

HU Extension
Handed out: 10/17/2015

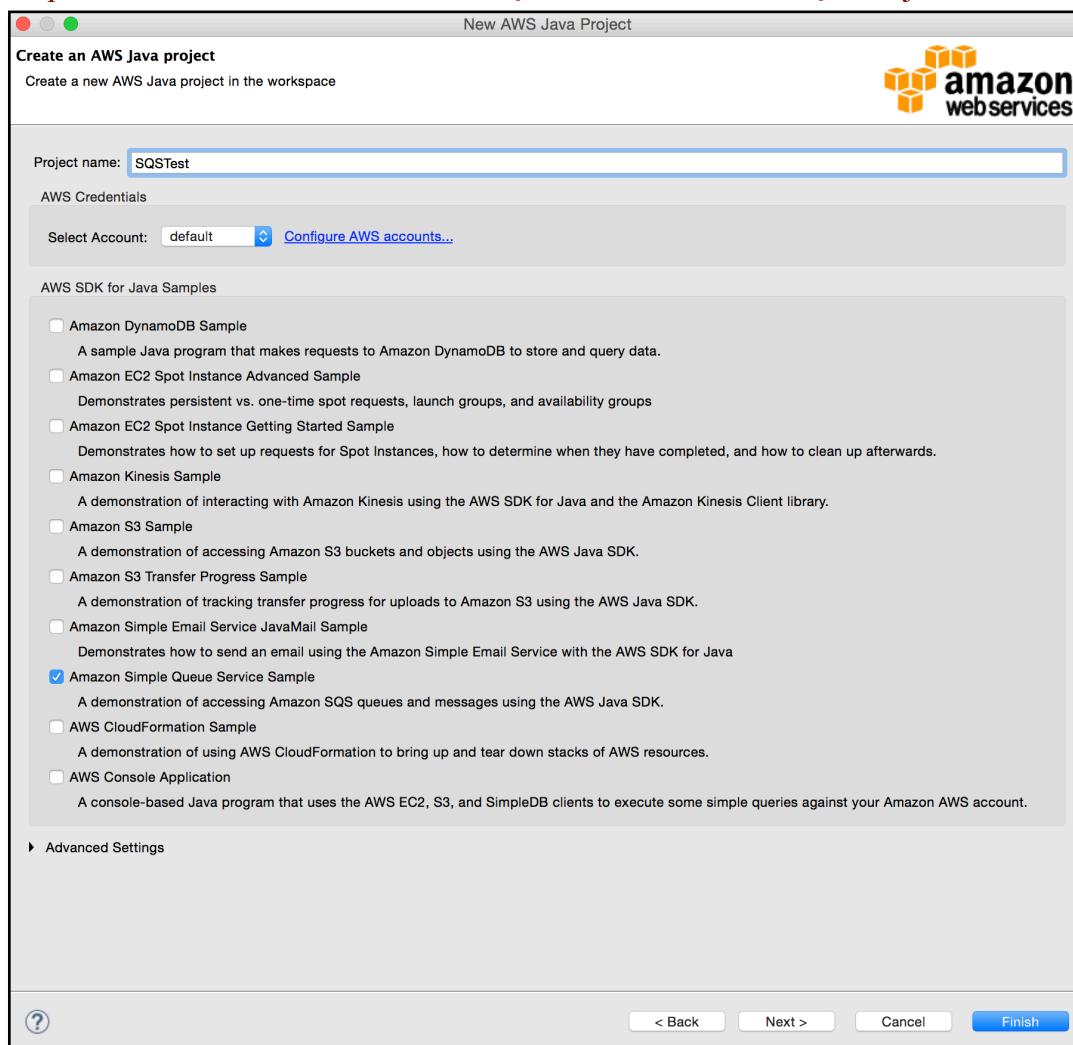
Assignment 07 E-90 Cloud Computing
Due by 11:59 PM on Friday, 10/23/2015
FINAL

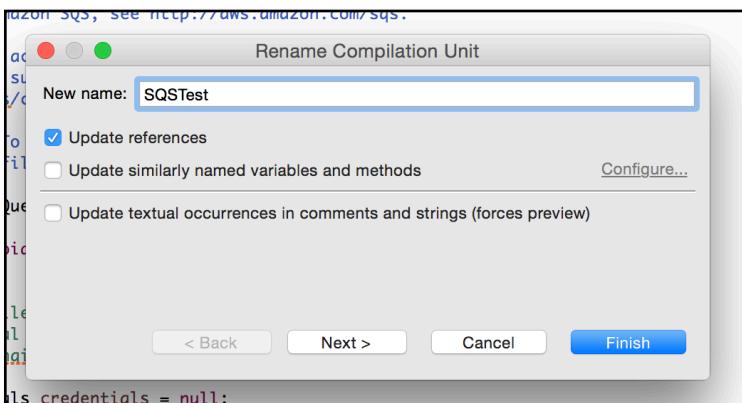
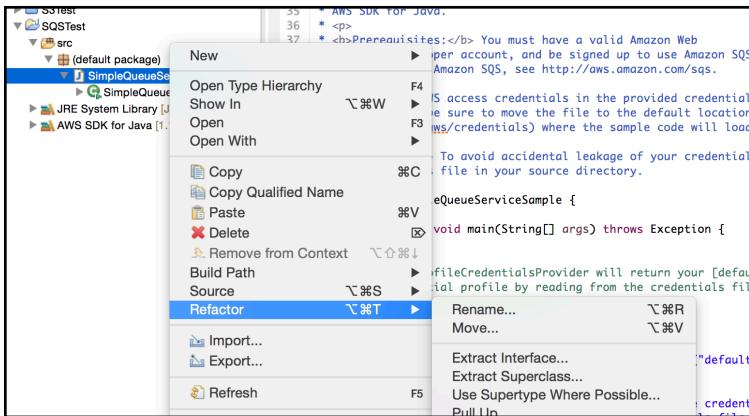
Place all of your narratives and illustrations in a single Word or PDF document named E90_LastNameFirstNameHW07.docx [.pdf]. Use this assignment as the initial template. Please implement code solutions as a separate class in one (Java, C#, Ruby, Python,...) project. Add your steps and your code below problem statements used for that problem. Upload your homework file and your working code (e.g., filename.java) into your Assignment 7 folder. Do not include executables. Please also do not zip your files.

Problem 1)

- 1) Create a queue using Amazon's SQS Service. Set the visibility timeout to 20 seconds.

Create a SQSTest project, use “Amazon Simple Queue Service Sample” as the template. Rename the class name to “SQSTest”. Java file is “SQSTest.java”.





Load the credential information. Set the region to “US_EAST_1”.

```
public static void main(String[] args) throws Exception {
    /*
     * The ProfileCredentialsProvider will return your [default]
     * credential profile by reading from the credentials file located at
     * (/Users/hqiu/.aws/credentials).
     */
    AWSredentials credentials = null;
    try {
        credentials = new ProfileCredentialsProvider("default").getCredentials();
    } catch (Exception e) {
        throw new AmazonClientException(
            "Cannot load the credentials from the credential profiles file. " +
            "Please make sure that your credentials file is at the correct " +
            "location (/Users/hqiu/.aws/credentials), and is in valid format.", e);
    }

    AmazonSQS sqs = new AmazonSQSClient(credentials);
    Region usEast1 = Region.getRegion(Regions.US_EAST_1);
    sqs.setRegion(usEast1);
}
```

Create the SQS queue using “CreateQueueRequest()”. Set the attribute “visibility timeout” to 20 seconds. Set the attribute using “addAttributeEntry()”. Get and list the QueueURL.

```

try {
    // Create a queue
    System.out.println("Creating a new SQS queue called MyQueue.\n");
    CreateQueueRequest createQueueRequest = new CreateQueueRequest("MyQueue");

    // Set the visibility timeout to 20 seconds
    createQueueRequest.addAttributesEntry("VisibilityTimeout", "20");

    String myQueueUrl = sqs.createQueue(createQueueRequest).getQueueUrl();

    // List queues
    System.out.println("Listing all queues in your account.\n");
    for (String queueUrl : sqs.listQueues().getQueueUrls()) {
        System.out.println(" QueueUrl: " + queueUrl);
    }
    System.out.println();
}

```

- 2) Insert a message and verify that you can retrieve and delete the message.

Send and receive the message. I encapsulate the receive message procedure into API called “receiveMessage()”. Thus we can reuse it later. Here I receive the message three times. The first time we can get the message. After that, the message is invisible from other users since the “Visibility Timeout” is 20 seconds. After 30 seconds, we can see and get the message again. At last, delete the message using “DeleteMessageRequest()”.

```

// Send a message
System.out.println("Sending the first message to MyQueue.\n");
sqs.sendMessage(new SendMessageRequest(myQueueUrl, "This is my first message text."));

// Receive messages
List<Message> messages = receiveMessage(sqs, myQueueUrl);

// Receive the messages again
messages = receiveMessage(sqs, myQueueUrl);

// Wait for 30 seconds until the message reappears in the queue
Thread.sleep(30000L);

// Delete the first message
messages = receiveMessage(sqs, myQueueUrl);
System.out.println("Deleting a message.\n");
String messageReceiptHandle = messages.get(0).getReceiptHandle();
sqc.deleteMessage(new DeleteMessageRequest(myQueueUrl, messageReceiptHandle));

```

```

private static List<Message> receiveMessage(AmazonSQS sqs, String myQueueUrl) {
    System.out.println("Receiving messages from MyQueue.\n");
    ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest(myQueueUrl);
    List<Message> messages = sqs.receiveMessage(receiveMessageRequest).getMessages();

    for (Message message : messages) {
        System.out.println(" Message");
        System.out.println("   MessageId: " + message.getMessageId());
        System.out.println("   ReceiptHandle: " + message.getReceiptHandle());
        System.out.println("   MD5OfBody: " + message.getMD5OfBody());
        System.out.println("   Body: " + message.getBody());
        for (Entry<String, String> entry : message.getAttributes().entrySet()) {
            System.out.println("     Attribute");
            System.out.println("       Name: " + entry.getKey());
            System.out.println("       Value: " + entry.getValue());
        }
    }
    System.out.println();
}

return messages;
}

```

```

Markers Properties Servers Data Source Explorer Snippets Problems Console
<terminated> SOSTest [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (Oct 22, 2015, 6:26:01 PM)
=====
Getting Started with Amazon SQS
=====

Creating a new SQS queue called MyQueue.

Listing all queues in your account.

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue

Sending the first message to MyQueue.

Receiving messages from MyQueue.

Message
MessageId: 25676f0d-5c9b-4c87-b865-c41d159d641d
ReceiptHandle: AQEB7VORI10CiFmojIiQZjg3hE4ZoQdqw3BEkyKXPZqCqLTSNhj/e1BaprlqlSdW/S/WtOPH1CF0JNs95Vf6ka9761Bzb9Y+D9nlu0JmhRyldqstNtQb0lc5jIpwG+kWN+0E8SX83dn
MD5OfBody: d95fd0a38c44b5459810e97212b0857e
Body: This is my first message text.

