Programmazione 22/2/2018

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

La morra si gioca in due e il gioco consiste nell'indovinare la somma dei numeri mostrati con le dita dai due giocatori. I due giocatori simultaneamente mostrano un numero di dita a scelta (da zero a cinque) mentre pronunciano un numero da uno a dieci. Se un giocatore indovina la somma e l'altro no, il primo vince e il secondo perde. Se entrambi indovinano la somma o se nessuno dei due la indovina la partita finisce in parità.

Scrivere una classe *Partita* che realizza quanto sopra descritto. La classe deve essere dotata di un metodo *Risultato mossa(int d, int t)* che consente ai giocatori di eseguire una giocata, dove *d* è il numero di dita e *t* il numero pronunciato (definire anche *Risultato*). Il primo giocatore che invoca il metodo rimane bloccato in attesa del secondo. Quando anche il secondo giocatore ha espresso i suoi valori entrambe le invocazioni di metodo restituiscono il risultato. Scrivere anche una classe *Giocatore* (thread) che genera un valore a caso per *d* e per *t*, chiama il metodo *mossa()*, e stampa il risultato.

Esercizio 2

Un sistema di votazione elettronica consente la votazione di proposte tramite la rete Internet. Si consideri un numero di *Votanti* non noto a priori, in comunicazione secondo una configurazione ad anello. Una votazione inizia con una *proposta* (il cui contenuto è rappresentato con del testo breve) votata e inviata da un votante al successivo, che a sua volta voterà e invierà la proposta al successivo, e cosi via. Quando il proponente riceverà la proposta, la elaborerà e visualizzerà il risultato, il quale verrà poi visualizzato e inviato da un votante al successivo, fino a giungere nuovamente al proponente.

La proposta può essere votata in tre modi possibili: SI, NO, NI (Non Interessante). Il risultato viene elaborato nel seguente modo: se il numero dei votanti che considerano la proposta non interessante è minore degli altri votanti, allora la proposta è considerata *accettata* o *rifiutata* a seconda che ci sia una maggioranza di SI o NO rispettivamente. In tutti gli altri casi la proposta viene considerata *ingiudicata*;

Si realizzi un'applicazione Java distribuita composta dalle classi *Proposta* e *Votante*. La classe *Proposta* viene istanziata con il contenuto della proposta, raccoglie le votazioni (anonime), elabora e restituisce il risultato. La classe *Votante* svolge le operazioni di trasmissione di istanze di *Proposta* tra un votante e il successivo, in accordo al protocollo di cui sopra.

Attenersi esattamente a quanto espresso dalle seguenti figure. Fig.1 mostra la 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 tre casi di test. Si noti che il proponente si distingue per avere in ingresso un parametro in più, che rappresenta il contenuto della proposta. Fig.5 mostra la sequenza di passi (ad alto livello) che le istanze di *Votante* svolgono in ogni caso di test. La logica di *Votante* e *Proposta* dovrà essere valida a prescindere dai numeri di porta e dal numero di votanti.

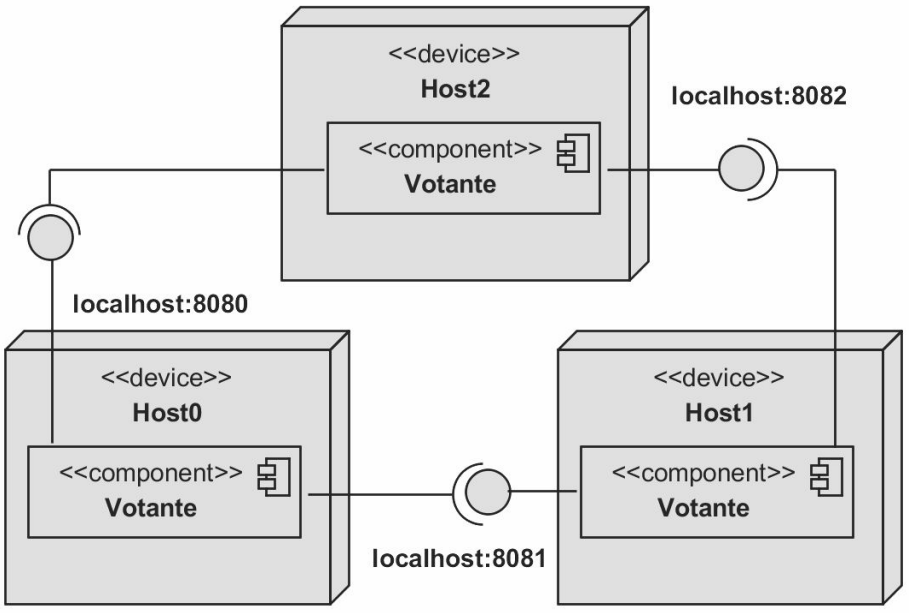
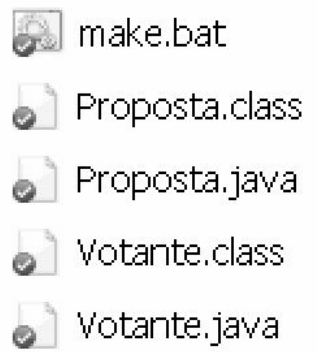


