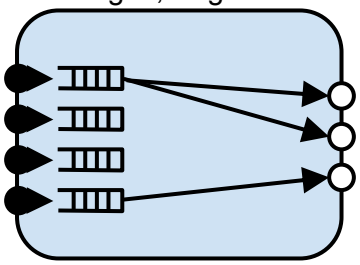
Programmazione 14/09/2015

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

Uno *Snodo* è caratterizzato da *i* ingressi e *u* uscite. Gli ingressi e le uscite sono numerati a partire da zero. Gli ingressi sono usati per inserire messaggi nello snodo, le uscite per estrarli. Ogni ingresso è dotato di una coda in cui vengono stoccati i messaggi fino a quando non vengono estratti. Ogni uscita può essere collegata a un solo ingresso, un ingresso può essere collegato a più uscite. Le uscite sono inizialmente tutte scollegate. La seguente figura rappresenta uno *Snodo* con 4 ingressi e 3 uscite (l'ingresso 0 è collegato alle uscite 0 e 1, gli ingressi 1 e 2 non sono collegati, l'ingresso 3 è collegato all'uscita 2).



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

- *Snodo(int i, int u)*: crea uno *Snodo* con *i* ingressi e *u* uscite. La coda associata a ogni ingresso ha una dimensione di default pari a 10 messaggi.
- *Snodo(int i, int u, int n)*: crea uno *Snodo* con *i* ingressi e *u* uscite. La coda associata a ogni ingresso ha dimensione pari a *n* messaggi.
- *void collega(int i, int u)*: collega l'uscita *u* all'ingresso *i*. Lancia *CollegamentoException*, di tipo unchecked, se l'uscita *u* o l'ingresso *i* non esistono. Definire anche la classe *CollegamentoException*.
- *void inserisci(Messaggio m, int i) throws InterruptedException*: inserisce il messaggio *m* nell'ingresso i. Il metodo è bloccante nel caso in cui la coda associata all'ingresso *i* sia piena.
- *Messaggio estrai(int u) throws InterruptedException*: estrae un messaggio usando l'uscita *u*. Il metodo è bloccante nel caso in cui non ci siano messaggi da estrarre. Lancia *CollegamentoException* nel caso in cui l'uscita *u* non sia collegata ad alcun ingresso.

Esercizio 2

Un sistema di *medical crowdsourcing* consente la diagnosi di malattie rare attraverso la consultazione efficiente di specialisti. Si consideri un numero di medici non noto a priori, in comunicazione attraverso una configurazione ad albero binario. Dati i *sintomi*, la ricerca di una *diagnosi* avviene in ordine anticipato: prima il medico posto sulla radice, poi quello posto nel ramo sinistro e poi quello posto nel ramo destro, e cosi via. A livello individuale, ogni medico riceverà i sintomi e potrà elaborare o meno una diagnosi, se è in grado. Non appena un medico produrrà una diagnosi, questa verrà lasciata inalterata e i giri di consultazione non andranno più verso i livelli sottostanti. Pertanto, dopo che sarà completato il giro del livello corrente, la diagnosi ritornerà al livello superiore, e via via fino alla radice. Una nuova richiesta è elaborata solo quando la richiesta corrente è ritornata alla radice.

Si realizzi un'applicazione Java distribuita, composta dalle classi *Consultazione* e *Medico*. La classe *Consultazione* mantiene ed elabora i *sintomi* e la *diagnosi*. La classe *Medico* svolge le operazioni di trasmissione di istanze di *Consultazione* tra medici, in accordo al protocollo di cui sopra. **Attenersi esattamente a quanto espresso dalle seguenti figure**. Fig.1 mostra una configurazione di test con cinque medici, con rispettive porte di ascolto. 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 medico posto sulla radice, e i medici in grado di diagnosticare, si distinguono per avere rispettivamente il termine "sintomi:" e "diagnosi:" nel terzo parametro in ingresso. Inoltre, se tra i parametri c'è la diagnosi allora il numero di porta del nodo di livello sottostante non serve, poichè il giro di consultazione non proseguirà verso tale nodo. Quindi un nodo si distingue per avere più di due parametri di cui il terzo non contiene il termini "sintomi: o "diagnosi:" (ma contiene un numero di porta). Fig.5 mostra la sequenza di passi (ad alto livello) che le istanze di Medico svolgono in un caso di test. La logica di Medico e Consultazione dovrà essere valida a prescindere dai numeri di porta e dal numero di medici.

