ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Technische Universität Berlin

Prof. Dr. Stefan Tai
Markus Klems

Your name: _____

Enterprise Computing (WS 2014)

Exercise 2 (3 Portfoliopunkte)

Info:

- The solution to this exercise must be handed in by Tuesday, Nov 11th 2014, 2PM to Markus Klems.
- The solution must be printed out. Please write your name on the solution sheet.

Task 1 – Web Services (30%)

Given the following excerpts from the AWS EC2 WSDL document *http://s3.amazonaws.com/ec2-downloads/ec2.wsdl*, please answer the three multiple choice questions below. Each correct answer gives you 10%; for each wrong answer you lose 5%. You cannot get less than 0% for this task.

```
<?xml version="1.0" encoding="UTF-8"?><definitions ...>
  <types>
...
      <xs:element name="RunInstances" type="tns:RunInstancesType"/>
      <xs:complexType name="RunInstancesType">
        <xs:sequence>
          <xs:element name="imageId" type="xs:string"/>
          <xs:element name="minCount" type="xs:int"/>
          <xs:element name="maxCount" type="xs:int"/>
          ...
        </xs:sequence>
      </xs:complexType>
...
      <xs:element name="RunInstancesResponse"
       type="tns:RunInstancesResponseType"/>
      <xs:complexType name="RunInstancesResponseType">
        <xs:sequence>
          <xs:element name="requestId" type="xs:string"/>
          ...
        </xs:sequence>
```

```
      </xs:complexType>
...
  <message name="RunInstancesRequestMsg">
    <part name="RunInstancesRequestMsgReq" element="tns:RunInstances"/>
  </message>
  <message name="RunInstancesResponseMsg">
    <part name="RunInstancesResponseMsgResp"
     element="tns:RunInstancesResponse"/>
  </message>
...
  <portType name="AmazonEC2PortType">
...
    <operation name="RunInstances">
      <input message="tns:RunInstancesRequestMsg"/>
      <output message="tns:RunInstancesResponseMsg"/>
    </operation>
...
  </portType>
  <binding name="AmazonEC2Binding" type="tns:AmazonEC2PortType">
    <soap:binding style="document"
     transport="http://schemas.xmlsoap.org/soap/http"/>
...
    <operation name="RunInstances">
      <soap:operation soapAction="RunInstances"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
...
  </binding>
  <service name="AmazonEC2">
    <port name="AmazonEC2Port" binding="tns:AmazonEC2Binding">
      <soap:address location="https://ec2.amazonaws.com/"/>
    </port>
  </service>
</definitions>
```

Which version of the WSDL specification is used for the EC2 Web Service?

✓	WSDL 1.x
	WSDL 2.0
	WSDL 3.1

Which messages are sent when the `RunInstances` operation is called?

	First a message by the web service, then a respone by the client.
	Only a message by the client.
	First a message by the client, then a response by the web service, then a response by the client.
✓	First a message by the client, then a response by the web service.

To which address does the web service client send messages?

	http://www.w3.org/2001/XMLSchema-instance
✓	https://ec2.amazonaws.com
	http://ec2.amazonaws.com/doc/2014-06-15/
	http://schemas.xmlsoap.org/soap/http

Task 2 – AWS S3 (40%)

- Clone the Eclipse project from https://gitlab.tubit.tu-berlin.de/klems/awss3
- Build the project with Maven

Prerequisites:
Set up your AWS credentials that were given to you by Markus as follows:
~/.aws/credentials
[default]
aws_access_key_id=enteryourkeyhere
aws_secret_access_key=enteryoursecrethere

(If you don't know how to do this, please read this: http://docs.aws.amazon.com/AWSSdkDocsJava/latest/DeveloperGuide/credentials.html or post a question in the ISIS2 forum for Enterprise Computing)

a) Now fill the blanks with your code (20%)

```
// TODO create a bucket with name "ise-tu-berlin-exercise2-",
// followed by your nickname (e.g., silversurfer)
log.info("Creating a bucket (if it does not exist, yet)");

String bucketName = "ise-tu-berlin-exercise2-batman";
if(!(s3.doesBucketExist(bucketName))){
        s3.createBucket(new CreateBucketRequest(bucketName));
}

// TODO Upload a text File object to your S3 bucket
// use the createSampleFile method to create the File object
log.info("Uploading an object");

File file = createSampleFile("batman");
String key = "batman-sample-file";
s3.putObject(new PutObjectRequest(bucketName, key, file));

// TODO Download the file from S3 and print it out using the
// displayTextInputStream method.
log.info("Downloading an object");

S3Object s3Object = s3.getObject(new GetObjectRequest(bucketName, key));
displayTextInputStream(s3Object.getObjectContent());
```

b) Which AWS S3 operation uses which HTTP method? (20%)

AWS operation	HTTP method
createBucket	HEAD
putObject	PUT
getObject	GET
deleteObject	DELETE

(Hint: Launch the Java program with JVM option "-Dlog4j.configuration=log4j.properties" and log4j.properties setting "log4j.logger.org.apache.http=DEBUG")

Task 3 – AWS SQS (30%)

Rewrite the (unmodified) borrower/lender example from 3b) of the previous exercise 1 by replacing JMS with AWS SQS. Most of the structure already exists but some pieces are missing. Fill out the blanks in the snippets below with your code.

Solution:

```
//  SqsBorrower.java
// TODO check response queue for matching responses
ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest();
receiveMessageRequest.setQueueUrl(responseQ);
receiveMessageRequest.setMessageAttributeNames(Arrays.asList("uuid"));
// Print out the response
System.out.println(          lenderResponseMessage.getBody()                    );

// delete the message from the queue

sqs.deleteMessage(new DeleteMessageRequest(responseQ, messageRecieptHandle));


// SqsLender.java
// TODO Prepare receive loan request message request.
ReceiveMessageRequest receiveLoanRequestMessageRequest =
                                                    new ReceiveMessageRequest();
receiveMessageRequest.setQueueUrl(requestQ);
receiveMessageRequest.setMessageAttributeNames(Arrays.asList("uuid"));
```

```
// TODO Check request queue for loan requests.
List<Message> messages = sqs.receiveMessage(receiveMessageRequest).getMessages();


// TODO Delete loan request message from queue

String messageReceiptHandle = loanRequestMessage.getReceiptHandle();
sqs.deleteMessage(new DeleteMessageRequest(requestQ, messageReceiptHandle));


// TODO Send out the response

sqs.sendMessage(loanResponseMessageRequest);
```