Programmazione 01/07/2015

Note sullo svolgimento della prova

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

Esercizio 1

Un documento contiene caratteri. Il numero massimo di caratteri di un documento è limitato ed è pari ad *n*. Il numero di caratteri effettivamente contenuti in un documento può essere minore del massimo. Inizialmente il documento è vuoto (il numero di caratteri in esso contenuti è zero). Un documento può essere editato da più thread contemporaneamente. Le operazioni di modifica (inserimento e cancellazione) possono essere eseguite solo fino a quando il documento non è reso definitivo. Realizzare una classe *Documento* dotata almeno dei seguenti costruttori e metodi:

- *Documento()*: crea un documento con dimensione massima pari a 100 caratteri.
- *Documento(int n)*: crea un documento avente la dimensione massima specificata.
- *void cancella(int pos, int d)*: rimuove dal documento *d* caratteri a partire da quello di posizione *pos*. Gli eventuali altri caratteri posizionati dopo quelli della regione "rimossa" vengono traslati per occuparne le posizioni.
- *void inserisci(int pos, String s)*: se nel documento c'è abbastanza spazio, inserisce i caratteri della stringa *s* nel documento a partire dalla posizione *pos*. Il numero di caratteri del documento aumenta di una quantità pari alla lunghezza di *s*. Se nel documento non c'è abbastanza spazio, si blocca fino a quando non ce n'è a sufficienza.
- *String versioneDefinitiva()*: restituisce, sotto forma di stringa, il contenuto del documento (i caratteri che lo compongono). Il metodo è bloccante nel caso in cui il documento non sia defintivo (e si sblocca quando lo diviene).
- v*oid rendiDefinitivo()*: rende definitivo il documento.

Esempio di operazioni:

```
Documento d = new Documento(10);
d.inserisci(0, "abc");          // abc
d.inserisci(1, "XYZW");         // aXYZWbc
d.cancella(2, 3);               // aXbc
```

I metodi *cancella()* e *inserisci()* lanciano una eccezione di tipo *NonModificabileException* nel caso in cui vengano invocati su un documento in stato definitivo (definire tale eccezione).

Definire inoltre:

- un thread *Inseritore* che inserisce 5 parole in un documento e poi lo rende definitivo;
- un thread *Visualizzatore* che attende che un documento sia reso definitivo e lo stampa a video;
- una classe di prova che crea un documento condiviso a un Inseritore e un Visualizzatore e li fa partire.

Esercizio 2

Un sistema di asta online consente di attribuire un prezzo di vendita ad un bene attraverso la rete Internet. Si consideri un numero di *Commercianti* non noto a priori, in comunicazione secondo una configurazione ad anello. Un'asta inizia con una *inserzione* creata e inviata da un commerciante (proponente) al successivo commerciante (ricevente). Una inserzione è composta dalla *descrizione del bene*, dalla *migliore offerta* (inizialmente pari alla base d'asta), dall'*incremento d'asta*, dalle *porte di ascolto* dei commercianti proponente e compratore (quest'ultima stabilita alla fine). Ogni commerciante ricevente, dopo aver visualizzato l'inserzione, effettuerà una offerta e invierà l'inserzione al commerciante successivo, e cosi via. Una nuova offerta per essere valida deve essere pari alla migliore offerta più un multiplo dell'incremento d'asta. Durante il primo giro, ogni offerta valida sostituisce la migliore offerta. Quando il proponente riceverà l'inserzione, la visualizzerà e la rimanderà al commerciante successivo, e cosi via per il secondo giro, che si ferma quando un commerciante ricevente rileva che la sua offerta fatta nel primo giro è pari alla migliore offerta. A quel punto il commerciante inserirà la propria porta di ascolto nella inserzione e la invierà direttamente al commerciante proponente.

