1.Install artemis in ur local machine

Go to bin folder and use my below command,where myLocalBroker is the folder name.

F:\sotfwares\apache-artemis-2.19.1\bin>**artemis create /myLocalBroker**

- Once command ran , then it will create some files in that above folder , go to the folder where u installed here it is **myLocalBroker**
- Inside we have a tool called **artemis** and then execute **run** command **on this**

  Example:

  F:\myLocalBroker\bin> artemis run

  Use crenetials as admin and pass as admin,

  U will find console at http://localhost:8161/console

You can now start the broker by executing:

```
  "F:\myLocalBroker\bin\artemis" run
```

Or you can setup the broker as Windows service and run it in the background:

```
  "F:\myLocalBroker\bin\artemis-service.exe" install
  "F:\myLocalBroker\bin\artemis-service.exe" start
```

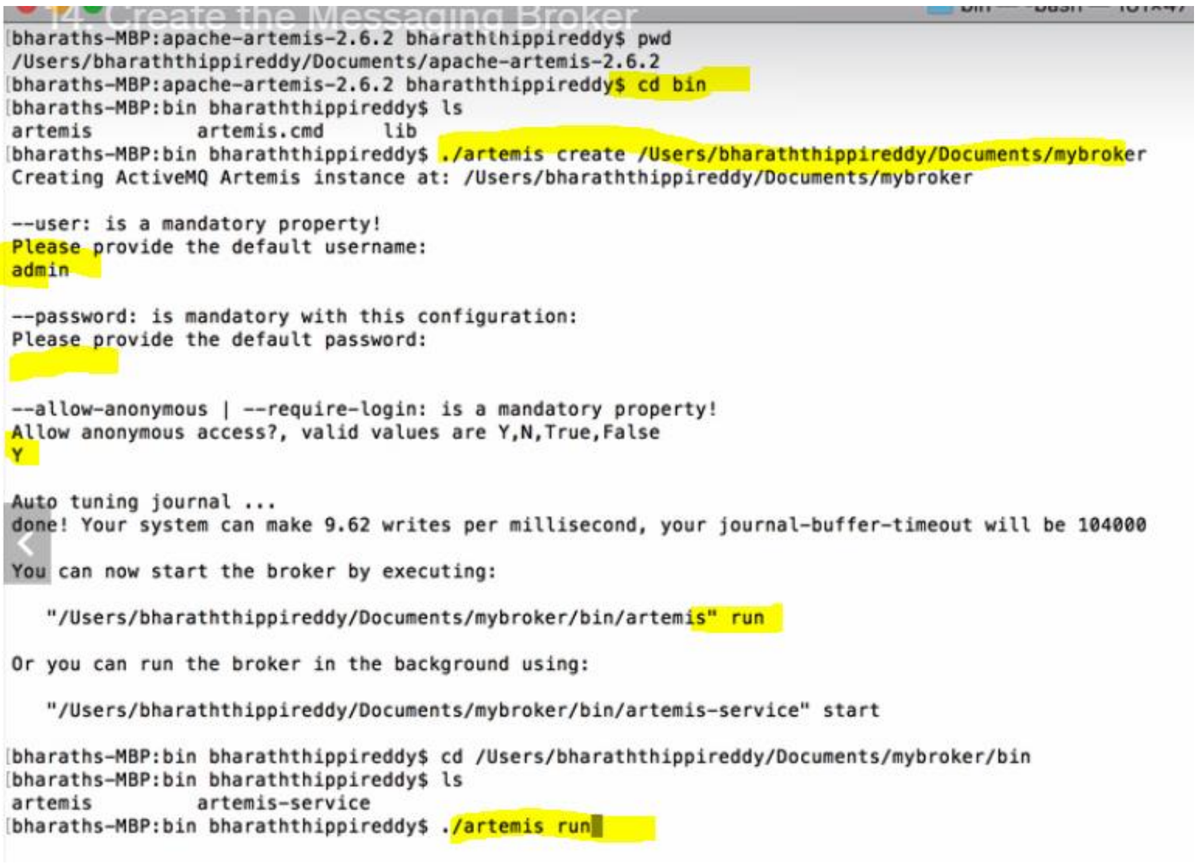
To stop the windows service:

```
    "F:\myLocalBroker\bin\artemis-service.exe" stop
```

