Programmazione Avanzata 16/09/2022

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 il caricamento 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

Il parcheggio di un autolavaggio ha una capienza pari a *n* posti. Quando una nuova auto arriva, se nel parcheggio c'è almeno un posto libero l'auto si ferma in attesa del proprio turno. Se invece non ci sono posti disponibili va via senza attendere. Un operaio lava le auto secondo il loro ordine di arrivo.
Realizzare una classe *Parcheggio* dotata almeno dei seguenti metodi:

- *void attendi()*: se ci sono posti liberi blocca l'auto fino a quando non è giunto il suo turno. Se non ci sono posti liberi viene lanciata *ParcheggioPienoException* (definire l'eccezione).
- *void lava()*: lava la prossima auto (secondo il loro ordine di arrivo). Se il parcheggio è vuoto, l'operaio si blocca fino a quando non arriva un'auto.

Definire anche le classi Auto e Operaio in modo che si comportino come segue.
Auto: cerca di entrare nel parcheggio e termina la propria esecuzione, o dopo il lavaggio o in caso di parcheggio pieno. Stampare un messaggio diverso nei due casi.
Operaio: lava 3 auto, si riposa 10 secondi e poi ricomincia.

Esercizio 2

Un sistema di *skill search* consente la ricerca di competenze specifiche, attraverso la consultazione di agenzie del lavoro, secondo un dato protocollo. Si consideri un numero di *agenzie del lavoro* non noto a priori, in comunicazione attraverso una configurazione ad anello. Il protocollo inizia con una *RichiestaDiCompetenze*, creata e inviata da una *AgenziaDelLavoro* (richiedente) alla successiva (ricevente). La *RichiestaDiCompetenze* è composta dalla descrizione delle competenze richieste, dalle porte di ascolto delle agenzie richiedente e proponente (quest'ultima stabilita nel primo giro di richiesta). Ogni agenzia ricevente, dopo aver visualizzato la richiesta, la invierà all'agenzia successiva se non ha una proposta, e cosi via. Non appena una agenzia ricevente ha una proposta, inserisce la propria porta di ascolto nell'oggetto *RichiestaDiCompetenze* e invia il medesimo all'agenzia richiedente invece che all'agenzia successiva. Quando l'agenzia richiedente riceverà la richiesta, la visualizzerà e la rimanderà all'agenzia ad essa successiva, e cosi via per il secondo giro. Nel secondo giro ogni agenzia riceve l'oggetto, lo visualizza, lo inoltra all'agenzia successiva e termina. Il secondo giro si ferma quando l'agenzia richiedente riceve nuovamente l'oggetto.

Si realizzi un'applicazione Java distribuita composta dalle classi *RichiestaDiCompetenze* e *AgenziaDelLavoro*. La classe *RichiestaDiCompetenze* mantiene ed elabora i dati suddetti, e raccoglie le offerte. La classe *AgenziaDelLavoro* svolge le operazioni di trasmissione di istanze di *RichiestaDiCompetenze* tra una agenzia e la successiva, in accordo al protocollo di cui sopra. La classe *AgenziaDelLavoro* svolge le operazioni di invio, ricezione e visualizzazione di istanze di *RichiestaDiCompetenze* esclusivamente in formato XML, in accordo al protocollo di cui sopra. Si assuma che le visualizzazioni di qualsiasi messaggio o contenuto non avvengano su console, ma tramite archiviazione su un database condiviso, visualizzato tramite MySQL Client. **Attenersi esattamente a quanto espresso dalle seguenti figure.** Fig.1 mostra una configurazione di test con tre agenzie. Fig.2 mostra lo schema 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 tre casi di test. Si noti che il richiedente si distingue per avere un "?" e il proponente per avere un "!" nelle rispettive descrizioni. Fig.5 mostra la sequenza di passi (ad alto livello) che le istanze di *AgenziaDelLavoro* svolgono in un caso di test. La logica di *AgenziaDelLavoro* e *RichiestaDiCompetenze* dovrà essere valida a prescindere dai numeri di porta e dal numero di agenzie.