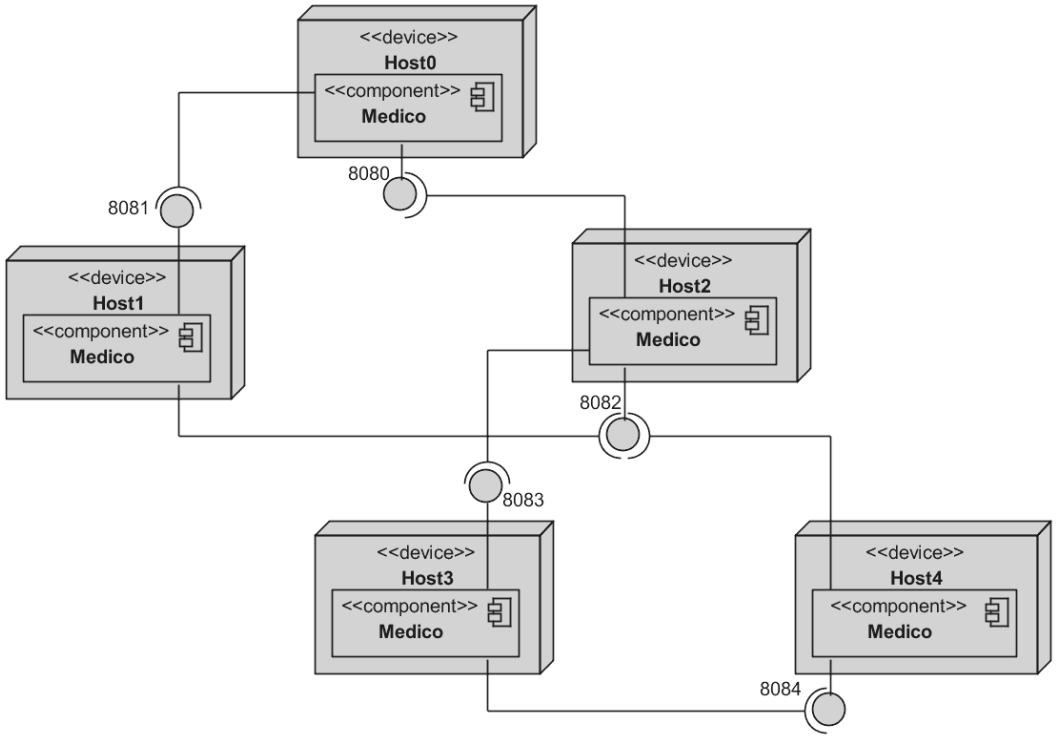
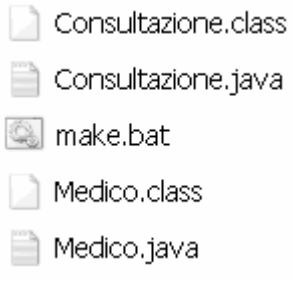


Fig.1 - Cinque istanze di *Medico* e relative porte di ascolto



(viene fornito solo make.bat)

