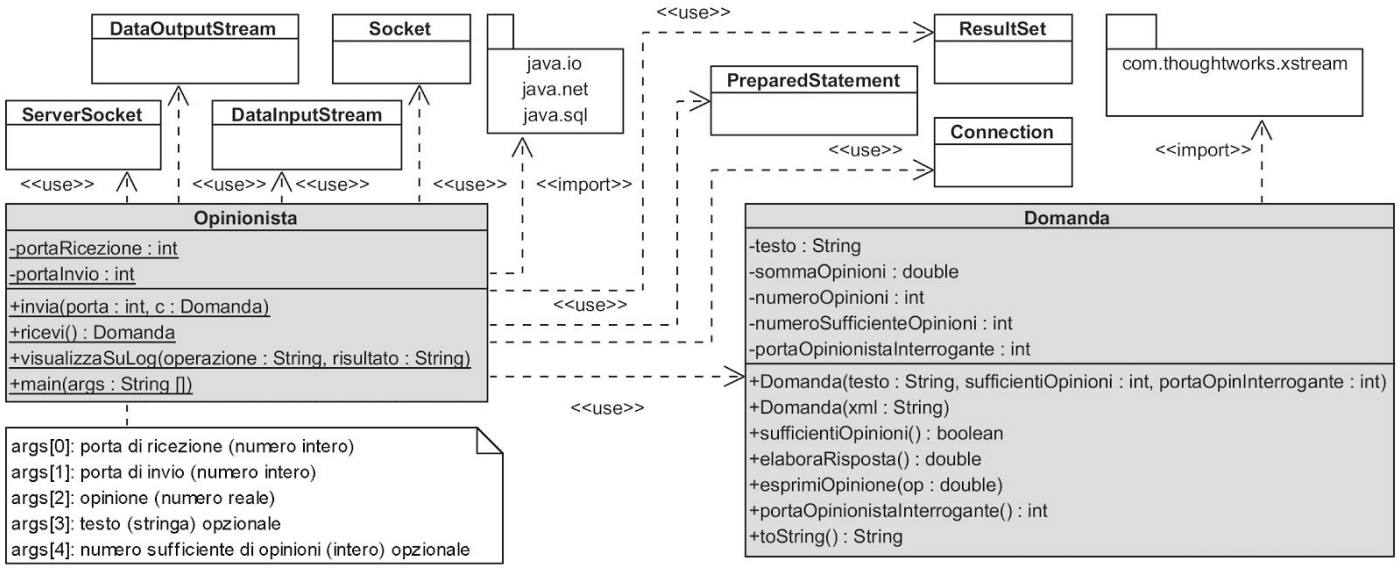

Fig.3 - Classi di cui deve essere composta l'applicazione (in grigio) e classi/package da usare (in bianco)

NOTE SU MYSQL CLIENT

A) Per poter rendere editabile la tabella *log* del database opinionista, in MySQL client :
Selezionare "Edit" > "Preferences" > "SQL Queries" > togliere la spunta da "Safe Updates"
Per svuotare il log ad ogni avvio: DELETE FROM opinionista.log;

B) Per ordinare le colonne in ordine di inserimento, cliccare sull'icona gialla sopra la tabella "Reset all sorted columns"

C) Per visualizzare il testo XML con indentazione cluccare sull'icona sopra la tabella accanto a "Wrap Cell Contents"