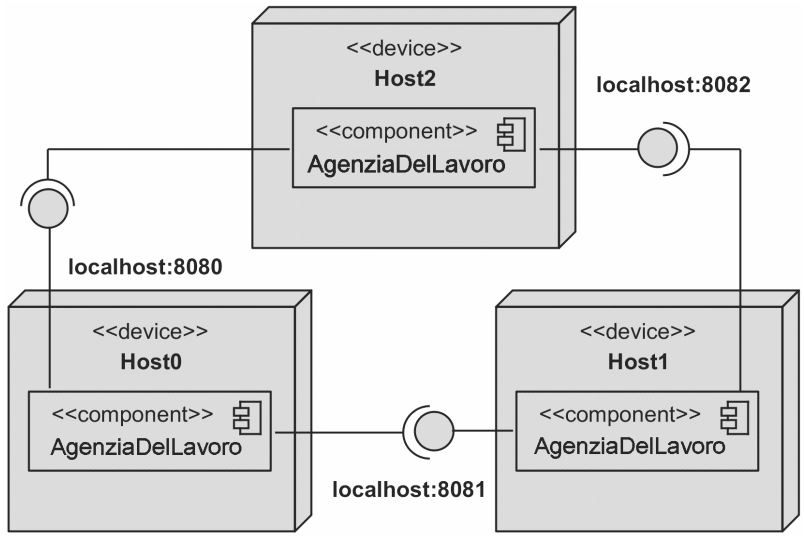
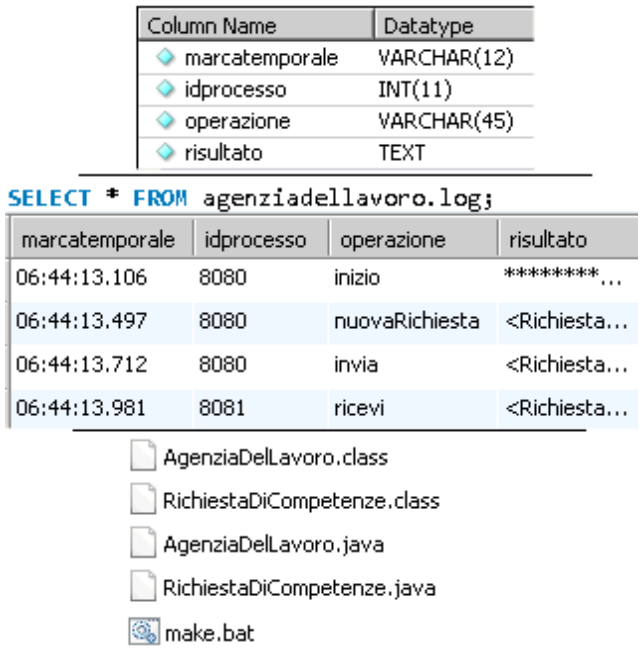


Fig.1 - Tre istanze di *AgenziaDelLavoro* e relative porte di ascolto



Column Name	Datatype
marcatemporale	VARCHAR(12)
idprocesso	INT(11)
operazione	VARCHAR(45)
risultato	TEXT

```
SELECT * FROM agenziadellavoro.log;
```

marcatemporale	idprocesso	operazione	risultato
06:44:13.106	8080	inizio	********...
06:44:13.497	8080	nuovaRichiesta	<Richiesta...
06:44:13.712	8080	invia	<Richiesta...
06:44:13.981	8081	ricevi	<Richiesta...

Fig.2 - Schema del database, tuple di esempio, e contenuto di c:\prg\myapps\esercizio2 di cui viene fornito solo make.bat

