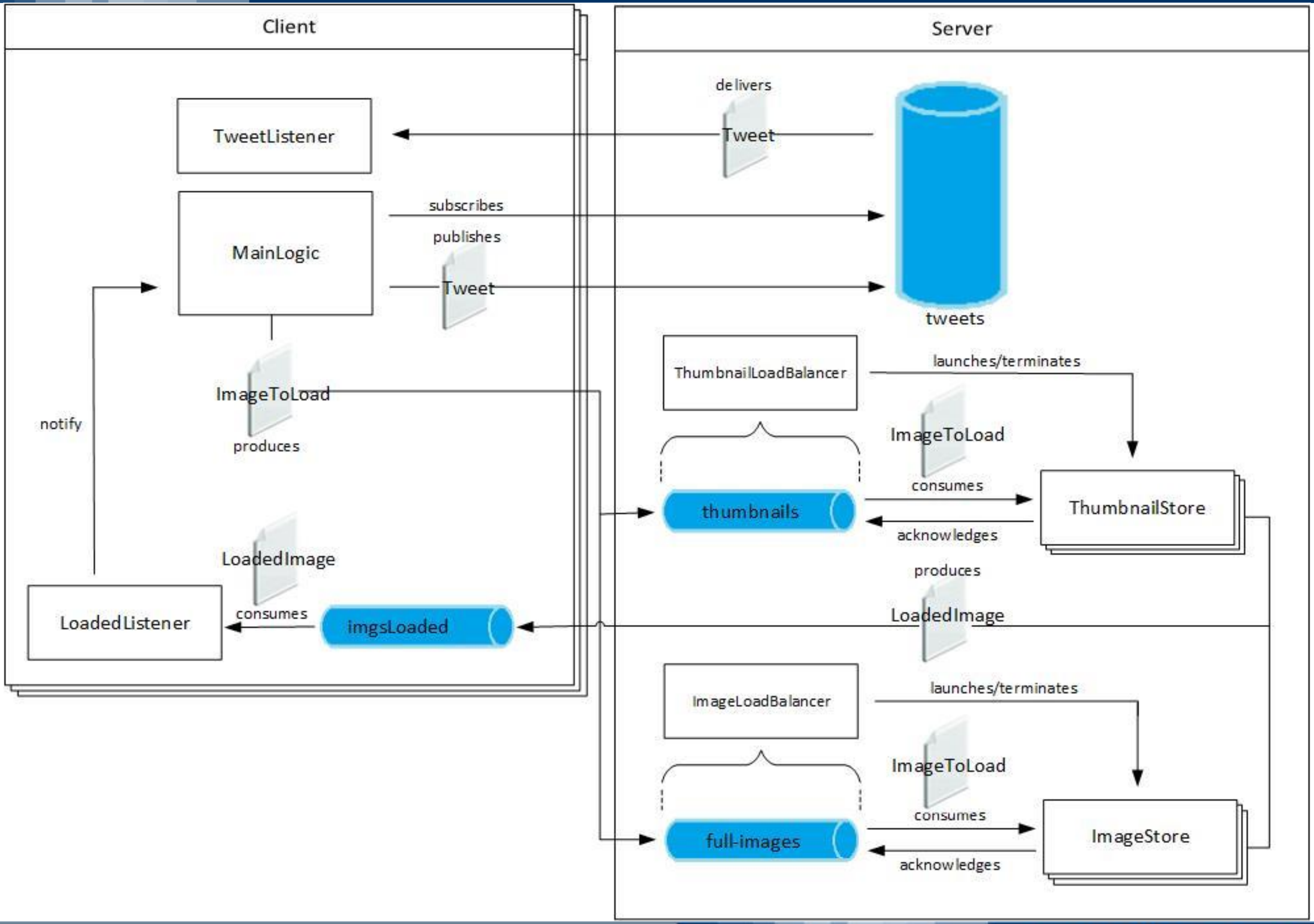
POLITECNICO DI MILANO

Projects presentation

Middleware Technologies for Distributed Systems

JMS Project - Architecture



JMS Project - Messages

- ImageToLoad:
  - String username: used to store the image in the client personal folder
  - int id: reflects client order of the images
  - String extension
  - byte[] img

- LoadedImage:
  - int id: id of the loaded image, to guarantee that the order is preserved
  - URL url: url to download the image
  - boolean thumbnail: tells if the loaded image is a thumbnail

