Programmazione 28/01/2015

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 ritirare il proprio elaborato.

Esercizio 1

Lo scenario è quello di un sistema produttori-consumatori in cui lo scambio di informazioni avviene mediante un buffer condiviso. I produttori inseriscono messaggi nel buffer. I messaggi possono essere prioritari o normali. I consumatori estraggono messaggi dal buffer. I messaggi vengono conservati nel buffer secondo una disciplina FIFO, indipendentemente dal fatto che siano prioritari o meno.
Realizzare una classe *Messaggio* dotata dei seguenti costruttori e metodi (il contenuto informativo del messaggio è per gli scopi dell'esercizio trascurabile):

- *Messaggio(boolean prio)* crea un messaggio; il valore di *prio* indica se il messaggio è proritario o meno.
- *boolean isPrioritario()* restituisce *true* se il messaggio è prioritario, *false* se normale.

Realizzare una classe *Buffer* che opera secondo le seguenti specifiche:

- *Buffer(int n)* crea un buffer di dimensione *n* (può contenere al più *n* messaggi).
- *Buffer()* crea un buffer con una dimensione predefinita pari a 10.
- *void inserisci(Messaggio m)* inserisce in coda al buffer il messaggio *m*; il metodo è bloccante nel caso in cui il buffer sia pieno.
- *Messaggio[] estrai(int q)* estrae dal buffer *q* messaggi (partendo dalla testa) e li restituisce al chiamante. Il metodo può estrarre e restituire meno di *q* messaggi solo se il buffer ne contiene meno di *q* e almeno uno è prioritario. Viceversa, il metodo è bloccante nel caso in cui nel buffer ci siano meno di *q* messaggi e nessuno di quelli presenti sia prioritario. Il metodo lancia *IllegalArgumentException* se *q* è maggiore della dimensione del buffer.

Esercizio 2

Un'azienda ha bisogno di eseguire il backup remoto di dati di valore, con un protocollo applicativo che ne garantisca l'integrità. L'applicazione di backup (*BancaDati*) quando viene accesa attende i dati, e dopo averli ricevuti li memorizza su file binario; quindi li legge dal file e invia la copia letta prima di spegnersi. L'applicazione che ha i dati originali (*SportelloDati*) li invia a una applicazione di backup e poi attende di ricevere la copia di backup; dopo averla ricevuta, prima di spegnersi confronta i dati originali con la copia di backup e stampa il risultato del confronto (vero/falso). Si realizzi in linguaggio Java un'applicazione distribuita composta dalle classi *Dati*, *BancaDati*, e *SportelloDati*. Fig. 1 presenta la sequenza di passi (ad alto livello) che le classi devono svolgere. Fig. 2 mostra i file dell'applicazione. Fig.3 mostra il contenuto del file di testing *make.bat*, la cui esecuzione produce su console la stampa di *true* o *false* a seconda dell'esito del confronto. Come traccia viene fornito solo il file make.bat (da non modificare). Fig. 4 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 anche la chiusura di flussi e connessioni aperti. Catturare solo le eccezioni obbligatorie e gestirle semplicemente stampando le relative informazioni.

