

Note sullo svolgimento della prova

- Non è consentito l'uso di vecchi testi d'esame, libri, appunti o dispositivi elettronici personali.
- 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

Un compito è un'operazione da eseguire in uno specifico istante del futuro. Una volta che un compito è stato eseguito è possibile recuperare il risultato dell'operazione. Un compito può essere definito a partire dalla seguente interfaccia:

```
public interface Compito {
    void esegui();
    Object dammiRisultato() throws Exception;
    boolean finito();
    int getTolleranza();
    long getQuando();
    void setException(Exception e);
}
```

Il metodo *getQuando()* permette di conoscere l'istante in cui il compito dovrà essere eseguito. Il tempo è espresso in millisecondi, secondo la convenzione di *System.currentTimeMillis()*. Ad ogni compito è associata una tolleranza: il compito può essere eseguito anche partendo con un certo ritardo rispetto all'istante previsto, a patto che il ritardo sia minore della tolleranza. La tolleranza è espressa in millisecondi. Se il ritardo è maggiore della tolleranza il compito non viene eseguito. Il metodo *finito()* restituisce *true* se il compito è terminato (con successo o con fallimento), *false* se deve ancora essere eseguito o è ancora in esecuzione. Il metodo *dammiRisultato()* consente di recuperare il risultato di un compito. Il metodo *dammiRisultato()* restituisce immediatamente un valore se possibile, altrimenti blocca il chiamante fino a quando il compito non è stato eseguito. Se l'esecuzione di un compito non va a buon fine, per eccessivo ritardo o per altri motivi, il metodo *dammiRisultato()* non restituisce alcun valore e lancia un'eccezione. Il tipo di eccezione da lanciare può essere impostato con il metodo *setException()*.

Scrivere una classe *NumeroPrimoCompito* che implementa opportunamente l'interfaccia *Compito*. La classe *NumeroPrimoCompito* ha un costruttore che permette di specificare l'istante di esecuzione, la tolleranza e un numero da analizzare. Il metodo *esegui()* verifica se il numero da analizzare è primo oppure no.

Realizzare una classe *EsecutoreFuturo* in grado di eseguire compiti usando un flusso di esecuzione proprio.

L'esecuzione di un compito può essere soggetta a ritardo se il compito precedente ha dei tempi di esecuzione troppo lunghi. La classe deve essere dotata di un metodo *void assegna(Compito c)* che permette di aggiungere il compito *c* all'insieme di compiti gestiti dall'esecutore.

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 così 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.

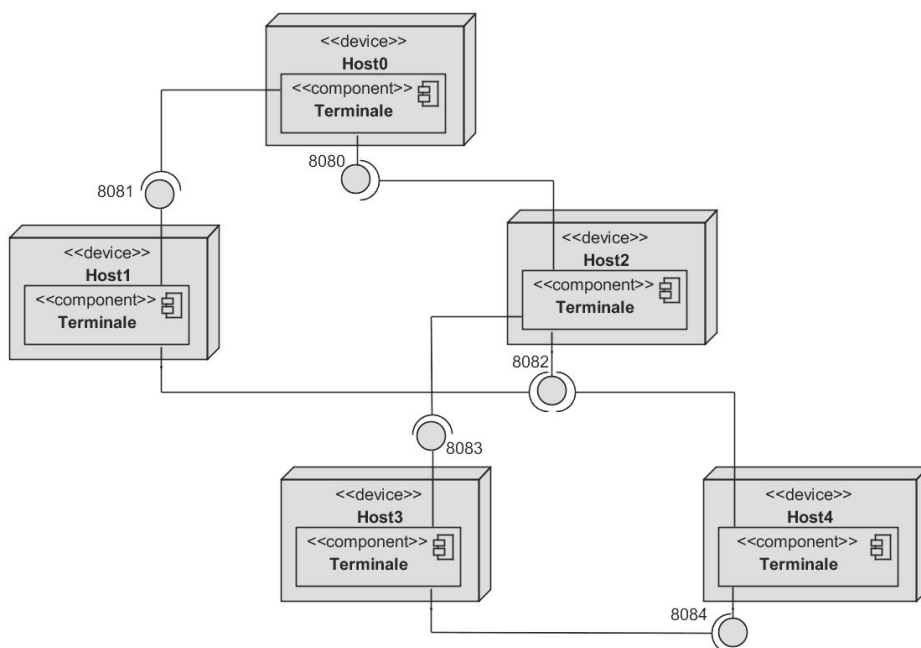


Fig.1 - Cinque istanze di *Terminale* e relative porte di ascolto

Column	Type
◇ idprocesso	int(11)
◇ operazione	varchar(45)
◇ risultato	text


```
SELECT * FROM terminale.log;
```

idprocesso	operazione	risultato
8080	inizio	***** ...
8080	nuovoParametro	<Parametro> ...
8080	invia	<Parametro> ...
8081	ricevi	<Parametro> ...