- Tweet:
  - String text
  - ArrayList<URL> images
  - ArrayList<URL> thumbnails

JMS Project – Queue/Topic

- Queue
  - thumbnails

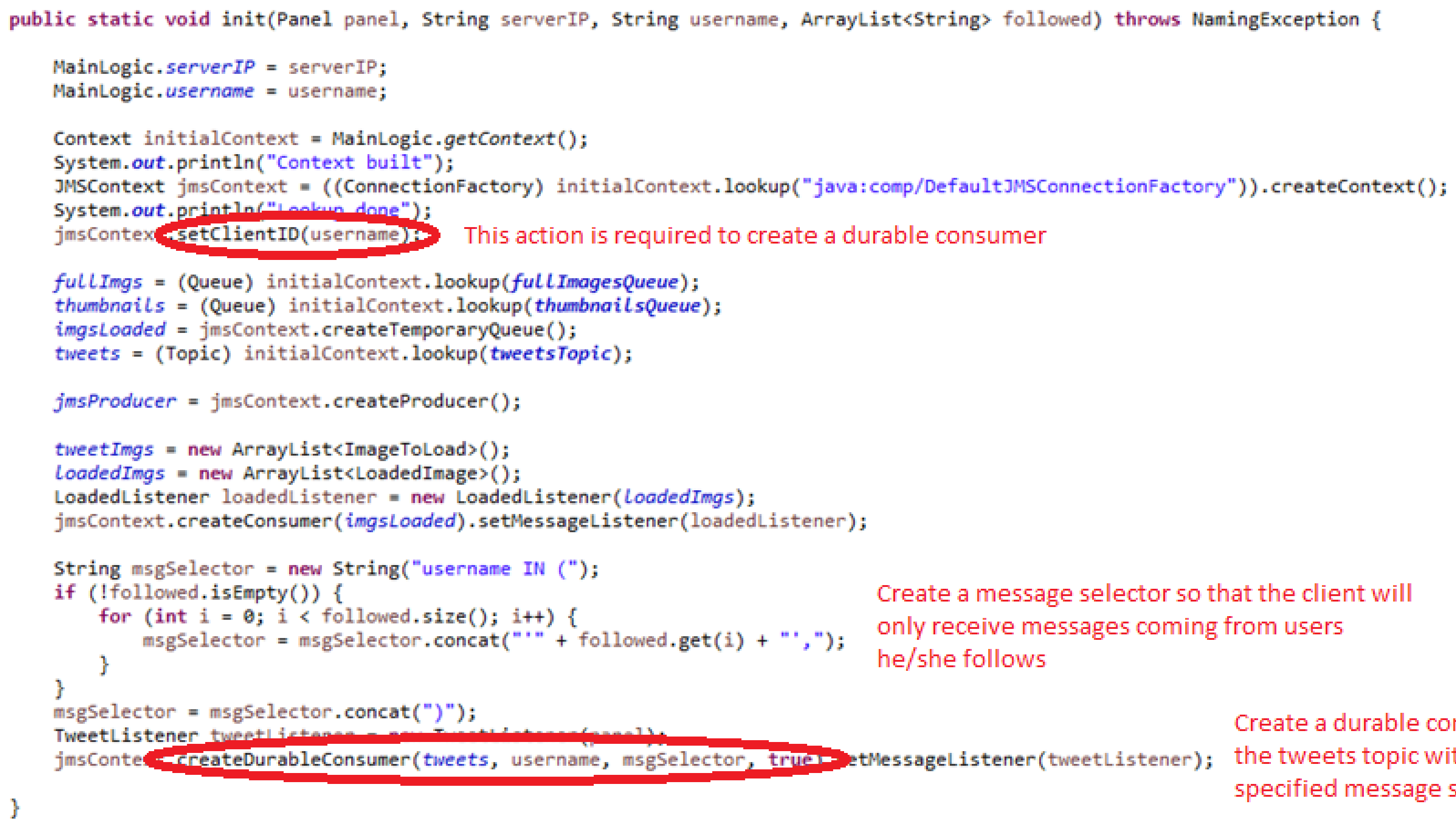
    queue used by the clients to send images to the thumbnail store service
  - full-images

    queue used by the clients to send images to the image store service
  - imagesLoaded

    temporary queue on the client used by the storage services to acknowledge the correct upload of the images
- Topic
  - Tweet

    topic that clients use to send tweets and subscribe to in a durable way (so to receive messages sent when they were offline)

