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

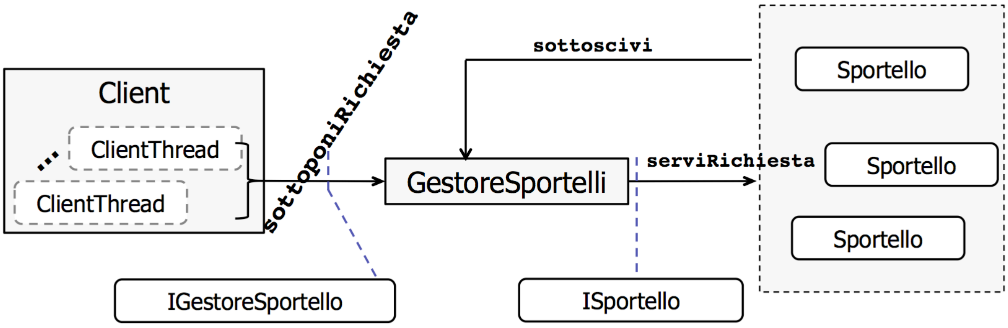
Esempio di prova pratica su Java RMI e pattern architetturali.

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 gestore di sportelli bancari. Il sistema implementa una politica di bilanciamento del carico tra gli sportelli, ed è composto da 3 tipologie di entità:

1. **Sportello**, che offre il servizio **`boolean serviRichiesta( int idCliente )`**. Ogni richiesta di servizio dura un tempo scelto a caso tra 1 e 5 secondi. Al termine della richiesta **`idCliente`** è salvato su un file. Lo Sportello, al proprio avvio, si sottoscrive al Gestore (invocazione di **`sottoscrivi`**).

Uno Sportello può servire in maniera concorrente 3 richieste. Ogni ulteriore richiesta è messa in attesa; tuttavia, al più 2 richieste possono essere accodate nell'attesa che lo Sportello abbia servito una richiesta precedente. Nel caso in cui la richiesta non può essere accodata, lo Sportello restituisce immediatamente esito **`false.`** Le richieste servite correttamente terminano con esito **`true.`**

2. **GestoreSportello:** prende in carico le richieste di servizio dei Client tramite **`boolean sottoponiRichiesta( int idCliente )`** e le smista verso un apposito Sportello. Gestore Sportello possiede una lista di Sportelli ordinata in base all'ordine delle sottoscizioni, ed invoca in sequenza ciascuno Sportello. La richiesta è servita dal primo sportello libero: al termine della richiesta il Gestore restituisce al Client esito **true.** Il Gestore restituisce esito **false** se ogni Sportello restituisce **false**.

3. **Client**: genera T thread, ognuno dei quali, allo scadere di un tempo di t secondi (con t scelto a caso tra 1 e 3), effettua una richiesta di prenotazione. Ogni client thread genera R richieste. Si imposti T pari a 10; R pari a 10. L' **`idClient`** è generato in maniera casuale tra 1 e 100.

Il candidato implementi il sistema utilizzando JAVA RMI. A tal fine, il candidato produca le opportune definizioni delle interfacce di Sportello e GestoreSportelli.