- make.bat
- Parametro.class
- Parametro.java
- Terminale.class
- Terminale.java

(viene fornito solo make.bat)

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

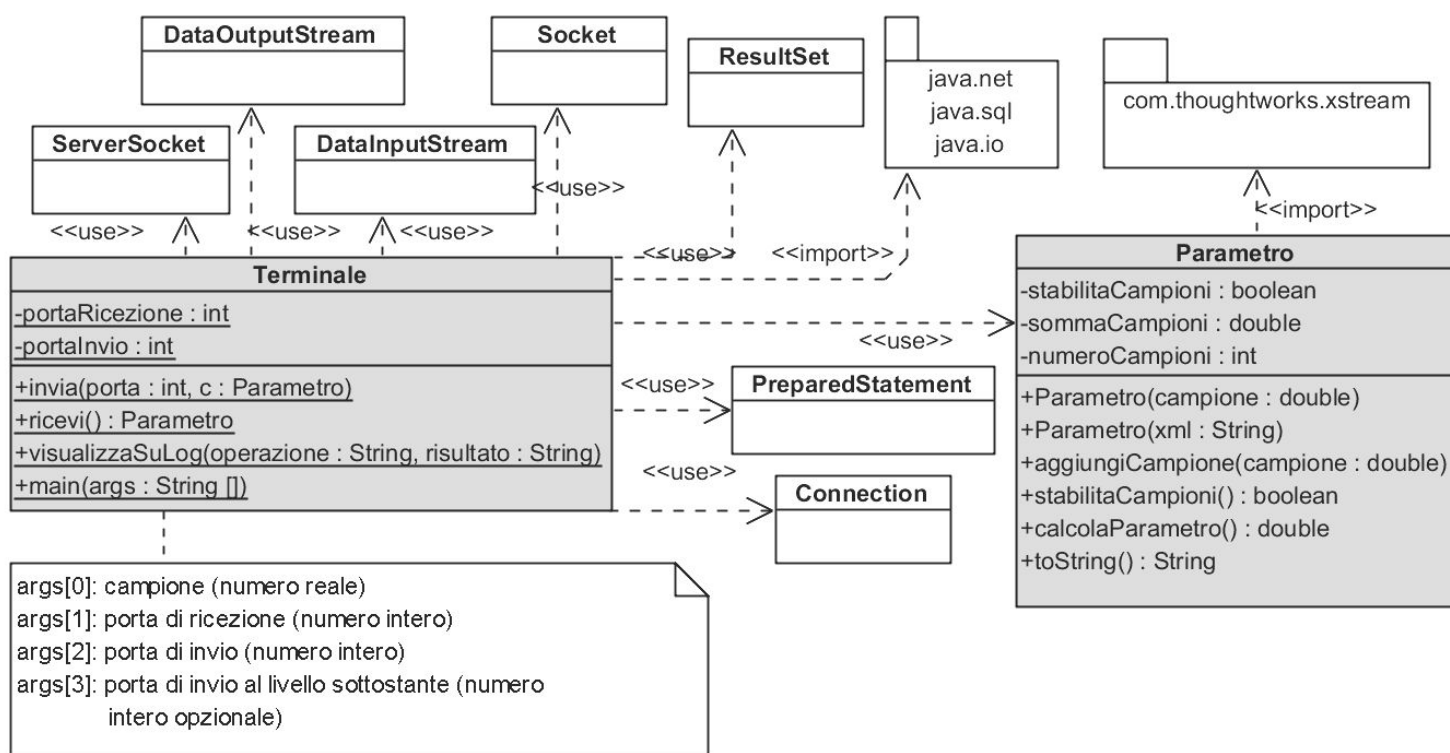


Fig.3 - Classi di cui deve essere composta l'applicazione (in grigio) e classi/package da usare (in bianco)

