Lab assignment 6: Integration patterns

Exercise 1: Split up components into applications

Take the webshop application provided in lab 5, and convert it into 5 separate applications:

- Customers
- Products
- Shopping
- Order
- Webshop client

Every application should be implemented in its own project.
You can either copy and paste the webshop project 4 times, and then rename the projects, or you create new projects, and copy and paste the classes into these new projects. Make sure you have the required libraries in the POM file.

The goal of this exercise is to make sure all 5 applications are completely separated from each other. It is not allowed to import a class that exist in another application. It is also not allowed to place one project on the classpath of another project.

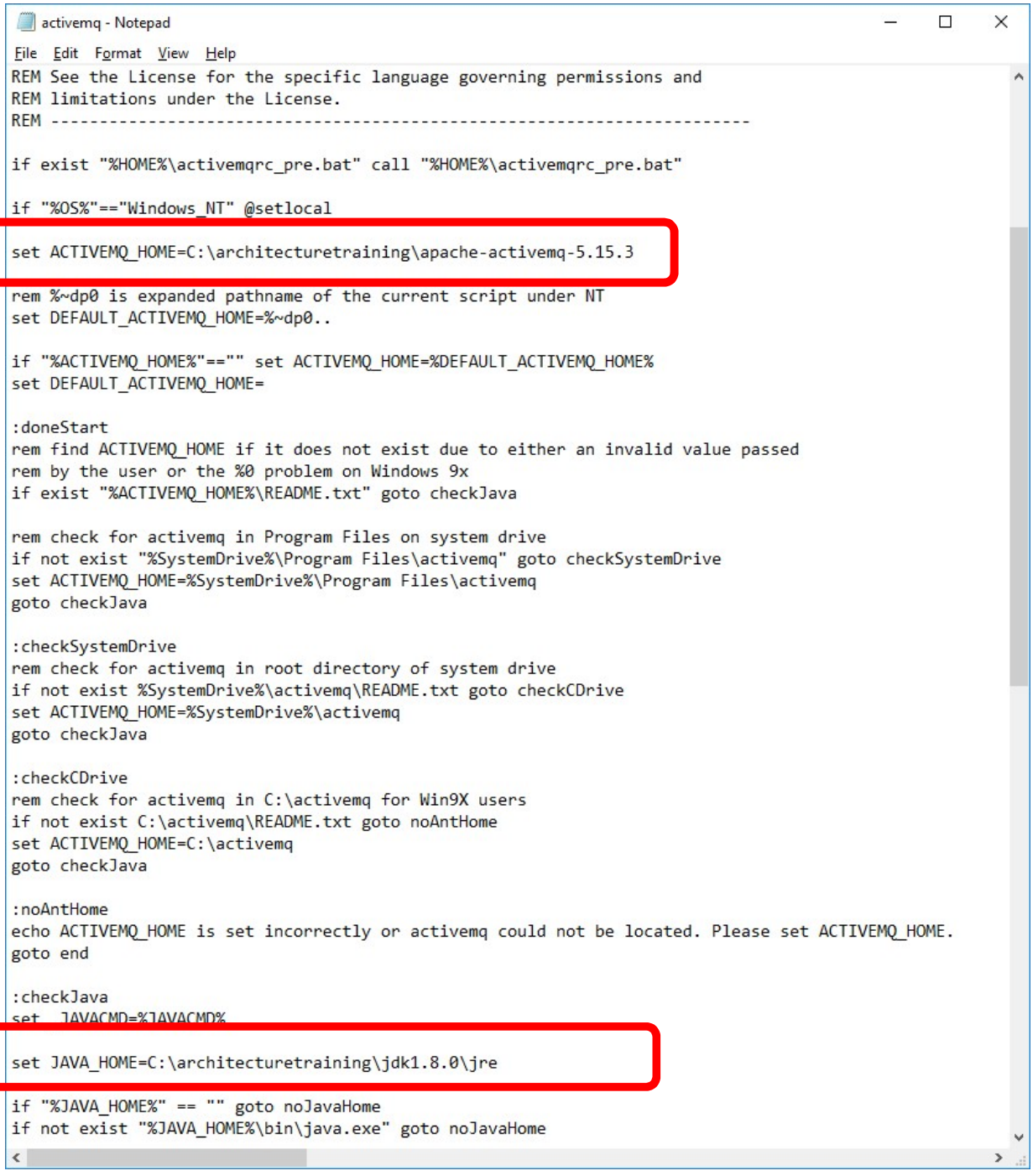
Start all 4 webshop applications and then run the webshop client which should then call the individual applications with REST calls.

