M7024E Laboratory 3

PROGRAMMING CLOUD
SERVICES - COMPUTE SERVICES

Ahmed Afif Monrat, Syed Asif Iqbal | M7024E: CLOUD SERVICES | 1/20/2018

1. Identify ways of creating the Amazon EC2 service clients.
(a) Explain in detail how the Amazon EC2 service clients are created by providing details of the packages and classes involved. Create a diagram of the dependencies involved.

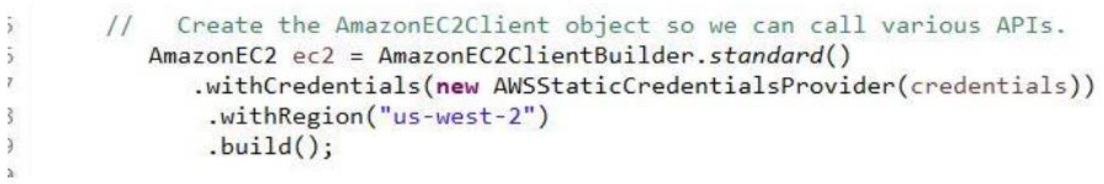
To create Amazon EC2 service client, the following steps need to be done:

I. Creating an Amazon Ec2 security group

II. Creating Key Pair

III. Run an Amazon EC2 instance

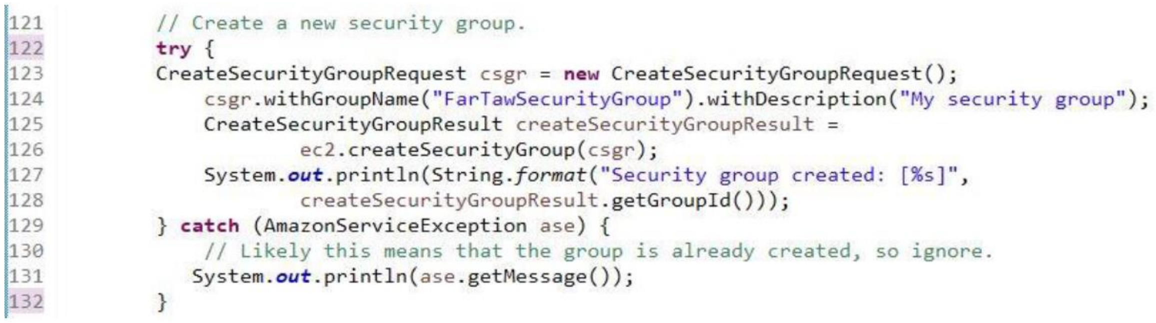
Create the AmazonEC2Client object so we can call various APIs.



```
    //   Create the AmazonEC2Client object so we can call various APIs.
       AmazonEC2 ec2 = AmazonEC2ClientBuilder.standard()
          .withCredentials(new AWSStaticCredentialsProvider(credentials))
           .withRegion("us-west-2")
           .build();
```

A. Creating an Amazon Ec2 Security group:

To create a service group, first we create and initialize a CreateSecurityGroupRequest instance. We used the withGroupName method to set the security group name, withDescription method to set the security group description, as follows: The security group name must be unique within the AWS region in which you initialize your Amazon EC2 client.

