Università degli Studi di Napoli Federico II
Esame di Advanced Computer Programming

Esempio di prova pratica su Java RMI e Socket
Durata della prova: 120 minuti

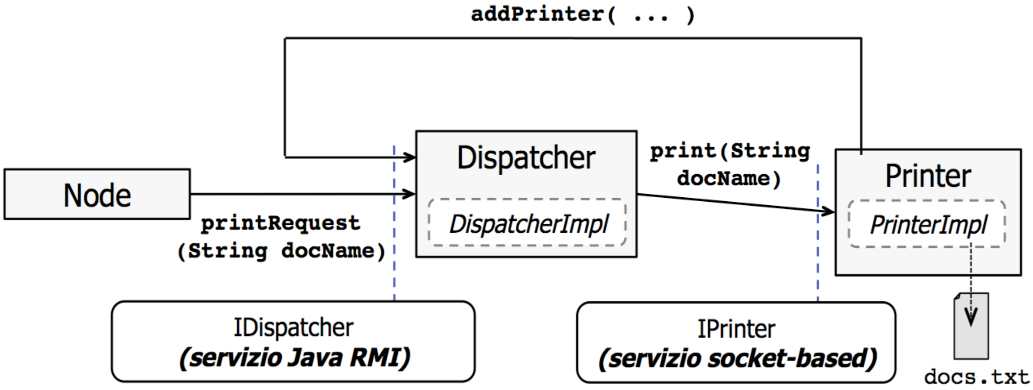
Cognome…………………….………. Nome………….………………. Matr……………

°°

Lo studente legga attentamente il testo e produca il programma ed i casi di test necessari per dimostrarne il funzionamento. La mancata compilazione completa dell'elaborato, la compilazione con errori o l'esecuzione errata daranno luogo alla valutazione come ***prova non superata****.*
Ricordarsi di indicare Cognome, Nome e matricola su questo stesso foglio, che dovrà essere in ogni caso consegnato alla Commissione. Al termine della prova lo studente dovrà far verificare il funzionamento del programma ad un membro della Commissione.

°°

Testo della prova



Il candidato realizzi un sistema per la gestione di stampanti remote (denominate *Printer* nel seguito). Il sistema è composto da 3 tipologie di entità, come illustrato in figura:

1. **Dispatcher**: offre i servizi **`addPrinter`** e **`printRequest`**. Il primo, **`addPrinter`**, consente l'<u>aggiunta</u> di una Printer al Dispatcher (il candidato scelga gli argomenti di *addPrinter* in maniera tale da supportare un meccanismo di <u>callback distribuita</u> basato su socket). Il Dispatcher mantiene un elenco di Printer registrate.
   Il metodo **`printRequest`** consente la richiesta di stampa del documento **`docName`**. Per ogni invocazione di **`printRequest`**, il Dispatcher invoca ad una ad una le Printer disponibili nell'elenco, finché ne viene individua una che non restituisca **`false`**. Se tutte le Printer restituiscono **`false`**, **`printRequest`** restituirà **`false`** a Node.

2. **Printer** offre il metodo **`print`**`(String docName)`**.** Ciascuna Printer può eseguire una sola **`print`** in maniera concorrente. Se Printer sta già eseguendo una richiesta, l'invocazione di **`print`** termina immediatamente con esito **`false`.**
   In caso contrario, **`print`** i) stampa a video e salva su file `docName` e ii) restituisce esito **`true`.**
   Una invocazione di **`print`** dura un tempo scelto a caso tra 5 e 10 secondi.

3. **Node:** emula un client che richiede la stampa di documenti. Node avvia 5 thread. Ciascun thread invoca 3 **`printRequest`** -con **`docName`** generato in maniera casuale- ogni 3 secondi (la stringa **`docName`** è generata concatenando il prefisso "doc" ed un intero casuale tra 1 e 50).

Il funzionamento del programma sarà verificato con <u>un numero arbitrario</u> di Printer lanciate all'avvio o durante l'esecuzione del programma. Una Printer è avviata come nel seguente esempio (<u>porta</u> e <u>nome del file</u> su cui saranno salvati i nomi dei documenti stampati, sono specificati da prompt):

```
java nomePackage.PrinterServer 8000  documentiPrinterUno.txt
```

Il candidato implementi il sistema utilizzando <u>Java RMI per il Dispatcher</u> e <u>socket</u> (*pattern proxy/skeleton ed implementazione per delega*) <u>per l'interazione tra Dispatcher e Printer.</u>