Programmazione 22/07/2015

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

Una lavagna è uno spazio condiviso da più utenti in cui possono essere inserite e cancellate forme geometriche. Prima di poter modificare una lavagna, un utente deve "entrare" nella lavagna; una volta terminate le modifiche l'utente "esce" dalla lavagna. Il numero di utenti che contemporaneamente possono essere "dentro" una lavagna è limitato ed è pari a *n*.

Creare una classe *Lavagna* (ed eventuali classi ausiliarie) dotata almeno dei seguenti costruttori e metodi:

- *Lavagna(int n)*: crea una lavagna di dimensione 100x100 che può contenere al più *n* utenti.
- *Lavagna(float x, float y, int n)*: crea una lavagna larga *x* e alta *y* che può contenere al più *n* utenti.
- *Utente[] entra(Utente u)*: l'utente *u* entra nella lavagna. Il metodo è bloccante nel caso in cui ci siano già *n* utenti nella lavagna. Il metodo restituisce tutti gli utenti attualmente dentro la lavagna.
- *void esci(Utente u)*: l'utente *u* esce dalla lavagna.
- *void inserisci(Forma f)*: inserisce nella lavagna la forma *f*. Lancia *IllegalArgumentException* nel caso in cui la forma da inserire non possa essere contenuta completamente nella lavagna.
- *void cancella(Forma f)*: rimuove dalla lavagna la forma *f* precedentemente inserita.
- *String toString()*: restituisce una rappresentazione testuale del contenuto attuale della lavagna, come indicato dal seguente esempio[1]:

  ```
  [ Q(5.0, 6.0) R(8.0, 9.0)]
  ```

- *String log()*: restituisce una stringa che riporta tutte le operazioni eseguite sulla lavagna come indicato dal seguente esempio[2]:

  ```
  Tue Jul 21 09:56:58 CEST 2015 INSERIMENTO Q(5.0, 6.0)
  Tue Jul 21 09:56:59 CEST 2015 INSERIMENTO R(8.0, 9.0)
  Tue Jul 21 09:58:00 CEST 2015 CANCELLAZIONE Q(5.0, 6.0)
  Tue Jul 21 09:58:23 CEST 2015 INSERIMENTO Q(5.0, 6.0)
  Tue Jul 21 09:59:30 CEST 2015 INSERIMENTO R(8.0, 9.0)
  Tue Jul 21 09:59:50 CEST 2015 CANCELLAZIONE Q(5.0, 6.0)
  ```

Realizzare inoltre le classi *Quadrato* e *Rettangolo*, sottotipi di *Forma* (quest'ultima è già definita).

---

[1] Q indica quadrato, R indica rettangolo; tra parentesi vengono indicate le coordinate della forma. Nell'esempio in questione la lavagna contiene solamente un rettangolo e un quadrato.

[2] Nell'esempio le date sono in Inglese, ma va bene anche l'Italiano. Q indica quadrato, R indica rettangolo; tra parentesi vengono indicate le coordinate della forma inserita/eliminata. Per memorizzare l'istante di una modifica è possibile usare la classe *java.util.Date*.

Esercizio 2

Un sistema di *crowdfunding* (finanziamento collettivo) consente di sponsorizzare un progetto attraverso microfinanziamenti. Si consideri un numero di *Sponsor* non noto a priori, in comunicazione secondo una configurazione ad anello. Uno *Sponsor* (proponente) crea, sponsorizza con una quota iniziale, visualizza e invia un *Progetto* al successivo *Sponsor* (ricevente). Un progetto è composto da: *obiettivo*, *finanziamento corrente*, *finanziamento minimo*, *numero corrente di Sponsor*, *numero minimo di Sponsor*, e *porta di ascolto dello Sponsor proponente*. Un progetto è *attivato* se il finanziamento corrente è maggiore o uguale a quello minimo e se il numero di Sponsor è maggiore o uguale a quello minimo. Ogni *Sponsor* ricevente, riceve il progetto, quindi può partecipare finanziando un importo oppure un importo pari a zero, dopodichè lo visualizza e lo invia allo *Sponsor* successivo, e cosi via. Durante il primo giro, ogni quota di partecipazione positiva incrementerà il numero di Sponsor e il finanziamento corrente. Non appena il progetto è attivato viene inviato al proponente, il quale ne verifica l'attivazione, ne calcola la propria quota percentuale, e lo invia allo Sponsor successivo, e cosi via per il secondo giro. Il secondo giro serve a ciascuno Sponsor finanziatore per calcolare la propria quota percentuale e visualizzarla localmente, fino a fermarsi allo Sponsor che prima ha verificato l'attivazione.

