Northeastern Illinois University

CS-207-2: Programming II

Research Lab: AES Encryption

Due: Friday November 30, 2018 at 11:59 PM

Goal:

The goal of this research lab is to investigate the AES encryption algorithm and encrypt a file using the Java implemented AES encryption algorithm while using exception handling techniques described in your course. In addition, students will become more familiar with referencing the Java 8 docs when working with new API classes.

The Problem:

You have been provided with a text file that you will eventually encrypt and decrypt using the AES Encryption Algorithm. This research lab will have no starter code as this will more accurately mimic the research process that you may encounter in a job situation or academic research situation. In addition, it will help to reinforce working with files, exception handling, and interfaces from the Java API.

Instructions:

- You should work in groups of 2-3 individuals. Groups of more than 3 are **not** permitted.
- Each group should submit ONE lab write-up. It is the responsibility of each group member to ensure that their name is on the write-up.
- The lab write-up should be typed! Type each question (and the question number) followed by your group's answer. **Convert your lab write-up to a .pdf.**
- You should use complete sentences and proper grammar in your write-up. Use spell-check! This counts as part of your grade.
- Do **not** copy/paste directly from your sources for your answers (this is called plagiarism). Instead, you should re-word the information in your own words.
- Submit the pdf and .java file (`AESEncryption.java`) to D2L by the specified due date.
- Each member of the group must turn in a **readable** digital copy of the peer assessment to an individual D2L folder by the assigned due date and time. The peer assessment counts as a significant part of your grade and you will receive a **zero** for that portion of the research lab grade if you do not turn it in.

Part A: Investigating the AES Encryption Algorithm

Use Google to investigate the AES encryption algorithm and answer the questions below.

Question #A.1

Who developed the AES Encryption algorithm? Provide some historical background on the AES Encryption algorithm.

Question #A.2

Describe some of the key aspects of the AES algorithm and its implementation.

Part B: Getting Started with AES in Java: Making an encrypt method

Question #B.1

Find the `Cipher` class in the Java 8 docs. In what package does it live? Import this package and create a class named `AESEncryption`. Create the `main` method, but leave it empty for now. Create a method named `encrypt` that does not return anything and does not take any parameters.

Question #B.2

The `Cipher` class refers to the different algorithms as transformations. However, the constructor to create a `Cipher` object is more complex than we would like. Instead, we can create a `Cipher` object using the `static` method `getInstance`. Click on the `getInstance` method that takes a `String` parameter. It provides you with a link to documentation for the `String` values you should use for different algorithms. What do you use for the `AES` algorithm? Use this static method with the correct parameter to create a `Cipher` object.

Question #B.3

Look at the `getInstance` method again. Does it throw any exceptions? What are they? Are they checked or unchecked? What happens if you try to compile your code at this point? Do you need to import any packages for these exceptions? Handle each exception by printing out a useful error message for the user. Compile your code.

Question #B.4

Next, you need to initialize your `Cipher` object. In the `Cipher` class find an initialization method (There are quite a few!!). All of them take an `int` as the first parameter. This `int` represents the operation mode (i.e. are you encrypting, decrypting, etc). Does the `Cipher` class have any constants that look like they might be useful? Pick the right one!

Question #B.5

Next, regardless of which initialization method you use, you have to provide a `Key` object - so we have to generate a key before we can initialize the `Cipher` object. To help you with this, you have been provided with the code below. Look up `SecretKey` in the Java 8 docs - what is it (i.e. class, abstract class, etc). What does it inherit from? Do any of the methods in the code below throw exceptions? Where should this code go relative to the creation of your `Cipher` object? Then, call the initialization method that uses two parameters to initialize the `Cipher` object. Compile!!

```
KeyGenerator keyGen = KeyGenerator.getInstance(“AES”);
keyGen.init(128);
SecretKey key = keyGen.generateKey();
```

Question #B.6
Which method in the `Cipher` class looks like it is used to encrypt or decrypt data? Pick the method that takes one parameter. What is the type of this parameter? What does the method return? In order to use this method, we will need to be able to read from the file that we are trying to encrypt and output an encrypted version of it (and save the key!).

Part C: Getting Started With Files

Since you will be working with encrypting files (and storing a key in a file), it is important to research the necessary Java 8 API classes for working with files.

Question #C.1
You need one specific Java package imported in order to work with `File` objects. What is it? Import this package before your class definition. Each file that you read from needs to be represented by a `File` object.