```

Check the message from the AWS console. Select “SQS” service. Check the queue in “Queues”. Select “Queue Actions” and then choose “View/Delete Messages” and click “Start Polling for Messages”.

Name	Messages Available	Messages in Flight	Created
MyQueue	1	0	2015-10-22 18:26:02 GMT-04:00

Details

Name: MyQueue
URL: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
ARN: arn:aws:sqs:us-east-1:217134905396:MyQueue
Created: 2015-10-22 18:26:02 GMT-04:00
Last Updated: 2015-10-22 18:26:02 GMT-04:00
Delivery Delay: 0 seconds

Default Visibility Timeout: 20 seconds
Message Retention Period: 4 days
Maximum Message Size: 256 KB
Receive Message Wait Time: 0 seconds
Messages Available (Visible): 1
Messages in Flight (Not Visible): 0
Messages Delayed: 0

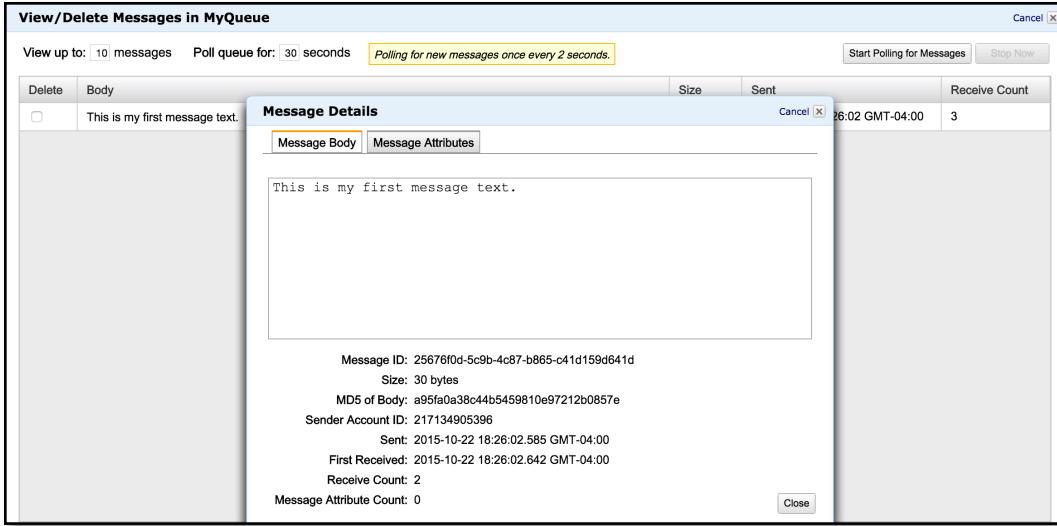
View up to: 10 messages **Poll queue for:** 30 seconds **Polling for new messages once every 2 seconds.**

Start Polling for Messages **Stop Now**

Delete	Body	Size	Sent	Receive Count
<p>View messages currently available in the queue by clicking the Start Polling for Messages button.</p> <ul style="list-style-type: none"> • Messages will come from the front of the queue unless other applications are also reading from the queue. • Messages displayed in the console will not be available to other applications until the console stops polling for messages. • The console will stop polling for messages as soon as the specified number of seconds have elapsed, the requested number of messages have been received, or the Stop Now button has been pressed. • Deleting a message from the console permanently removes it from the queue. <p><input type="checkbox"/> Don't show this again. Start Polling for Messages</p> <p>This progress bar indicates whether messages displayed above are available to applications.</p>				

Delete Messages **Close**

We successfully send the message into the queue.



Results from Eclipse console. The first time we get the message. The second time we couldn't get anything since the message is invisible. After 30 seconds, we can get the message from the queue again.

```

Markers Properties Servers Data Source Explorer Snippets Problems Console 
<terminated> SQSTest [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (Oct 23, 2015, 1:21:26 AM)
=====
Getting Started with Amazon SQS
=====

Creating a new SQS queue called MyQueue.

Listing all queues in your account.

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue

Sending the first message to MyQueue.

Receiving messages from MyQueue.

Message
MessageId: 48e8552e-dfb9-43f6-adbe-80d08104d61c
ReceiptHandle: AQEBntramZYC/xd4u98XFyzqTdbDqNQ5x8GB4vQtRA2uTcNykAkMncJFrkRhGuGj2Whi0Rcv+wIgLQcumSbZo+PAV1ZxIRA17EGhTed4LMxwU1bHAppo04vbh1JksfJhxw2pHKijSK
MD5OfBody: a95fa0a38c44b5459810e97212b0857e
Body: This is my first message text.

Receiving messages from MyQueue.

Receiving messages from MyQueue.

Message
MessageId: 48e8552e-dfb9-43f6-adbe-80d08104d61c
ReceiptHandle: AQEBntramZYC/xd4u98XFyzqTdbDqNQ5x8GB4vQtRA2uTcNykAkMncJFrkRhGuGj2Whi0Rcv+wIgLQcumSbZo+PAV1ZxIRA17EGhTed4LMxwU1bHAppo04vbh1JksfJhxw2pHKijSK
MD5OfBody: a95fa0a38c44b5459810e97212b0857e
Body: This is my first message text.

Deleting a message.

```

- 3) Subsequently insert another message, and retrieve that second message. Do not delete the second message but rather wait for 30 seconds (use Java `Thread.sleep()` or whatever you find suitable for that purpose).
- 4) Show that the message reappeared in the queue.

We have done the same thing in step 2). Here we receive the message three times. The first time we can retrieve the message. Then the message is invisible from other users and we couldn't retrieve anything, until it passes the “visibility timeout”. After 30 seconds, it passes the “visibility timeout” and the message reappeared in the queue. In the AWS console, there are two columns “Message Available” and “Message in Flight”. The “Message in Flight” will record the number of invisible messages.

```
// Send another message
System.out.println("Sending the second message to MyQueue.\n");
sqc.sendMessage(new SendMessageRequest(myQueueUrl, "This is my second message text."));

// Retrieve the second message
messages = receiveMessage(sqc, myQueueUrl);

// Retrieve the second message again
messages = receiveMessage(sqc, myQueueUrl);

// Wait for 30 seconds, receive the messages again
Thread.sleep(30000L);
messages = receiveMessage(sqc, myQueueUrl);
```

```
Markers Properties Servers Data Source Explorer Snippets Problems Console <terminated> SQSTest [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (Oct 23, 2015, 1:21:26 AM)
Sending the second message to MyQueue.

Receiving messages from MyQueue.

Message
MessageId: 92ed4b16-3f9b-4ac6-87f9-4da680011140
ReceiptHandle: AQEBzF2indmAy8eQInm0-SY-BtvWeRQuqe9Hzv3gKKubZi6SFxu5KaDFePwtfMs8T4ylvr5EM5Qmfv6t1NEQjYKYkaryls+cxauQCLeE40IfJS07KSSk/rRtGembG0+Q4oMDaw6u9yzz
MD5OfBody: 8e881d3de27f22d57b6ec75028cab1ac
Body: This is my second message text.

Receiving messages from MyQueue.

Receiving messages from MyQueue.

Message
MessageId: 92ed4b16-3f9b-4ac6-87f9-4da680011140
ReceiptHandle: AQEBzF2indmAy8eQInm0-KAVN/Ze4xckgsnL86wL14cakv1o1CFPIUb00RS9hJ3yv6qRuya3AmKh8XNtlhe3rsHx8ISxnBxYu2w3mRpggM9Pvpch2S+xy3sHUuaBB7VPG/pAJNGM1SENnaR
MD5OfBody: 8e881d3de27f22d57b6ec75028cab1ac
Body: This is my second message text.

Sending five messages to MyQueue.
```

- 5) Add five more messages to the queue and then execute `GetQueueAttributes` call. Retrieve and display all attributes of your queue. In particular report on the average number of messages in the queue.

Add five more messages.

```
// Send five more messages to the queue
System.out.println("Sending five messages to MyQueue.\n");
for (int i = 1; i <= 5; i++) {
    String messageStr = "This is my " + i + "/5 message text.";
    sqs.sendMessage(new SendMessageRequest(myQueueUrl, messageStr));
}

// Receive all the messages
// messages = receiveMessage(sqs, myQueueUrl);
// Thread.sleep(30000L);
```

Get all attributes of the queue, especially report on the approximate number of messages. Set the parameters in “withAttributeNames()” to “All” to get all the attributes. We can also set it to a particular attribute name.

```
// Retrieve all attributes of the queue.
GetQueueAttributesRequest getQAResult = new GetQueueAttributesRequest().withQueueUrl(myQueueUrl);
GetQueueAttributesResult getQAResult = sqs.getQueueAttributes(getQAResult.withAttributeNames("All"));
Map <String, String> attributeMap = getQAResult.getAttributes();

// Display all attributes, especially the average number of messages in the queue.
System.out.println();
System.out.println("Number of messages in queue " + myQueueUrl + " is " +
    attributeMap.get("ApproximateNumberOfMessages"));

System.out.println("Number of invisible messages in queue is " +
    attributeMap.get("ApproximateNumberOfMessagesNotVisible"));

System.out.println("\nDisplay all the attributes:");
for (String key : attributeMap.keySet()) {
    System.out.println(" " + key + " is: " + attributeMap.get(key));
}
```

We add 7 messages and delete 1 before. So we get 6 messages here.

The screenshot shows two windows. The top window is titled 'Queues' and lists a single queue named 'MyQueue'. It shows 6 messages available, 0 messages in flight, and was created on 2015-10-23 01:15:14 GMT-04:00. The bottom window is titled 'View/Delete Messages in MyQueue' and displays 6 messages in the queue. Each message has a checkbox labeled 'Delete', a 'Body' column containing text snippets, and columns for 'Size', 'Sent', and 'Receive Count'.

Delete	Body	Size	Sent	Receive Count
<input type="checkbox"/>	This is my second message text.	31 bytes	2015-10-23 01:15:45 GMT-04:00	3
<input type="checkbox"/>	This is my 2/5 message text.	28 bytes	2015-10-23 01:16:15 GMT-04:00	1
<input type="checkbox"/>	This is my 1/5 message text.	28 bytes	2015-10-23 01:16:15 GMT-04:00	1
<input type="checkbox"/>	This is my 3/5 message text.	28 bytes	2015-10-23 01:16:15 GMT-04:00	1
<input type="checkbox"/>	This is my 4/5 message text.	28 bytes	2015-10-23 01:16:15 GMT-04:00	1
<input type="checkbox"/>	This is my 5/5 message text.	28 bytes	2015-10-23 01:16:15 GMT-04:00	2

- 6) Finally, delete the queue.