Si realizzi un'applicazione Java distribuita composta dalle classi *Progetto* e *Sponsor*. La classe *Progetto* mantiene ed elabora i dati suddetti. La classe *Sponsor* svolge le operazioni di trasmissione di istanze di *Progetto* tra uno sponsor 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 sponsor. 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 alcuni 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 Sponsor svolgono in un caso di test. La logica di Sponsor e Progetto dovrà essere valida a prescindere dai numeri di porta e dal numero di sponsor.

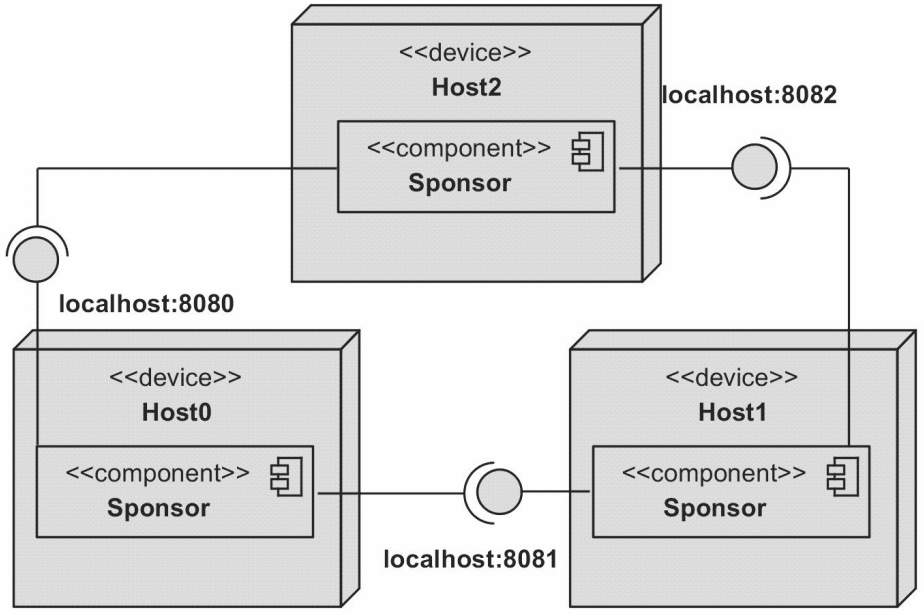
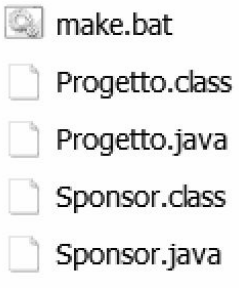


Fig.1 - Tre istanze di *Sponsor* e relative porte di ascolto



(viene fornito solo make.bat)