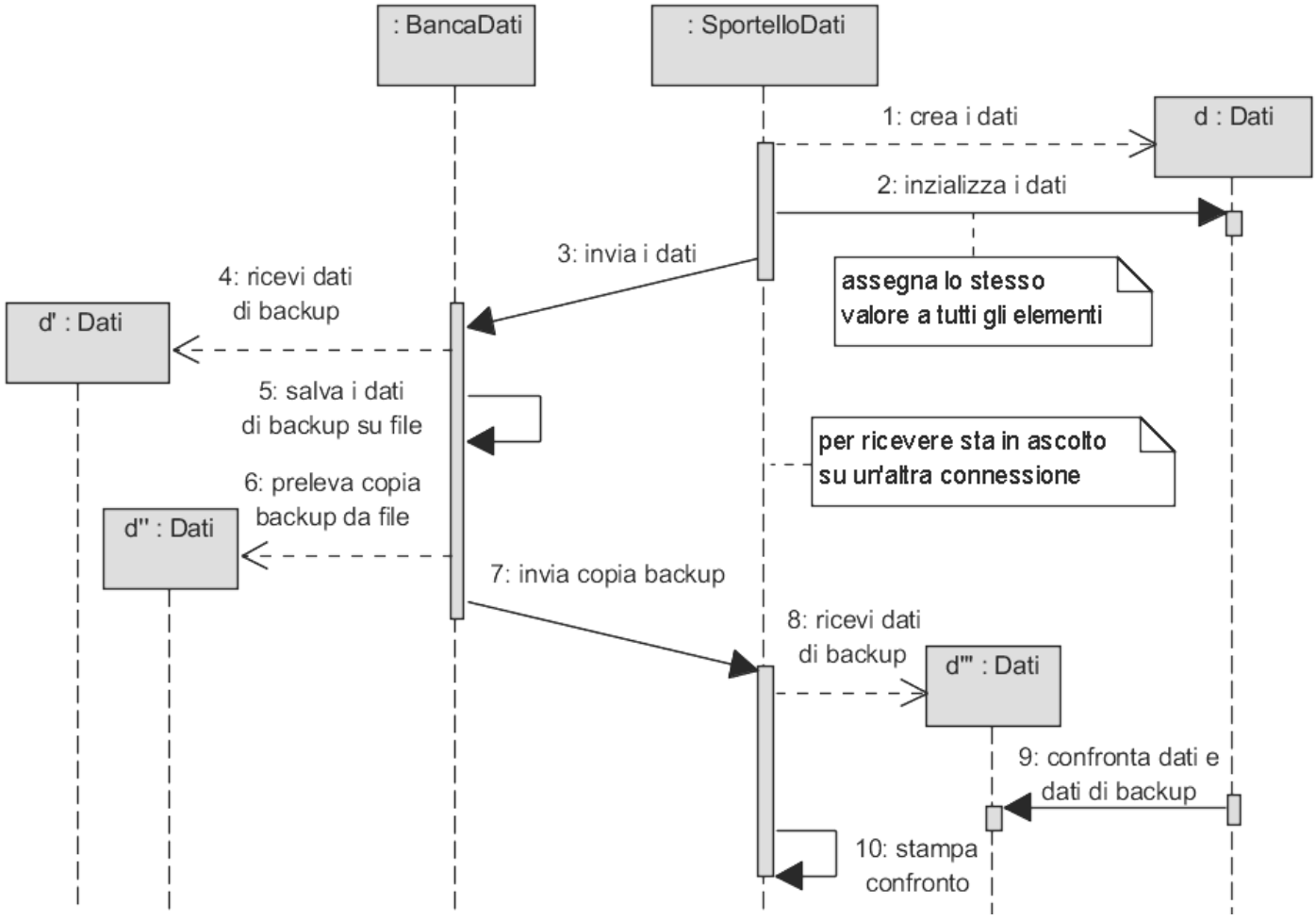

Fig.1 - Sequenza di passi che *SportelloDati* e *BancaDati* devono svolgere.

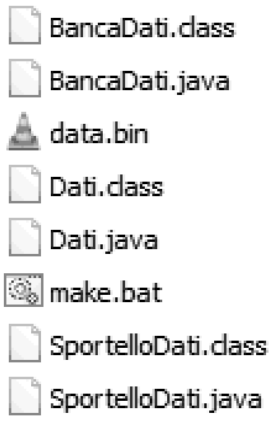

Fig.2 - c:\prg\esercizio2

```
c:\prg\jdk8\bin\javac *.java
pause
start cmd /k "c:\prg\jdk8\bin\java BancaDati
pause
c:\prg\jdk8\bin\java SportelloDati 50 3.14
pause
```

