Programmazione Avanzata 28/01/2022

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

Un mazzo contiene 40 carte diverse. Le carte hanno numeri che vanno da 1 a 10 e i seguenti semi: picche, fiori, quadri, cuori. Due giocatori siedono uno di fronte all'altro e ci sono **due** mazzi di carte. Il mazzo che si trova sulla sinistra del giocatore 1 è sulla destra del giocatore 2, mentre il mazzo a sinistra del giocatore 2 è sulla destra del giocatore 1. Ogni giocatore, senza attendere turni, opera come segue: controlla se ha delle coppie di carte uguali (sia il numero che il seme) e le rimuove dal gioco; scarta una carta tra quelle che ha in mano mettendola nel mazzo alla propria sinistra; prende dal mazzo alla propria destra tante carte quante gliene sevono per arrivare ad averne 5. Se il mazzo da cui un giocatore pesca le carte (quello alla sua destra) si svuota, il giocatore deve attendere che l'altro giocatore scarti qualche carta. Ogni coppia fatta vale 1 punto. Il giocatore che per primo arriva a 15 punti ha vinto. Scrivere i tipi Seme, Carta, Mazzo, Giocatore e Lancia in modo da ottenenere quanto descritto (la classe Lancia fa tutto quello che serve per avviare la partita). Evitare attese attive. Esempio di esecuzione:

```
. . .
G1: ho 5 carte [4F, 7F, 10Q, 10P, 4Q]
G1: ho 5 carte [4F, 7F, 10P, 4Q, 7C]
G1: ho 5 carte [4F, 10P, 4Q, 7C, 4Q]
G1: coppia di 4Q
G2: ho 5 carte [10C, 4F, 1F, 6F, 1C]
G2: ho 5 carte [10C, 1F, 6F, 1C, 10P]
G2: ho 5 carte [1F, 6F, 1C, 10P, 6C]
G2: ho 5 carte [1F, 6F, 1C, 10P, 6P]
G2: ho 5 carte [6F, 1C, 10P, 6P, 2C]
G2: ho 5 carte [1C, 10P, 6P, 2C, 5F]
G2: ho 5 carte [1C, 10P, 2C, 5F, 6Q]
G2: ho 5 carte [10P, 2C, 5F, 6Q, 8C]
G1: ho 5 carte [4F, 7C, 1Q, 4F, 10C]
G1: coppia di 4F
G2: ho 5 carte [10P, 2C, 5F, 6Q, 5C]
G2: ho perso, ho 7 punti
G1: ho vinto, ho 15 punti
```

Esercizio 2

Un sistema di *environmental crowdsensing* consente il rilevamento di un parametro ambientale (es. il livello di inquinamento dell'aria) attraverso misure provenienti da terminali mobili distribuiti nel territorio. Per maggiore affidabilità ci sono molti terminali per ogni area, in comunicazione secondo una configurazione ad albero binario. I terminali sono interrogati in ordine anticipato (radice, ramo sinistro, ramo destro), e cosi via in profondità finché il valor medio dei campioni rilevati, arrotondato al valore intero più vicino, non risulta identico tra due successivi campioni. A quel punto tale valor medio arrotondato diventa il parametro rilevato, e quindi la visita dell'albero binario completerà il livello corrente, ritornerà al livello superiore, e via via fino alla radice. Una nuova richiesta di calcolo è elaborata solo quando la richiesta corrente è ritornata alla radice.