To uninstall the windows service

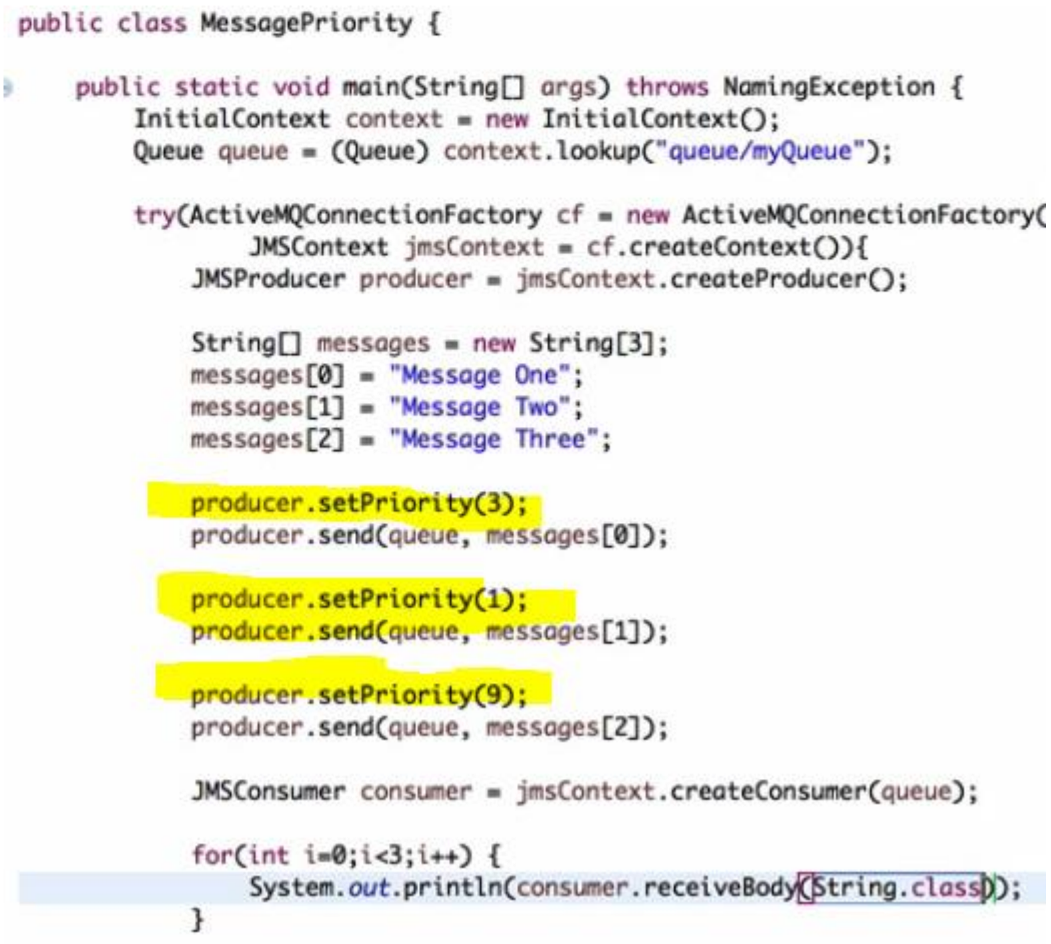
```
    "F:\myLocalBroker\bin\artemis-service.exe" uninstall
```

14. Create the Messaging Broker



```
bharaths-MBP:apache-artemis-2.6.2 bharaththippireddy$ pwd
/Users/bharaththippireddy/Documents/apache-artemis-2.6.2
bharaths-MBP:apache-artemis-2.6.2 bharaththippireddy$ cd bin
bharaths-MBP:bin bharaththippireddy$ ls
artemis         artemis.cmd     lib
bharaths-MBP:bin bharaththippireddy$ ./artemis create /Users/bharaththippireddy/Documents/mybroker
Creating ActiveMQ Artemis instance at: /Users/bharaththippireddy/Documents/mybroker

--user: is a mandatory property!
Please provide the default username:
admin

--password: is mandatory with this configuration:
Please provide the default password:


--allow-anonymous | --require-login: is a mandatory property!
Allow anonymous access?, valid values are Y,N,True,False
Y

Auto tuning journal ...
done! Your system can make 9.62 writes per millisecond, your journal-buffer-timeout will be 104000

You can now start the broker by executing:

   "/Users/bharaththippireddy/Documents/mybroker/bin/artemis" run

Or you can run the broker in the background using:

   "/Users/bharaththippireddy/Documents/mybroker/bin/artemis-service" start

bharaths-MBP:bin bharaththippireddy$ cd /Users/bharaththippireddy/Documents/mybroker/bin
bharaths-MBP:bin bharaththippireddy$ ls
artemis         artemis-service
bharaths-MBP:bin bharaththippireddy$ ./artemis run
```

