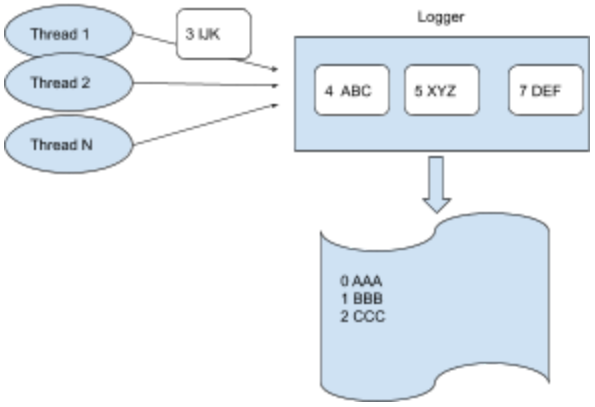
Programmazione Avanzata 25/02/2020

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 il caricamento 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

Un insieme di thread produce dei messaggi di log. Ogni messaggio è caratterizzato da un numero d'ordine intero diverso da tutti gli altri. Un *Logger* viene usato dai thread per salvare i messaggi in un file secondo l'ordine specificato (a partire da 0). Il Logger è in grado di mantenere al suo interno un certo numero di messaggi che non possono essere trasferiti sul file in quanto fuori ordine (il messaggio con numero d'ordine X può essere trasferito su file solo quando quest'ultimo contiene, come ultimo elemento, il messaggio con numero d'ordine X-1).



La figura mostra la situazione in cui i messaggi con numero d'ordine 0, 1 e 2 sono già stati trasferiti su file e nel Logger sono contenuti i messaggi con numeri d'ordine 4, 5 e 7. Tali messaggi non possono essere trasferiti su file in quanto manca il messaggio con numero d'ordine 3. Quando il messaggio con tale numero d'ordine viene loggato dal Thread 1 i messaggi con numeri d'ordine 3, 4 e 5 vengono trasferiti su file, mentre quello con numero d'ordine 7 rimane nel Logger perché manca quello con numero d'ordine 6.

Realizzare la classe Logger secondo le seguenti specifiche:

- *Logger(String fn, int n)*: crea un Logger che salva i messaggi nel file di nome *fn*; il Logger è in grado di bufferizzare al più *n* messaggi.
- *void log(String r, int o)*: viene chiamato da un thread per salvare il messaggio *r*. L'argomento *o* indica il numero d'ordine del messaggio in questione. Il metodo può essere bloccante nel caso in cui non sia possibile trasferire immediatamente su file il messaggio *r* (perché fuori ordine) e allo stesso tempo non sia possibile mantenerlo nel Logger (perché già pieno). Viceversa il metodo non è bloccante quando è possibile trasferire immediatamente il messaggio nel file o se è possibile bufferizzarlo nel Logger.

Il file che contiene i messaggi ha il formato illustrato dal seguente esempio:

```
0     Operazione fallita
1     Provo a connettermi con il server
2     L'utente ha selezionato il menu File
3     Connessione con il server attiva
...    …
```

Realizzare anche una sottoclasse di Thread (*Produttore*) che inserisce messaggi casuali con numeri d'ordine compresi tra *start* e *end* specificati attraverso il suo costruttore.

Esercizio 2

Un *ArchivioDistribuito* è un documento XML, contenente una sequenza di valori reali, segmentato in vari file XML ciascuno su un diverso host/processo Java. I processi sono comunicanti secondo uno schema ad anello (es. Fig.1), e il loro numero non è noto a priori. Per gestire in modo coordinato i segmenti, si adopera una istanza di *ArchivioDistribuito* per ogni segmento. Ogni oggetto *ArchivioDistribuito* può accedere esclusivamente al proprio file XML. La Fig.2 mostra (in alto) tre esempi di file XML generati da tre distinti processi Java, posti per semplicità sul medesimo host e su porte di ascolto diverse: *8080_archivio.xml*, *8081_archivio.xml*, *8082_archivio.xml*. La Fig.2 mostra (in basso) il contenuto di un file XML. Le operazioni da svolgere riguardano sempre l'archivio nel suo insieme, e possono essere richieste da qualsiasi oggetto *ArchivioDistribuito* (richiedente).