Fig.2 - c:\prg\esercizio2

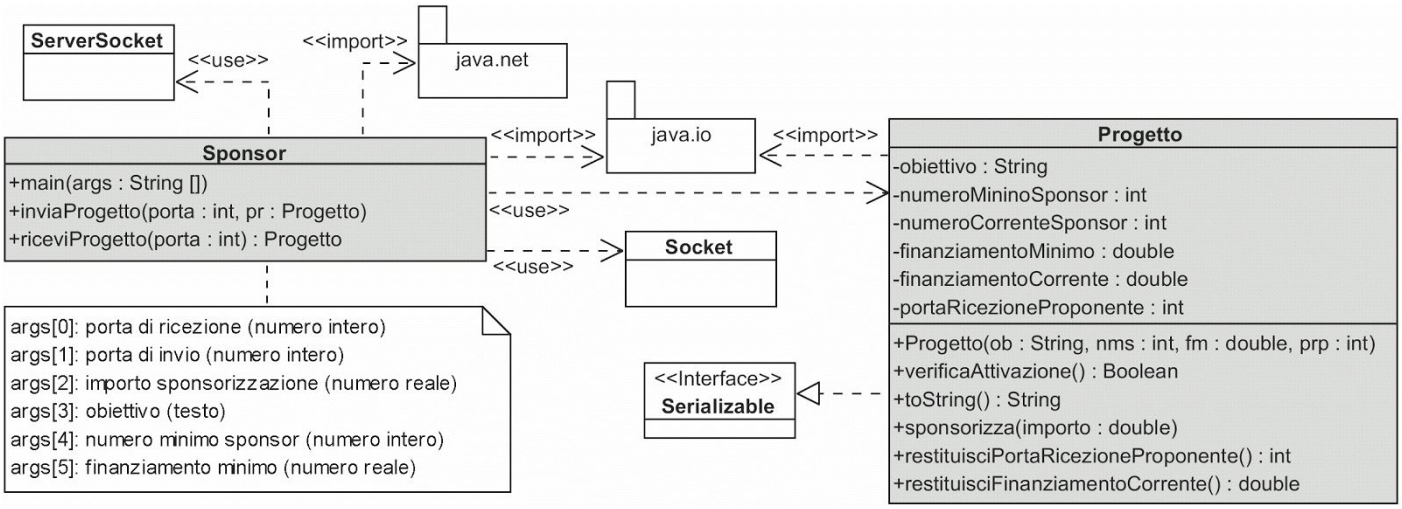


Fig.3 - Classi di cui deve essere composta l'applicazione (in grigio) e classi/package da usare (in bianco)

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

start cmd /k "color 2F && c:\prg\jdk8\bin\java Sponsor 8082 8080 3.6"
pause
start cmd /k "color 2F && c:\prg\jdk8\bin\java Sponsor 8081 8082 7.2"
pause
start cmd /k "color 2F && c:\prg\jdk8\bin\java Sponsor 8080 8081 5.1 "Tablet solidale" 2 10.0"
pause
taskkill /f /im "java.exe"
rem come risultato il progetto e' attivato senza il contributo di 8082
pause

start cmd /k "color 3F && c:\prg\jdk8\bin\java Sponsor 8082 8080 6.2"
pause
start cmd /k "color 3F && c:\prg\jdk8\bin\java Sponsor 8081 8082 0.0"
pause
start cmd /k "color 3F && c:\prg\jdk8\bin\java Sponsor 8080 8081 5.1 "Tablet solidale" 2 10.0"
pause
taskkill /f /im "java.exe"
rem come risultato il progetto e' attivato senza il contributo di 8081
pause

start cmd /k "color 4F && c:\prg\jdk8\bin\java Sponsor 8082 8080 3.6"
pause
start cmd /k "color 4F && c:\prg\jdk8\bin\java Sponsor 8081 8082 2.8"
pause
start cmd /k "color 4F && c:\prg\jdk8\bin\java Sponsor 8080 8081 5.1 "Tablet solidale" 3 5.0"
pause
taskkill /f /im "java.exe"
rem come risultato il progetto  e' attivato con il contributo di tutti
pause


