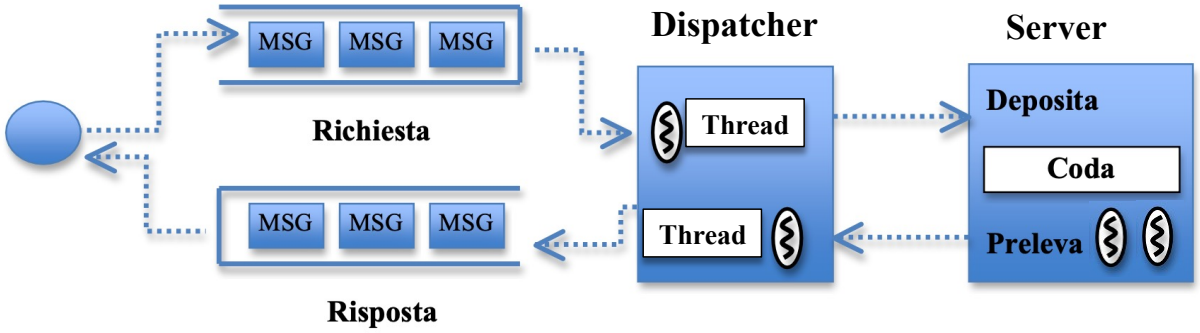
Università degli Studi di Napoli Federico II
Advanced Computer Programming

Esercitazione Interoperabilità Python/Java

Si realizzi un sistema per la gestione di un magazzino basato su **code di messaggi**. Il sistema implementa un deposito di articoli e si compone di 3 entità mostrate in figura:



STOMP **Proxy-Skeleton**

- **Client (Python).** Il client invia N messaggi sulla coda ***Richiesta.*** Ogni messaggio contiene 2 informazioni: (i) *tipo di richiesta* (**deposita** o **preleva**) e (ii) **id_articolo** (rappresentato da un intero, usato per richieste di tipo deposita). Una volta inviati i messaggi di richiesta, il Client si metterà in ricezione asincrona su una coda ***Risposta*** delle risposte alle richieste di deposito (il messaggio di risposta contiene la stringa "deposited") e di prelievi (il messaggio di risposta contiene il valore prelevato). I messaggi di richiesta devono contenere il riferimento alla coda di risposta da far utilizzare al Dispatcher (*hint: utilizzare l'header reply-to*)
- **Dispatcher (Java).** Questa entità funge da intermediario tra le richieste inviate dal client al server. Il *dispatcher* si occupa di prelevare le richieste del client dalla coda *Richiesta* (gestita con STOMP), e le inoltra al **Server.** Il *dispatcher* è un'applicazione multithread che riceve in maniera asincrona le richieste da parte di **Client** sulla coda *Richiesta*, ne estrae le informazioni (*tipo richiesta, valore*), ed invoca (attraverso un nuovo thread) il corrispondente metodo *preleva* o *deposita* fornito da *Server*.
- **Server (Python).** Il server è un'applicazione multiprocesso Python che implementa i metodi di *preleva* e *deposita* invocati da *Dispatcher* (ricevuti da Client) per prelevare e depositare articoli in una coda. La dimensione della coda è pari a 5. L'accesso alla coda è disciplinato attraverso il problema produttore/consumatore.

Si crei 1 Client che genera 10 messaggi. Tipo di richiesta e id_articolo siano generati in maniera casuale. La comunicazione tra Dispatcher e Server deve essere implementata attraverso socket TCP e pattern Proxy/Skeleton con Skeleton implementato per ereditarietà.

Proprietà naming-JNDI per JMS:

```
"java.naming.factory.initial" -> "org.apache.activemq.jndi.ActiveMQInitialContextFactory"
"java.naming.provider.url" -> "tcp://127.0.0.1:61616"
```