Information System Security

HW#1

컴퓨터공학부

20143079 오예진

1.

* 소스코드

//컴퓨터공학부 20143079 오예진

package aes;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.security.InvalidAlgorithmParameterException;

import java.security.InvalidKeyException;

import java.security.Key;

import java.security.KeyFactory;

import java.security.NoSuchAlgorithmException;

import java.security.SecureRandom;

import java.util.Scanner;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.PBEKeySpec;

import javax.crypto.spec.SecretKeySpec;

public class AesMain {

public static void main(String[] args) throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, InvalidAlgorithmParameterException, IllegalBlockSizeException, BadPaddingException {

//////////////////////////////////////////////////

//암호화하고자 하는 파일 읽기

//////////////////////////////////////////////////

String file\_location = "";

Scanner file\_location\_scan = new Scanner(System.in);

file\_location = file\_location\_scan.nextLine(); //file이름을 사용자로부터 받는다

File input\_file = new File("C:\\Users\\Yejin\\eclipse-workspace\\HW1\_AES\\src\\aes\\"+file\_location); //암호화할 파일 읽기

BufferedReader input\_text = null;

String input = ""; //초기의 input.txt에서 읽은 결과를 저장하는 String

try{

input\_text = new BufferedReader(new FileReader(input\_file));

input = input\_text.readLine(); //input.txt에서 읽은 내용을 저장

}catch (FileNotFoundException e) {

e.printStackTrace();

}catch (IOException e) {

e.printStackTrace();

}

//AES KEY(대칭키)를 사용자로부터 입력받아 생성 (Generate an AES KEY)

//임의의 키가 아닌 사용자가 원하는 키를 사용하기 위해 다음과 같이 구현

String user\_key = "";

Scanner user\_input = new Scanner(System.in);

user\_key = user\_input.nextLine();

Key aes\_key = new SecretKeySpec(user\_key.getBytes(),"AES");

//초에 사용하는 Initial Vector 생성. Random하게 생성하며 16byte여야 함

SecureRandom s\_random = SecureRandom.getInstance("SHA1PRNG");

byte[] iv = new byte[16];

s\_random.nextBytes(iv);

IvParameterSpec AES\_IV = new IvParameterSpec(iv);

//AES, CBC모드, Padding=PKCS5

//(Create a cipher object for AES in ECB mode and PKCS5 padding)

Cipher aes\_Cipher;

aes\_Cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");

//\*\*\*plaintext1 => 초기 input.txt에 있는 text \*\*\*

//\*\*\*plaintext2 => input.txt의 text를 암호화 한 후 다시 복호화 한 text. 따라서 plaintext1과 동일 \*\*\*

//////////////////////////////////////////////////

//암호화 Encrypt

//////////////////////////////////////////////////

aes\_Cipher.init(Cipher.ENCRYPT\_MODE, aes\_key, AES\_IV); //IV포함하므로 끝 parameter AES\_IV 추가

byte[] plaintext1 = input.getBytes(); //(input.txt에 있던 내용) to (byte)

byte[] ciphertext = aes\_Cipher.doFinal(plaintext1);

//////////////////////////////////////////////////

//복호화 Decrypt

//////////////////////////////////////////////////

aes\_Cipher.init(Cipher.DECRYPT\_MODE, aes\_key, AES\_IV); //IV포함하므로 끝 parameter AES\_IV 추가

byte[] plaintext2 = aes\_Cipher.doFinal(ciphertext); //암호화된 ciphertext를 복호화

//////////////////////////////////////////////////

//암호화해서 file에 쓰기

//////////////////////////////////////////////////

File Encrypt\_file = new File("C:\\Users\\Yejin\\Desktop\\Encrypt\_file.txt"); //C:\\Users\\Yejin\\Desktop에 Encrypt\_file.txt을 생성하여 해당 내용을 저장

BufferedWriter encrypt\_write = null;

try{

encrypt\_write = new BufferedWriter(new FileWriter(Encrypt\_file));

encrypt\_write.write(ciphertext.toString()); //byte array는 .txt파일에 제대로 쓰이지 않기 때문에 toString()사용

encrypt\_write.flush();

}catch (IOException e) {

e.printStackTrace();