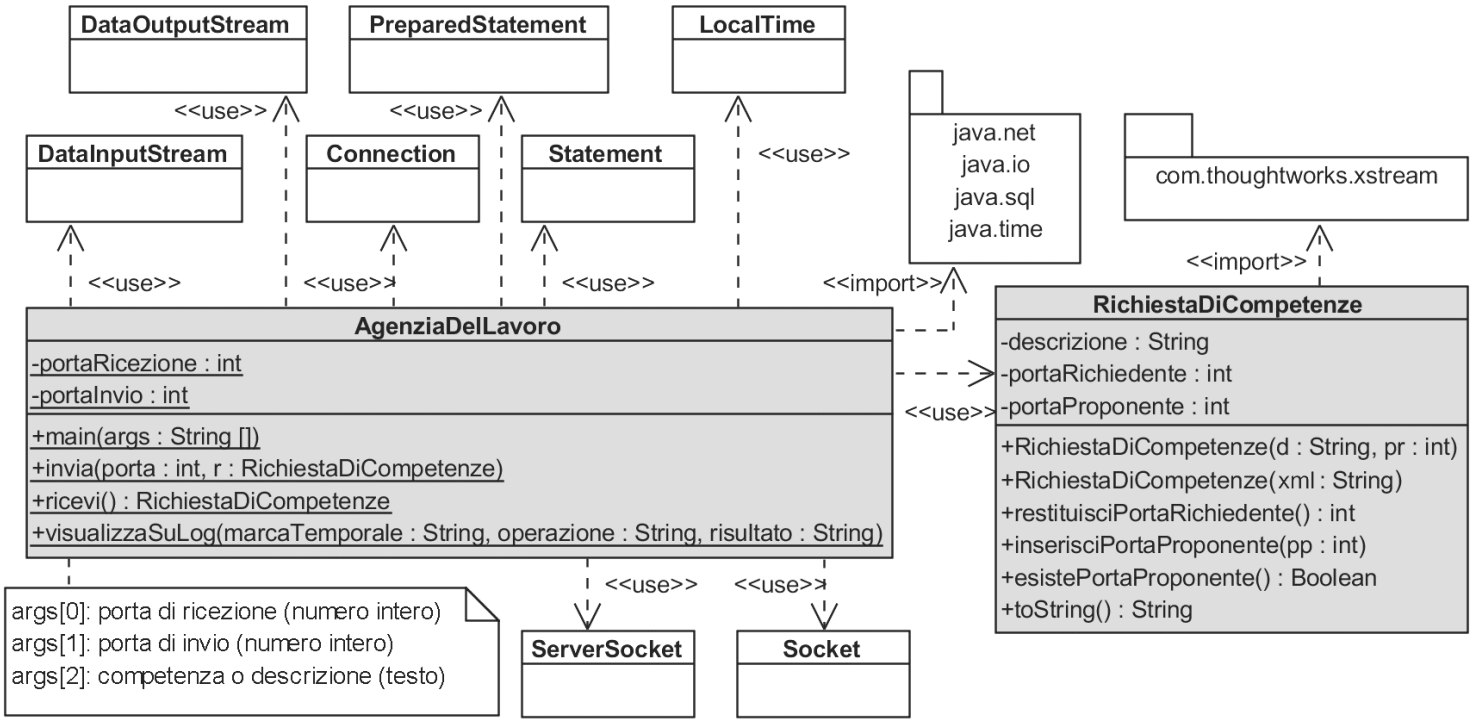


Fig.3 - Classi di cui deve essere composta l'applicazione (in grigio) e classi/package da usare (in bianco)

```
@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% AgenziaDelLavoro 8082 8080"
start cmd /k "color 2F && c:\prg\jdk8\bin\java -classpath %LIBS% AgenziaDelLavoro 8081 8082"
start cmd /k "color 2F && c:\prg\jdk8\bin\java -classpath %LIBS% AgenziaDelLavoro 8080 8081 "Programmatore C?""
rem competenza non disponibile, due giri completi di protocollo
pause

start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% AgenziaDelLavoro 8082 8080"
start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% AgenziaDelLavoro 8081 8082 "Programmatore C!""
start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% AgenziaDelLavoro 8080 8081 "Programmatore C?""
rem competenza disponibile in Agenzia1, al primo giro Agenzia2 non viene interpellata
pause

start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% AgenziaDelLavoro 8082 8080 "Programmatore C!""
start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% AgenziaDelLavoro 8081 8082"
start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% AgenziaDelLavoro 8080 8081 "Programmatore C?""
rem competenza disponibile in Agenzia2, due giri completi di protocollo
pause
```

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

NOTE SU MYSQL CLIENT

A) Per poter rendere editabile la tabella *log* del database opinionista, in MyQSL client:
   Selezionare "Edit" > "Preferences" > "SQL Queries" > togliere la spunta da "Safe Updates"
   Per svuotare il log ad ogni avvio: tasto destro sull'icona della tabelle *log* > "Truncate Table"

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 cliccare sull'icona sopra la tabella accanto a "Wrap Cell Contents"

