

Lab - JMS

On the machine of the instructor, IP address 192.168.50.97, the following servants are running:

- a JMS topic named `dynamicTopics/Quotazioni` in which messages are sent with the following features:
 - a property `Nome` of type `String`
 - a property `Valore` of type `Float`
 - a textual body
- a JMS provider, namely Apache ActiveMQ (version 5.14.x) - see [here](#) for download, instructions, etc. Download the `.jar` files needed for interacting with it on your client (middleware libraries) - [1](#).
- a JMS producer sending messages with a predefined frequency.

The student is required to write a client Java subscribing to the topic, in order to asynchronously retrieve messages and to show (on the console output, in a panel, etc.) such messages.

Then the student, after acquiring the knowledge on such messages, should modify the client in order to filter such messages, e.g., by selecting only those ones dealing with a certain `Nome` (e.g., `Barilla`).

Useful code snippets for JMS

```
Context jndiContext = null;
ConnectionFactory connectionFactory = null;
Connection connection = null;
Session session = null;
Destination destination = null;
String destinationName = "dynamicTopics/Quotazioni";

...

/*
 * Create a JNDI API InitialContext object
 */

Properties props = new Properties();
props.setProperty(Context.INITIAL_CONTEXT_FACTORY, "org.apache.activemq.jndi.ActiveMQInitialContextFactory");
props.setProperty(Context.PROVIDER_URL, "tcp://192.168.49.73:61616");
jndiContext = new InitialContext(props);

...

connectionFactory = (ConnectionFactory) jndiContext.lookup("ConnectionFactory");
destination = (Destination) jndiContext.lookup(destinationName);

...

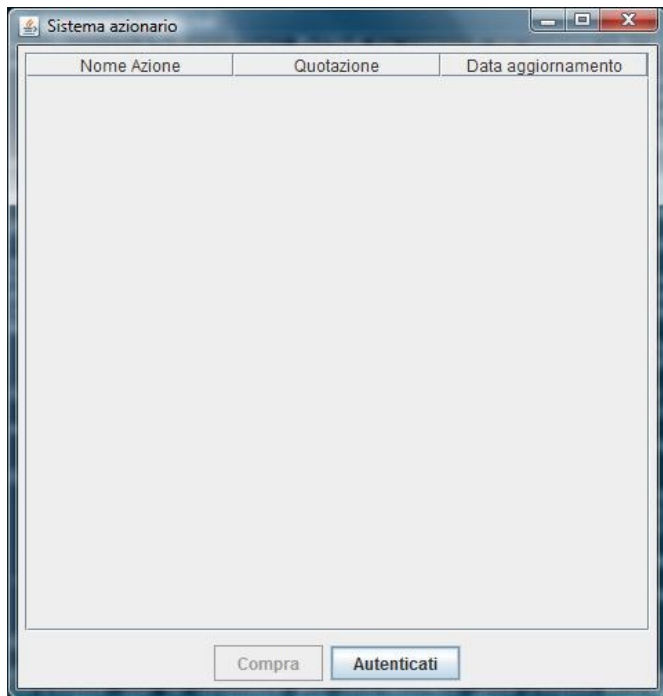
connection = connectionFactory.createConnection();
session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
```

Complex application

The student should realize an application for managing the purchasing system of a stock exchange. The business logic is realized by means of JMS technology; on the server system, it provides a JMS publisher which sends on the topic `Quotes` a text message characterized by specific properties:

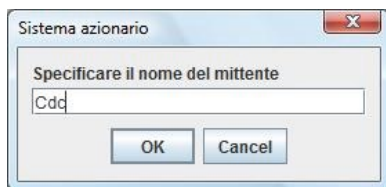
- `Nome` - the name of the given company, type `String`.
- `Valore` - the quotation of the given company, type `Float`.

On the client side, messages should be received by a subscriber, and shown through a window as shown below

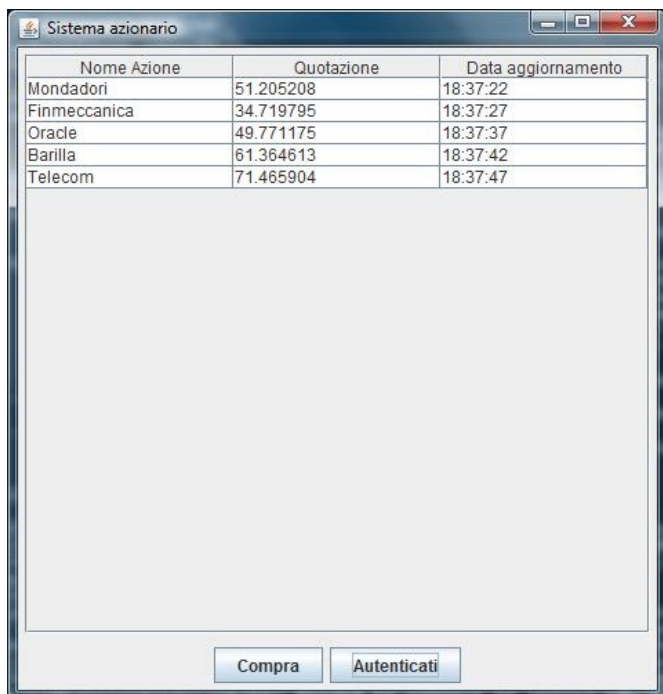


From the window you can then select a stock and buy it, specifying the purchase price and quantity. The request for purchase must be made by sending an appropriate JMS message (specific properties later in this text) to be sent on a second topic called `Ordini`. Upon receipt of the purchase message, the servant responds by sending into the same topic (`Ordini`) a notification message (which can be positive or negative); this notification message must be received by the client and must be displayed in a `JOptionPane`.

The main window is made as previously shown (unauthenticated user). In particular, when pressing the key [`Autenticati`], it is required to open a `JOptionPane` which allows to insert a string that identifies the user (see Figure).



After the insertion of a string, it starts receiving JMS messages with quotes. The JMS listener reads the message from the topic `Quotazioni`, pulls out the properties and updates the table (see Figure).



Now you can buy the shares of a stock by selecting the row of the corresponding table and pressing the [`Compra`] button. This brings up a new window, shown in the Figure below, to place purchase orders. The window allows you to enter a real number representing the price at which you want to buy the stock certificate and an integer representing the amount of shares to be purchased.

Effettua Acquisti

Utente: Claudio

Nome:

Prezzo: €


Quantità:

Importo: 10.00 €

After pressing [Ordina] it should happens as it follows:

- appropriately transform the values contained in the two text fields;
- calculate the total amount (to be shown as in the figure) as product price \times quantity taking care that the resulting number is represented with two digits after the decimal point;
- create a text message which has the properties
 - Utente, type String, with the account of the user (as previously inserted);
 - Nome, type String, with the name of the chosen company;
 - Prezzo, type Float, with the prize at which you may want to buy stock certificates;
 - Quantita, type Integer, number of stock certificates you want to buy;
- publish the message on the topic Ordini;
- create a subscriber that listens on the topic waiting for a notification message, and upon receipt of such a message extracts the Boolean property named Status and suitably displays the value of the notification in a JOptionPane (see Figure)

Effettua Acquisti

 L'acquisto è andato a buon fine