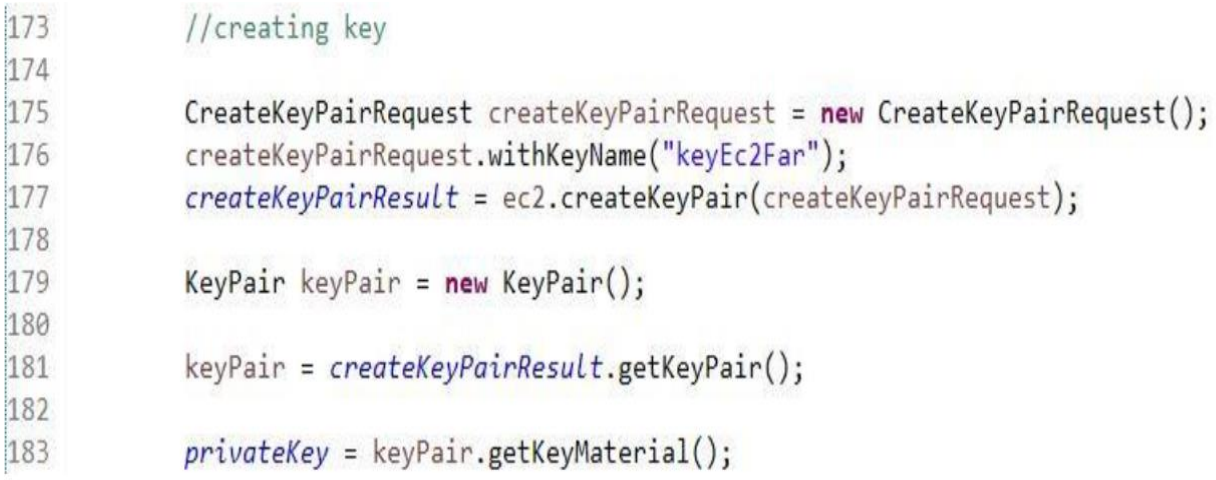


```
        // Create a new security group.
        try {
        CreateSecurityGroupRequest csgr = new CreateSecurityGroupRequest();
            csgr.withGroupName("FarTawSecurityGroup").withDescription("My security group");
            CreateSecurityGroupResult createSecurityGroupResult =
                    ec2.createSecurityGroup(csgr);
            System.out.println(String.format("Security group created: [%s]",
                    createSecurityGroupResult.getGroupId()));
        } catch (AmazonServiceException ase) {
            // Likely this means that the group is already created, so ignore.
           System.out.println(ase.getMessage());
        }
```

If we attempt to create a security group with the same name as an existing security group, createSecurityGroup throws an exception.

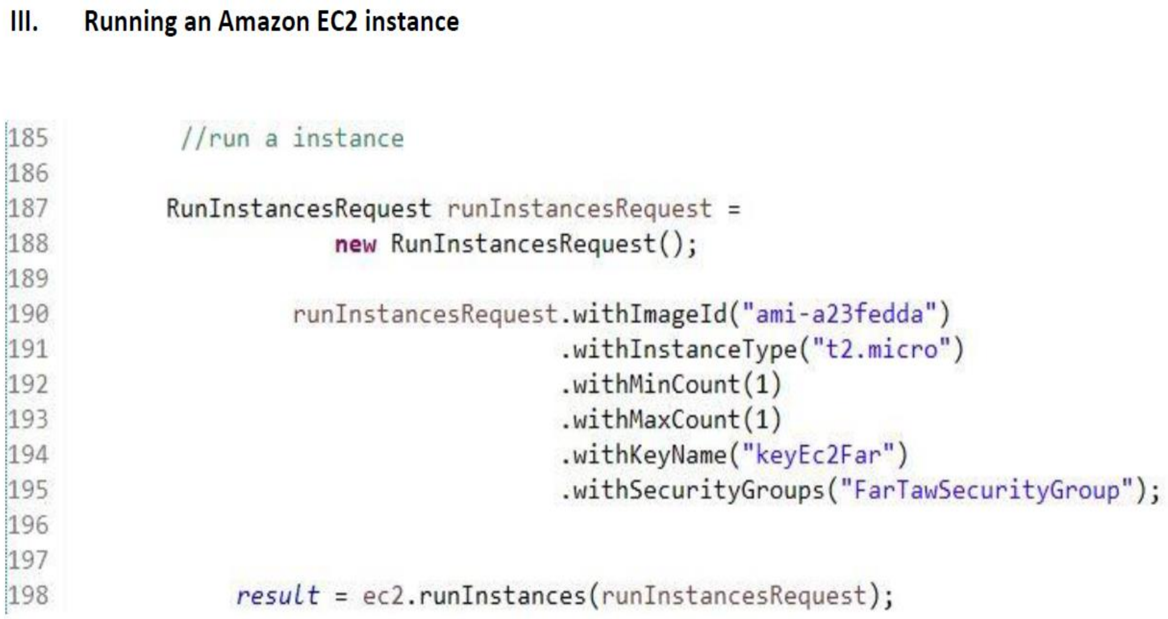
II. Creating Key Pair:

First, we created and initialized a CreateKeyPairRequest instance and used the withKeyName method to set the key pair name. Then, we passed the request object to the createKeyPair method. The method returns a CreateKeyPairResult instance. Then, we called the result object's getKeyPair method to obtain a KeyPair object. Then, we called the KeyPair object's getKeyMaterial method to obtain the unencrypted PEM-encoded private key, as follows:



```
        //creating key

        CreateKeyPairRequest createKeyPairRequest = new CreateKeyPairRequest();
        createKeyPairRequest.withKeyName("keyEc2Far");
        createKeyPairResult = ec2.createKeyPair(createKeyPairRequest);

        KeyPair keyPair = new KeyPair();

        keyPair = createKeyPairResult.getKeyPair();

        privateKey = keyPair.getKeyMaterial();
```

III. Running an Amazon EC2 instance



III. Running an Amazon EC2 instance

```
         //run a instance

        RunInstancesRequest runInstancesRequest =
                    new RunInstancesRequest();

                runInstancesRequest.withImageId("ami-a23fedda")
                                  .withInstanceType("t2.micro")
                                  .withMinCount(1)
                                  .withMaxCount(1)
                                  .withKeyName("keyEc2Far")
                                  .withSecurityGroups("FarTawSecurityGroup");


            result = ec2.runInstances(runInstancesRequest);
```

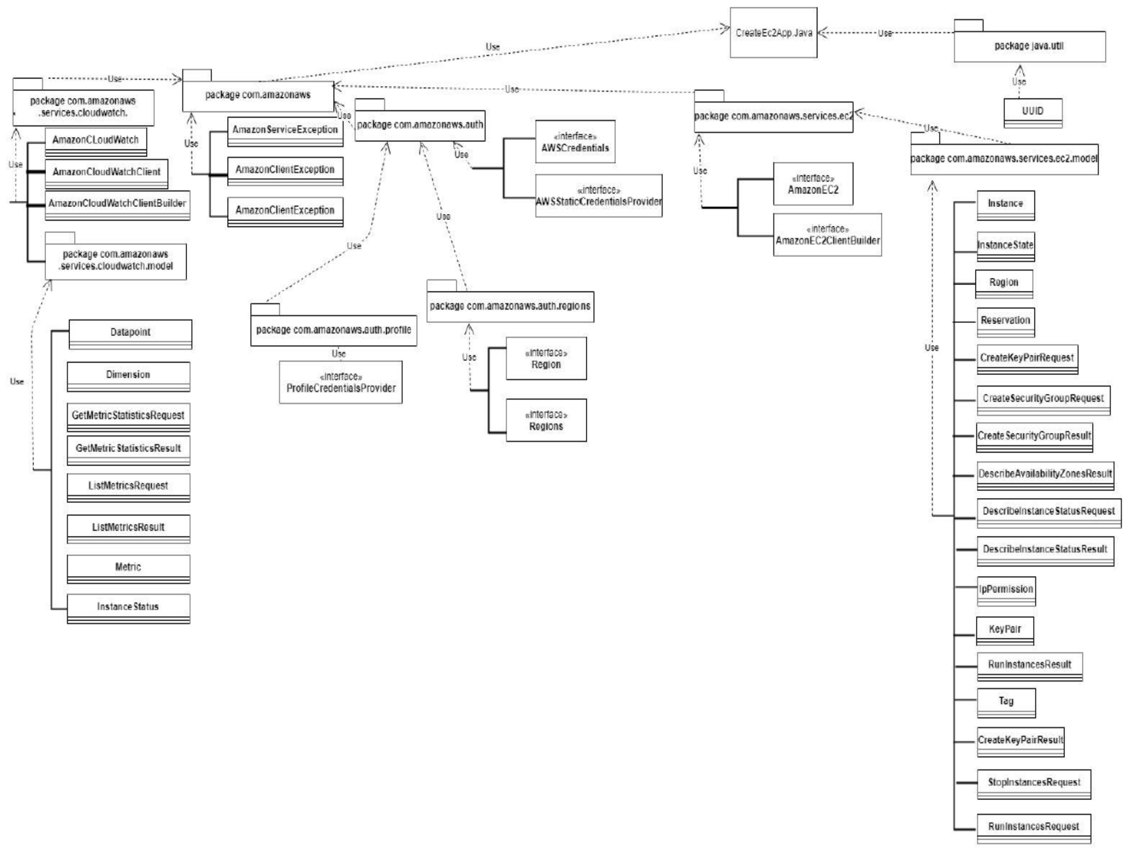
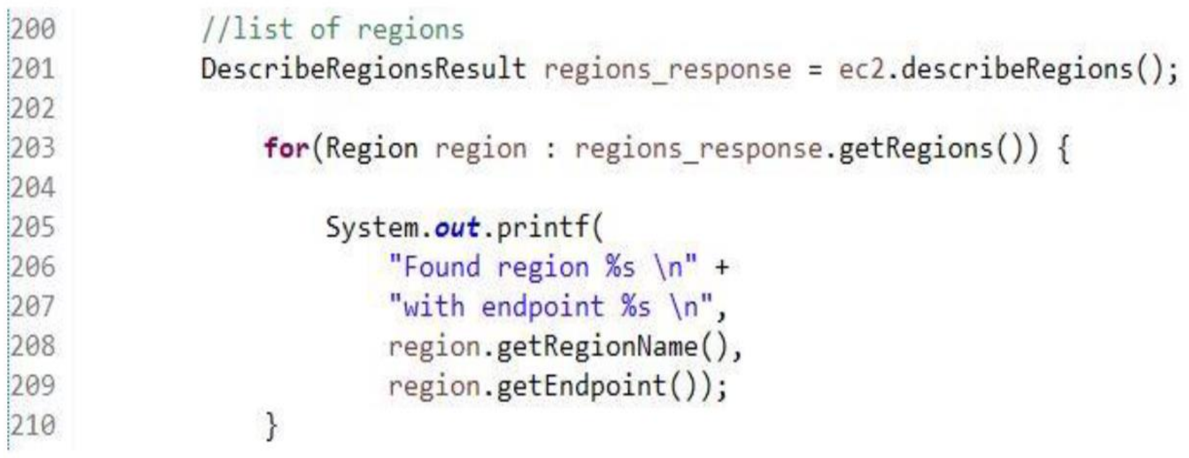