Do all of this using either Java, C# or some other AWS API you find convenient.

```
// Delete a queue
System.out.println("\nDeleting the test queue.\n");
sqc.deleteQueue(new DeleteQueueRequest(myQueueUrl));
```

All results from Eclipse console.

The screenshot shows the Eclipse IDE's Console view with the following output:

```
<terminated> SQSTest [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (Oct 23, 2015, 1:21:26 AM)
=====
Getting Started with Amazon SQS
=====

Creating a new SQS queue called MyQueue.

Listing all queues in your account.

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue

Sending the first message to MyQueue.

Receiving messages from MyQueue.

Message
MessageId: 48e8552e-dfb9-43f6-adbe-80d08104d61c
ReceiptHandle: AQEBrLtramZYC/xdu498XFeyzqTdbTDqNQSx8GB4vQtRA2uTcNyAkMncJFrkRhGuGJ2Whi0Rcv+wIgLQcumSbZo+PAV1ZxIRAl7EGHrTed4LMxwU1bHAppo04vbhtlJksfJhxwx2pHKIjk5
MD5OfBody: 095fa0a38c44b5459810e97212b0857e
Body: This is my first message text.

Receiving messages from MyQueue.

Receiving messages from MyQueue.

Message
MessageId: 48e8552e-dfb9-43f6-adbe-80d08104d61c
ReceiptHandle: AQEBr8a53g4a25L6wl06HPAD274chfIJQdDQ21p5TuIXIkV5k18NN9wrCa0G5T2qv2SFumG84UTcm1jFUA3txn4wAlkxUBLkEDr3vjCdqKI8Vh2zm8swXISexm+A+xMDTP9P/JNtYisaTrUx
MD5OfBody: 095fa0a38c44b5459810e97212b0857e
Body: This is my first message text.

Deleting a message.
```

```

Markers Properties Servers Data Source Explorer Snippets Problems Console 
<terminated> SQSTest [Java Application] /Library/Java/JavaVirtualMachines/dk1.8.0_45.jdk/Contents/Home/bin/java (Oct 23, 2015, 1:21:26 AM)
Sending the second message to MyQueue.

Receiving messages from MyQueue.

Message
MessageID: 92ed4b16-3f9b-4ac6-87f9-4da680011140
ReceiptHandle: AQBz2Mh10Vm/AM0zr0KAVN/Ze4xkCgsn18wL14cakv1o1CFPIUb00RS9hJ3yv6qRuya3AmKnh8XNt1he3rsHx8IS5xnBxYu2w3mRpggM9PvpcTH2S+xy3sHUuaBB7VPG/pAJNGM1SENmR
MD5OfBody: 8e881d3de27f22d57b6ec75028ca1bac
Body: This is my second message text.

Receiving messages from MyQueue.

Receiving messages from MyQueue.

Message
MessageID: 92ed4b16-3f9b-4ac6-87f9-4da680011140
ReceiptHandle: AQBz2Mh10Vm/AM0zr0KAVN/Ze4xkCgsn18wL14cakv1o1CFPIUb00RS9hJ3yv6qRuya3AmKnh8XNt1he3rsHx8IS5xnBxYu2w3mRpggM9PvpcTH2S+xy3sHUuaBB7VPG/pAJNGM1SENmR
MD5OfBody: 8e881d3de27f22d57b6ec75028ca1bac
Body: This is my second message text.

Sending five messages to MyQueue.

Receiving messages from MyQueue.

Message
MessageID: 6aa9689d-454f-429a-9268-263b56db2d29
ReceiptHandle: AQBQUuwk3XVm/cjkjMhCK4jvYE+DQ4rkDL19mpZKd0Trfv8bLrTG5XAV5E0iFHKJd1CRYbgS6iFmt0G7nDR1MKkZgHxk6Vb0ISngbn3XVK9ybWW+HuI6PK1L4VDiju+fx1HyaryDT95RbdRW
MD5OfBody: 0e987b2596eed8628d51761ad343dc2a
Body: This is my 1/5 message text.

Number of messages in queue https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue is 6
Number of invisible messages in queue is 0

```

```

Markers Properties Servers Data Source Explorer Snippets Problems Console 
<terminated> SQSTest [Java Application] /Library/Java/JavaVirtualMachines/dk1.8.0_45.jdk/Contents/Home/bin/java (Oct 23, 2015, 1:31:02 AM)
Number of messages in queue https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue is 6
Number of invisible messages in queue is 0

Display all the attributes:
ApproximateNumberOfMessagesDelayed is: 0
ReceiveMessageWaitTimeSeconds is: 0
CreatedTimestamp is: 1445578263
DelaySeconds is: 0
MessageRetentionPeriod is: 345600
MaximumMessageSize is: 262144
VisibilityTimeout is: 20
ApproximateNumberOfMessages is: 6
ApproximateNumberOfMessagesNotVisible is: 0
LastModifiedTimestamp is: 1445578263
QueueArn is: arn:aws:sqs:us-east-1:217134905396:MyQueue

Deleting the test queue.

```

Points: 25

Problem 2)

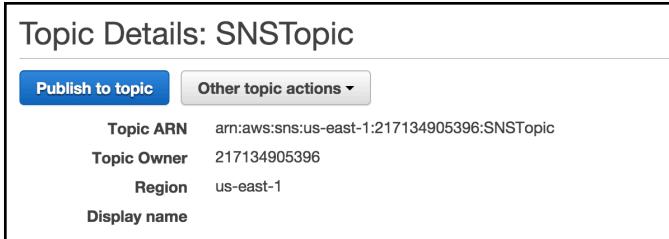
- 1) Create an SNS topic.

Use AWS CLI “create-topic”. Verify from the AWS console at the same time.

```

hqi@bos-mpdei>> aws sns create-topic --name SNSTopic
{
    "TopicArn": "arn:aws:sns:us-east-1:217134905396:SNSTopic"
}
hqi@bos-mpdei>> aws sns list-topics
{
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-east-1:217134905396:SNSTopic"
        }
    ]
}

```



Topics		
	Publish to topic	Create new topic
Filter		Actions ▾
	Name	ARN
<input type="checkbox"/>	SNSTopic	arn:aws:sns:us-east-1:217134905396:SNSTopic

- 2) Subscribe one of your emails and one of your phones.

To subscribe the phone, we should set the “DisplayName” of the queue in advance.

```
hqiu@bos-mpdei> aws sns set-topic-attributes --topic-arn arn:aws:sns:us-east-1:217134905396:SNSTopic --attribute-name DisplayName --attribute-value SNSTopicHQ
hqiu@bos-mpdei> aws sns get-topic-attributes --topic-arn arn:aws:sns:us-east-1:217134905396:SNSTopic
{
    "Attributes": {
        "SubscriptionsConfirmed": "0",
        "DisplayName": "SNSTopicHQ",
        "SubscriptionsDeleted": "0",
        "EffectiveDeliveryPolicy": "{\"http\":{\"defaultHealthyRetryPolicy\":{\"minDelayTarget\":20,\"maxDelayTarget\":20,\"numRetries\":3,\"numMaxDelayRetries\":0,\"numNoDelayRetries\":0,\"numMinDelayRetries\":0},\"backoffFunction\":\"linear\"},\"disableSubscriptionOverrides\":false}}",
        "Owner": "217134905396",
        "Policy": "{\"Version\":\"2008-10-17\",\"Id\":\"__default_policy_ID\",\"Statement\":[{\"Sid\":\"__default_statement_ID\",\"Effect\":\"Allow\",\"Principal\":\"*\",\"Action\":[\"SNS:Subscribe\",\"SNS>ListSubscriptionsByTopic\",\"SNS>DeleteTopic\",\"SNS:GetTopicAttributes\",\"SNS:Publish\",\"SNS:RemovePermission\",\"SNS>AddPermission\",\"SNS:Receive\",\"SNS:SetTopicAttributes\"],\"Resource\":\"arn:aws:sns:us-east-1:217134905396:SNSTopic\", \"Condition\":{\"StringEquals\":{\"AWS:SourceOwner\":\"217134905396\"}}}],\"TopicArn\": \"arn:aws:sns:us-east-1:217134905396:SNSTopic\",
        "SubscriptionsPending": "0"
    }
}
```

Check from the AWS console. Select SNS service. Navigate to the display name through “Topics->Actions->Edit topic display name”.

Topic Details: SNSTopic

Edit display name

The topic display name is required for SMS subscriptions. It is also present in the "From:" field of notifications sent to the given topic.

Display name	SNSTopicHQ	<small>i</small>
--------------	------------	------------------

[Cancel](#) [Set display name](#)

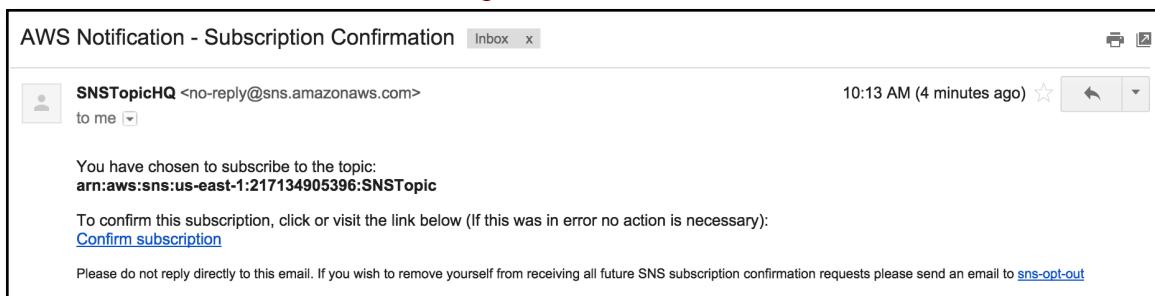
Subscribe email and phone. For mobile phone, the protocol is “sms”. For email, the protocol is “email”.