}

//////////////////////////////////////////////////

//복호화해서 file에 쓰기

//////////////////////////////////////////////////

File Decrypt\_file = new File("C:\\Users\\Yejin\\Desktop\\Decrypt\_file.txt"); //C:\\Users\\Yejin\\Desktop에 Decrypt\_file.txt을 생성하여 해당 내용을 저장

BufferedWriter decrypt\_write = null;

try{

decrypt\_write = new BufferedWriter(new FileWriter(Decrypt\_file));

decrypt\_write.write(new String(plaintext2)); //byte array를 확인하기 위해 string으로 변환

decrypt\_write.flush();

}catch (IOException e) {

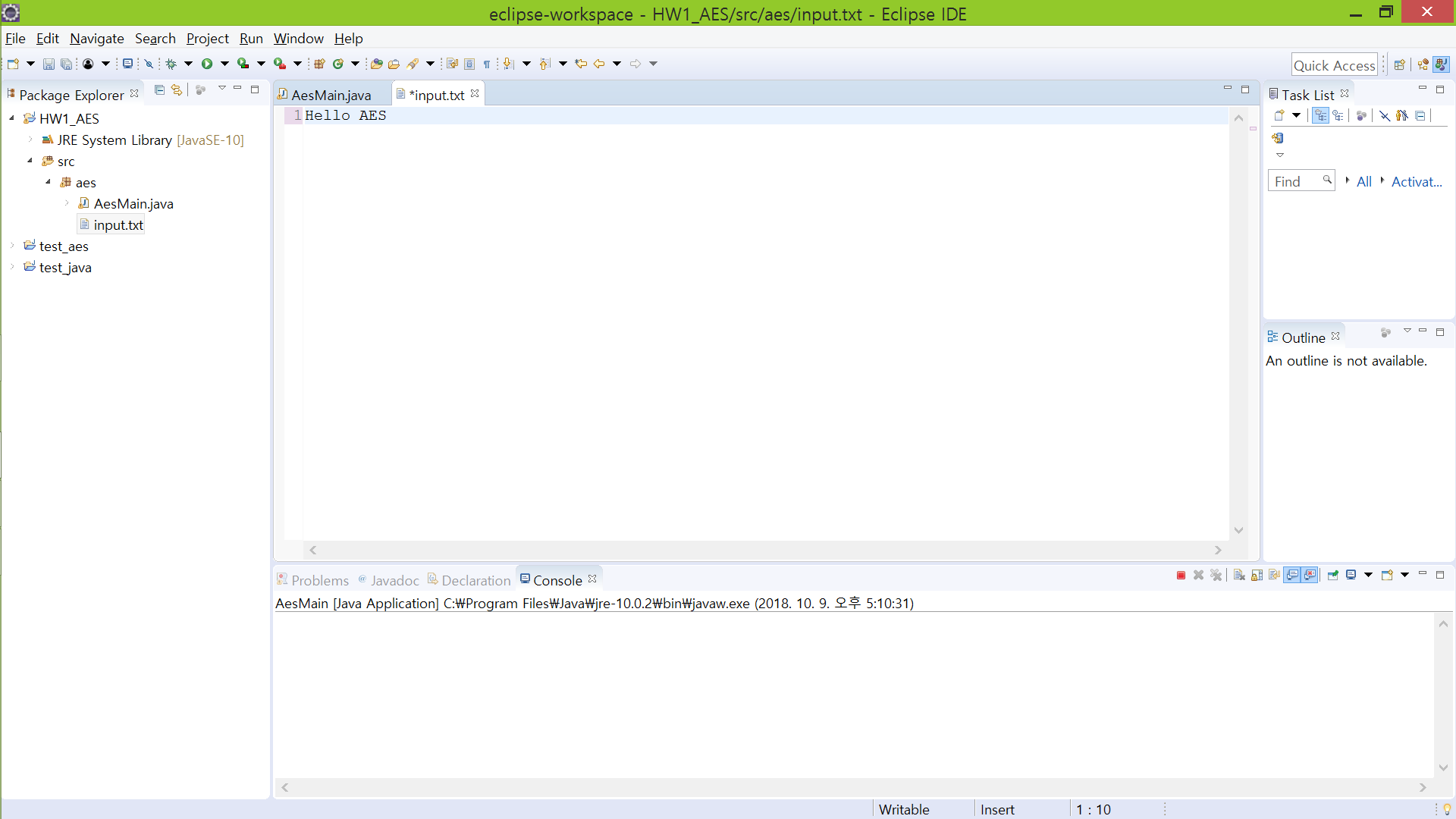
e.printStackTrace();