CreateEc2App.Java
package java.util
UUID
package com.amazonaws.services.cloudwatch.
AmazonCLoudWatch
AmazonCloudWatchClient
AmazonCloudWatchClientBuilder
package com.amazonaws.services.cloudwatch.model
Datapoint
Dimension
GetMetricStatisticsRequest
GetMetric StatisticsResult
ListMetricsRequest
ListMetricsResult
Metric
Instance Status
package com.amazonaws
AmazonServiceException
AmazonClientException
AmazonClientException
package com.amazonaws.auth
«interface» AWSCredentials
«interface» AWSStaticCredentialsProvider
package com.amazonaws.auth.profile
«interface» ProfileCredentialsProvider
package com.amazonaws.auth.regions
«interface» Region
«interface» Regions
package com.amazonaws.services.ec2
«interface» AmazonEC2
«interface» AmazonEC2ClientBuilder
package com.amazonaws.services.ec2.model
Instance
InstanceState
Region
Reservation
CreateKeyPairRequest
CreateSecurityGroupRequest
CreateSecurityGroupResult
DescribeAvailabilityZonesResult
DescribeInstanceStatusRequest
DescribeInstanceStatusResult
IpPermission
KeyPair
RunInstancesResult
Tag
CreateKeyPairResult
StopInstancesRequest
RunInstancesRequest
Use

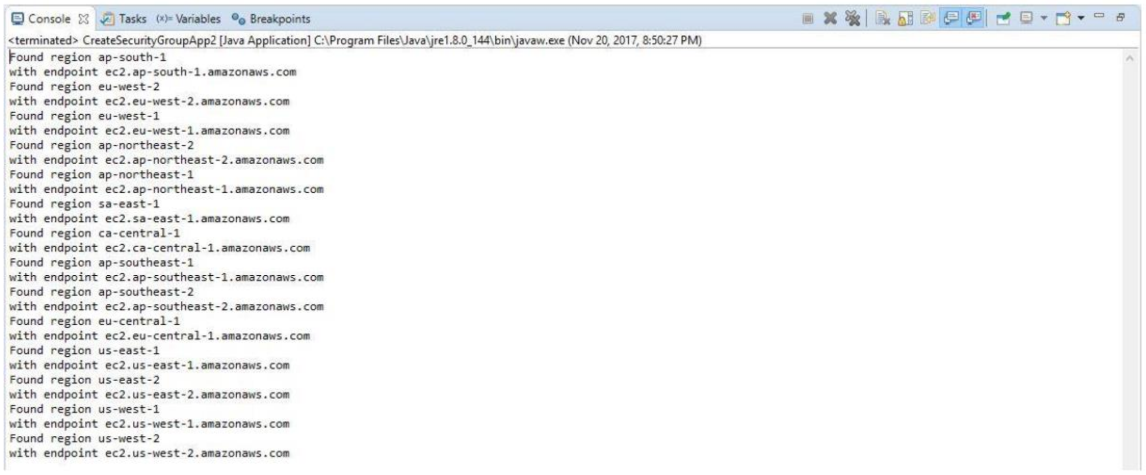
Figure 1 : Dependency Diagram of packages of classes of CreateEc2App2.java