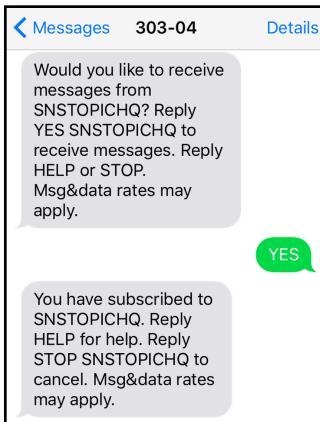
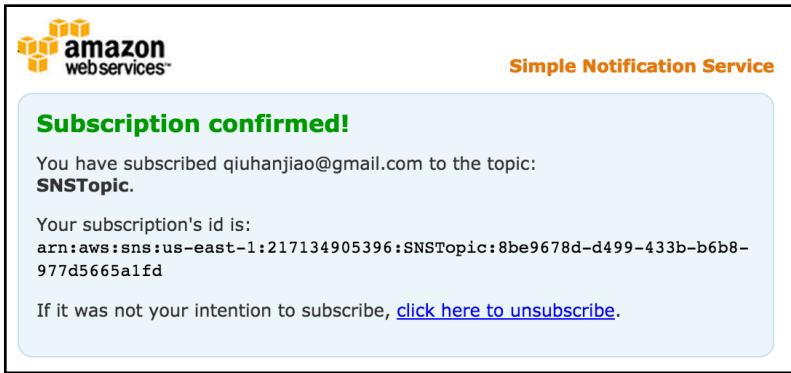
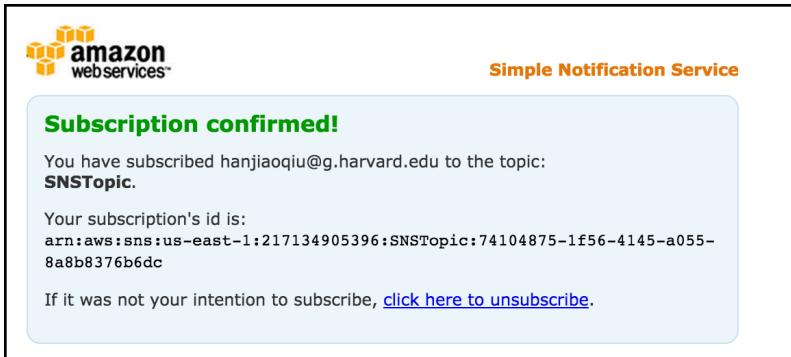
```
hqiu@bos-mpdei>> aws sns set-topic-attributes --topic-arn arn:aws:sns:us-east-1:217134905396:SNSTopic --attribute-name DisplayName --attribute-value SNSTopicHQ
hqiu@bos-mpdei>> aws sns subscribe --topic-arn arn:aws:sns:us-east-1:217134905396:SNSTopic --protocol email --notification-endpoint hanjiaoqiu@g.harvard.edu
{
    "SubscriptionArn": "pending confirmation"
}
hqiu@bos-mpdei>> aws sns subscribe --topic-arn arn:aws:sns:us-east-1:217134905396:SNSTopic --protocol email --notification-endpoint qiuhanjiao@gmail.com
{
    "SubscriptionArn": "pending confirmation"
}
hqiu@bos-mpdei>> aws sns subscribe --topic-arn arn:aws:sns:us-east-1:217134905396:SNSTopic --protocol sms --notification-endpoint 1-617-955-7630
{
    "SubscriptionArn": "pending confirmation"
}
```

```
hqiu@bos-mpdei>> aws sns list-subscriptions
{
    "Subscriptions": [
        {
            "Owner": "217134905396",
            "Endpoint": "16179557630",
            "Protocol": "sms",
            "TopicArn": "arn:aws:sns:us-east-1:217134905396:SNSTopic",
            "SubscriptionArn": "PendingConfirmation"
        },
        {
            "Owner": "217134905396",
            "Endpoint": "hanjiaoqiu@g.harvard.edu",
            "Protocol": "email",
            "TopicArn": "arn:aws:sns:us-east-1:217134905396:SNSTopic",
            "SubscriptionArn": "PendingConfirmation"
        },
        {
            "Owner": "217134905396",
            "Endpoint": "qiuhanjiao@gmail.com",
            "Protocol": "email",
            "TopicArn": "arn:aws:sns:us-east-1:217134905396:SNSTopic",
            "SubscriptionArn": "PendingConfirmation"
        }
    ]
}
```

- 3) Confirm subscription to the topic manually from both the email and the phone.

Go to mailbox, click “confirm subscription”.





We can also check the subscribers from AWS console. Select “SNS home->Subscriptions”. After we confirm the subscription, the subscription ID of each account is showed below. It corresponds to the subscription’s ID during the confirmation. For example, the subscription’s ID for qiuhanjiao@gmail.com is “arn:aws:sns:us-east-1:217134905396:SNSTopic:8be9678d-d499-433b-b6b8-977d5665a1fd”.

Topic Details: SNSTopic

Publish to topic		Other topic actions ▾	
Topic ARN	arn:aws:sns:us-east-1:217134905396:SNSTopic	Topic Owner	217134905396
Region	us-east-1	Display name	SNSTopicHQ

Subscriptions

Create Subscription		Request confirmations		Confirm Subscription		Other Subscription Actions ▾			
Filter									
<input type="checkbox"/>	Subscription ID	<input type="checkbox"/>	Protocol	<input type="checkbox"/>	Endpoint	<input type="checkbox"/>	Subscriber	<input type="checkbox"/>	
<input type="checkbox"/>	arn:aws:sns:us-east-1:217134905396:SNSTopic:74104875-1f56-4145-a055-8a8b8376b6dc	<input type="checkbox"/>	email	<input type="checkbox"/>	hanjiaoqiu@g.harvard.edu	<input type="checkbox"/>	217134905396	<input type="checkbox"/>	
<input type="checkbox"/>	arn:aws:sns:us-east-1:217134905396:SNSTopic:8be9678d-d499-433b-b6b8-977d5665a1fd	<input type="checkbox"/>	email	<input type="checkbox"/>	qiuhanjiao@gmail.com	<input type="checkbox"/>	217134905396	<input type="checkbox"/>	
<input type="checkbox"/>	arn:aws:sns:us-east-1:217134905396:SNSTopic:e3d88388-0c43-4c42-bdf5-43d862da3bf6	<input type="checkbox"/>	sms	<input type="checkbox"/>	16179557630	<input type="checkbox"/>	217134905396	<input type="checkbox"/>	

- 4) Demonstrate that you can send the same message to both media.

Send two messages to all the media, using AWS CLI “aws sns publish”.

```
hqiu@bos-mpdei>> aws sns publish --topic-arn arn:aws:sns:us-east-1:217134905396:SNSTopic --subject "Most Recent News From Hanjiao" --message "This is a message sent from Hanjiao"
{
    "MessageId": "d00da64f-e098-5838-9cc6-e8eba4d607ca"
}
hqiu@bos-mpdei>> aws sns publish --topic-arn arn:aws:sns:us-east-1:217134905396:SNSTopic --subject "Most Recent News From Hanjiao" --message "This is a second message sent from Hanjiao"
{
    "MessageId": "3fee6571-07c9-57d4-9de9-2bd543ba4fc4"
}
```

- 5) Capture your actions on the email side.

Get the first message “This is a message sent from Hanjiao”.

Most Recent News From Hanjiao

Inbox

SNSTopicHQ <no-reply@sns.amazonaws.com> to me 10:19 AM (0 minutes ago)

This is a message sent from Hanjiao

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
<https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:217134905396:SNSTopic:8be9678d-d499-433b-b6b8-977d5665a1fd&Endpoint=qiuhanjiao@gmail.com>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at
<https://aws.amazon.com/support>

Get the second message “This is a second message sent from Hanjiao”.



- 6) Capture all subscription arn-s and all message arn-s.

List all subscriptions.

```
hqiu@bos-mpdei:> aws sns list-subscriptions
{
  "Subscriptions": [
    {
      "Owner": "217134905396",
      "Endpoint": "16179557630",
      "Protocol": "sms",
      "TopicArn": "arn:aws:sns:us-east-1:217134905396:SNSTopic",
      "SubscriptionArn": "arn:aws:sns:us-east-1:217134905396:SNSTopic:e3d88388-0c43-4c42-bdf5-43d862da3bf6"
    },
    {
      "Owner": "217134905396",
      "Endpoint": "hanjiaoqiu@g.harvard.edu",
      "Protocol": "email",
      "TopicArn": "arn:aws:sns:us-east-1:217134905396:SNSTopic",
      "SubscriptionArn": "arn:aws:sns:us-east-1:217134905396:SNSTopic:74104875-1f56-4145-a055-8a8b8376b6dc"
    },
    {
      "Owner": "217134905396",
      "Endpoint": "qiuhanjiao@gmail.com",
      "Protocol": "email",
      "TopicArn": "arn:aws:sns:us-east-1:217134905396:SNSTopic",
      "SubscriptionArn": "arn:aws:sns:us-east-1:217134905396:SNSTopic:8be9678d-d499-433b-b6b8-977d5665a1fd"
    }
  ]
}
```

Check from AWS console.

Subscriptions				
		Create Subscription	Request confirmations	Actions ▾
<input type="button" value="Filter"/> <input type="text"/>				
Subscription ARN	Protocol	Endpoint	Topic ARN	
arn:aws:sns:us-east-1:217134905396:SNSTopic:e3d88388-0c43-4c42-bdf5-43d862da3bf6	sms	16179557630	arn:aws:sns:us-east-1:217134905396:SNSTopic	
arn:aws:sns:us-east-1:217134905396:SNSTopic:74104875-1f56-4145-a055-8a8b8376b6dc	email	hanjiaoqiu@g.harvard.edu	arn:aws:sns:us-east-1:217134905396:SNSTopic	
arn:aws:sns:us-east-1:217134905396:SNSTopic:8be9678d-d499-433b-b6b8-977d5665a1fd	email	qiuhanjiao@gmail.com	arn:aws:sns:us-east-1:217134905396:SNSTopic	

- 7) [Optional] if you can, take a picture of your SMS message on your phone – or forward it to your email and demo it that way.
 Please, perform all tasks using AWS CLI. Only if AWS CLI really could not do the work (you must explain) you are allowed to use the AWS Management Console.

Get the message from mobile phone.



In the above graph, the first two messages only show the subject name. They haven't shown the content. The last message shows the content. I get it from the AWS console following the steps below. The message has to be the JSON format.

Select the topic, click on "Publish to topic". Enter the subject name, then click on "JSON" message generator. Only select "SMS" like below. Click "Generate JSON" after finish.

JSON message generator

The JSON message generator tool allows you to convert your messages to the appropriate JSON format.

Message

Message from console = Sky is blue.

Email SQS Lambda
 HTTP HTTPS SMS

Target Platforms

<input type="checkbox"/> iOS Prod	<input checked="" type="checkbox"/> iOS Dev	<input checked="" type="checkbox"/> VoIP Prod
<input checked="" type="checkbox"/> VoIP Dev	<input checked="" type="checkbox"/> MacOS Prod	<input checked="" type="checkbox"/> MacOS Dev
<hr/>		
<input checked="" type="checkbox"/> Android	<input checked="" type="checkbox"/> Amazon FireOS	<input checked="" type="checkbox"/> Baidu
<input checked="" type="checkbox"/> Windows MPNS <input checked="" type="checkbox"/> Windows 8.1+		

Cancel **Generate JSON**

Publish a message

Amazon SNS enables you to publish notifications to all subscriptions associated with a topic as well as to an individual endpoint associated with a platform application.