}

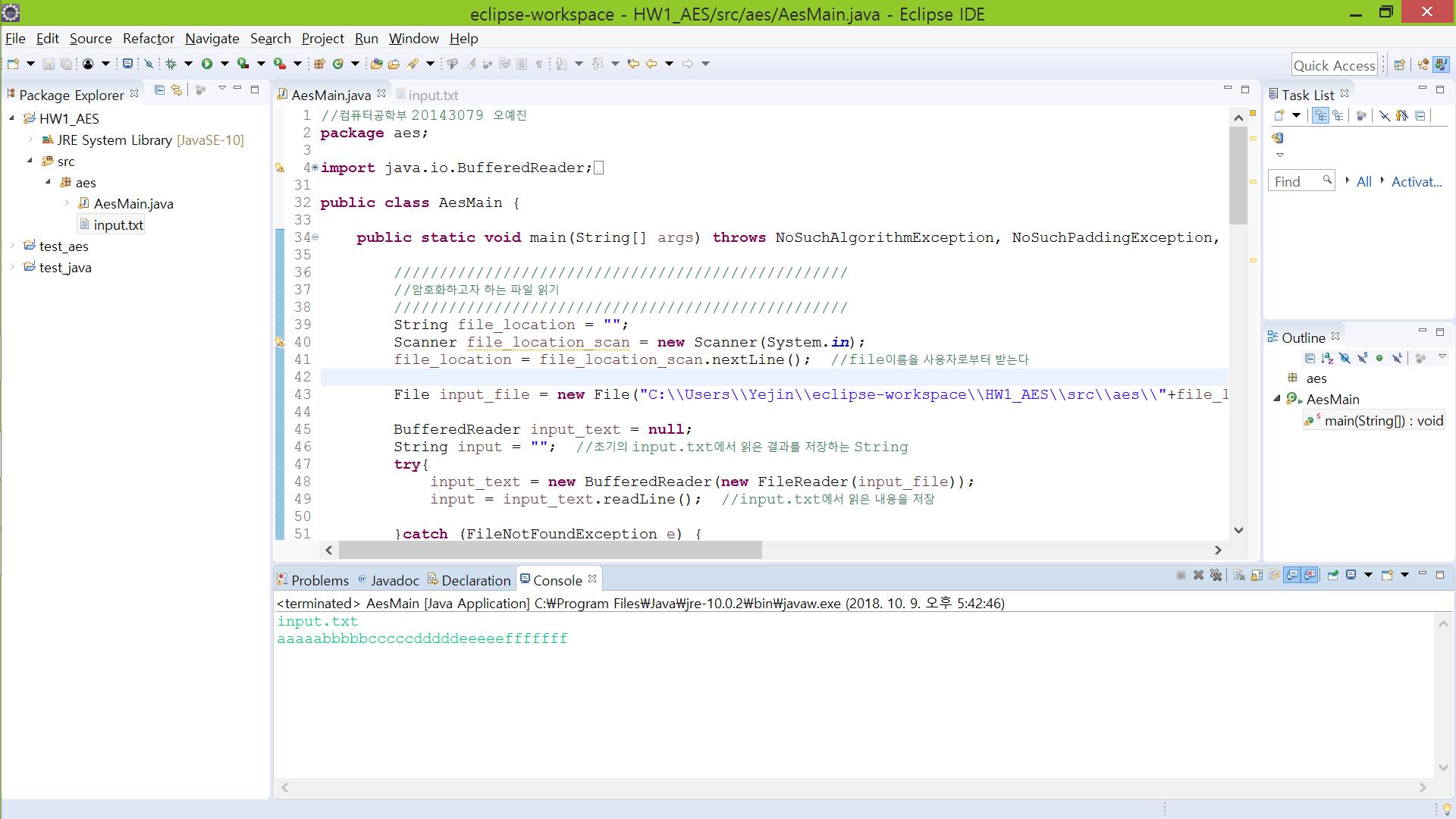
}

}