Fig.1 - Tre istanze di *Votante* e relative porte di ascolto



(viene fornito solo make.bat)

Fig.2 - c:\prg\esercizio2

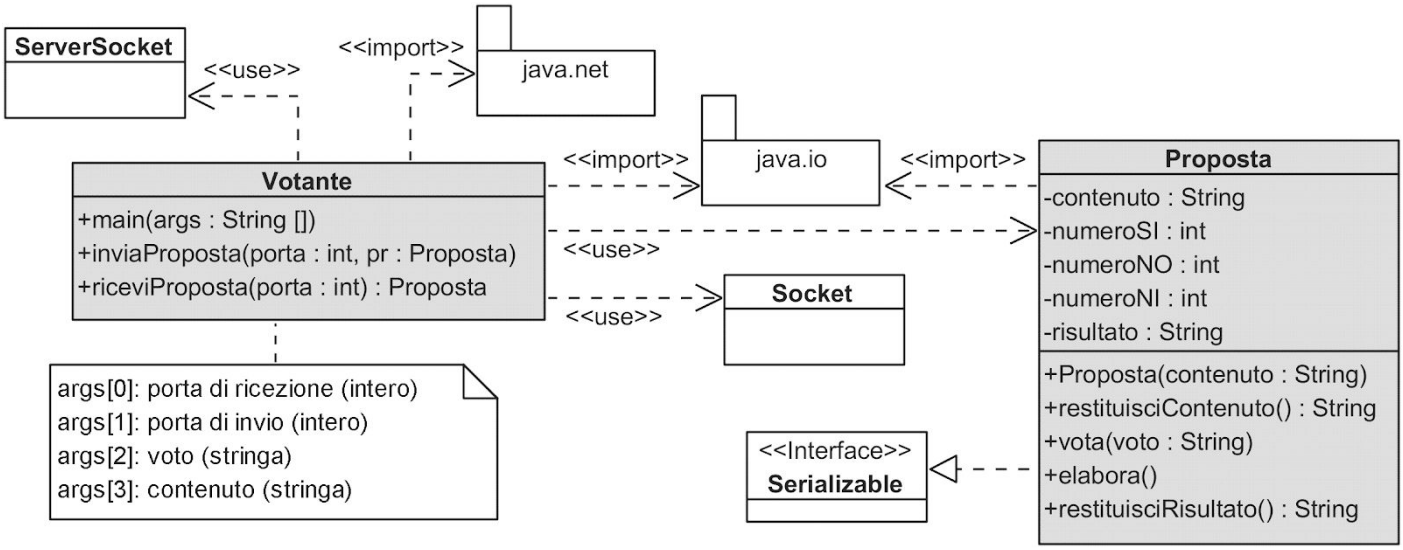


Fig.3 - Classi di cui deve essere composta l'applicazione (in grigio) e classi/package da usare (in bianco)

```
c:\prg\jdk8\bin\javac *.java
pause
start cmd /k "c:\prg\jdk8\bin\java Votante 8082 8080 SI"
pause
start cmd /k "c:\prg\jdk8\bin\java Votante 8081 8082 NO"
pause
start cmd /k "c:\prg\jdk8\bin\java Votante 8080 8081 SI "Propongo di introdurre gli OGM""
pause
rem come risultato stampa "accettata"
pause
start cmd /k "c:\prg\jdk8\bin\java Votante 8082 8080 NO"
pause
start cmd /k "c:\prg\jdk8\bin\java Votante 8081 8082 NO"
pause
start cmd /k "c:\prg\jdk8\bin\java Votante 8080 8081 SI "Propongo di abolire il limite di velocita""
pause
rem come risultato stampa "rifiutata"
pause
start cmd /k "c:\prg\jdk8\bin\java Votante 8082 8080 NI"
pause
start cmd /k "c:\prg\jdk8\bin\java Votante 8081 8082 NO"
pause
start cmd /k "c:\prg\jdk8\bin\java Votante 8080 8081 SI "Propongo di introdurre l'eutanasia""
pause
rem come risultato stampa "ingiudicata"
```