2. Create a Java program to manage your EC2 instance. Start with listing region names and their endpoints.



```
        //list of regions
        DescribeRegionsResult regions_response = ec2.describeRegions();

            for(Region region : regions_response.getRegions()) {

                System.out.printf(
                    "Found region %s \n" +
                    "with endpoint %s \n",
                    region.getRegionName(),
                    region.getEndpoint());
            }
```

Output:

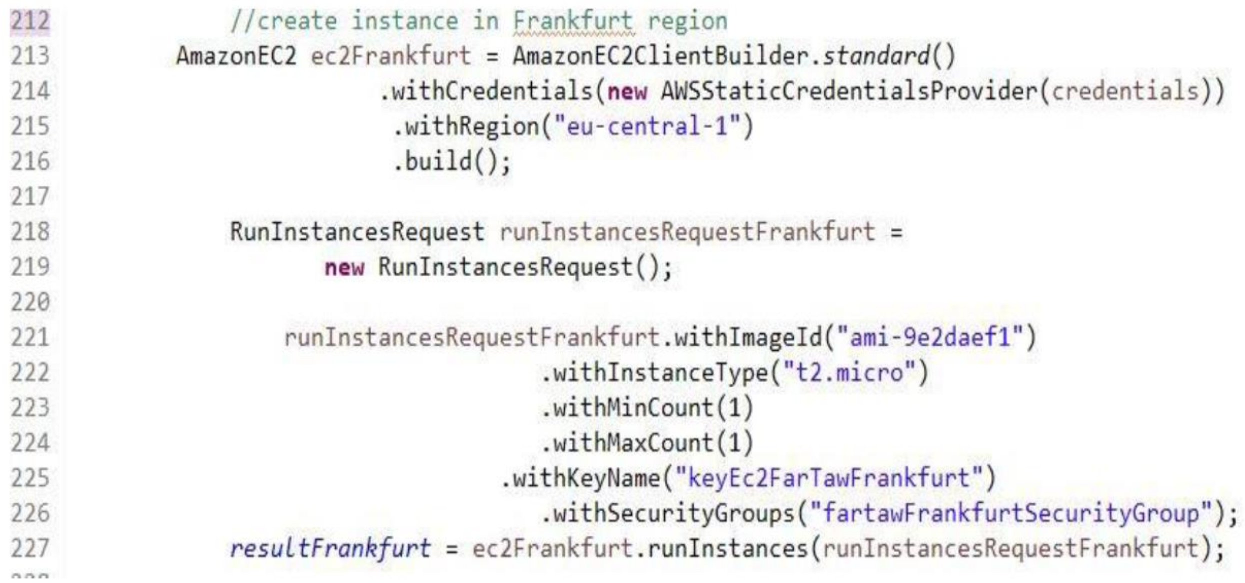


```
<terminated> CreateSecurityGroupApp2 [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Nov 20, 2017, 8:50:27 PM)
Found region ap-south-1
with endpoint ec2.ap-south-1.amazonaws.com
Found region eu-west-2
with endpoint ec2.eu-west-2.amazonaws.com
Found region eu-west-1
with endpoint ec2.eu-west-1.amazonaws.com
Found region ap-northeast-2
with endpoint ec2.ap-northeast-2.amazonaws.com
Found region ap-northeast-1
with endpoint ec2.ap-northeast-1.amazonaws.com
Found region sa-east-1
with endpoint ec2.sa-east-1.amazonaws.com
Found region ca-central-1
with endpoint ec2.ca-central-1.amazonaws.com
Found region ap-southeast-1
with endpoint ec2.ap-southeast-1.amazonaws.com
Found region ap-southeast-2
with endpoint ec2.ap-southeast-2.amazonaws.com
Found region eu-central-1
with endpoint ec2.eu-central-1.amazonaws.com
Found region us-east-1
with endpoint ec2.us-east-1.amazonaws.com
Found region us-east-2
with endpoint ec2.us-east-2.amazonaws.com
Found region us-west-1
with endpoint ec2.us-west-1.amazonaws.com
Found region us-west-2
with endpoint ec2.us-west-2.amazonaws.com
```