Si realizzi un'applicazione Java distribuita composta dalle classi *Inserzione* e *Commerciante*. La classe *Inserzione* mantiene ed elabora i dati suddetti, e raccoglie le offerte. La classe *Commerciante* svolge le operazioni di trasmissione di istanze di *Inserzione* tra un commerciante e il successivo, in accordo al protocollo di cui sopra.

Attenersi esattamente a quanto espresso dalle seguenti figure. Fig.1 mostra una configurazione di test con soli tre votanti. Fig.2 mostra 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 quattro casi di test. Si noti che il proponente si distingue per avere in ingresso più di tre parametri. Fig.5 mostra la sequenza di passi (ad alto livello) che le istanze di *Commerciante* svolgono in un caso di test. La logica di *Commerciante* e *Inserzione* dovrà essere valida a prescindere dai numeri di porta e dal numero di votanti.

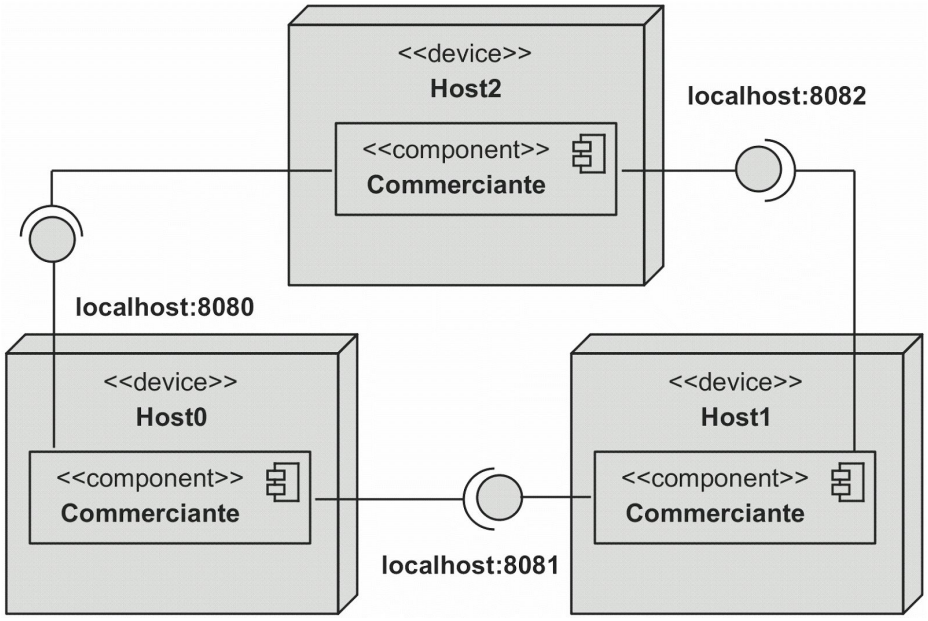
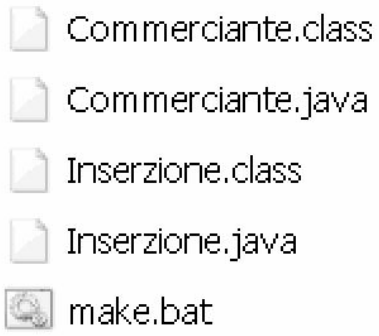


Fig.1 - Tre istanze di *Commerciante* e relative porte di ascolto



(viene fornito solo make.bat)