Topic ARN	arn:aws:sns:us-east-1:217134905396:SNSTopic	?
Subject	Most Recent News From Hanjiao	?
Message format	<input type="radio"/> Raw <input checked="" type="radio"/> JSON	
Message	<pre>{ "default": "Message from console = Sky is blue.", "sms": "Message from console = Sky is blue.", "APNS": "{\"aps\":{\"alert\":\"Message from console = Sky is blue.\\"}}", "APNS_SANDBOX": "{\"aps\":{\"alert\":\"Message from console = Sky is blue.\\"}}", "APNS_VOIP": "{\"aps\":{\"alert\":\"Message from console = Sky is blue.\\"}}", "APNS_VOIP_SANDBOX": "{\"aps\":{\"alert\":\"Message from console = Sky is blue.\\"}}", "MACOS": "{\"aps\":{\"alert\":\"Message from console = Sky is blue.\\"}}", "MACOS_SANDBOX": "{\"aps\":{\"alert\":\"Message from console = Sky is blue.\\"}}", "GCM": "{\"data\": {\"message\": \"Message from console = Sky is blue.\"}}", "ADM": "{\"data\": {\"message\": \"Message from console = Sky is blue.\"}}", "BAIDU": "{\"title\":\"Message from console = Sky is blue.\",\"description\":\"Message from console = Sky is blue.\\"}", "MPNST": "<?xml version='1.0' encoding='utf-8'?><wp:Notification xmlns:wp='WPNotification'><wp:Title><wp:Count>ENTER COUNT</wp:Count><wp:Title>Message from console = Sky is blue.</wp:Title></wp:Notification>", "WNS": "<badge version='1' value='23' />" }</pre>	
JSON message generator		
Time to live (TTL)	?	
		Cancel Publish message

Publish the message, we will receive the content on our phone.

Unsubscribe the email and phone.

```
hqiu@bos-mpdei:> aws sns unsubscribe --subscription-arn arn:aws:sns:us-east-1:217134905396:SNSTopic:e3d88388-0c43-4c42-bdf5-43d862da3bf6
hqiu@bos-mpdei:> aws sns unsubscribe --subscription-arn arn:aws:sns:us-east-1:217134905396:SNSTopic:74104875-1f56-4145-a055-8a8b8376b6dc
hqiu@bos-mpdei:> aws sns unsubscribe --subscription-arn arn:aws:sns:us-east-1:217134905396:SNSTopic:8be9678d-d499-433b-b6b8-977d5665a1fd
```

Subscriptions			
Create Subscription Request confirmations Actions ▾ ?			
<input style="width: 100px; margin-right: 10px;" type="text"/> Filter			
Subscription ARN	Proto...	Endpoint	Topic ARN

Delete the topic.

```
hqiu@bos-mpdei:> aws sns delete-topic --topic-arn arn:aws:sns:us-east-1:217134905396:SNSTopic
```

Points: 15

Problem 3)

Do the same work as in Problem 2, but use Java/C# or any other programming language supported by SNS this time.

Attached files: CreateTopic.java and Publish.java have practically all elements you need.

Source code: CreateTopic.java, Publish.java, OptIn.java.

```

public static void main(String[] args) throws Exception {
    AWSCredentials credentials = null;
    try {
        credentials = new ProfileCredentialsProvider("default").getCredentials();
    } catch (Exception e) {
        throw new AmazonClientException(
            "Cannot load the credentials from the credential profiles file. " +
            "Please make sure that your credentials file is at the correct " +
            "location (/Users/hqiu/.aws/credentials), and is in valid format.", e);
    }
}

```

1) Create a “SNS Topic” named “MyTopicHQ”. Get the “Topic Arn”.

```

AmazonSNS sns = new AmazonSNSClient(credentials);
try {
    // Create a Topic
    System.out.println("Creating MyTopicHQ.\n");
    CreateTopicRequest createTopicRequest = new CreateTopicRequest().withName("MyTopicHQ");

    // Retrieve Amazon Resource Name
    String myTopicArn = sns.createTopic(createTopicRequest).getTopicArn();
    System.out.println("Topic created: " + myTopicArn);

    Thread.sleep(1000);
    // List Topics
    System.out.println("List topics: ");
    for (Topic topic : sns.listTopics().getTopics()) {
        System.out.println("TopicArn: " + topic.getTopicArn());
    }
}

```

2) Subscribe the emails and phones.

```

// Set Topic Name for subscribing my phone
String topicName = "TopicHQ";
System.out.println("\nSet topic name: " + topicName + "\n");
SetTopicName(sns, myTopicArn, topicName);

// Subscribe my email and phone to MyTopicHQ
subscribeEmail(sns, myTopicArn, "hanjiaoqiu@g.harvard.edu");
subscribeEmail(sns, myTopicArn, "qiuhanjiao@gmail.com");
subscribePhone(sns, myTopicArn, "1-617-955-7630");

```

Set the “DisplayName” needed by subscribing phones.

```

private static void SetTopicName(AmazonSNS sns, String myTopicArn, String topicName) {
    SetTopicAttributesRequest setTARequest = new SetTopicAttributesRequest().withTopicArn(myTopicArn);
    setTARequest.withAttributeName("DisplayName").setAttributeValue(topicName);
    sns.setTopicAttributes(setTARequest);
}

```

The protocol to subscribe emails is “email”. The protocol to subscribe phone is “sms”.

```

private static void subscribeEmail(AmazonSNS sns, String myTopicArn, String email) {
    SubscribeRequest subReq = new SubscribeRequest(myTopicArn, "email", email);
    SubscribeResult subRes = sns.subscribe(subReq);
    String subscribedTopicArn = subRes.getSubscriptionArn();
    System.out.println("Subscribed " + email + " to topic: " + subscribedTopicArn);
}

private static void subscribePhone(AmazonSNS sns, String myTopicArn, String phone) {
    SubscribeRequest subReq = new SubscribeRequest(myTopicArn, "sms", phone);
    SubscribeResult subRes = sns.subscribe(subReq);
    String subscribedTopicArn = subRes.getSubscriptionArn();
    System.out.println("Subscribed " + phone + " to topic: " + subscribedTopicArn);
}

```

Display all the attributes of the SNS Topic.

```
// Fetch Topic attributes
System.out.println("\nTopic attributes: ");
fetchAttr(sns, myTopicArn);

private static void fetchAttr(AmazonSNS sns, String myTopicArn) {
    GetTopicAttributesRequest getTAResult = new GetTopicAttributesRequest().withTopicArn(myTopicArn);
    GetTopicAttributesResult getTAResult = sns.getTopicAttributes(getTAResult);
    Map<String, String> attributes = new HashMap<String, String>();
    attributes = getTAResult.getAttributes();
    for (String key : attributes.keySet()) {
        System.out.println(key + ": " + attributes.get(key));
    }
}
```

The results from Eclipse console. We have three pending subscriptions. And it also shows we've unsubscribed three users in previous step.

The screenshot shows the Eclipse IDE's Console view. The output text is as follows:

```
<terminated> CreateTopic [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (Oct 23, 2015, 11:04:15 AM)
Creating MyTopicHQ.

Topic created: arn:aws:sns:us-east-1:217134905396:MyTopicHQ
List topics:
TopicArn: arn:aws:sns:us-east-1:217134905396:MyTopicHQ

Set topic name: TopicHQ

Subscribed hanjiaoqiu@g.harvard.edu to topic: pending confirmation
Subscribed qiuhanjiao@gmail.com to topic: pending confirmation
Subscribed 1-617-955-7630 to topic: pending confirmation

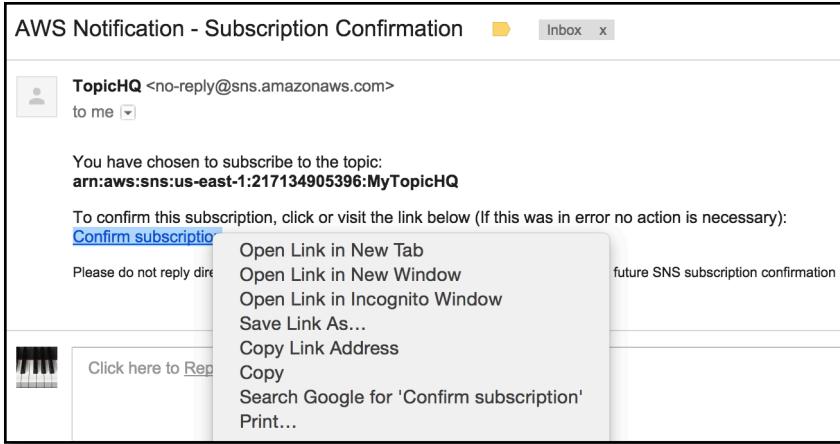
Topic attributes:
Policy: {"Version": "2008-10-17", "Id": "__default_policy_ID", "Statement": [{"Sid": "__default_statement_ID", "Effect": "Allow", "Principal": {"AWS": "*"}, "Action": ["SNS:SubOwner": 217134905396
SubscriptionsPending: 3
TopicArn: arn:aws:sns:us-east-1:217134905396:MyTopicHQ
EffectiveDeliveryPolicy: {"http": {"defaultHealthyRetryPolicy": {"minDelayTarget": 20, "maxDelayTarget": 20, "numRetries": 3, "numMaxDelayRetries": 0, "numNoDelayRetries": 0
SubscriptionsConfirmed: 0
DisplayName: TopicHQ
SubscriptionsDeleted: 3
```

Check from the AWS console.

The screenshot shows the AWS SNS Topic Details page for 'MyTopicHQ'. The 'Topic Details' section shows the ARN, owner, region, and display name. The 'Subscriptions' section lists three pending confirmations:

Subscription ID	Protocol	Endpoint	Subscriber
PendingConfirmation	email	hanjiaoqiu@g.harvard.edu	217134905396
PendingConfirmation	sms	16179557630	217134905396
PendingConfirmation	email	qiuhanjiao@gmail.com	217134905396

- 3) Confirm subscription. Get the token from the confirmation email. Right click the “Confirm subscription”, choose “Copy Link Address”. Then we will get a string. Copy the “tokenID” field out and paste that into our code.



Confirm the subscription.

```

try {
    String topicArn = "arn:aws:sns:us-east-1:217134905396:MyTopicHQ";
    // Endpoint=giuhanjiao@gmail.com
    String token = "2336412f37fb687f5d51e6e241d7700bdeeb300d6040051d539adc048f86c09b33fe3d34f6ce360db1f7390c59c4481fe1db4cde12e
    // Endpoint=hanjiaoqiu@g.harvard.edu
    // String token = "2336412f37fb687f5d51e6e241d7700bdeeb300d6040041e70e47fd455e1b00a6bf96ce9758e95ecfa10f1a59287405fc8558803
    ConfirmSubscriptionResult conSubRes = sns.confirmSubscription(
        new ConfirmSubscriptionRequest(topicArn, token));
    String subscribedTopicArn = conSubRes.getSubscriptionArn();
}

// List Topic subscriptions
ListSubscriptionsResult listSubResult = sns.listSubscriptions();
List<Subscription> subscriptions = listSubResult.getSubscriptions();
for (Subscription sub : subscriptions) {
    System.out.println(sub.getEndpoint() + " " + sub.getOwner() +
        " " + sub.getTopicArn());
}

```

Results from Eclipse console.

```

Markers Properties Servers Data Source Explorer Snippets Problems Console
<terminated> OptIn [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (Oct 23, 2015, 11:09:42 AM)
=====
Accepting subscription, Opting-In
=====

hanjiaoqiu@g.harvard.edu 217134905396 arn:aws:sns:us-east-1:217134905396:MyTopicHQ
16179557630 217134905396 arn:aws:sns:us-east-1:217134905396:MyTopicHQ

```

Check from the AWS console. I confirm my phone through replying the text message instead of using the Java program. We get all three accounts subscribed here.