3. Write a method to run an instance from the list of regions ("Frankfurt/Ireland") from the previous step.

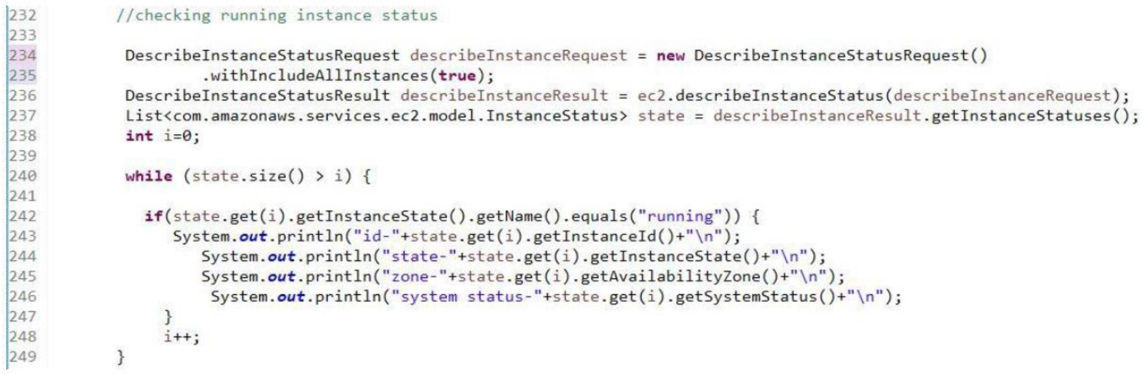
(i) **Use <keypair, security groups, number of instances, etc> as parameters to start an instance.**

In order to create an ec2 instance in a different region, at first we copied the AMI to the new region. In this case, we copied it to Frankfurt region. After that we created security group and keypair for that region. Then by declaring the region, instance type, AMI ID, key pair and security group, we created the ec2 instance in Frankfurt.



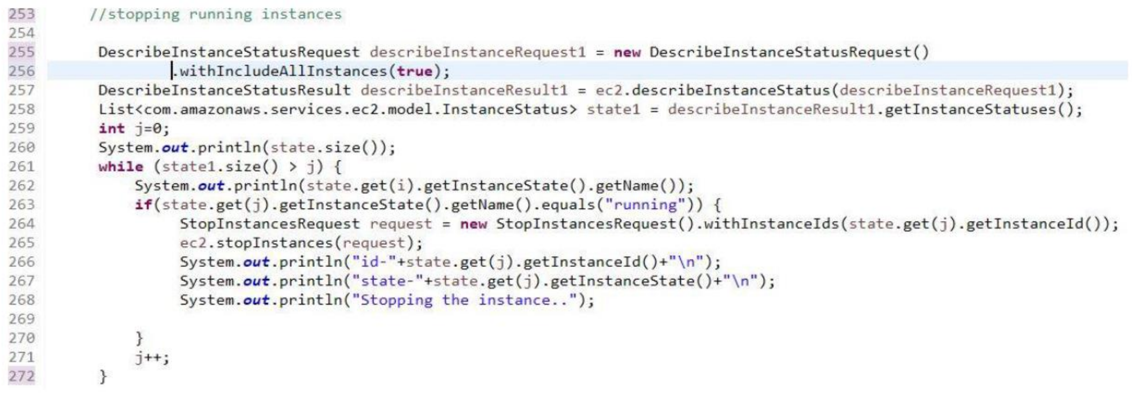
```java
            //create instance in Frankfurt region
        AmazonEC2 ec2Frankfurt = AmazonEC2ClientBuilder.standard()
                      .withCredentials(new AWSStaticCredentialsProvider(credentials))
                       .withRegion("eu-central-1")
                       .build();

            RunInstancesRequest runInstancesRequestFrankfurt =
                   new RunInstancesRequest();

                runInstancesRequestFrankfurt.withImageId("ami-9e2daef1")
                                  .withInstanceType("t2.micro")
                                  .withMinCount(1)
                                  .withMaxCount(1)
                               .withKeyName("keyEc2FarTawFrankfurt")
                                  .withSecurityGroups("fartawFrankfurtSecurityGroup");
            resultFrankfurt = ec2Frankfurt.runInstances(runInstancesRequestFrankfurt);
```