27.Prioritise messages



```
public class MessagePriority {

    public static void main(String[] args) throws NamingException {
        InitialContext context = new InitialContext();
        Queue queue = (Queue) context.lookup("queue/myQueue");

        try(ActiveMQConnectionFactory cf = new ActiveMQConnectionFactory(
                JMSContext jmsContext = cf.createContext()){
            JMSProducer producer = jmsContext.createProducer();

            String[] messages = new String[3];
            messages[0] = "Message One";
            messages[1] = "Message Two";
            messages[2] = "Message Three";

            producer.setPriority(3);
            producer.send(queue, messages[0]);

            producer.setPriority(1);
            producer.send(queue, messages[1]);

            producer.setPriority(9);
            producer.send(queue, messages[2]);

            JMSConsumer consumer = jmsContext.createConsumer(queue);

            for(int i=0;i<3;i++) {
                System.out.println(consumer.receiveBody(String.class));
            }
```

s

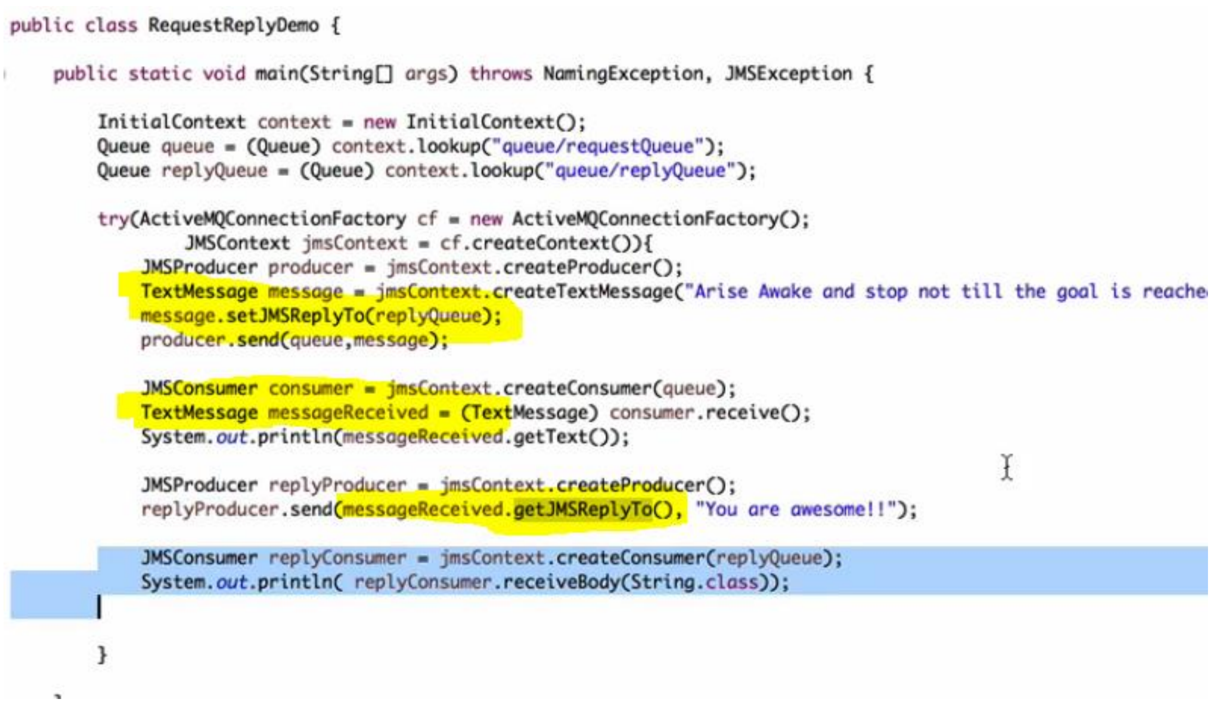
30. Dynamically Replying to the Queue using header of the received message

While sending a message u set the header to which that consumer should respond.