Per coordinarsi, gli oggetti *ArchivioDistribuito* si scambiano un oggetto *Operazione* (dal contenuto immutabile durante lo svolgimento di una operazione). Gli accessi al file XML avvengono nel seguente modo: si trasforma il contenuto del file XML in un oggetto temporaneo *ArrayList* e si svolgono le operazioni su tale oggetto; quindi si ritrasforma l'oggetto in XML e si sovrascrive il file. Poichè lo stato dell'archivio viene mantenuto interamente su file, gli attributi e le operazioni della classe *ArchivioDistribuito* sono statici. Le operazioni possibili sono le seguenti:

- **crea**: crea un file XML contenente un segmento senza dati in tutti i processi. In particolare: il richiedente genera un *ArrayList* vuoto, lo archivia su file XML, lo visualizza, inoltra l'operazione al processo successivo, che fa le medesime azioni, e cosi via fino al processo richiedente, che si limiterà a ricevere l'operazione;
- **inserisci**: inserisce un dato valore nel primo file XML di lunghezza minima trovato. In particolare: il richiedente controlla se il proprio segmento è vuoto, nel qual caso inserisce il valore; altrimenti inoltra l'operazione al processo successivo, che fa le medesime azioni, e cosi via. Se nessuno dei processi ha un segmento vuoto, l'operazione ritorna al richiedente. A quel punto si esegue il medesimo protocollo, controllando questa volta se il proprio segmento ha un solo elemento. Se nessuno dei processi ha un segmento di un solo elemento, l'operazione ritorna al richiedente. A quel punto si esegue il medesimo protocollo, controllando questa volta se il proprio segmento ha due soli elementi. Procedendo in questo modo si troverà prima o poi un segmento di dimensione minima dove inserire il nuovo valore. A quel punto l'operazione verrà eseguita, il segmento locale sarà visualizzato, e l'operazione non sarà più propagata;
- **estrai**: estrae il valore dal primo segmento che lo contiene, se esiste. In particolare: il richiedente controlla se il proprio segmento contiene il valore, nel qual caso lo estrae e stampa il contenuto del segmento; altrimenti inoltra l'operazione al processo successivo, che fa le medesime azioni, e cosi via. Se nessuno dei segmenti ha il valore, l'operazione ritorna al richiedente, che si limiterà a ricevere l'operazione.

Si realizzi un'applicazione Java distribuita composta dalle classi *ArchivioDistribuito* e *Operazione*. Le visualizzazioni, gli invii e le ricezioni dell'operazione siano in formato XML. **Attenersi esattamente a quanto espresso dalle seguenti figure**. Fig.1 mostra una configurazione di test con soli tre host. 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. La logica delle due classi dovrà essere valida a prescindere dai numeri di porta e di processi, e dal processo richiedente.

```
@echo off
c:\prg\jdk8\bin\javac -classpath c:\prg\libs\xstream-1.4.7.jar *.java
pause

set L=c:\prg\libs\
set LIBS=%L%xstream-1.4.7.jar;%L%xmlpull-1.1.3.1.jar;%L%xpp3_min-1.1.4c.jar;

start cmd /k "@echo off && color 2F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8082 8080"
start cmd /k "@echo off && color 2F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8081 8082"
start cmd /k "@echo off && color 2F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8080 8081 crea 0"
pause
taskkill /f /im "java.exe"
rem risultato: tutti gli host creano un file senza dati
type 8080_archivio.xml
type 8081_archivio.xml
type 8082_archivio.xml
pause

start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8082 8080"
start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8081 8082 inserisci 3.14"
start cmd /k "color 3F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8080 8081"
pause
taskkill /f /im "java.exe"
rem risultato: lo host 8081 inserisce il dato 3.14
type 8080_archivio.xml
type 8081_archivio.xml
type 8082_archivio.xml
pause

start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8082 8080"
start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8081 8082 inserisci 1.72"
start cmd /k "color 4F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8080 8081"
pause
taskkill /f /im "java.exe"
rem risultato: lo host 8082 inserisce il dato 1.72
type 8080_archivio.xml
type 8081_archivio.xml
type 8082_archivio.xml
pause

start cmd /k "color 5F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8082 8080"
start cmd /k "color 5F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8081 8082 inserisci 5.92"
start cmd /k "color 5F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8080 8081"
pause
taskkill /f /im "java.exe"
rem risultato: lo host 8080 inserisce il dato 5.92
type 8080_archivio.xml
type 8081_archivio.xml
type 8082_archivio.xml
```

```
pause

start cmd /k "color 6F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8082 8080"
start cmd /k "color 6F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8081 8082 inserisci 2.16"
start cmd /k "color 6F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8080 8081"
pause
taskkill /f /im "java.exe"
rem risultato: lo host 8081 inserisce il dato 2.16
type 8080_archivio.xml
type 8081_archivio.xml
type 8082_archivio.xml
pause

start cmd /k "color 7F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8082 8080"
start cmd /k "color 7F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8081 8082 estrai 5.92"
start cmd /k "color 7F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8080 8081"
pause
taskkill /f /im "java.exe"
rem risultato: lo host 8080 rimuove il dato 5.92
type 8080_archivio.xml
type 8081_archivio.xml
type 8082_archivio.xml
pause

start cmd /k "color 8F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8082 8080"
start cmd /k "color 8F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8081 8082 estrai 5.9"
start cmd /k "color 8F && c:\prg\jdk8\bin\java -classpath %LIBS% ArchivioDistribuito 8080 8081"
pause
taskkill /f /im "java.exe"
rem risultato: nessuno estrae
type 8080_archivio.xml
type 8081_archivio.xml
type 8082_archivio.xml
pause
```

Fig.4 - File make.bat, l'unico ad essere fornito, da non modificare.