4. Write a method to retrieve the status of your running instance(s).



```
        //checking running instance status

         DescribeInstanceStatusRequest describeInstanceRequest = new DescribeInstanceStatusRequest()
                 .withIncludeAllInstances(true);
         DescribeInstanceStatusResult describeInstanceResult = ec2.describeInstanceStatus(describeInstanceRequest);
         List<com.amazonaws.services.ec2.model.InstanceStatus> state = describeInstanceResult.getInstanceStatuses();
         int i=0;

         while (state.size() > i) {

           if(state.get(i).getInstanceState().getName().equals("running")) {
              System.out.println("id-"+state.get(i).getInstanceId()+"\n");
                 System.out.println("state-"+state.get(i).getInstanceState()+"\n");
                 System.out.println("zone-"+state.get(i).getAvailabilityZone()+"\n");
                  System.out.println("system status-"+state.get(i).getSystemStatus()+"\n");
             }
             i++;
        }
```

5. Write a method to stop an instance(s) that you started.



```
        //stopping running instances

         DescribeInstanceStatusRequest describeInstanceRequest1 = new DescribeInstanceStatusRequest()
                 .withIncludeAllInstances(true);
         DescribeInstanceStatusResult describeInstanceResult1 = ec2.describeInstanceStatus(describeInstanceRequest1);
         List<com.amazonaws.services.ec2.model.InstanceStatus> state1 = describeInstanceResult1.getInstanceStatuses();
         int j=0;
         System.out.println(state.size());
         while (state1.size() > j) {
             System.out.println(state.get(i).getInstanceState().getName());
             if(state.get(j).getInstanceState().getName().equals("running")) {
                  StopInstancesRequest request = new StopInstancesRequest().withInstanceIds(state.get(j).getInstanceId());
                  ec2.stopInstances(request);
                  System.out.println("id-"+state.get(j).getInstanceId()+"\n");
                  System.out.println("state-"+state.get(j).getInstanceState()+"\n");
                  System.out.println("Stopping the instance..");

             }
             j++;
         }
```

Exercise b: Identify ways of creating the CloudWatch clients.

There are two ways to create a cloudwatch client using one of the following techniques.