Fig.2 - c:\prg\esercizio2

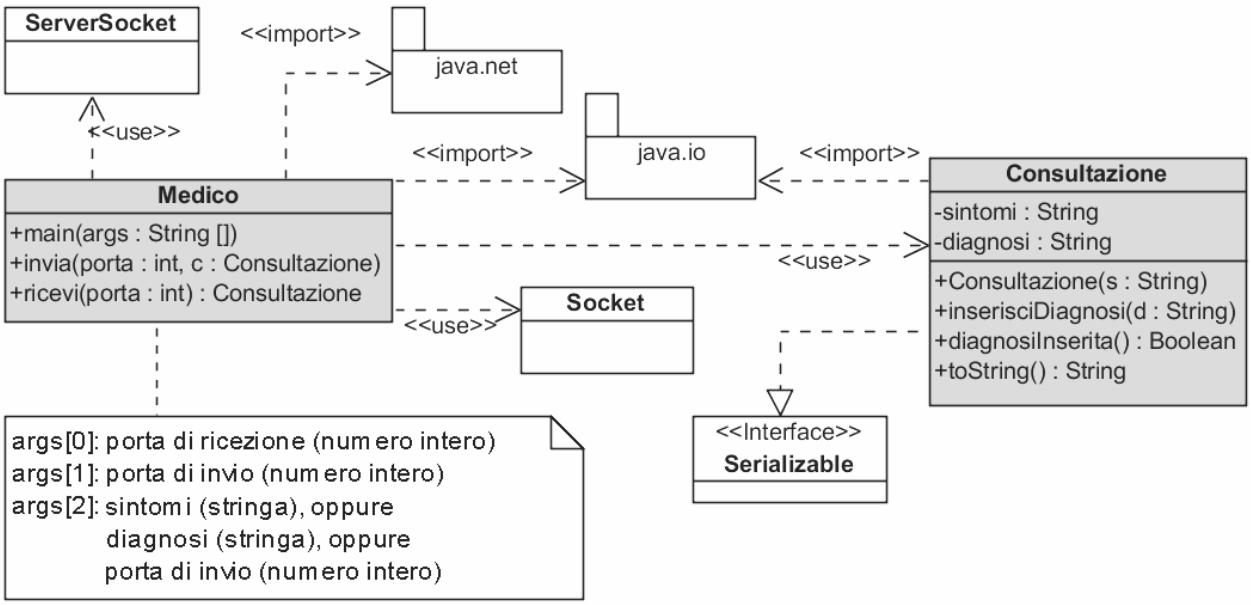


Fig.3 - Classi di cui deve essere composta l'applicazione (in grigio) e classi/package da usare (in bianco)

```
@echo off
c:\prg\jdk8\bin\javac *.java
pause
start cmd /k "color 5F && c:\prg\jdk8\bin\java Medico 8084 8082 "diagnosi: Draculismo""
pause
start cmd /k "color 5F && c:\prg\jdk8\bin\java Medico 8083 8084 "diagnosi: Draculismo""
pause
start cmd /k "color 5F && c:\prg\jdk8\bin\java Medico 8082 8080 8083"
pause
start cmd /k "color 5F && c:\prg\jdk8\bin\java Medico 8081 8082"
pause
start cmd /k "color 5F && c:\prg\jdk8\bin\java Medico 8080 8081 "sintomi: Canini lunghi""
pause
taskkill /f /im "java.exe"
rem come risultato la diagnosi è fatta da Host3
pause

start cmd /k "color 2F && c:\prg\jdk8\bin\java Medico 8084 8082"
pause
start cmd /k "color 2F && c:\prg\jdk8\bin\java Medico 8083 8084 "diagnosi: Draculismo""
pause
start cmd /k "color 2F && c:\prg\jdk8\bin\java Medico 8082 8080 "diagnosi: Draculismo""
pause
start cmd /k "color 2F && c:\prg\jdk8\bin\java Medico 8081 8082"
pause
start cmd /k "color 2F && c:\prg\jdk8\bin\java Medico 8080 8081 "sintomi: Canini lunghi""
pause
taskkill /f /im "java.exe"
rem come risultato la diagnosi è fatta da Host2
pause

start cmd /k "color 4F && c:\prg\jdk8\bin\java Medico 8084 8082"
pause
start cmd /k "color 4F && c:\prg\jdk8\bin\java Medico 8083 8084 "
pause
start cmd /k "color 4F && c:\prg\jdk8\bin\java Medico 8082 8080 "diagnosi: Draculismo""
pause
start cmd /k "color 4F && c:\prg\jdk8\bin\java Medico 8081 8082 "diagnosi: Draculismo""
pause
start cmd /k "color 4F && c:\prg\jdk8\bin\java Medico 8080 8081 "sintomi: Canini lunghi""
pause
taskkill /f /im "java.exe"
rem come risultato la diagnosi è fatta da Host1
pause
```