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

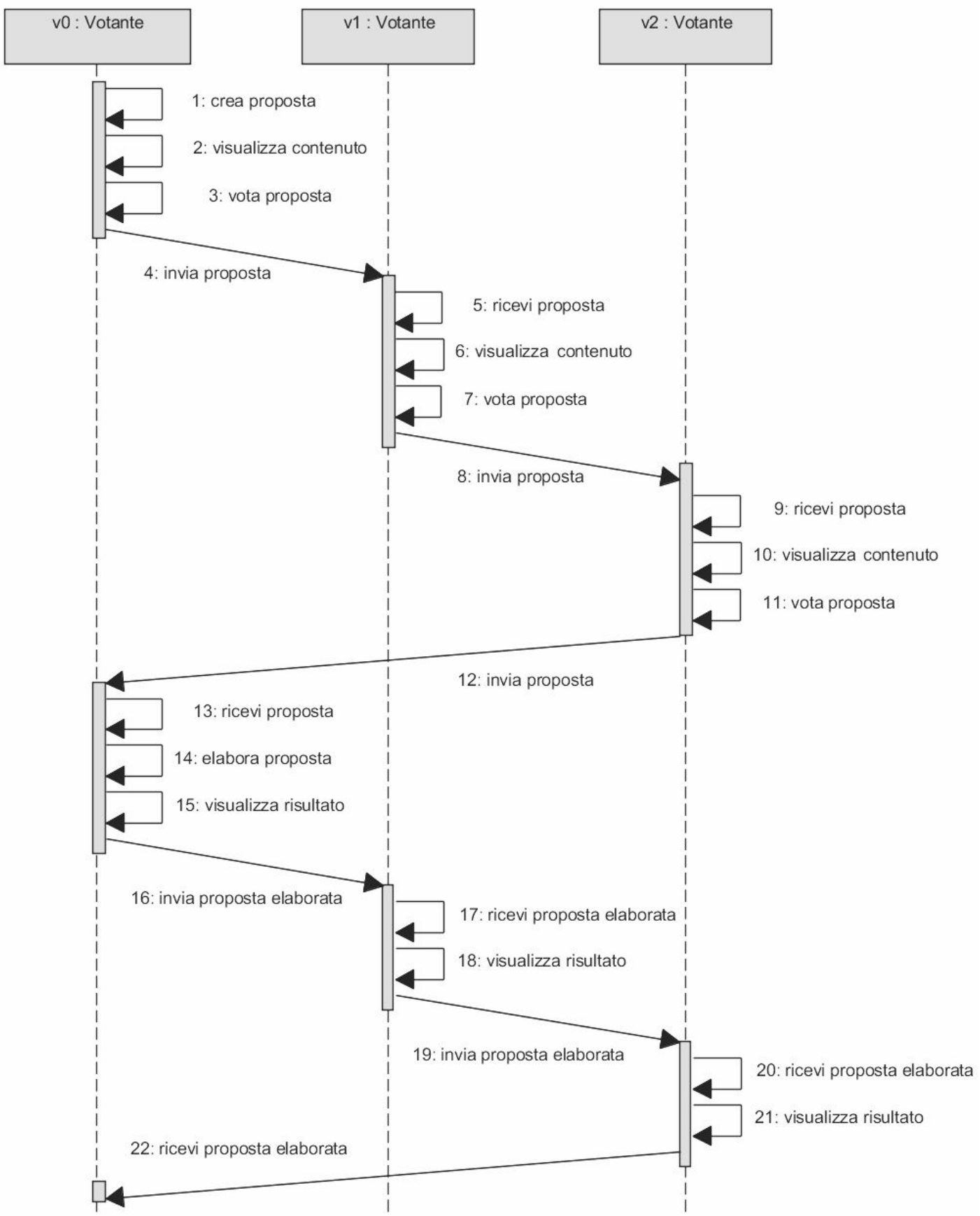


Fig.4 - Sequenza di passi che le istanze di *Votante* devono svolgere. Per brevità non viene rappresentata la classe *Proposta*.