Topic Details: MyTopicHQ

Publish to topic **Other topic actions ▾**

Topic ARN	arn:aws:sns:us-east-1:217134905396:MyTopicHQ
Topic Owner	217134905396
Region	us-east-1
Display name	TopicHQ

Subscriptions

Create Subscription **Request confirmations** **Confirm Subscription** **Other Subscription Actions ▾** **Filter**

Subscription ID	Protocol	Endpoint	Subscriber
arn:aws:sns:us-east-1:217134905396:MyTopicHQ:346ee071-a549-4c7a-abab-43085d922...	email	hanjiaoqiu@g.harvard.edu	217134905396
arn:aws:sns:us-east-1:217134905396:MyTopicHQ:b556b219-26a5-4a5f-aadb-84f09a0fd07f	sms	16179557630	217134905396
arn:aws:sns:us-east-1:217134905396:MyTopicHQ:6b19c1fe-007d-4b88-ad57-962f35e79...	email	qiuhanjiao@gmail.com	217134905396

- 4) Publish messages to all subscriptions. Capture the actions on both email and phone side.

```
for (Topic topic : topicList) {
    String myTopicArn = topic.getTopicArn();
    sns.publish(new PublishRequest()
        .withTopicArn(myTopicArn)
        .withMessage("This is my message to all topics.")
        .withSubject("Message sent to " + myTopicArn));
}
```

Message sent to arn:aws:sns:us-east-1:217134905396:MyTopicHQ **Inbox** x

TopicHQ <no-reply@sns.amazonaws.com> to me 11:18 AM (11 minutes ago) **Star** **Print** **Forward** **Reply** **Compose**

This is my message to all topics.

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
<https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:217134905396:MyTopicHQ:346ee071-a549-4c7a-abab-43085d9221a0&Endpoint=hanjiaoqiu@g.harvard.edu>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at
<https://aws.amazon.com/support>

Message sent to arn:aws:sns:us-east-1:217134905396:MyTopicHQ **Inbox** x

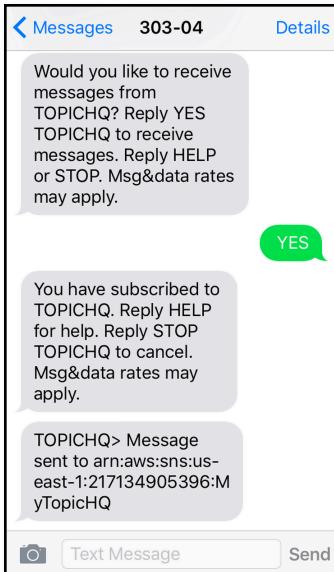
TopicHQ <no-reply@sns.amazonaws.com> to me 11:18 AM (10 minutes ago) **Star** **Print** **Forward** **Reply** **Compose**

This is my message to all topics.

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
<https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:217134905396:MyTopicHQ:6b19c1fe-007d-4b88-ad57-962f35e79911&Endpoint=qiuhanjiao@gmail.com>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at
<https://aws.amazon.com/support>



Points: 20

Problem 4)

Demonstrate that you can send a message from an SNS Topic to an SQS queue using SNS and SQS Consoles. Please note that you will have to “Subscribe” SQS Queue to an SNS Topic on both sides, the Topic side and Queue side. In other words, on the side of SQS Console you do not confirm subscription the way you did it when the protocol was email, but rather “Subscribe to the Topic”. Capture all relevant screen shots.

- 1) Create the SQS queue and SNS topic. Get the ARN of the queue and the topic.

Please enter a name for your new queue. Queue names must be 1-80 characters in length and be composed of alphanumeric characters, hyphens (-), and underscores (_). Your queue will be created in the US East (N. Virginia) region.

Region: US East (N. Virginia)
Queue Name: MySQSQueue

Configure your new queue by setting queue attributes (optional).

Queue Settings

- Default Visibility Timeout:** 20 seconds Value must be between 0 seconds and 12 hours.
- Message Retention Period:** 4 days Value must be between 1 minute and 14 days.
- Maximum Message Size:** 256 KB Value must be between 1 and 256 KB.
- Delivery Delay:** 0 seconds Value must be between 0 seconds and 15 minutes.
- Receive Message Wait Time:** 0 seconds Value must be between 0 and 20 seconds.

Dead Letter Queue Settings

- Use Redrive Policy:**
- Dead Letter Queue:** Value must be an existing queue name.
- Maximum Receives:** Value must be between 1 and 1000.

Buttons: Cancel, Create Queue

Queues

Create New Queue Queue Actions

Filter by Prefix:

Name	Messages Available	Messages in Flight	Created
MySQSQueue	0	0	2015-10-23 11:42:09 GMT-04:00

1 SQS Queue selected.

Details **Permissions** **Redrive Policy**

Name: MySQSQueue
URL: https://sqs.us-east-1.amazonaws.com/217134905396/MySQSQueue
ARN: arn:aws:sqs:us-east-1:217134905396:MySQSQueue
Created: 2015-10-23 11:42:09 GMT-04:00
Last Updated: 2015-10-23 11:42:09 GMT-04:00
Delivery Delay: 0 seconds

Default Visibility Timeout: 20 seconds
Message Retention Period: 4 days
Maximum Message Size: 256 KB
Receive Message Wait Time: 0 seconds
Messages Available (Visible): 0
Messages in Flight (Not Visible): 0
Messages Delayed: 0

Topic Details: MySNSTopic

Publish to topic **Other topic actions ▾**

Topic ARN: arn:aws:sns:us-east-1:217134905396:MySNSTopic
Topic Owner: 217134905396
Region: us-east-1
Display name: SNSTopicHQ

MySQSQueue: arn:aws:sqs:us-east-1:217134905396:MySQSQueue
 MySNSTopic: arn:aws:sns:us-east-1:217134905396:MySNSTopic

- 2) Give permission to Amazon SNS topic to send messages to the Amazon SQS queue.

First set a SendMessage policy on a queue. Select the box for the queue “MySQSQueue”, click the “Permissions” tab, and then click “Add a Permission.”

1 SQS Queue selected.

Details **Permissions** **Redrive Policy**

Effect	Principals	Actions	Conditions
<i>This queue has an empty SQS Queue Access Policy. This means that only the queue owner is allowed to use it. You can Add a Permission to grant another account access to this queue.</i>			

Add a Permission **Edit Policy Document (Advanced)** **What's an SQS Queue Access Policy?**

In the “Add a Permission” dialog box, select “Allow” for Effect, select Everybody (*) for Principal, and then select “SendMessage” from the “Actions” drop-down.

Add a condition that allows the action for the topic. Click “Add Conditions (optional)”, select “ArnEquals” for Condition, select “aws:SourceArn” for Key, and paste in the SNS topic ARN for Value. Click “Yes. Add Condition”. The new condition should appear at the bottom of the box.

Click Add Permission.

Add a Permission to MySQSQueue

Permissions enable you to control which operations a user can perform on a queue. [Click here](#) to learn more about access control concepts.

Effect: Allow Deny

Principal: aws account number(s) Everybody (*)
Use commas between multiple values.

Actions: All SQS Actions (SQS:*)

SendMessage
 ReceiveMessage
 PurgeQueue
 DeleteMessage
 ChangeMessageVisibility
 GetQueueAttributes
 GetQueueUrl

Cancel **Add Permission**

Add a Permission to MySQSQueue

Permissions enable you to control which operations a user can perform on a queue. [Click here](#) to learn more about access control concepts.

Effect: Allow Deny

Principal: aws account number(s) Everybody (*)
Use commas between multiple values.

Actions: All SQS Actions (SQS:*)

Conditions (optional) **Hide**

Conditions specify additional restrictions on when a permission can take effect. For more information about using conditions, see the [description of the Condition element](#).

Qualifier: None

Condition: ArnEquals

Key: aws:SourceArn

Value: arn:aws:sns:us-east-1:217134905396:MySNSTopic
Use commas between multiple values.

Add Condition

Cancel **Add Permission**

Add a Permission to MySQSQueue

Permissions enable you to control which operations a user can perform on a queue. [Click here](#) to learn more about access control concepts.

Effect: Allow Deny

Principal: aws account number(s) Everybody (*)
Use commas between multiple values.

Actions: All SQS Actions (SQS:*)

Conditions (optional) [Hide](#)

Conditions specify additional restrictions on when a permission can take effect. For more information about using conditions, see the [description of the Condition element](#).

Qualifier: None

Condition: ArnEquals

Key: aws:SourceArn

Value: arn:aws:sns:us-east-1:217134905396:MySNSTopic
Use commas between multiple values.

Confirm Add Condition

It looks like you entered the value of a condition. Did you want to add this condition before saving your added permission

1 SQS Queue selected.

Effect	Principals	Actions	Conditions
Allow	• Everybody (*)	• SQS:SendMessage	• ArnEquals o aws:SourceArn: "arn:aws:sns:us-east-1:217134905396:MySNSTopic"

[What's an SQS Queue Access Policy?](#)

- 3) Subscribe the queue to the Amazon SNS topic. In the navigate pane, select the topic. Click “Create Subscription”.

Topic Details: MySNSTopic

Publish to topic		Other topic actions ▾
Topic ARN	arn:aws:sns:us-east-1:217134905396:MySNSTopic	
Topic Owner	217134905396	
Region	us-east-1	
Display name	SNSTopicHQ	

Subscriptions

Create Subscription	Request confirmations	Confirm Subscription	Other Subscription Actions ▾
Filter			
<input type="checkbox"/> Subscription ID		Protocol	Endpoint
<input checked="" type="checkbox"/> arn:aws:sns:us-east-1:217134905396:MySNSTopic:e9aae9fe-29f3-43d4-9402-07909ae98...	sqs	arn:aws:sqs:us-east-1:21...	

Select “Amazon SQS” for Protocol, paste in the ARN for MySQSQueue.