JMS Project – Client initialization



```java
public static void init(Panel panel, String serverIP, String username, ArrayList<String> followed) throws NamingException {

    MainLogic.serverIP = serverIP;
    MainLogic.username = username;

    Context initialContext = MainLogic.getContext();
    System.out.println("Context built");
    JMSContext jmsContext = ((ConnectionFactory) initialContext.lookup("java:comp/DefaultJMSConnectionFactory")).createContext();
    System.out.println("Lookup done");
    jmsContext.setClientID(username);

    fullImgs = (Queue) initialContext.lookup(fullImagesQueue);
    thumbnails = (Queue) initialContext.lookup(thumbnailsQueue);
    imgsLoaded = jmsContext.createTemporaryQueue();
    tweets = (Topic) initialContext.lookup(tweetsTopic);

    jmsProducer = jmsContext.createProducer();

    tweetImgs = new ArrayList<ImageToLoad>();
    loadedImgs = new ArrayList<LoadedImage>();
    LoadedListener loadedListener = new LoadedListener(loadedImgs);
    jmsContext.createConsumer(imgsLoaded).setMessageListener(loadedListener);

    String msgSelector = new String("username IN (");
    if (!followed.isEmpty()) {
        for (int i = 0; i < followed.size(); i++) {
            msgSelector = msgSelector.concat("'" + followed.get(i) + "',");
        }
    }
    msgSelector = msgSelector.concat(")");
    TweetListener tweetListener = new TweetListener(panel);
    jmsContext.createDurableConsumer(tweets, username, msgSelector, true).setMessageListener(tweetListener);

}
```

This action is required to create a durable consumer

Create a message selector so that the client will only receive messages coming from users he/she follows

Create a durable consumer on the tweets topic with the specified message selector

JMS Project – Send tweet

```java
public static void sendTweet(String tweetText) throws InterruptedException {

    ArrayList<URL> imgsURL = null;
    ArrayList<URL> thumbnailsURL = null;

    if (!tweetImgs.isEmpty()) {
        imgsURL = new ArrayList<URL>();
        thumbnailsURL = new ArrayList<URL>();
        for (ImageToLoad imgToLoad : tweetImgs) {
            jmsProducer.setJMSReplyTo(imgsLoaded).send(fullImgs, imgToLoad);
            jmsProducer.setJMSReplyTo(imgsLoaded).send(thumbnails, imgToLoad);
        }

        // wait for the images to be loaded
        int nResponses = 2 * tweetImgs.size();
        tweetImgs = new ArrayList<ImageToLoad>();
        synchronized (loadedImgs) {
            while(loadedImgs.size() < nResponses) {
                loadedImgs.wait();
            }
        }

        // sort the images as they were selected by the user
        Collections.sort(loadedImgs, new Comparator<LoadedImage>() {
            @Override
            public int compare(LoadedImage li1, LoadedImage li2) {
                return li1.getId() - li2.getId();
            }
        });

        for (int i = 0; i < loadedImgs.size(); i++) {
            LoadedImage loadedImg = loadedImgs.get(i);
            if (!loadedImg.isThumbnail()) {
                imgsURL.add(loadedImg.getURL());
            } else {
                thumbnailsURL.add(loadedImg.getURL());
            }
        }

        loadedImgs.removeAll(loadedImgs);
    }

    jmsProducer.setProperty("username", username).send(tweets, new Tweet(tweetText, imgsURL, thumbnailsURL));
}
```

Set the reply to the client owned temporary queue and send the images attached to the tweet to the storage services