Fig.3 - File make.bat

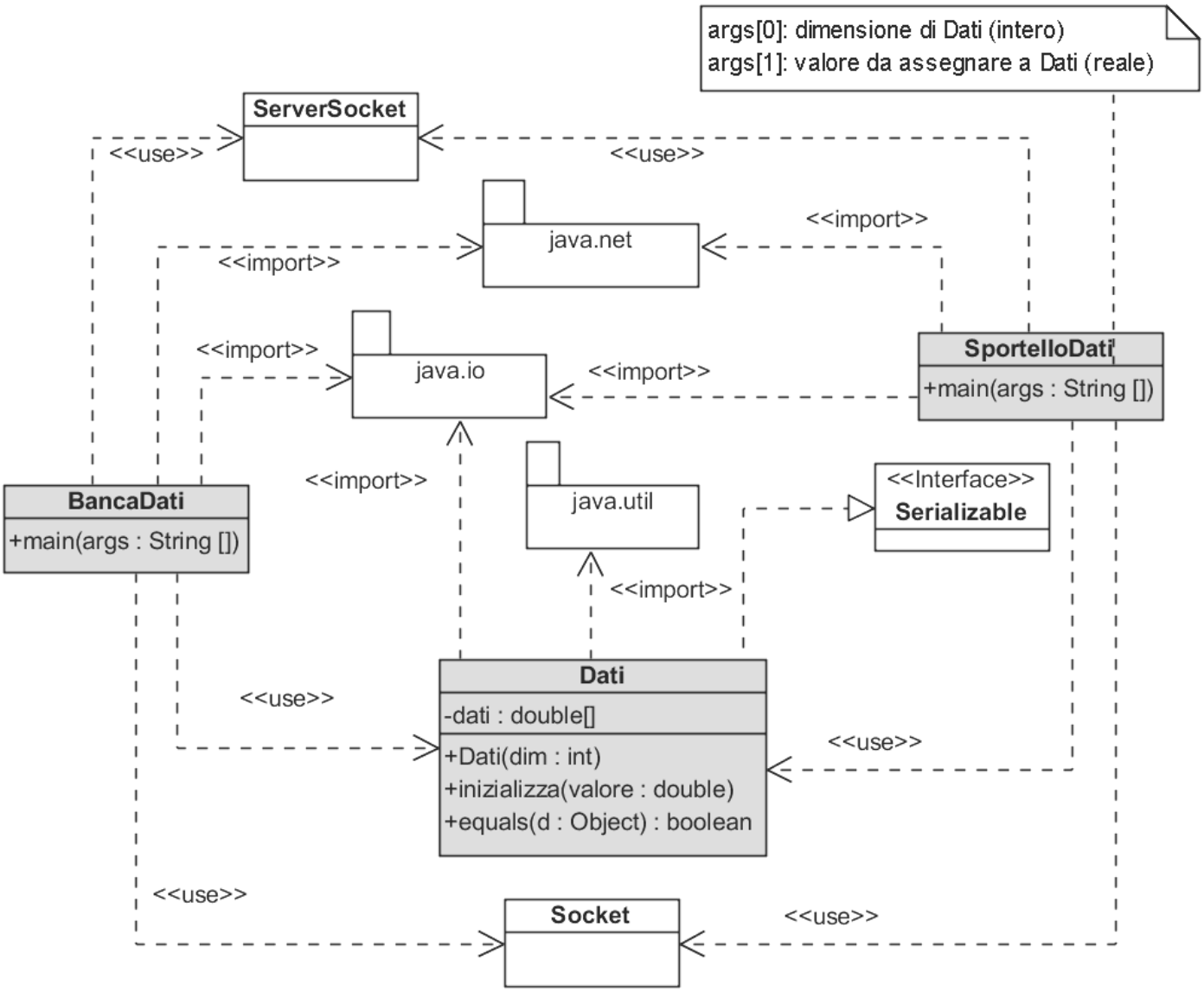

Fig.4 - Classi di cui deve essere composta l'applicazione (in grigio) e classi da usare (in bianco)