```java
package uber.oracle.apps.ap.touchless.aws;

import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.S3ClientOptions;
import com.amazonaws.services.s3.model.*;

import java.io.*;
import java.nio.file.Files;
import java.util.LinkedList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public final class AWSUtil {
  private static final Logger LOGGER = Logger.getLogger(AWSUtil.class.getName());
  private static AWSCredentials awsCredentials;
  private static final String CONTENT_MD5_DIDNOT_MATCH = "The Content-MD5 you specified did not match what we received.
  private static final String INVALID_AWS_CREDENTIALS = "Cannot proceed with null/invalid AWS Credentials";
  private static final String INVALID_CREDENTIALS_FILE = "Cannot load the credentials from the credential profiles file
                                                         "Please make sure that your credentials file is at the cor
                                                         "location (~/.aws/credentials), and is in valid format.";
  private static final String AWSSERVICE_EXC_MSG = "Encountered  AmazonServiceException and retried more than 4 times f
                                                   "ErrorType: %s ;HTTP StatusCode: %s ;RequestID: %s" ;

  static {
    try {
      awsCredentials = getAWSCredentials();
      validateAWSCredentials();
    } catch (Exception e) {
      LOGGER.log(Level.SEVERE, "Exception " + e.getMessage() + " occurred while getting AWS Credentials!");
    }
  }

  //We should not be instantiating this class
  private AWSUtil() {
    LOGGER.log(Level.SEVERE, "AWSUtil private constructor should never be instantiated. Throwing RuntimeException.");
    throw new RuntimeException("AWSUtil private constructor should never be instantiated");
  }

  private static void validateAWSCredentials() throws Exception {
    if (awsCredentials == null || awsCredentials.getAWSAccessKeyId().isEmpty()
                               && awsCredentials.getAWSSecretKey().isEmpty()) {
      throw new RuntimeException(INVALID_AWS_CREDENTIALS);
    }
  }

  private static AWSCredentials getAWSCredentials() throws IOException {
    try {
      awsCredentials = new ProfileCredentialsProvider().getCredentials();
      return awsCredentials;
    } catch (Exception e) {
      throw new AmazonClientException( INVALID_CREDENTIALS_FILE ,e);
    }
  }

  private static void printAWSErrorMessage(final AmazonServiceException ase) {
    LOGGER.log(Level.SEVERE, "Error Message:    " + ase.getMessage());
    LOGGER.log(Level.SEVERE, "HTTP Status Code: " + ase.getStatusCode());
    LOGGER.log(Level.SEVERE, "AWS Error Code:   " + ase.getErrorCode());
    LOGGER.log(Level.SEVERE, "Error Type:       " + ase.getErrorType());
    LOGGER.log(Level.SEVERE, "Request ID:       " + ase.getRequestId());
```

```
  }

  public static List<Bucket> listS3Buckets(Region region) throws IOException, RuntimeException {
    try {
      validateAWSCredentials();
      final AmazonS3 s3Client = new AmazonS3Client(awsCredentials);
      s3Client.setRegion(region);
      return s3Client.listBuckets();
    } catch (final AmazonServiceException ase) {
      printAWSErrorMessage(ase);
      throw ase;
    } catch (final Exception e) {
      throw new RuntimeException(String.format("Exception occurred while listing buckets for %s",
          region));
    }
  }

  public static List<S3ObjectSummary> listS3Objects(Region region, String bucketName, String prefix, int limit, String
    try {
      validateAWSCredentials();
      // final AmazonS3 s3Client = new AmazonS3Client(awsCredentials); //this way of initializing client has been depre
      // s3Client.setRegion(region);
      final AmazonS3 s3Client = getS3Client(region.getName());
      ListObjectsV2Request listObjReq = new ListObjectsV2Request().withMaxKeys(limit).withPrefix(prefix);
      ListObjectsV2Result result;
      List<S3ObjectSummary> objectSummaryList = new LinkedList<S3ObjectSummary>();
      if (bucketName != null)
        listObjReq.withBucketName(bucketName);
      if (limit > 0)
        listObjReq.withMaxKeys(limit);
      if (startAfter != null)
        listObjReq.withStartAfter(startAfter);
      if (prefix != null)
        listObjReq.withPrefix(prefix);
      do {
        result = s3Client.listObjectsV2(listObjReq);
        for (S3ObjectSummary objectSummary : result.getObjectSummaries()) {