Consumer, after receiving the message , he will check the header and reply to that

Real example:- in olden days, if u wanna reply to the letter, we will reply to the from address right

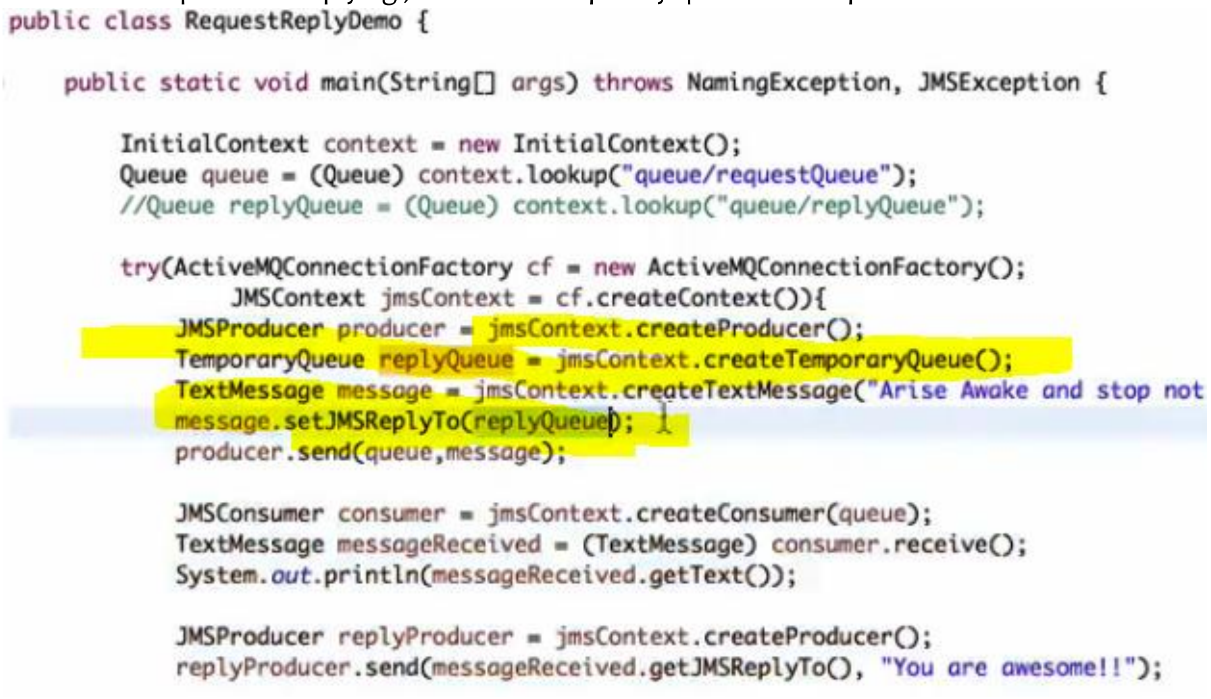
Same here, we have to set the header where it consist of the response queue.



```
public class RequestReplyDemo {

    public static void main(String[] args) throws NamingException, JMSException {

        InitialContext context = new InitialContext();
        Queue queue = (Queue) context.lookup("queue/requestQueue");
        Queue replyQueue = (Queue) context.lookup("queue/replyQueue");

        try(ActiveMQConnectionFactory cf = new ActiveMQConnectionFactory();
                JMSContext jmsContext = cf.createContext()){
            JMSProducer producer = jmsContext.createProducer();
            TextMessage message = jmsContext.createTextMessage("Arise Awake and stop not till the goal is reache
            message.setJMSReplyTo(replyQueue);
            producer.send(queue,message);

            JMSConsumer consumer = jmsContext.createConsumer(queue);
            TextMessage messageReceived = (TextMessage) consumer.receive();
            System.out.println(messageReceived.getText());

            JMSProducer replyProducer = jmsContext.createProducer();
            replyProducer.send(messageReceived.getJMSReplyTo(), "You are awesome!!");

            JMSConsumer replyConsumer = jmsContext.createConsumer(replyQueue);
            System.out.println( replyConsumer.receiveBody(String.class));

        }
```