Create Subscription

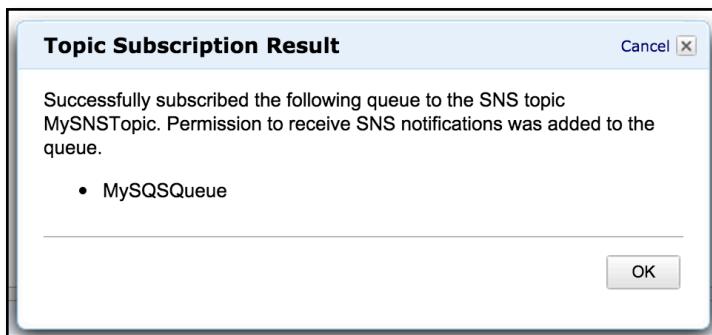
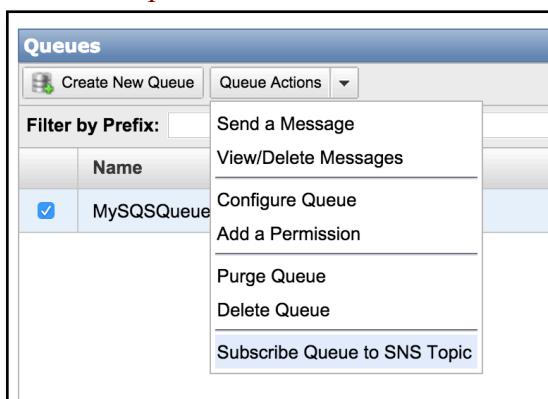
Topic ARN: arn:aws:sns:us-east-1:217134905396:MySNSTopic

Protocol: Amazon SQS

Endpoint: arn:aws:sqs:us-east-1:217134905396:MySQSQueue

Create Subscription

On MySQSQueue side, Select “Subscribe Queue to SNS Topic” from the Queue Actions drop-down list.



The subscription ID appears in the “Subscriptions” list of MySNSTopic.

Subscriptions

Create Subscription	Request confirmations	Confirm Subscription	Other Subscription Actions	
<input type="checkbox"/>	Subscription ID	Protocol	Endpoint	Subscriber
<input type="checkbox"/>	arn:aws:sns:us-east-1:217134905396:MySNSTopic:d2cc0af6-1a39-4bdd-bffa-d9c0838dc...	sqs	arn:aws:sqs:us-east-1:217134905396:MySQSQueue	217134905396

Subscriptions			
Create Subscription		Request confirmations	Actions ▾
Filter <input type="text"/>			
Subscription ARN	Proto...	Endpoint	Topic ARN
arn:aws:sns:us-east-1:217134905396:MySNSTopic:e9aae9fe-29f3-43d4-9402-07909ae98...	sqs	arn:aws:sqs:us-east-1:217134905396:MySQSQueue	arn:aws:sns:us-east-1:217134905396:MySNSTopic

- 4) Publish to a topic using the Amazon SNS console. In the navigation pane, select the topic and click “Publish to Topic”.

In the Subject box, enter a subject (for example, Testing publish to queue) in the Message box, enter some text (for example, Hello from Hanjiao!), and click “Publish Message”. Publish another message “Another message from Hanjiao”.

Publish a message

Amazon SNS enables you to publish notifications to all subscriptions associated with a topic as well as to an individual endpoint associated with a platform application.

Topic ARN	arn:aws:sns:us-east-1:217134905396:MySNSTopic	?
Subject	Testing publish to queue	?
Message format	<input checked="" type="radio"/> Raw <input type="radio"/> JSON	
Message	Hello from Hanjiao!	
JSON message generator		
Time to live (TTL)		?
Cancel		Publish message

Publish a message

Amazon SNS enables you to publish notifications to all subscriptions associated with a topic as well as to an individual endpoint associated with a platform application.

Topic ARN	arn:aws:sns:us-east-1:217134905396:MySNSTopic	?
Subject	Testing publish to queue	?
Message format	<input checked="" type="radio"/> Raw <input type="radio"/> JSON	
Message	Another message from Hanjiao!	

- 5) View the message from the topic using the Amazon SQS console.

From the “Queue Action” drop-down, select “View/Delete Messages” and click “Start Polling for Messages”. Two messages appear in the box. In the “Body”

column, click “More Details”. The Message Details box contains a JSON document that contains the subject and message.

The first screenshot shows the 'View/Delete Messages in MySQSQueue' interface with two messages listed. The second screenshot is a 'Message Details' modal for the first message, showing its JSON content and metadata. The third screenshot shows the queue after one message has been processed, with the message details modal still open.

Delete	Body	Size	Sent	Receive Count
<input type="checkbox"/>	{ "Type" : "Notification", "MessageId" : "99e1494d-bfac-52b3-bddf-07e9cdf4633e", "TopicArn" : "arn:aws:sns:us-east-1:217134905396:MySNSTopic", "Subject" : "Testing publish to queue", "Message" : "Hello from Hanjiao!", "Timestamp" : "2015-10-23T17:36:24.591Z", "SignatureVersion" : "1", "Signature" : "Y5YmUYe8FQE4gdMqcwvNTDPM+FzXNwp7w+k+XzbBhFqpBqf2GH08Jk3f4q1RXWBK" }	1.2 KB	2015-10-23 13:36:24 GMT-04:00	3
<input type="checkbox"/>	{ "Type" : "Notification", "MessageId" : "c821beb1-80c9-51d7-b9c6-f015f269344e", "TopicArn" : "arn:aws:sns:us-east-1:217134905396:MySNSTopic", "Subject" : "Testing publish to queue", "Message" : "Another message from Hanjiao!", "Timestamp" : "2015-10-23T17:37:36.509Z", "SignatureVersion" : "1", "Signature" : "NNZIUsd/BMMx9JH9Mb6ov1Hsst8SZtuH2Or8n9Nusighxtkg79Umo/mFEeCK0HYd5" }	1.2 KB	2015-10-23 13:37:36 GMT-04:00	1

Message Details

Message Body

```
{
  "Type" : "Notification",
  "MessageId" : "99e1494d-bfac-52b3-bddf-07e9cdf4633e",
  "TopicArn" : "arn:aws:sns:us-east-1:217134905396:MySNSTopic",
  "Subject" : "Testing publish to queue",
  "Message" : "Hello from Hanjiao!",
  "Timestamp" : "2015-10-23T17:36:24.591Z",
  "SignatureVersion" : "1",
  "Signature" : "Y5YmUYe8FQE4gdMqcwvNTDPM+FzXNwp7w+k+XzbBhFqpBqf2GH08Jk3f4q1RXWBK"
}
```

Message ID: 9d34b81b-7c48-4c3b-998b-733eb0be0ccf
Size: 1.2 KB
MD5 of Body: a235ca0125a5794f738b474f5fc54558
Sender Account ID: 443302527238
Sent: 2015-10-23 13:36:24.675 GMT-04:00
First Received: 2015-10-23 13:36:46.799 GMT-04:00
Receive Count: 3
Message Attribute Count: 0

View/Delete Messages in MySQSQueue

Message Details

Message Body

```
{
  "Type" : "Notification",
  "MessageId" : "c821beb1-80c9-51d7-b9c6-f015f269344e",
  "TopicArn" : "arn:aws:sns:us-east-1:217134905396:MySNSTopic",
  "Subject" : "Testing publish to queue",
  "Message" : "Another message from Hanjiao!",
  "Timestamp" : "2015-10-23T17:37:36.509Z",
  "SignatureVersion" : "1",
  "Signature" : "NNZIUsd/BMMx9JH9Mb6ov1Hsst8SZtuH2Or8n9Nusighxtkg79Umo/mFEeCK0HYd5"
}
```

Message ID: 00c7cd2e-37b2-460f-b2db-e7d6c29e6883
Size: 1.2 KB
MD5 of Body: 3fef66829fc31d0356ba5934db247cc7
Sender Account ID: 443302527238
Sent: 2015-10-23 13:37:36.577 GMT-04:00
First Received: 2015-10-23 13:37:57.321 GMT-04:00
Receive Count: 1
Message Attribute Count: 0

Stopped after polling the queue at 0.6 receives/second for 10.1 seconds. Messages shown above are now available to other consumers.

Points: 15

Problem 5)

Implement above task programmatically using Java, C# or any other AWS SNS supported language. This problem might require some literature (Internet) searching.
Java source code file: SNSToSQS.java

1) Create a SQS instance and a SNS Topic instance.

```
AmazonSQS sqs = new AmazonSQSClient(credentials);
AmazonSNS sns = new AmazonSNSClient(credentials);
```

Get the QueueArn and TopicArn. The QueueArn can be get from the queue attributes.

```
// Create a SQS queue
System.out.println("Creating a new SQS queue called HqiuSQSQueue.\n");
CreateQueueRequest createQueueRequest = new CreateQueueRequest("HqiuSQSQueue");
String myQueueUrl = sqs.createQueue(createQueueRequest).getQueueUrl();

// Retrieve SQS Amazon Resource Name
GetQueueAttributesRequest getQAResult = new GetQueueAttributesRequest().withQueueUrl(myQueueUrl);
GetQueueAttributesResult getQAResult = sqs.getQueueAttributes(getQAResult.withAttributeNames("QueueArn"));
Map<String, String> attributeMap = getQAResult.getAttributes();
String myQueueArn = attributeMap.get("QueueArn");
System.out.println("Queue created: " + myQueueArn);
```

```
// Create a SNS Topic
System.out.println("Creating HqiuSNSTopic.\n");
CreateTopicRequest createTopicRequest = new CreateTopicRequest().withName("HqiuSNSTopic");

// Retrieve SNS Amazon Resource Name
String myTopicArn = sns.createTopic(createTopicRequest).getTopicArn();
System.out.println("Topic created: " + myTopicArn);
```

2) Subscribe the SQS queue to the SNS Topic. There are two different ways to do this.