Fig.2 - c:\prg\esercizio2

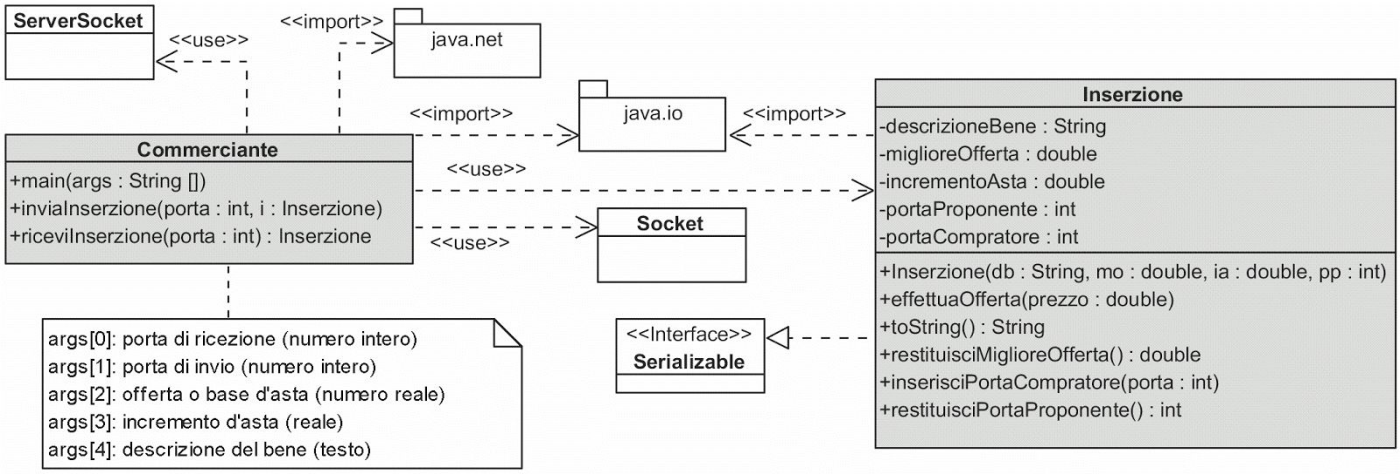


Fig.3 - Classi di cui deve essere composta l'applicazione (in grigio) e classi/package da usare (in bianco)

```
@echo off
c:\prg\jdk8\bin\javac *.java
pause

start cmd /k "color 2F && c:\prg\jdk8\bin\java Commerciante 8082 8080 30"
pause
start cmd /k "color 2F && c:\prg\jdk8\bin\java Commerciante 8081 8082 40"
pause
start cmd /k "color 2F && c:\prg\jdk8\bin\java Commerciante 8080 8081 20 10 "Mouse di Alan Turing""
pause
taskkill /f /im "java.exe"
rem come risultato stampa "compratore 8081"
pause

start cmd /k "color 3F && c:\prg\jdk8\bin\java Commerciante 8082 8080 30"
pause
start cmd /k "color 3F && c:\prg\jdk8\bin\java Commerciante 8081 8082 45"
pause
start cmd /k "color 3F && c:\prg\jdk8\bin\java Commerciante 8080 8081 20 10 "Mouse di Alan Turing""
pause
taskkill /f /im "java.exe"
rem come risultato stampa "compratore 8082"
pause

start cmd /k "color 4F && c:\prg\jdk8\bin\java Commerciante 8082 8080 40"
pause
start cmd /k "color 4F && c:\prg\jdk8\bin\java Commerciante 8081 8082 40"
pause
start cmd /k "color 4F && c:\prg\jdk8\bin\java Commerciante 8080 8081 20 10 "Mouse di Alan Turing""
pause
taskkill /f /im "java.exe"
rem come risultato stampa "compratore 8081"
pause

start cmd /k "color 5F && c:\prg\jdk8\bin\java Commerciante 8082 8080 20"
pause
start cmd /k "color 5F && c:\prg\jdk8\bin\java Commerciante 8081 8082 20"
pause
start cmd /k "color 5F && c:\prg\jdk8\bin\java Commerciante 8080 8081 20 10 "Mouse di Alan Turing""
pause
taskkill /f /im "java.exe"
rem come risultato stampa "compratore 8081"
pause
```