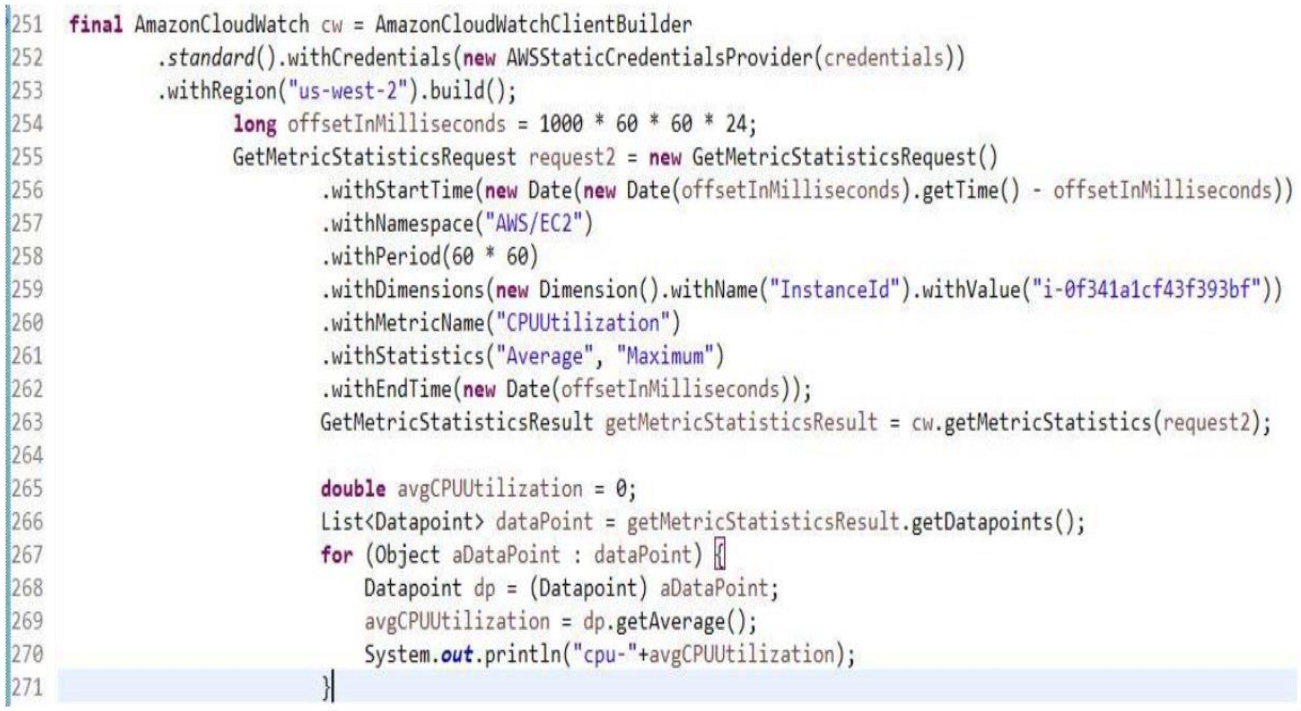
Factory method

The easiest way to create and run cloudwatch client quickly is to use the AWS cloudwatch factory method and provide user's credential profile, which identifies the set of credentials user want to use from the AWS credential file and credential profile. A region parameter is also required for this method.

Service builder

A more robust way to connect to Amazon CloudWatch is through the service builder. This allows user to specify credentials and other configuration settings in a configuration file. These settings can then be shared across all clients so that user only have to specify their settings once.

Write a Java program to monitor the status of your EC2 instances



```
final AmazonCloudWatch cw = AmazonCloudWatchClientBuilder
        .standard().withCredentials(new AWSStaticCredentialsProvider(credentials))
        .withRegion("us-west-2").build();
              long offsetInMilliseconds = 1000 * 60 * 60 * 24;
              GetMetricStatisticsRequest request2 = new GetMetricStatisticsRequest()
                      .withStartTime(new Date(new Date(offsetInMilliseconds).getTime() - offsetInMilliseconds))
                      .withNamespace("AWS/EC2")
                      .withPeriod(60 * 60)
                      .withDimensions(new Dimension().withName("InstanceId").withValue("i-0f341a1cf43f393bf"))
                      .withMetricName("CPUUtilization")
                      .withStatistics("Average", "Maximum")
                      .withEndTime(new Date(offsetInMilliseconds));
                      GetMetricStatisticsResult getMetricStatisticsResult = cw.getMetricStatistics(request2);

                      double avgCPUUtilization = 0;
                      List<Datapoint> dataPoint = getMetricStatisticsResult.getDatapoints();
                      for (Object aDataPoint : dataPoint) {
                          Datapoint dp = (Datapoint) aDataPoint;
                          avgCPUUtilization = dp.getAverage();
                          System.out.println("cpu-"+avgCPUUtilization);
                      }
```

References:

[1] "Run an Amazon EC2 Instance - AWS SDK for Java", Docs.aws.amazon.com, 2017. [Online]. Available: http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/run-instance.html. [Accessed: 16- Nov- 2017].

[2] "Create a Key Pair - AWS SDK for Java", Docs.aws.amazon.com, 2017. [Online]. Available: http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/create-key-pair.html. [Accessed: 17- Nov- 2017].

[3] "Create an Amazon EC2 Security Group - AWS SDK for Java", Docs.aws.amazon.com, 2017. [Online]. Available: http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/create-security-group.html. [Accessed: 18- Nov- 2017].

[4] G. API, "Getting state of EC2 instance Java API", Stackoverflow.com, 2017. [Online]. Available: https://stackoverflow.com/questions/8828964/getting-state-of-ec2-instance-java-api. [Accessed: 18- Nov- 2017].

[5] "Amazon CloudWatch — AWS SDK for PHP 2.8.12 documentation", Docs.amazonaws.cn, 2017. [Online]. Available: http://docs.amazonaws.cn/en_us/aws-sdk-php/guide/latest/service-cloudwatch.html. [Accessed: 18- Nov- 2017].