          objectSummaryList.add(objectSummary);
        }
        System.out.println("Next Continuation Token : " + result.getNextContinuationToken());
        listObjReq.setContinuationToken(result.getNextContinuationToken());
      } while (result.isTruncated() == true);
      return objectSummaryList;
    } catch (final AmazonServiceException ase) {
      printAWSErrorMessage(ase);
      throw ase;
    } catch (final Exception e) {
      throw new RuntimeException(String.format("Exception occurred while listing buckets for %s , %s",
          region, e));
    }
  }

  public static FileOutputStream getS3Object(String bucketName,
                                             String keyName,
                                             Region region,
                                             String localPath) throws RuntimeException {
    FileOutputStream fos = null;
    try {
      validateAWSCredentials();
//      final AmazonS3 s3Client = new AmazonS3Client(awsCredentials);
//      s3Client.setRegion(region);
      final AmazonS3 s3Client = getS3Client(region.getName());
      S3Object obj = s3Client.getObject(bucketName, keyName);
      S3ObjectInputStream s3is = obj.getObjectContent();
      String[] keyNameArray = keyName.split(File.separator);
      final String fileName = keyNameArray[keyNameArray.length - 1];
      fos = new FileOutputStream(localPath + fileName);
      byte[] read_buf = new byte[1024];
      int read_len = 0;
      while ((read_len = s3is.read(read_buf)) > 0) {
        fos.write(read_buf, 0, read_len);
      }
```

```
      s3is.close();
      fos.close();
      return fos;
    } catch (AmazonServiceException ase) {
      printAWSErrorMessage(ase);
      throw ase;
    } catch (FileNotFoundException e) {
      throw new RuntimeException(String.format("Unable to find file  %s , %s, %s",
          bucketName, keyName, region));
    } catch (Exception e) {
      throw new RuntimeException(String.format("Exception occurred while listing buckets for %s , %s",
          keyName, e));
    } finally {
      if (fos != null) {
        try {
          fos.close();
        } catch (IOException e) {
          e.printStackTrace();
        }
      }

    }
  }

  public static InputStream getS3ObjectStream(String bucketName,
                                              String keyName,
                                              Region region,
                                              oracle.apps.fnd.cp.request.CpContext cpContext) {

    try {
      long start = System.currentTimeMillis();

      validateAWSCredentials();
      final AmazonS3 s3Client = getS3Client();
      s3Client.setRegion(region);

      cpContext.getLogFile().writeln("create client: " + Long.toString(System.currentTimeMillis() - start), 1);
      start = System.currentTimeMillis();

      S3Object obj = s3Client.getObject(bucketName, keyName);

      cpContext.getLogFile().writeln("get object: " + Long.toString(System.currentTimeMillis() - start), 1);
      start = System.currentTimeMillis();
      return obj.getObjectContent();
    } catch (Exception e) {
      throw new RuntimeException(String.format("Exception occurred while getting input stream for %s, %s", keyName, e))
    }
  }

  public static void putS3File(String bucketName, String keyName, Region region, File object, ObjectMetadata metadata)
    try {
      validateAWSCredentials();
//      final AmazonS3 s3Client = new AmazonS3Client(awsCredentials);
      final AmazonS3 s3Client = getS3Client(region.getName());
//      s3Client.setRegion(region);
      PutObjectRequest request = new PutObjectRequest(bucketName, keyName, object);
      request.withMetadata(metadata);
      s3Client.setS3ClientOptions(S3ClientOptions.builder().disableChunkedEncoding().build());
      s3Client.putObject(request);
    } catch (Exception e) {
      throw new RuntimeException(String.format("Exception occurred while putting object for %s, %s", keyName, e));
    }
  }

  public static void putS3File(String bucketName, String s3Key, Region region, File input) {
    try (final InputStream stream = new FileInputStream(input)) {
      validateAWSCredentials();
      final AmazonS3 s3Client = getS3Client(region.getName());
      ObjectMetadata metadata = new ObjectMetadata();
      metadata.setContentLength(input.length());