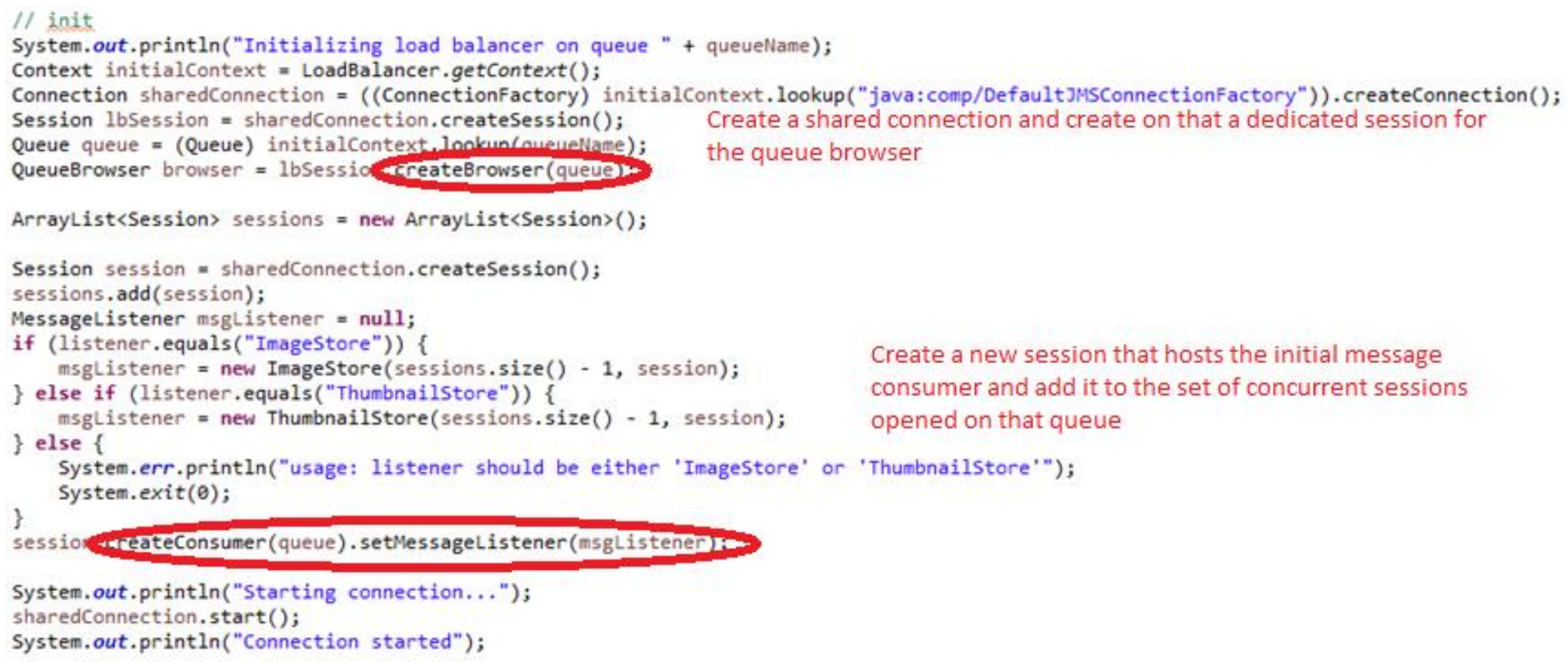
Wait for the LoadedListener to receive all the replies from the storage services

This action is needed since the storage services might process messages out-of-order due to load balancing

Set the username property, so that the message will go through the message selector of the followers and send the tweet