start cmd /k "color 5F && c:\prg\jdk8\bin\java Sponsor 8082 8080 3.6"
pause
start cmd /k "color 5F && c:\prg\jdk8\bin\java Sponsor 8081 8082 2.8"
pause
start cmd /k "color 5F && c:\prg\jdk8\bin\java Sponsor 8080 8081 5.1 "Tablet solidale" 3 20.0"
pause
taskkill /f /im "java.exe"
rem come risultato il progetto non e' attivato
pause
```

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

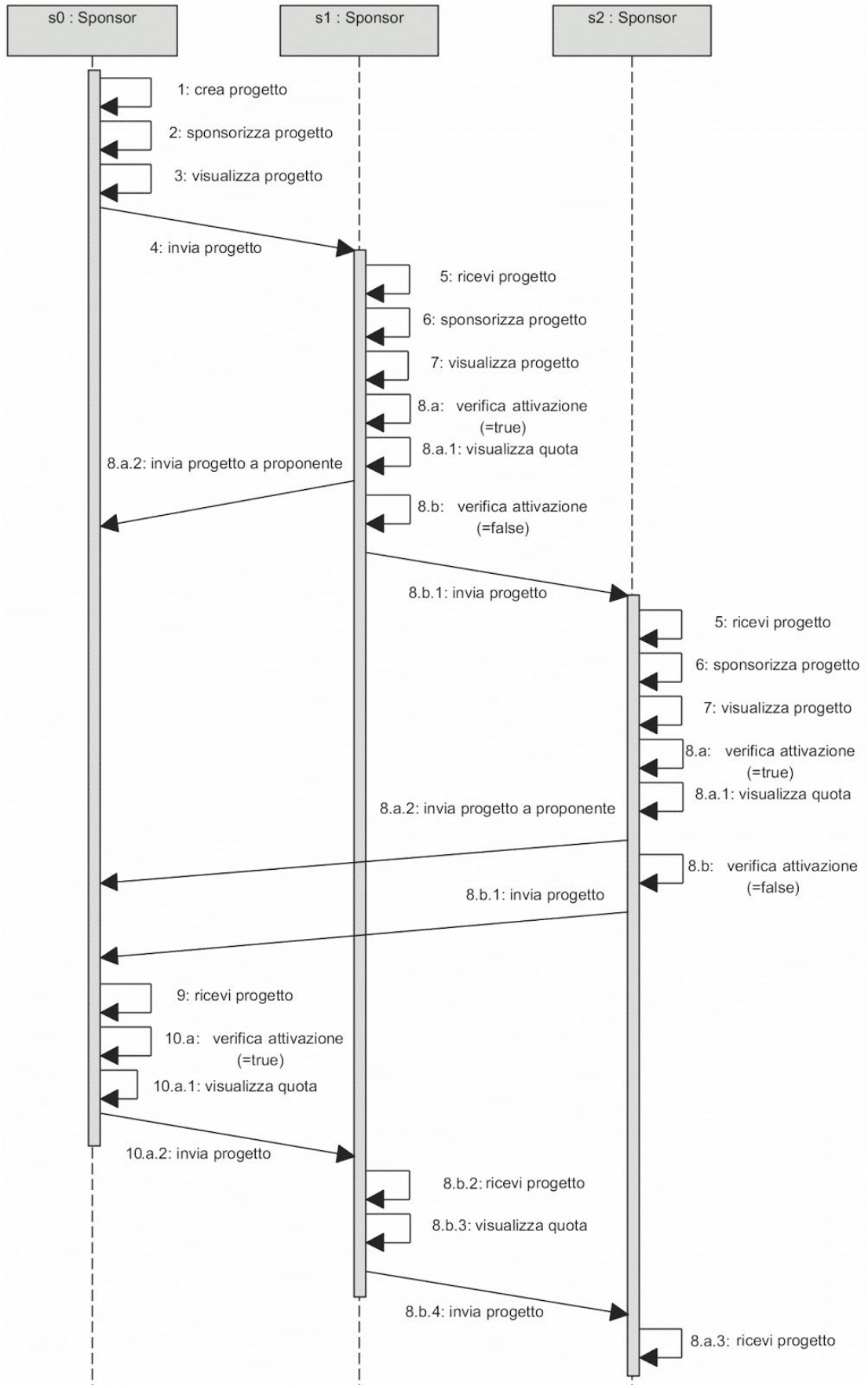


Fig.4 - Sequenza di passi che le istanze di *Sponsor* devono svolgere in un caso di test. Per brevità non viene rappresentata la classe *Progetto*.