Esercitazione di laboratorio

RMI – Progettazione di un sistema di callback

Un server fornisce ai propri client un servizio di *notifica* per mezzo del quale un client può sincronizzarsi con il verificarsi sul server di un evento di un certo tipo. Il servizio di notifica implementa l' interfaccia remota

```
public interface Notification extends Remote {
  void register(EventType e, Callback c) throws RemoteException;
}
```

con

```
public interface Callback extends Remote {
  Event notifyEvent(Event e) throws RemoteException;
}
```

La classe **`EventType`** descrive un *tipo* di evento, mentre la classe **`Event`** descrive un *evento* specificandone il tipo ed un messaggio eventualmente associato (**`String`**).

PROGETTARE E REALIZZARE

1. La classe **`NotificationImpl`**, implementazione *thread-safe* dell' interfaccia **`Notification`**.
2. La classe **`Server`**, che implementa il programma server in accordo alle seguenti specifiche:
   a. *crea* ed *esporta* il servizio di notifica sulla porta **`PORT`** (inizialmente la porta **`2000`**) e
   b. lo registra presso l'RMI Registry con il nome **`NomeServizio`** (inizialmente **`"MyRegistrationService"`**);
   c. supporta i tipi {0, 1};
   d. genera in modo casuale eventi di tali tipi e li notifica ai client interessati.

3. La classe **`CallBackImpl`**, implementazione thread-safe dell'interfaccia **`Callback`**, prevedendo anche un metodo **`public Event waitForEvent()`** per mezzo del quale un client aspetta il verificarsi di un evento. L'operazione ritorna l'evento che si è verificato.
4. La classe Client che implementa il programma client in accordo alle seguenti specifiche:
    a. Crea ed esporta il servizio di callback;
    b. si sincronizza con un evento di uno dei tipi supportati;
    c. scrive su standard output l'evento che si è verificato.