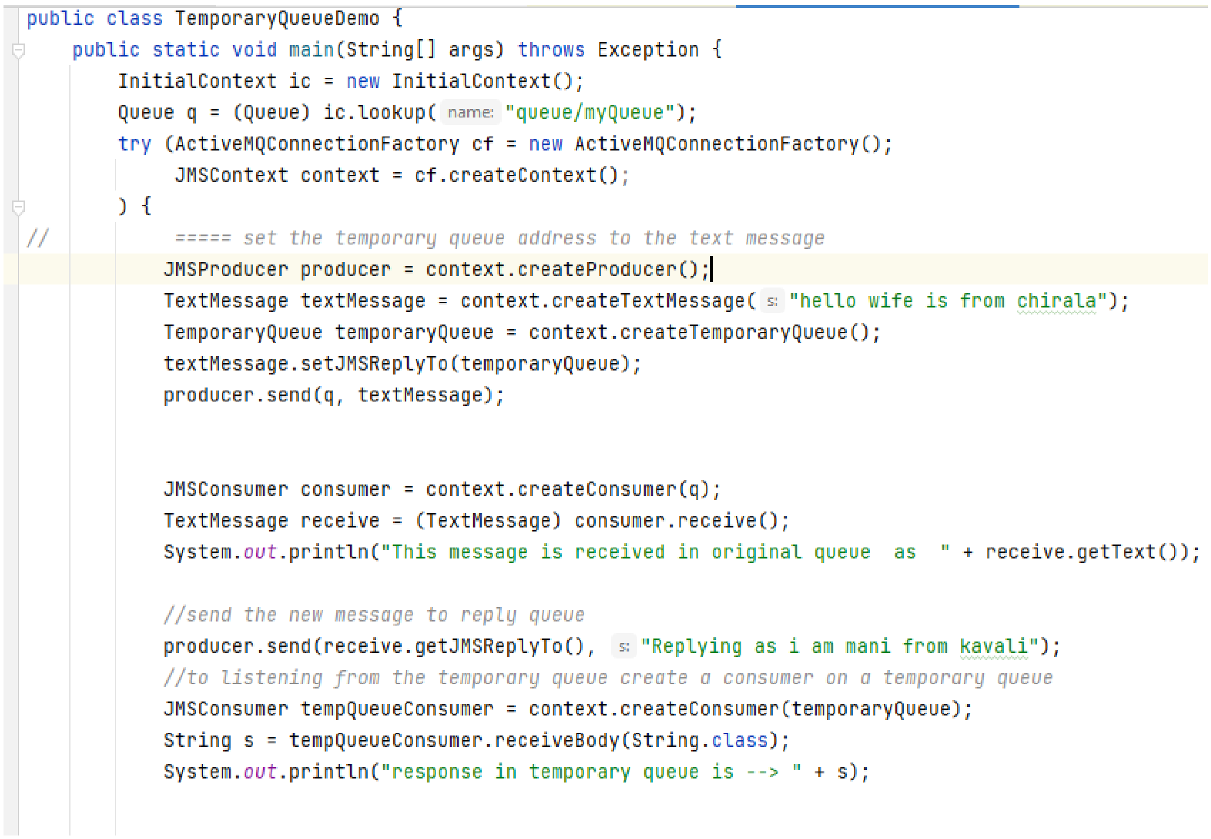
31.Replying to a temporary queue

Don't create queue for replying , u can use temporary queues concept



```
public class RequestReplyDemo {

    public static void main(String[] args) throws NamingException, JMSException {

        InitialContext context = new InitialContext();
        Queue queue = (Queue) context.lookup("queue/requestQueue");
        //Queue replyQueue = (Queue) context.lookup("queue/replyQueue");

        try(ActiveMQConnectionFactory cf = new ActiveMQConnectionFactory();
                JMSContext jmsContext = cf.createContext()){
            JMSProducer producer = jmsContext.createProducer();
            TemporaryQueue replyQueue = jmsContext.createTemporaryQueue();
            TextMessage message = jmsContext.createTextMessage("Arise Awake and stop not
            message.setJMSReplyTo(replyQueue);
            producer.send(queue,message);

            JMSConsumer consumer = jmsContext.createConsumer(queue);
            TextMessage messageReceived = (TextMessage) consumer.receive();
            System.out.println(messageReceived.getText());

            JMSProducer replyProducer = jmsContext.createProducer();
            replyProducer.send(messageReceived.getJMSReplyTo(), "You are awesome!!");
```

Listening on a temporary queue



```
public class TemporaryQueueDemo {
    public static void main(String[] args) throws Exception {
        InitialContext ic = new InitialContext();
        Queue q = (Queue) ic.lookup( name: "queue/myQueue");
        try (ActiveMQConnectionFactory cf = new ActiveMQConnectionFactory();
             JMSContext context = cf.createContext();
        ) {
//           ===== set the temporary queue address to the text message
            JMSProducer producer = context.createProducer();
            TextMessage textMessage = context.createTextMessage( s: "hello wife is from chirala");
            TemporaryQueue temporaryQueue = context.createTemporaryQueue();
            textMessage.setJMSReplyTo(temporaryQueue);
            producer.send(q, textMessage);


            JMSConsumer consumer = context.createConsumer(q);
            TextMessage receive = (TextMessage) consumer.receive();
            System.out.println("This message is received in original queue  as  " + receive.getText());

            //send the new message to reply queue
            producer.send(receive.getJMSReplyTo(), s: "Replying as i am mani from kavali");
            //to listening from the temporary queue create a consumer on a temporary queue
            JMSConsumer tempQueueConsumer = context.createConsumer(temporaryQueue);
            String s = tempQueueConsumer.receiveBody(String.class);
            System.out.println("response in temporary queue is --> " + s);
```