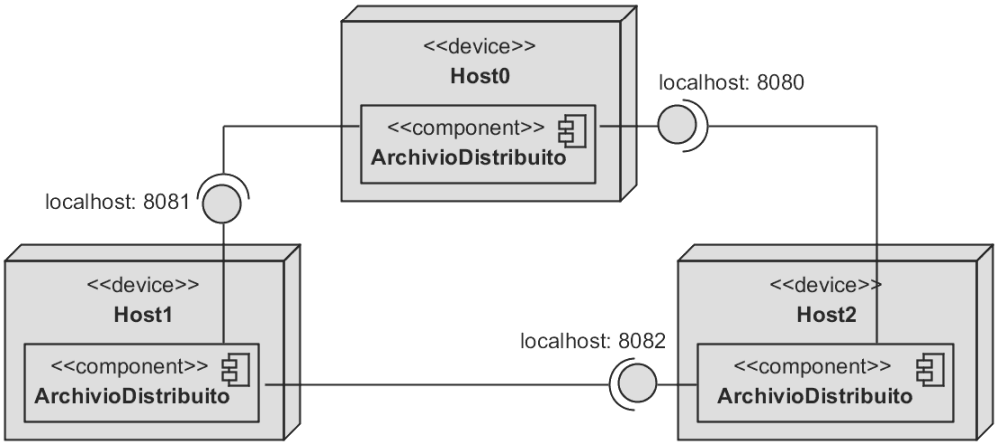


Fig.1 - Tre istanze di *ArchivioDistribuito* e relative porte di ascolto

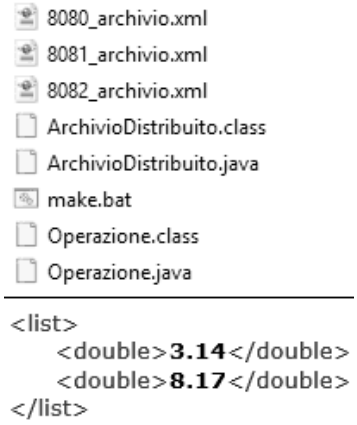


Fig.2 - c:\prg\esercizio2

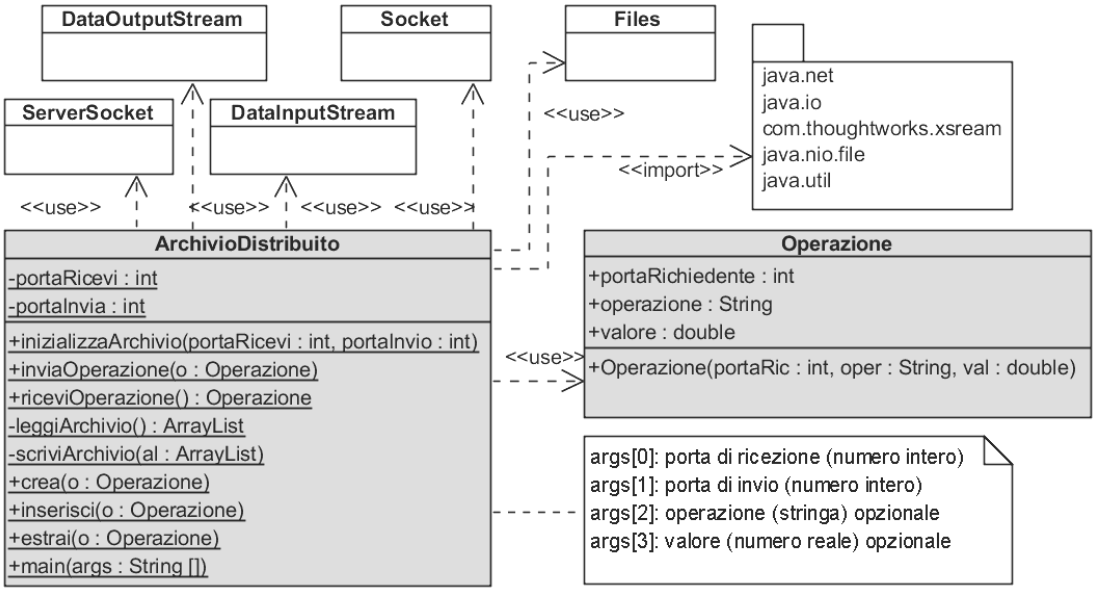


Fig.3 - Classi di cui deve essere composta l'applicazione (in grigio) e classi/package da usare (in bianco)