Si realizzi un' applicazione Java distribuita, composta dalle classi *Parametro* e *Terminale*. La classe *Parametro* mantiene ed elabora i campioni per ottenere il parametro. La classe *Terminale* svolge le operazioni di trasmissione in formato XML di istanze di *Parametro*, 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 cinque terminali, con rispettive porte di ascolto. 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 alcuni casi di test. Si noti che il terminale posto sulla radice si distingue per avere la porta di ascolto 8080, mentre i nodi interni dell'albero con figli si distinguono per avere quattro variabili di ingresso. Fig.5 mostra la sequenza di passi (ad alto livello) che le istanze di Terminale svolgono in un caso di test. La logica di Terminale e Parametro dovrà essere valida a prescindere dai numeri di porta diversi da 8080 e dal numero di terminali.

```
@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% Terminale 10.4 8084 8082"
start cmd /k "color 2F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 10.0 8083 8084"
start cmd /k "color 2F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 10.2 8082 8080 8083"
start cmd /k "color 2F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 10.3 8081 8082"
start cmd /k "color 2F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 10.2 8080 8081"
pause
taskkill /f /im "java.exe"
rem campioni stabili a media 10.0 su Host1
pause

start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 15.4 8084 8082"
start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 11.0 8083 8084"
start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 17.8 8082 8080 8083"
start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 19.3 8081 8082"
start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 7.5 8080 8081"
pause
taskkill /f /im "java.exe"
rem campioni stabili a media 14.0 su Host4
pause

start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 30.4 8084 8082"
start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 50.0 8083 8084"
start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 17.8 8082 8080 8083"
start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 19.3 8081 8082"
start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% Terminale 7.5 8080 8081"
pause
taskkill /f /im "java.exe"
rem i campioni a media 25.0 su Host0, ma non stabili
pause
```

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

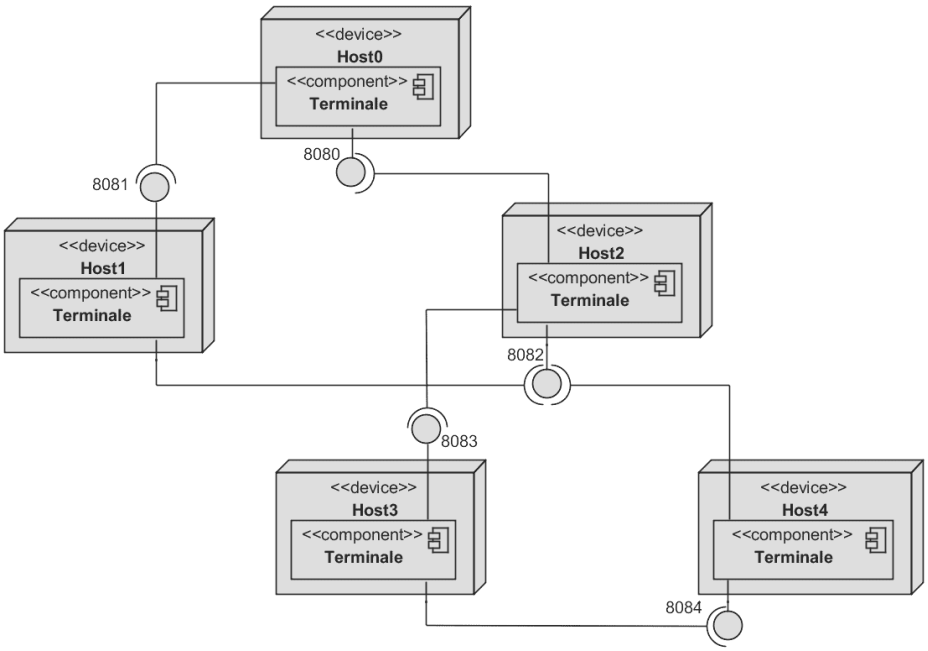
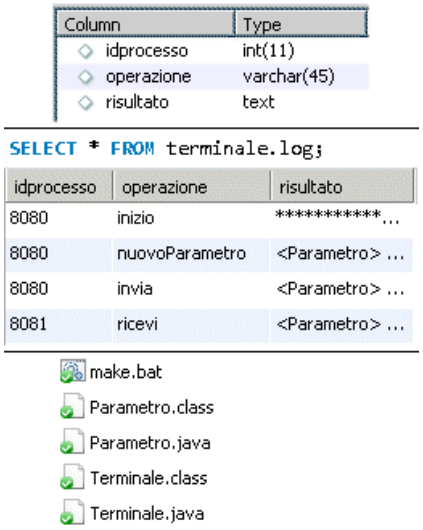


Fig.1 - Cinque istanze di *Terminale* e relative porte di ascolto



(viene fornito solo make.bat)