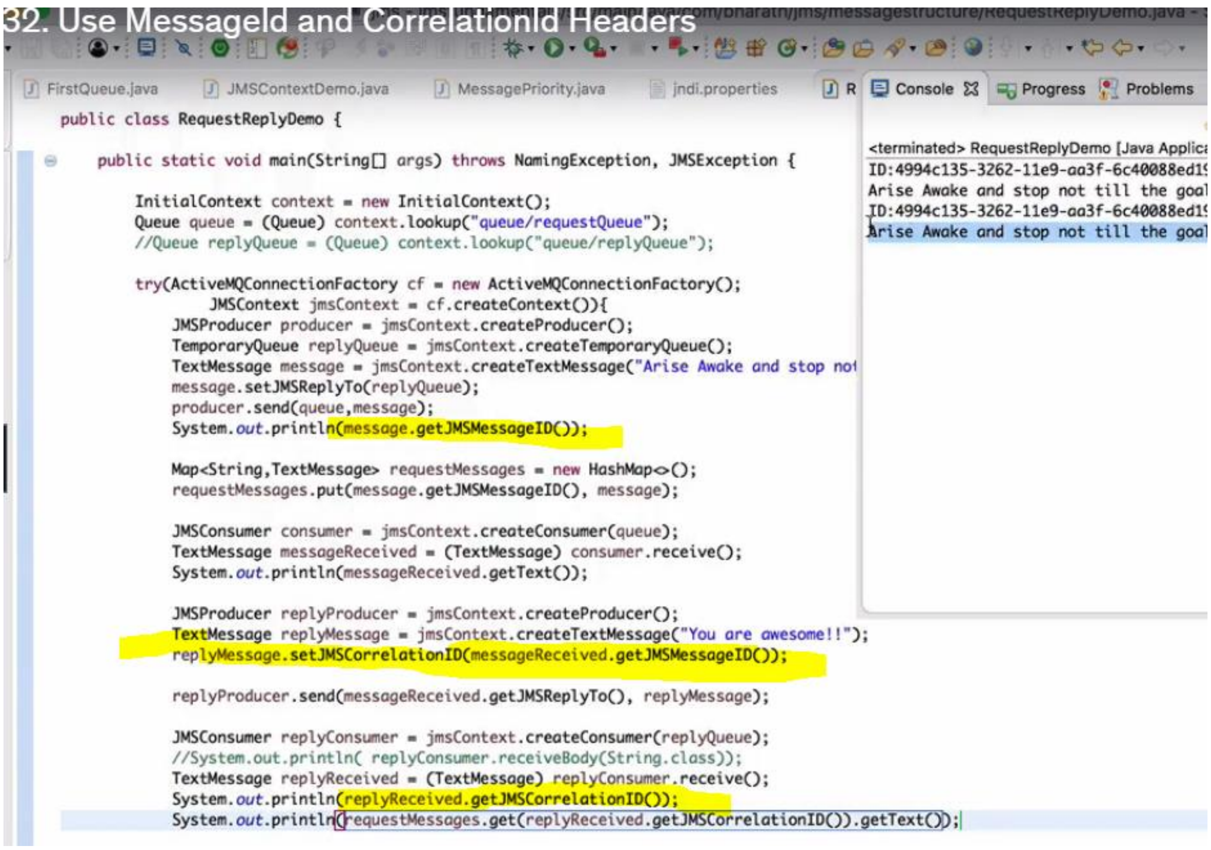
32.using co-relation id and message ID

Why?— **just to link the request and response**

See for every message , jms will provide an unique message ID

U will send a message and u will get the response and **if u want to know, the response is for which message** , then while sending the reply back , u should set the value for co-relation id field , this corelation better set same as message id u received