The first method is to use the “Topics” class. It provides the method “subscribeQueue()” to do the subscription in only one step. The second method is like problem 2, which invokes “subscribe()” method of “sns client” class. The protocol type is “sqs”. Both the methods work.

```
// Subscribe SQS queue to SNS topic on both sides
// Method 1:
Topics.subscribeQueue(sns, sqs, myTopicArn, myQueueUrl);

// Method 2:
// Send subscribe request
// SubscribeRequest subReq = new SubscribeRequest(myTopicArn, "sqs", myQueueArn);
// SubscribeResult subRes = sns.subscribe(subReq);

System.out.println("\nSubscribed " + myQueueArn + " to topic: " + myTopicArn + "\n");
```

3) Set policy on both topic (to allow subscriptions) and queue (to allow messages to be published to it).

```
// Set policy on topic to allow open subscriptions
Policy snsPolicy = new Policy().withStatements(
    new Statement(Effect.Allow)
        .withPrincipals(Principal.AllUsers)
        .withActions(SNSActions.Subscribe));
sns.setTopicAttributes(new SetTopicAttributesRequest(myTopicArn, "Policy", snsPolicy.toJson()));

// Set the queue policy to allow SNS to publish messages
Policy sqsPolicy = new Policy().withStatements(
    new Statement(Effect.Allow)
        .withPrincipals(Principal.AllUsers)
        .withResources(new Resource(myQueueArn))
        .withActions(SQSActions.SendMessage)
        .withConditions(ConditionFactory.newSourceArnCondition(myTopicArn)));

HashMap<String, String> queueAttributes = new HashMap<String, String>();
queueAttributes.put("Policy", sqsPolicy.toJson());
sqs.setQueueAttributes(new SetQueueAttributesRequest(myQueueUrl, queueAttributes));
```

4) Public messages to SNS topic. Check the message in the SQS.

```
// Public message from SNS Topic to SQS queue
sns.publish(new PublishRequest()
    .withTopicArn(myTopicArn)
    .withMessage("This is my message to SQS queue.")
    .withSubject("Message sent to " + myTopicArn));
```



```
// Receive messages from SQS queue
List<Message> messages = receiveMessage(sqs, myQueueUrl);
```



```
private static List<Message> receiveMessage(AmazonSQS sqs, String myQueueUrl) {

    System.out.println("\nReceiving messages from MyQueue.\n");
    ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest(myQueueUrl);
    List<Message> messages = sqs.receiveMessage(receiveMessageRequest).getMessages();

    for (Message message : messages) {
        System.out.println(" Message");
        System.out.println("   MessageId: " + message.getMessageId());
        System.out.println("   ReceiptHandle: " + message.getReceiptHandle());
        System.out.println("   MD5OfBody: " + message.getMD5OfBody());
        System.out.println("   Body: " + message.getBody());
        for (Entry<String, String> entry : message.getAttributes().entrySet()) {
            System.out.println("     Attribute");
            System.out.println("       Name: " + entry.getKey());
            System.out.println("       Value: " + entry.getValue());
        }
    }
    System.out.println();

    return messages;
}
```

Check from AWS SNS and SQS console. HqiuSQSQueue has subscribed HqiuSNSTopic. I first use method 1 to do the subscription and send one message.

Topic Details: HqiuSNSTopic

Subscriptions

Subscriptions				
		Protocol	Endpoint	Subscriber
<input type="checkbox"/>	Subscription ID	sqs	arn:aws:sqs:us-east-1:217134905396:HqiuSQSQueue	217134905396

Queues

Name	Messages Available	Messages in Flight	Created
HqiuSQSQueue	1	0	2015-10-23 17:48:18 GMT-04:00
MySQSQueue	2	0	2015-10-23 11:42:09 GMT-04:00

Details

Name: HqiuSQSQueue **Default Visibility Timeout:** 30 seconds
URL: https://sqs.us-east-1.amazonaws.com/217134905396/HqiuSQSQueue **Message Retention Period:** 4 days
ARN: arn:aws:sqs:us-east-1:217134905396:HqiuSQSQueue **Maximum Message Size:** 256 KB
Created: 2015-10-23 17:48:18 GMT-04:00 **Receive Message Wait Time:** 0 seconds
Last Updated: 2015-10-23 17:55:12 GMT-04:00 **Messages Available (Visible):** 1
Delivery Delay: 0 seconds **Messages in Flight (Not Visible):** 0 **Messages Delayed:** 0

Queues

Filter by Prefix: HqiuSQSQueue

Name	Messages Available	Messages in Flight	Created
HqiuSQSQueue	1	0	2015-10-23 17:48:18 GMT-04:00
MySQSQueue	2	0	2015-10-23 11:42:09 GMT-04:00

View/Delete Messages in HqiuSQSQueue

View up to: 10 messages Poll queue for: 30 seconds *Polling for new messages once every 2 seconds.*

Delete	Body	Size	Sent	Receive Count
<input type="checkbox"/>	Message Details Message Body Message Attributes <pre>{ "Type" : "Notification", "MessageId" : "995e4b30-6933-5782-8c55-0594c633a487", "TopicArn" : "arn:aws:sns:us-east-1:217134905396:HqiuSNSTopic", "Subject" : "Message sent to arn:aws:sns:us-east-1:217134905396:HqiuSNSTopic", "Message" : "This is my message to SQS queue.", "Timestamp" : "2015-10-23T21:55:12.186Z", "SignatureVersion" : "1", "Signature" : }</pre> <p>Message ID: 5df2168-1cf6-4134-ad68-a3c29ee2a05c Size: 1,022 bytes MD5 of Body: c2ef82e76f4af73c779b820850d257b4 Sender Account ID: 443302527238 Sent: 2015-10-23 17:55:12.250 GMT-04:00 First Received: 2015-10-23 17:56:20.267 GMT-04:00 Receive Count: 1 Message Attribute Count: 0</p>	55:12 GMT-04:00		1

Stopped after polling the queue at 0.7 receives/second for 4.1 seconds. Messages shown above are now available to other consumers.

[Delete Messages](#) [Close](#)

We can get the message “This is my message to SQS queue” from HqiuSQSQueue. Then I unsubscribe HqiuSQSQueue and redo the subscription through Method 2. I send the message again. The HqiuSQSQueue successfully get the message.

Queues

Filter by Prefix: HqiuSQSQueue

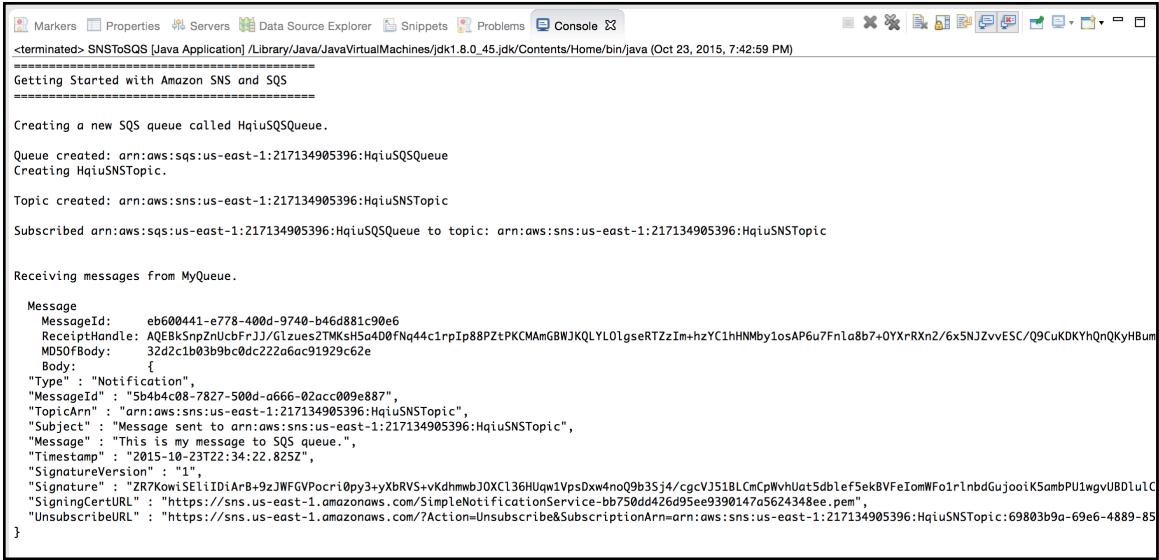
Name	Messages Available	Messages in Flight	Created
HqiuSQSQueue	2	0	2015-10-23 17:48:18 GMT-04:00
MySQSQueue	2	0	2015-10-23 11:42:09 GMT-04:00

View/Delete Messages in HqiuSQSQueue

View up to: 10 messages Poll queue for: 30 seconds *Polling for new messages once every 2 seconds.*

Delete	Body	Size	Sent	Receive Count
<input type="checkbox"/>	Message Details Message Body Message Attributes <pre>{ "Type" : "Notification", "MessageId" : "5b4b4c08-7827-500d-a666-02acc009e887", "TopicArn" : "arn:aws:sns:us-east-1:217134905396:HqiuSNSTopic" }</pre> <p>Message ID: 5b4b4c08-7827-500d-a666-02acc009e887 Size: 1,022 bytes MD5 of Body: 1.022 bytes Sent: 2015-10-23 18:34:22 GMT-04:00 First Received: 2015-10-23 18:34:22 GMT-04:00 Receive Count: 2</p>	1.022 bytes	2015-10-23 18:34:22 GMT-04:00	2
<input type="checkbox"/>	Message Details Message Body Message Attributes <pre>{ "Type" : "Notification", "MessageId" : "995e4b30-6933-5782-8c55-0594c633a487", "TopicArn" : "arn:aws:sns:us-east-1:217134905396:HqiuSNSTopic" }</pre> <p>Message ID: 995e4b30-6933-5782-8c55-0594c633a487 Size: 1,022 bytes MD5 of Body: 1.022 bytes Sent: 2015-10-23 17:55:12 GMT-04:00 First Received: 2015-10-23 17:55:12 GMT-04:00 Receive Count: 4</p>	1.022 bytes	2015-10-23 17:55:12 GMT-04:00	4

Results from Eclipse console.



```

Markers Properties Servers Data Source Explorer Snippets Problems Console
<terminated> SNSToSQS [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (Oct 23, 2015, 7:42:59 PM)
=====
Getting Started with Amazon SNS and SQS
=====

Creating a new SQS queue called HqiuSQSQueue.

Queue created: arn:aws:sqs:us-east-1:217134905396:HqiuSQSQueue
Creating HqiuSNSTopic.

Topic created: arn:aws:sns:us-east-1:217134905396:HqiuSNSTopic

Subscribed arn:aws:sqs:us-east-1:217134905396:HqiuSQSQueue to topic: arn:aws:sns:us-east-1:217134905396:HqiuSNSTopic

Receiving messages from MyQueue.

Message
{
  "MessageId": "eb600441-e778-400d-9740-b46d881c90e6",
  "ReceiptHandle": "AQEBkSnpZnUcbFrJJ/GIzes2TMKsh5o4D0fNq44c1rpIp88PZtPKCMAmGBWJKQLYl0lgseRTZzIm+hzYC1hHNMy1osAP6u7Fnla8b7+0YXrRXn2/6x5NJZvvESC/Q9CuKDKYhQnQKyHBum",
  "MD5OfBody": "32d2c1b03b9bc0dc222a6ac91929c62e",
  "Body": {
    "Type": "Notification",
    "MessageId": "5b4b4c08-7827-500d-a666-02acc009e887",
    "TopicArn": "arn:aws:sns:us-east-1:217134905396:HqiuSNSTopic",
    "Subject": "Message sent to arn:aws:sns:us-east-1:217134905396:HqiuSNSTopic",
    "Message": "This is my message to SQS queue.",
    "Timestamp": "2015-10-23T22:34:22.825Z",
    "SignatureVersion": "1",
    "Signature": "ZR7KowiSEliIDiArB+9zJWFGVpocri0py3+yXbRVs+vKdhmwbdJ0XC136HUqw1VpsDxw4noQ9b3Sj4/cgcVJ51BLcmCpVhUlut5dblef5ekBVFeIomWFo1rlnbdGujooiK5ambPU1wgvUBDlulC",
    "SigningCertURL": "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-bb750dd426d95ee9390147a5624348ee.pem",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:217134905396:HqiuSNSTopic:69803b9a-69e6-4889-85"
  }
}

```

Points: 25