D) Indirizzo per connettersi al database: "jdbc:mysql://localhost:3306/agenziadellavoro"

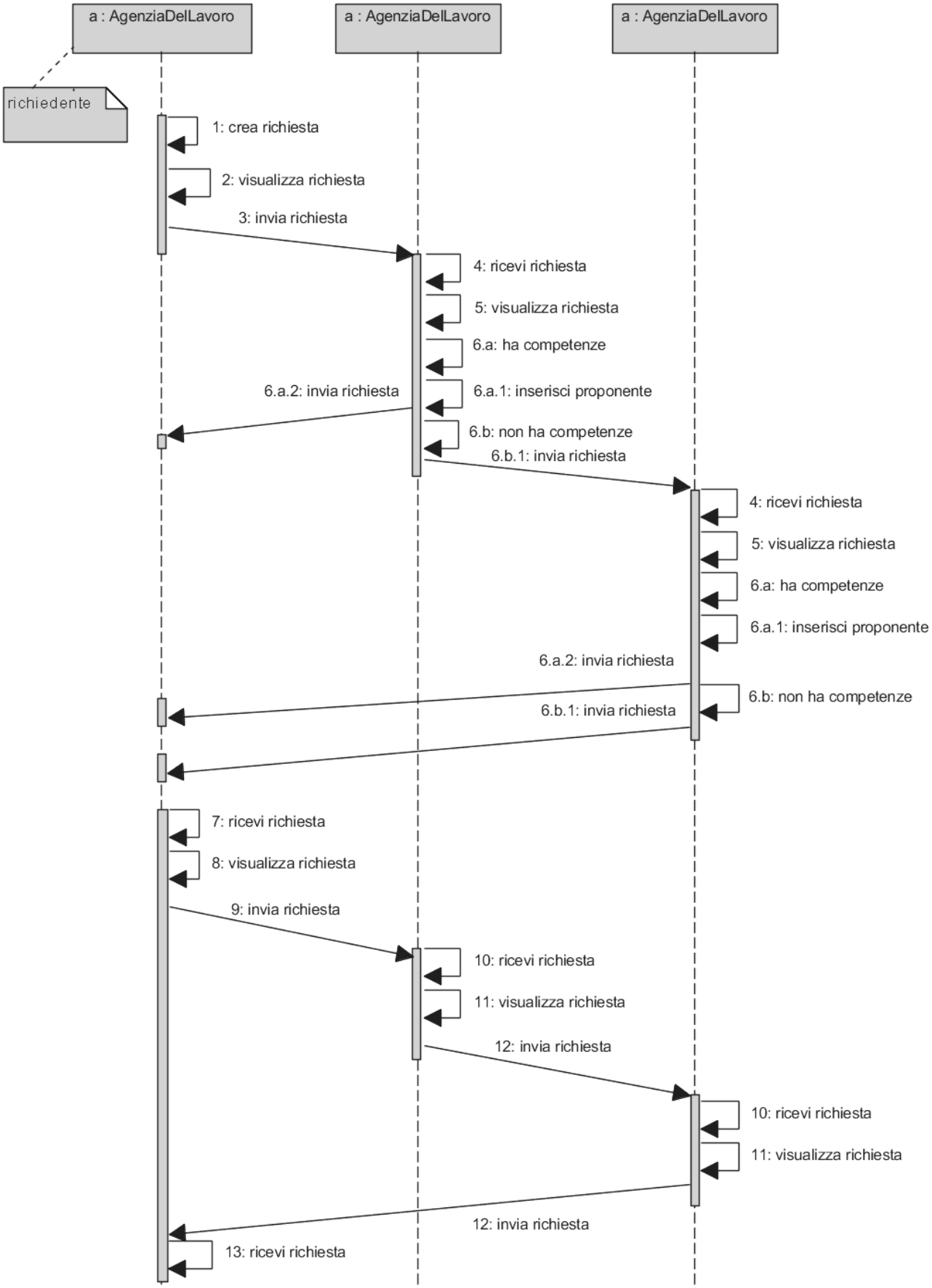


Fig.5 - Sequenza di passi che le istanze di *AgenziaDelLavoro* devono svolgere in un caso di test. Per brevità non viene rappresentata la classe *RichiestaDiCompetenze*.