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

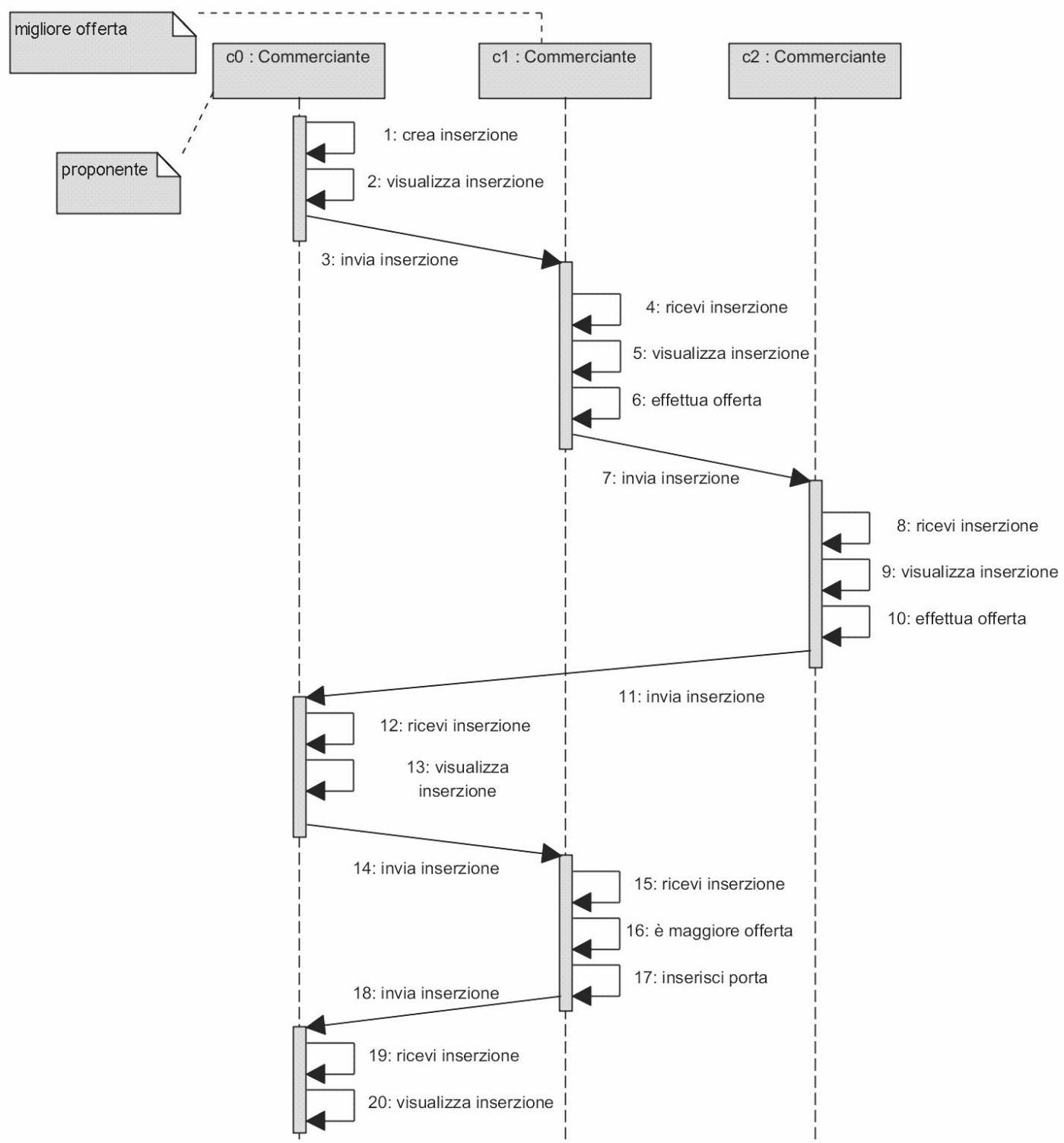


Fig.4 - Sequenza di passi che le istanze di *Commerciante* devono svolgere in un caso di test. Per brevità non viene rappresentata la classe *Inserzione*.