Question #C.2
Download the `data.txt` file from the NeededFiles.zip file and save it in the same folder as your Java file. Look at the data in this file so that you can see what it contains (for when you decrypt its encrypted version). Do you recognize this famous speech? Who gave this speech (also, when and where)?

Question #C.3
You will need to be able to read from your `File` objects. To do this, you will be using the `FileInputStream` class. Why is it necessary to use the `FileInputStream` class instead of the `Scanner` class (Hint: Look at the method that you have to use in the `Cipher` class to encrypt/decrypt data)? Find the constructor for the `FileInputStream` class in the Java 8 docs that takes a `File` object as a parameter. What type/name of the exception does it throw? Why do you need to know this before you create a `FileInputStream` object? Create a `FileInputStream` object for your `File` object. Make sure to handle the `FileNotFoundException`. Hint: Sure hope you're using multiple catch blocks as opposed to a bunch of separate try-catch statements. Compile.

Question #C.4
The `read` method of the `FileInputStream` class requires an initialized byte array for a parameter. Create and initialize a byte array. How do you know the `length` of the array? (Hint: Read the Java docs for the `File` class. Remember, in order to declare and create an array, you need an `int` for the size - you may find casting helpful!). Make sure to handle any exceptions and print out helpful error messages. Compile.

Question #C.5 - Coding Only
Read from your `data.txt` file. Make sure to handle exceptions. Compile.

Question #C.6
Now that you have the data from the file, use the method from #B.6 to encrypt the data (make sure to assign it to the correct return type). Does this method throw any exceptions? What are they? Do they all inherit from the same Exception as any other Exception you have already caught? If so, find a way to use this superclass exception for these particular exceptions!

Part D: Writing the Encrypted Data

Question #D.1
You need to write the encrypted byte array to a file using a `FileOutputStream` object. Why is it necessary to use the `FileOutputStream` class instead of the `Scanner` class? Create a `FileOutputStream` object and write the encrypted data to a new file (yes, you will need a method from the `FileOutputStream` class to write the data). Compile.

Question #D.2
In order to decrypt, you will need the secret key that you generated - so you have to save it. Find the `Key` interface in the Java docs. Is there a method that returns the encoded key? What is the type that is returned from this method? Create a `FileOutputStream` object and use the appropriate method to write the key to a new file. Close all your files. Compile.

Question #D.3 - Coding Only
Run your code and call the `encrypt` method in the `main` method. Verify that the contents of the encrypted file look like nonsense and that the contents of the file for the key look like nonsense.

Part E: The decrypt Method

Decrypting an encrypted file is very similar to encrypting, however you need to supply the unique key for decrypting (which is why we saved it in a file).

Question #E.1 - Coding Only
Create a method named `decrypt` that does not take any parameters and does not return anything. Using the same methodology as for Part D, read (and save using `byte` arrays) the encrypted file and the key. Since you already have a key, it is easy to create a key object using the following syntax (where `keyBytes` is the name of the byte array for your key):

```
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
```

Yes - compile. Compile frequently.

Question #E.2
Create a `Cipher` object the same way that as in the `encrypt` method. However, when initializing the `Cipher` object, which operational mode should you use? After initializing, use your `Cipher` object to perform the decryption (same method call as in the `encrypt` method). Create a `FileOutputStream` object and use the appropriate method to write the decrypted data to a new file. Close all your files. Compile. Comment out your `encrypt` method in `main` and call the `decrypt` method.

Question #E.3 - Coding Only

At this point, you have two methods: `encrypt()` and `decrypt().` However, your code is not user-friendly. It is hard-coded to only read from specifically-named files. And you have to uncomment and comment the methods that you want to use. Write code to make this user-friendly by implementing the following changes:

1. Prompt the user to enter `E` for encryption and `D` for decryption.
2. If they enter `E` for encryption, use the `encrypt()` method and prompt them to enter the complete file name that is to be decrypted.
3. If they enter `D` for decryption, use the `decrypt()` method and prompt them to enter the complete file name that is to be decrypted, followed by a prompt to enter the complete file name for the key file.

Part F: Final Summary

Whenever you complete a project, it is important to do a project retrospection (i.e, what went well, what needed improvement, etc).

Question #F.1

Which questions were you able to answer on your own (i.e. as a group, by researching the Java docs, etc.) and which questions required you to seek help from a peer leader?

Question #F.2

What would you do/add to make the resulting output even more user-friendly?

Question #F.3

What was a main lecture topic from class that was reinforced by this research lab?

Question #F.4

Based on your experience with this project, how would you approach future projects similar in nature to this one, assuming you were not provided with instructions?