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

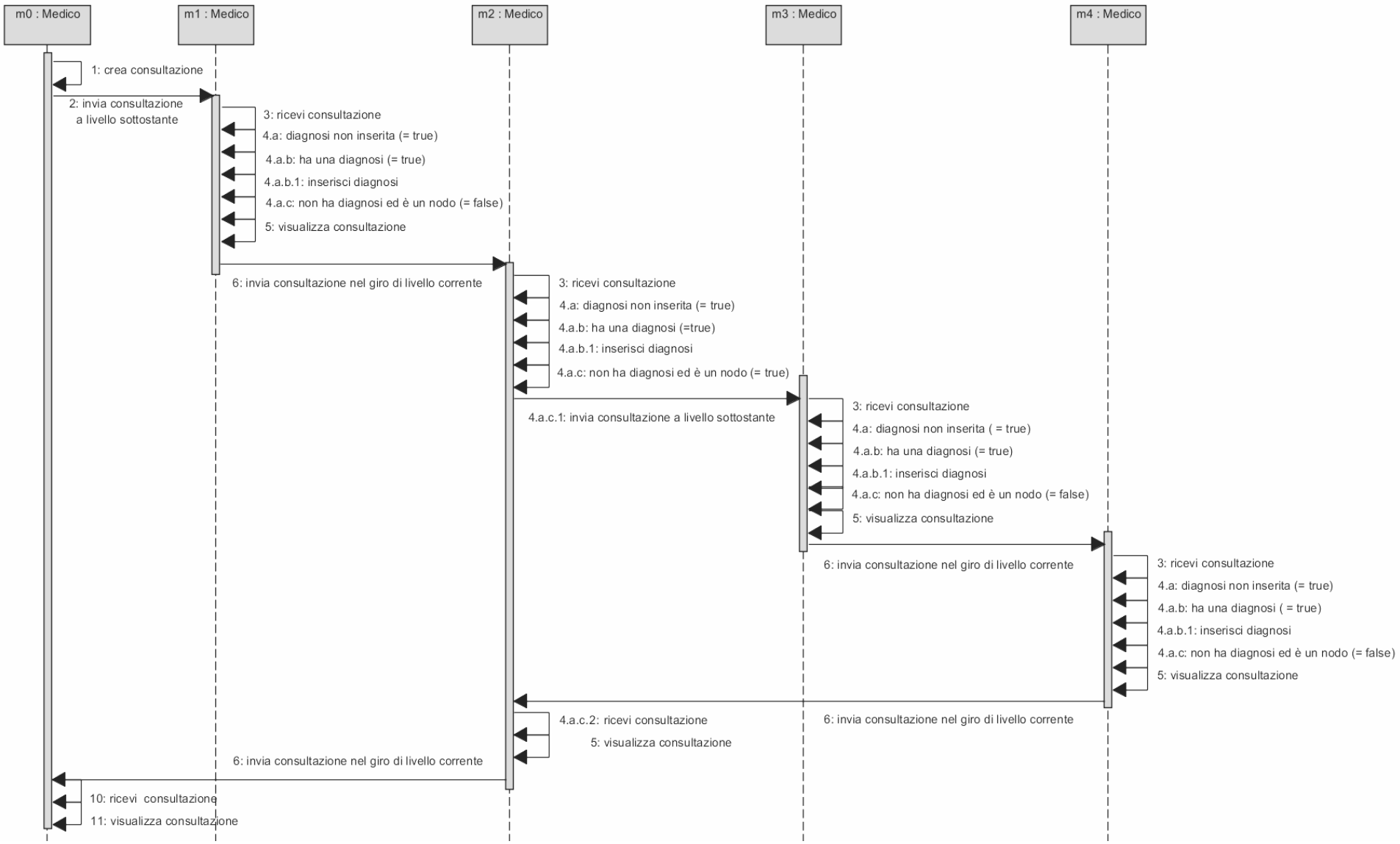


Fig.5 - Sequenza di passi che le istanze di *Medico* devono svolgere nel un caso di test. Per brevità non viene rappresentata la classe *Consultazione*.