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

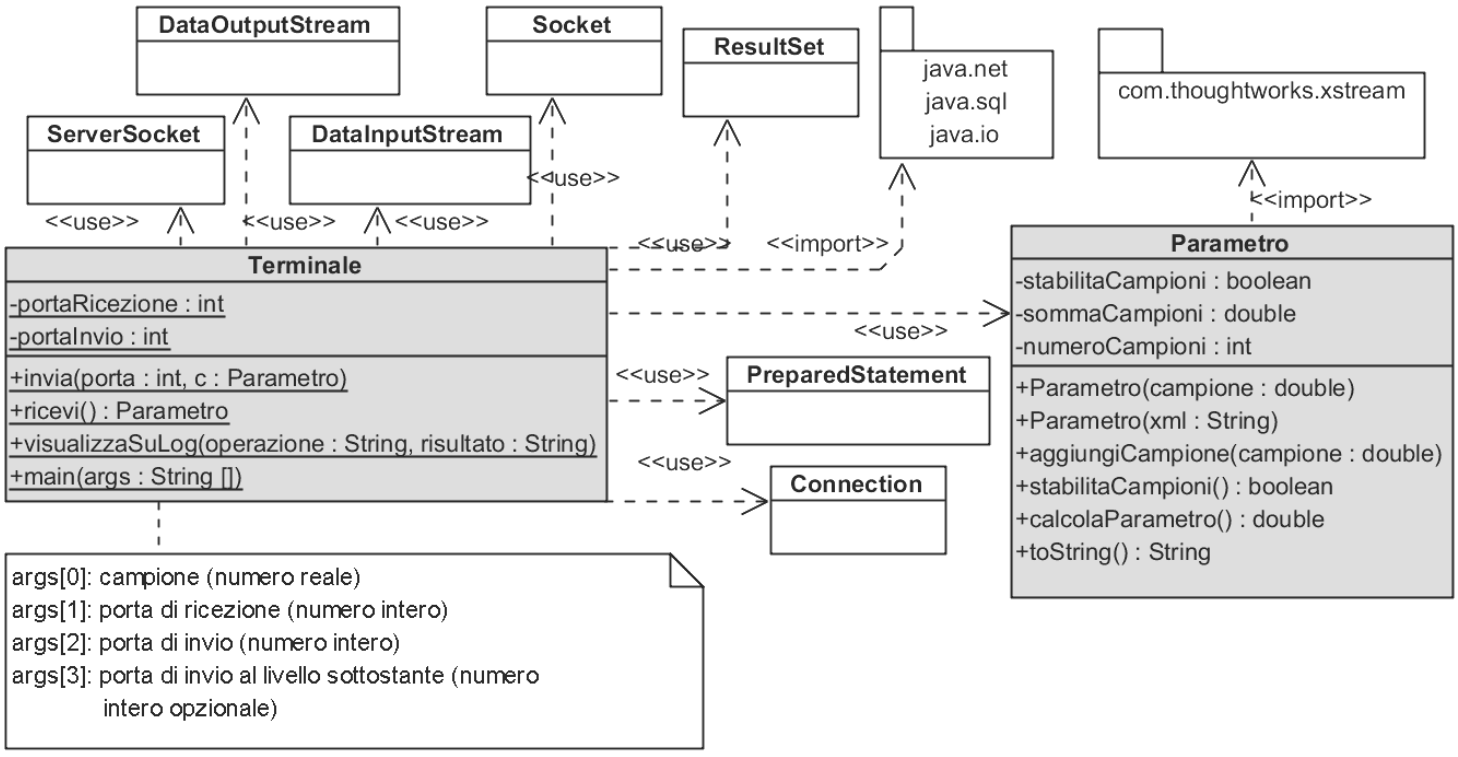


Fig.3 - Classi di cui deve essere composta l'applicazione (in grigio) e classi/package da usare (in bianco)

NOTE SU MYSQL CLIENT

A) Per poter rendere editabile la tabella *log* del database opinionista, in MyQSL client :
Selezionare "Edit" > "Preferences" > "SQL Queries" > togliere la spunta da "Safe Updates"
Per svuotare il log ad ogni avvio: DELETE FROM terminale.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 cliccare sull'icona sopra la tabella accanto a "Wrap Cell Contents"