POLITECNICO DI MILANO

JMS Project – Server initialization

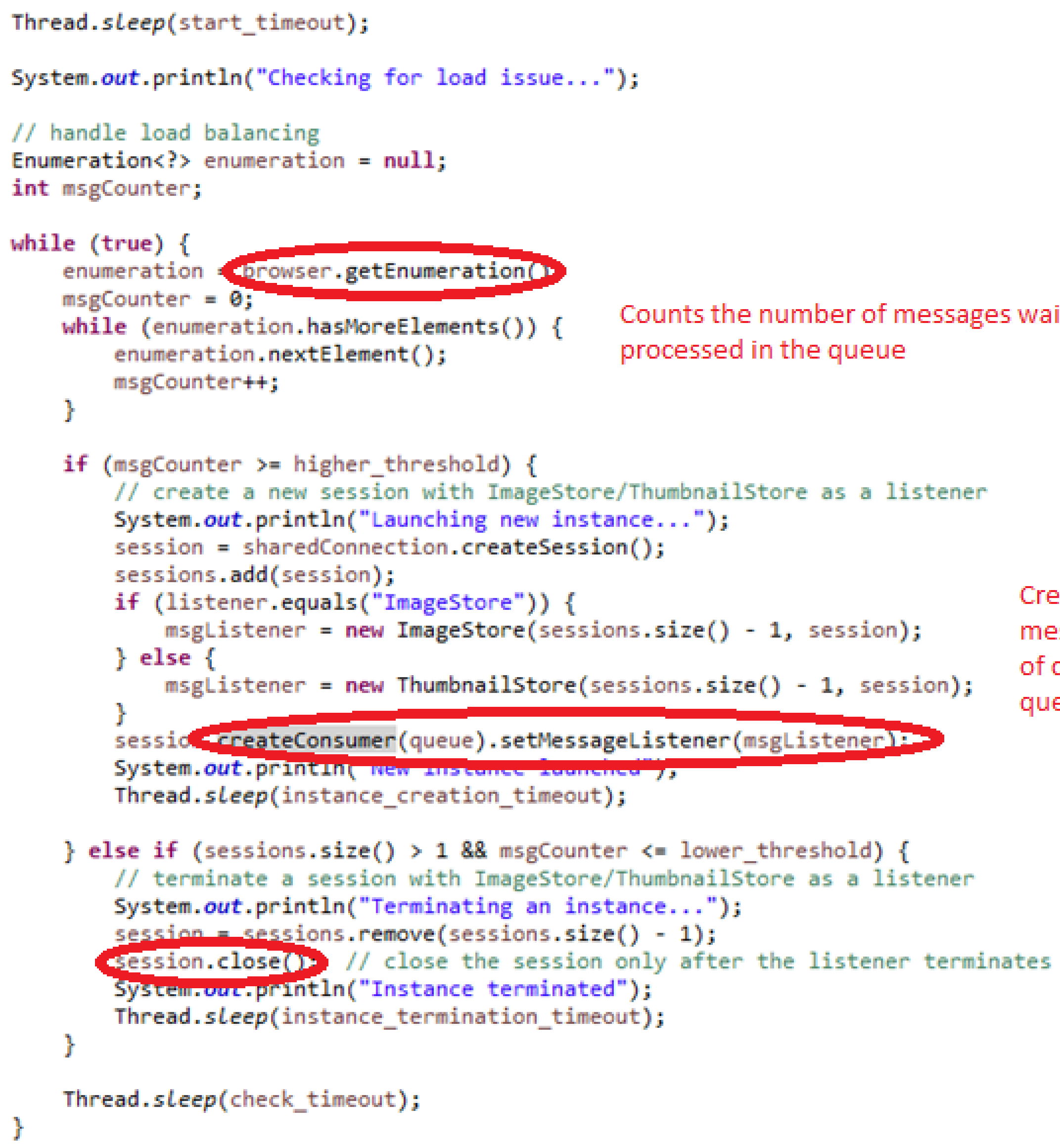


```java
// init
System.out.println("Initializing load balancer on queue " + queueName);
Context initialContext = LoadBalancer.getContext();
Connection sharedConnection = ((ConnectionFactory) initialContext.lookup("java:comp/DefaultJMSConnectionFactory")).createConnection();
Session lbSession = sharedConnection.createSession();
Queue queue = (Queue) initialContext.lookup(queueName);
QueueBrowser browser = lbSession.createBrowser(queue);

ArrayList<Session> sessions = new ArrayList<Session>();

Session session = sharedConnection.createSession();
sessions.add(session);
MessageListener msgListener = null;
if (listener.equals("ImageStore")) {
    msgListener = new ImageStore(sessions.size() - 1, session);
} else if (listener.equals("ThumbnailStore")) {
    msgListener = new ThumbnailStore(sessions.size() - 1, session);
} else {
    System.err.println("usage: listener should be either 'ImageStore' or 'ThumbnailStore'");
    System.exit(0);
}
session.createConsumer(queue).setMessageListener(msgListener);

System.out.println("Starting connection...");
sharedConnection.start();
System.out.println("Connection started");
```

Create a shared connection and create on that a dedicated session for the queue browser

Create a new session that hosts the initial message consumer and add it to the set of concurrent sessions opened on that queue

JMS Project – Load balancing



```
Thread.sleep(start_timeout);

System.out.println("Checking for load issue...");

// handle load balancing
Enumeration<?> enumeration = null;
int msgCounter;

while (true) {
    enumeration = browser.getEnumeration();
    msgCounter = 0;
    while (enumeration.hasMoreElements()) {
        enumeration.nextElement();
        msgCounter++;
    }

    if (msgCounter >= higher_threshold) {
        // create a new session with ImageStore/ThumbnailStore as a listener
        System.out.println("Launching new instance...");
        session = sharedConnection.createSession();
        sessions.add(session);
        if (listener.equals("ImageStore")) {
            msgListener = new ImageStore(sessions.size() - 1, session);
        } else {
            msgListener = new ThumbnailStore(sessions.size() - 1, session);
        }
        session.createConsumer(queue).setMessageListener(msgListener);
        System.out.println("New instance launched");
        Thread.sleep(instance_creation_timeout);

    } else if (sessions.size() > 1 && msgCounter <= lower_threshold) {
        // terminate a session with ImageStore/ThumbnailStore as a listener
        System.out.println("Terminating an instance...");
        session = sessions.remove(sessions.size() - 1);
        session.close(); // close the session only after the listener terminates
        System.out.println("Instance terminated");
        Thread.sleep(instance_termination_timeout);
    }

    Thread.sleep(check_timeout);
}
```

Counts the number of messages waiting to be processed in the queue

Create a new session that hosts a new message consumer and add it to the set of concurrent sessions opened on that queue

Terminates the last session added to the set and so reduces the number of cuncurrent listeners on the queue