      metadata.setContentType(Files.probeContentType(input.toPath()));
      s3Client.putObject( new PutObjectRequest(bucketName, s3Key, stream, metadata) );
```

```
    } catch (IOException e) {
      e.printStackTrace();
      throw new RuntimeException(String.format("Exception uploading file %s : %s",input.getName(), e));
    } catch (AmazonServiceException ase) {
      throw new RuntimeException(String.format("Encountered  AmazonServiceException for %s:- errorMessage: %s ;AWSError
                                               "ErrorType: %s ;HTTP StatusCode: %s ;RequestID: %s" ,s3Key, ase.getM
          ,ase.getErrorCode() ,ase.getErrorType() ,ase.getStatusCode() ,ase.getRequestId() ));
    } catch (Exception ace) {
      throw new RuntimeException(String.format("Encountered  AmazonClientException for %s:- errorMessage: %s"
          ,s3Key, ace.getMessage()));
    }
  }


  //TODO move this method to generic FileUtils
  public static File createFoldersIfNotExist(final String folderPathString) throws IOException {
    final File folderPath = new File(folderPathString);
    if (!folderPath.exists()) {
      System.out.println("Creating path: " + folderPath);
      final boolean directoryCreated = folderPath.mkdirs();
      if (!directoryCreated) {
        throw new IOException("Failed to create the directory: " + folderPathString);
      }
    }
    return folderPath;
  }


  //TODO move this method to generic FileUtils
  public static File createFileIfNotExist(final String filePathString) throws IOException {
    final File filePath = new File(filePathString);
    final String folderPathString = filePath.getParent();
    final File path = createFoldersIfNotExist(folderPathString);

    if (!filePath.exists()) {
      System.out.println("Creating file: " + filePath);
      filePath.createNewFile();
    }
    return filePath;
  }


  public static AmazonS3 getS3Client(String region) {
    return AmazonS3ClientBuilder.standard()
        .withCredentials(new AWSStaticCredentialsProvider(awsCredentials))
        .withRegion(region)
//        .withRegion(Regions.US_WEST_2.getName())
        .build();
  }

  public  static AmazonS3 getS3Client() {
    return AmazonS3ClientBuilder.standard()
            .withCredentials(new AWSStaticCredentialsProvider(awsCredentials))
            .withRegion(Regions.DEFAULT_REGION).build();
  }

  public static AmazonS3 getS3Client (final String profile , final String region) {
    AmazonS3 client = null;

    if (profile != null && !profile.isEmpty() && region != null && !region.isEmpty()) {
      client = getSpecificS3Client(profile, region);
    } else {

      if (region != null && !region.isEmpty())
        client = getS3Client(region);
      else
        client = getS3Client();
    }
    return client;
  }
```

```
  private static AmazonS3 getSpecificS3Client(final String profile ,final String region) {
    return AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider(profile))
                .withRegion(region)
                .build();
  }

  public static void deleteS3File(AmazonS3 s3Client, String region, String bucketName, String keyName)
      throws SdkClientException {
    if (s3Client == null) {
      s3Client = getS3Client(region);
    }
    s3Client.deleteObject(new DeleteObjectRequest(bucketName, keyName));
  }


  public static void deleteS3File(AmazonS3 s3Client, String bucketName, String keyName)
      throws SdkClientException {
    if (s3Client == null) {
      s3Client = getS3Client();
    }
    s3Client.deleteObject(new DeleteObjectRequest(bucketName, keyName));
  }


  public static void moveS3File(AmazonS3 s3Client ,String fromBucket, String fromKey, String toBucket, String toKey)
      throws AmazonServiceException {
    try {
      if (s3Client == null) {
        s3Client = getS3Client(Regions.US_EAST_1.getName());
      }
      s3Client.copyObject(fromBucket, fromKey, toBucket, toKey);
      s3Client.deleteObjects(fromBucket,fromKey);
    } catch (AmazonServiceException e) {
      e.printStackTrace();
      throw new RuntimeException(String.format("Encountered  ASE during moveS3File for %s:- errorMessage: %s ;AWSErrorC
                                               "ErrorType: %s ;HTTP StatusCode: %s ;RequestID: %s" ,fromKey, e.getM
          ,e.getErrorCode() ,e.getErrorType() ,e.getStatusCode() ,e.getRequestId() ));
    }
  }


}
```