NOTE SU MYSQL CLIENT

- 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 terminale.log;
- Per ordinare le colonne in ordine di inserimento, cliccare sull'icona gialla sopra la tabella "Reset all sorted columns"
- 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"

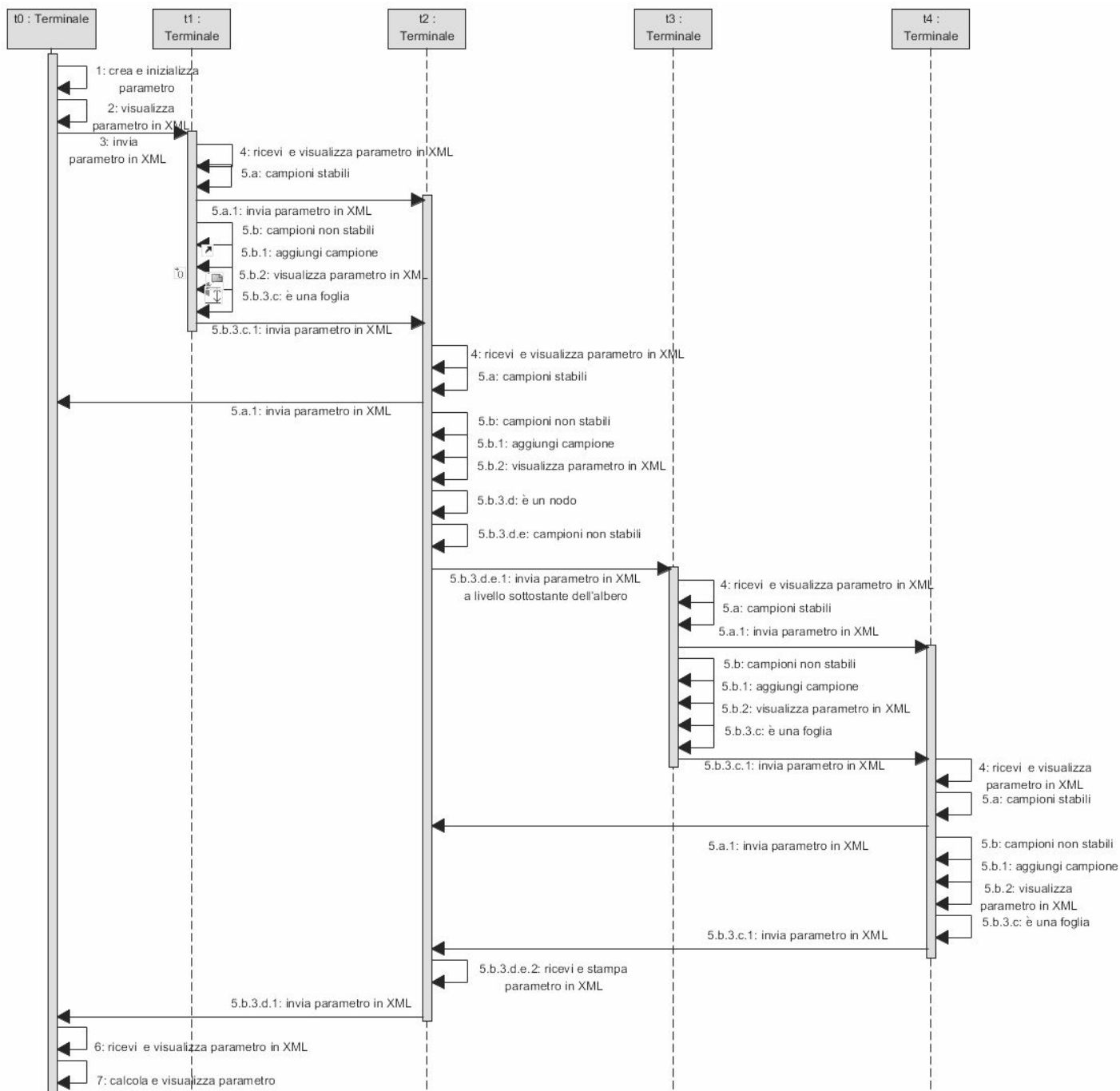


Fig.5 - Sequenza di passi che le istanze di *Terminale* devono svolgere nel un caso di test. Per brevità non viene rappresentata la classe *Parametro*.