Note :- u cant set the message id ,but u can set the co relation id

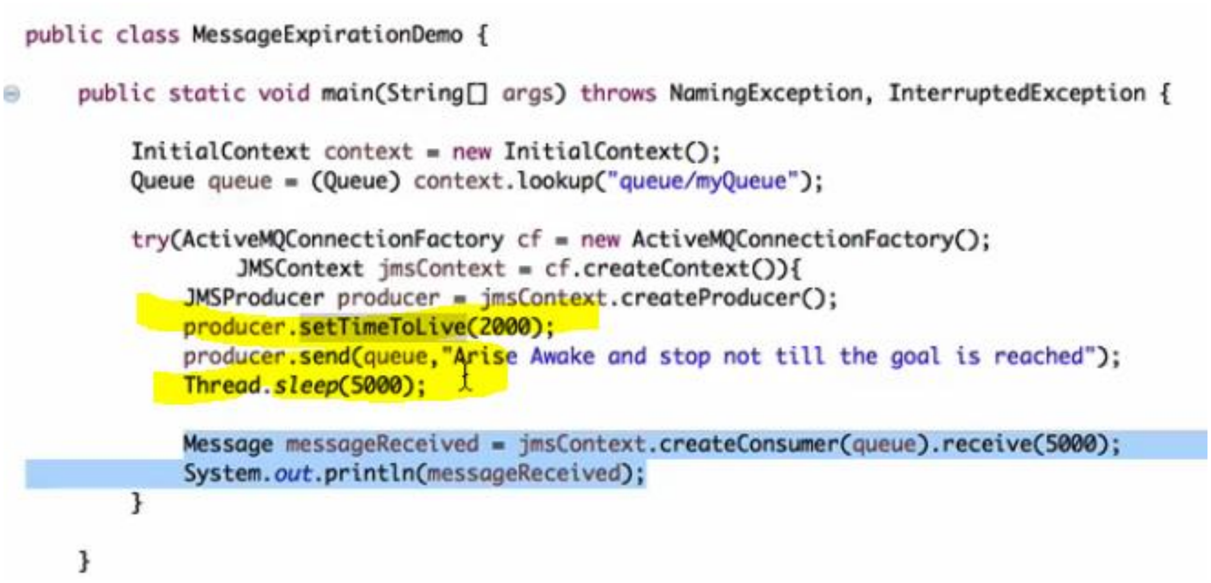

32. Use MessageId and CorrelationId Headers

```
public class RequestReplyDemo {

    public static void main(String[] args) throws NamingException, JMSException {

        InitialContext context = new InitialContext();
        Queue queue = (Queue) context.lookup("queue/requestQueue");
        //Queue replyQueue = (Queue) context.lookup("queue/replyQueue");

        try(ActiveMQConnectionFactory cf = new ActiveMQConnectionFactory();
                JMSContext jmsContext = cf.createContext()){
            JMSProducer producer = jmsContext.createProducer();
            TemporaryQueue replyQueue = jmsContext.createTemporaryQueue();
            TextMessage message = jmsContext.createTextMessage("Arise Awake and stop not
            message.setJMSReplyTo(replyQueue);
            producer.send(queue,message);
            System.out.println(message.getJMSMessageID());

            Map<String,TextMessage> requestMessages = new HashMap<>();
            requestMessages.put(message.getJMSMessageID(), message);

            JMSConsumer consumer = jmsContext.createConsumer(queue);
            TextMessage messageReceived = (TextMessage) consumer.receive();
            System.out.println(messageReceived.getText());

            JMSProducer replyProducer = jmsContext.createProducer();
            TextMessage replyMessage = jmsContext.createTextMessage("You are awesome!!");
            replyMessage.setJMSCorrelationID(messageReceived.getJMSMessageID());

            replyProducer.send(messageReceived.getJMSReplyTo(), replyMessage);

            JMSConsumer replyConsumer = jmsContext.createConsumer(replyQueue);
            //System.out.println( replyConsumer.receiveBody(String.class));
            TextMessage replyReceived = (TextMessage) replyConsumer.receive();
            System.out.println(replyReceived.getJMSCorrelationID());
            System.out.println(requestMessages.get(replyReceived.getJMSCorrelationID()).getText());
```

```
<terminated> RequestReplyDemo [Java Applica
ID:4994c135-3262-11e9-aa3f-6c40088ed19
Arise Awake and stop not till the goal
ID:4994c135-3262-11e9-aa3f-6c40088ed19
Arise Awake and stop not till the goal
```