Exercise 2: JMS

For this exercise we will use ActiveMQ as JMS middleware so we first have to start ActiveMQ.

First edit the file **C:\springtraining\apache-activemq-5.15.3\bin\activemq.bat**

Make the following changes:



Then save **activemq.bat**

Then run the file **startactivemq.bat** and activemq will start:

Now that we have started ActiveMQ, we can continue by importing the projects.

Import the given **SpringJMSSender** project and the **SpringJMSReceiver** project.
First run the SpringJmsReceiverApplication.java in the **SpringJMSReceiver** project.
Then run the SpringJmsSenderApplication in the **SpringJMSSender** project.

You should now see the following in the sender console:

```
Sending a JMS message.
Sending a JMS message.
```

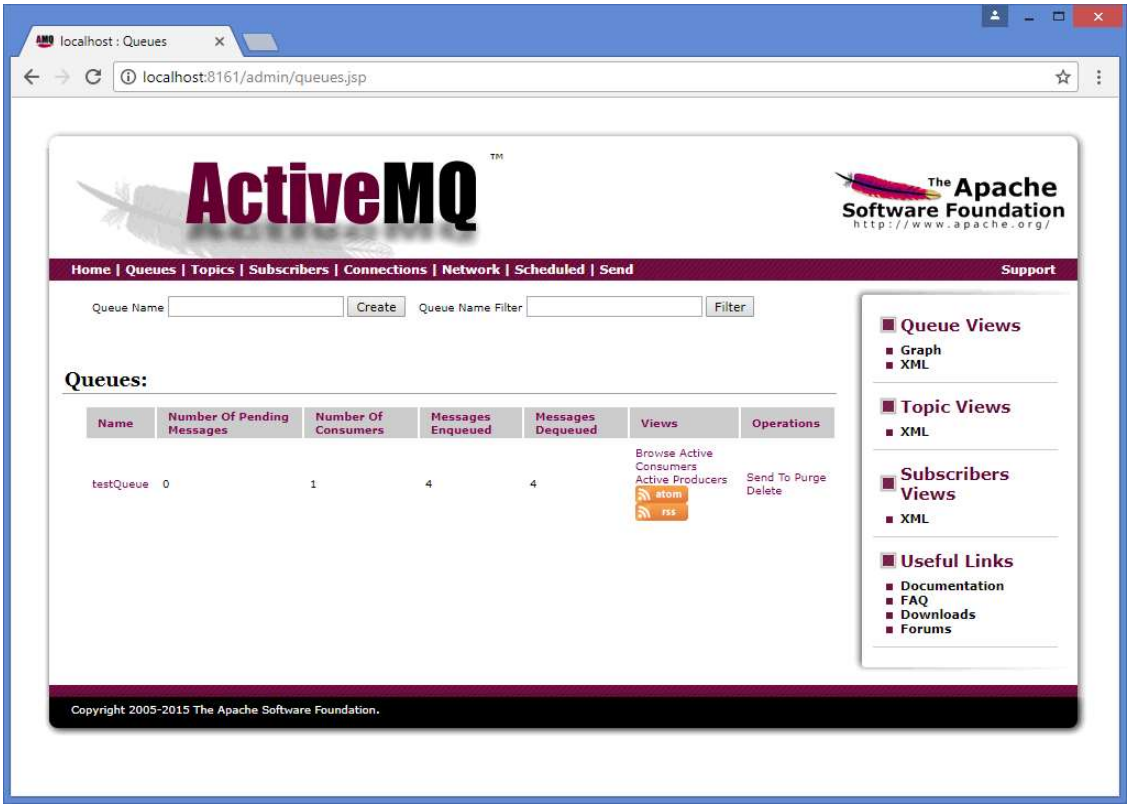
And you should see the following in the receiver console:

```
Receiver has started ...
JMS receiver received message:Frank Brown
JMS receiver received message:Mary Smith
```

Then open the ActiveMQ console at http://localhost:8161/admin.