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

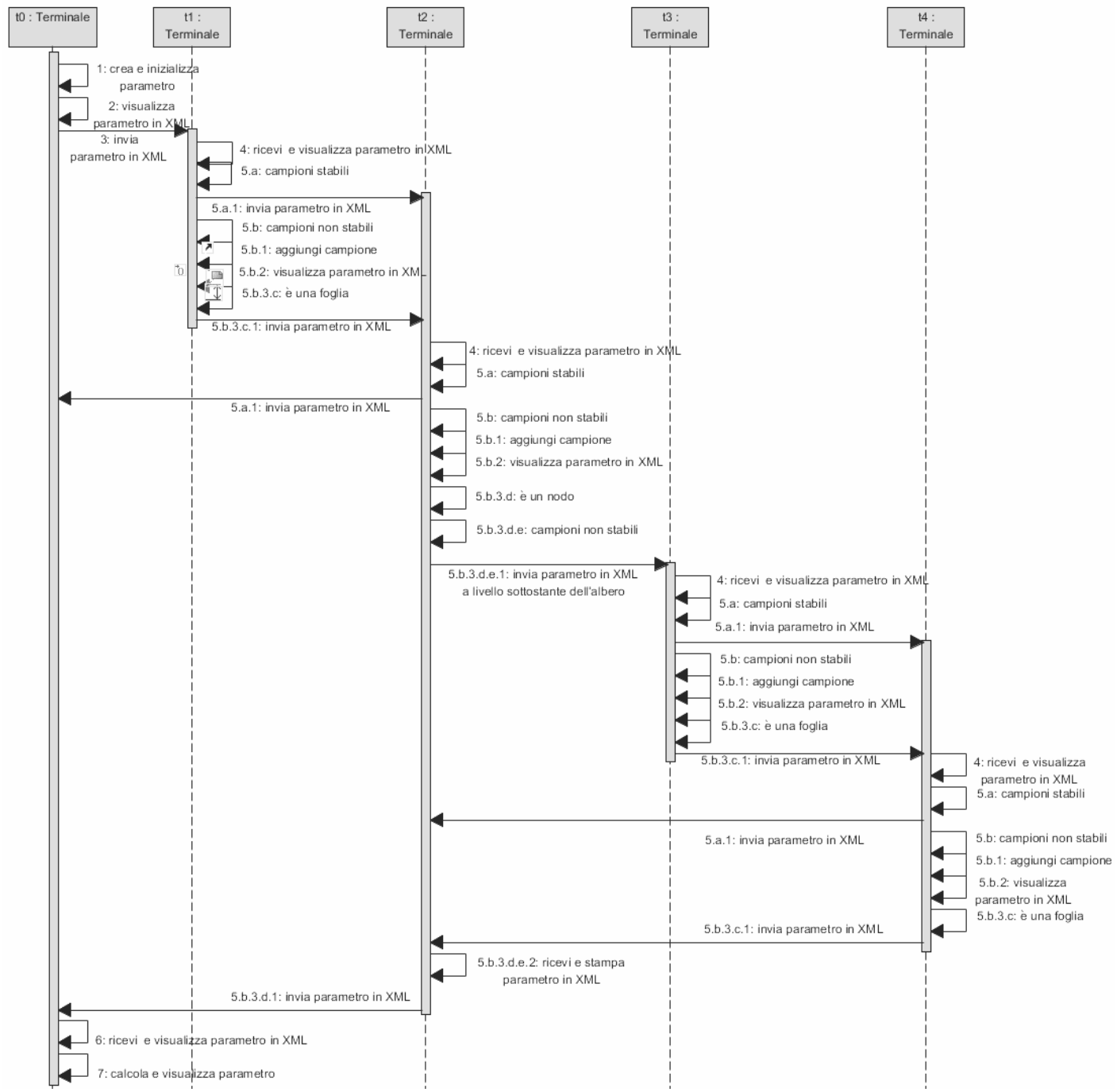


Fig.5 - Sequenza di passi che le istanze di *Terminale* devono svolgere nel un caso di test. Per brevità non viene rappresentata la classe *Parametro*.