33. Set Message expiry

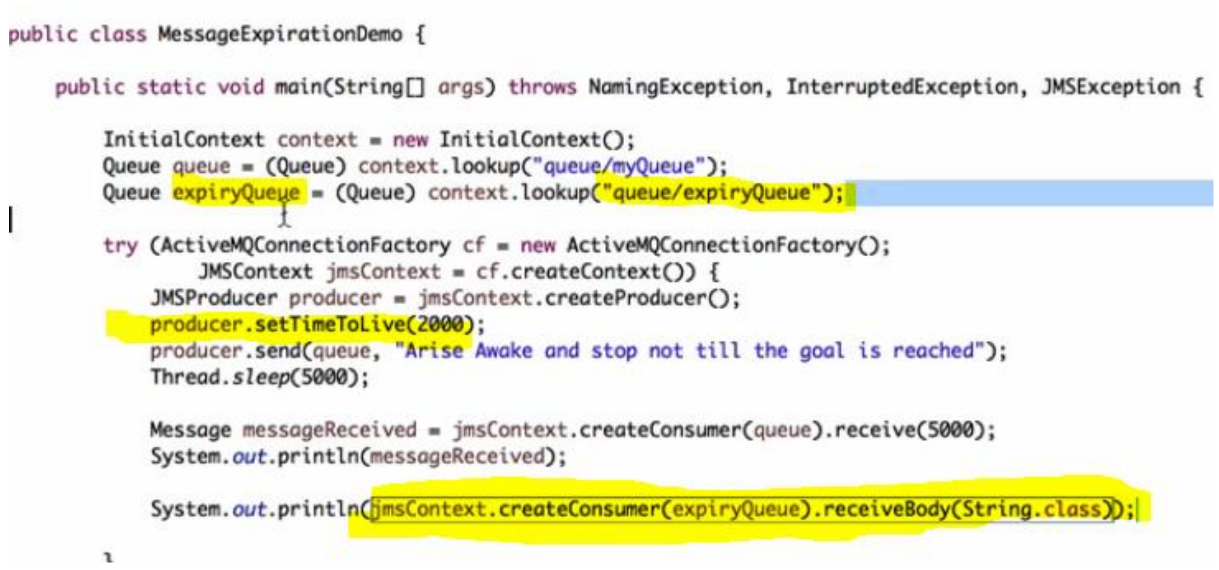
Once the message expired , u cant receive the message ,the expired msg will go to the expired queue.



```
public class MessageExpirationDemo {

    public static void main(String[] args) throws NamingException, InterruptedException {

        InitialContext context = new InitialContext();
        Queue queue = (Queue) context.lookup("queue/myQueue");

        try(ActiveMQConnectionFactory cf = new ActiveMQConnectionFactory();
                JMSContext jmsContext = cf.createContext()){
            JMSProducer producer = jmsContext.createProducer();
            producer.setTimeToLive(2000);
            producer.send(queue,"Arise Awake and stop not till the goal is reached");
            Thread.sleep(5000);

            Message messageReceived = jmsContext.createConsumer(queue).receive(5000);
            System.out.println(messageReceived);
        }

    }
```

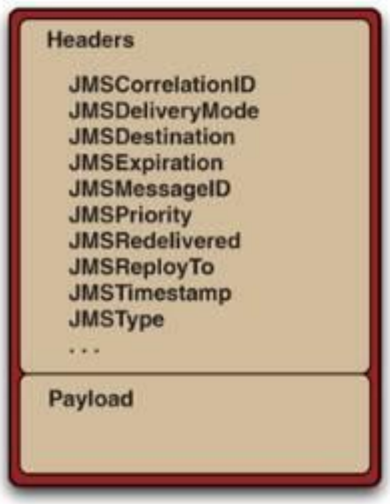
34.access expired msg

Expired messages are not lost , they were just moved to a separate queue called expiry queue, u can find the expired queue name in some xml file , and try fetching the message from there



```
public class MessageExpirationDemo {

    public static void main(String[] args) throws NamingException, InterruptedException, JMSException {

        InitialContext context = new InitialContext();
        Queue queue = (Queue) context.lookup("queue/myQueue");
        Queue expiryQueue = (Queue) context.lookup("queue/expiryQueue");

        try (ActiveMQConnectionFactory cf = new ActiveMQConnectionFactory();
                JMSContext jmsContext = cf.createContext()) {
            JMSProducer producer = jmsContext.createProducer();
            producer.setTimeToLive(2000);
            producer.send(queue, "Arise Awake and stop not till the goal is reached");
            Thread.sleep(5000);

            Message messageReceived = jmsContext.createConsumer(queue).receive(5000);
            System.out.println(messageReceived);

            System.out.println(jmsContext.createConsumer(expiryQueue).receiveBody(String.class));

        }
```

JMs Message anatomy



- JMS Message consist of mainly two parts **Headers** and **Pay-loads.**
- **Headers** consists of **metadata** of the message which is used by both clients and JMS Providers.
- **The Payload** consists of the actual body of the message (which can be binary or textual).
- The complexity of the JMS Message lies in headers.