You can login with username **admin** and password **admin**

Select the Queues page from the menu:

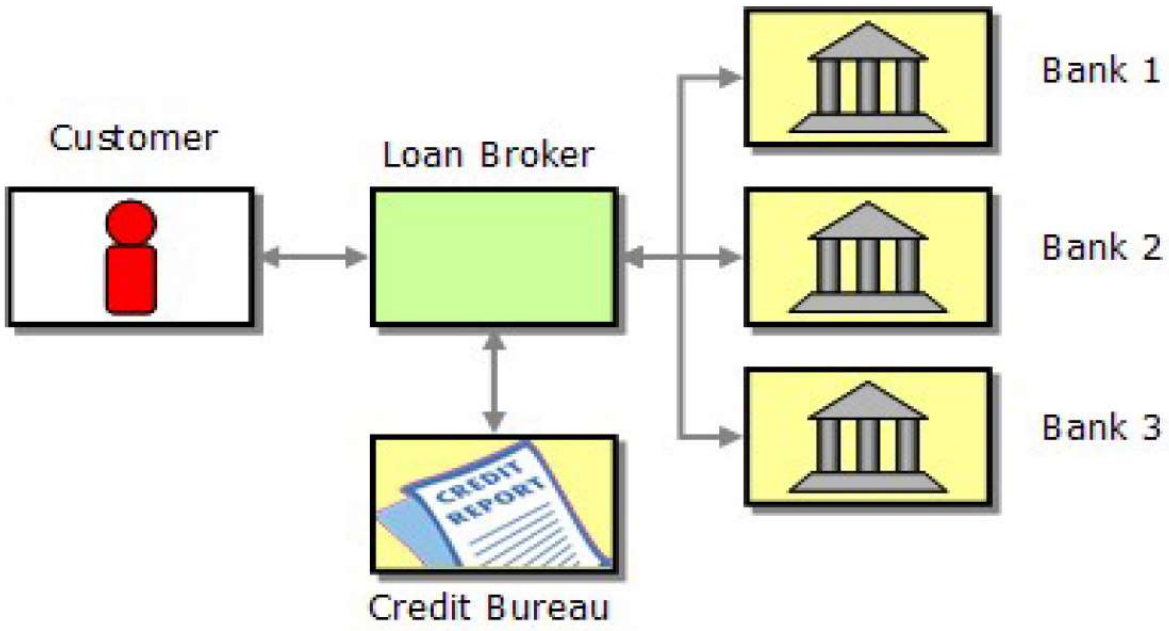


You see that the queue with name **testqueue** has one consumer, and 4 message have been received and consumed.

Now implement the following with JMS in your webshop of exercise 1:
When the user confirms the Order we need to update the stock quantity in the ProductCatalog.
Implement this with JMS. Make sure you add the required dependencies in the POM file.

Exercise 3: Integration patterns

We have to design the integration of a loan broker between the following entities:



- The Test Client makes requests for loan quotes.
- The Loan Broker acts as the central process manager and coordinates the communication between the credit bureau and the banks.
- The Credit Bureau provides a service to the Loan Broker, computing customer's credit scores.
- Each Bank receives a quote request from the Loan Broker and submits an interest rate quote according to the loan parameters.

We have the following requirements:

- Management Console:
  - We want a single front-end that displays the health of all components and allows us to take compensating actions if something goes wrong.
- Loan Broker Quality of Service:
  - We want to monitor the loan broker's response times between quote request and response and send it to the management console.
- Verify the Credit Bureau Operation:
  - The Credit Bureau is an external service provided by a third party. We want to ensure the correct operation of this service by periodically sending test messages.
- Credit Bureau Failover:
  - If the Credit Bureau malfunctions we want to temporarily redirect the credit request messages to another service provider.

Draw the design of your integration solution using the integration patterns.