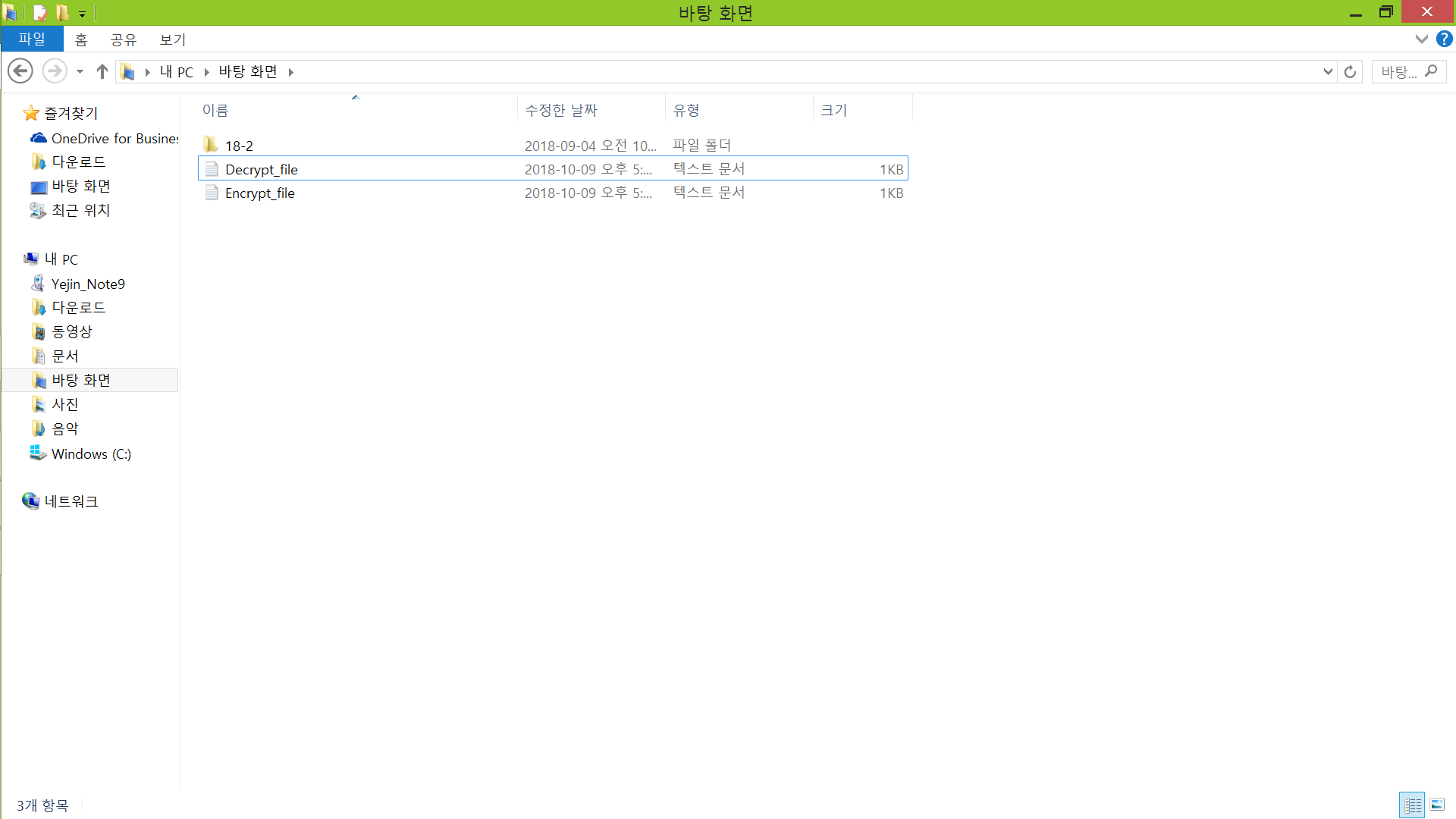
* 프로그램 시연



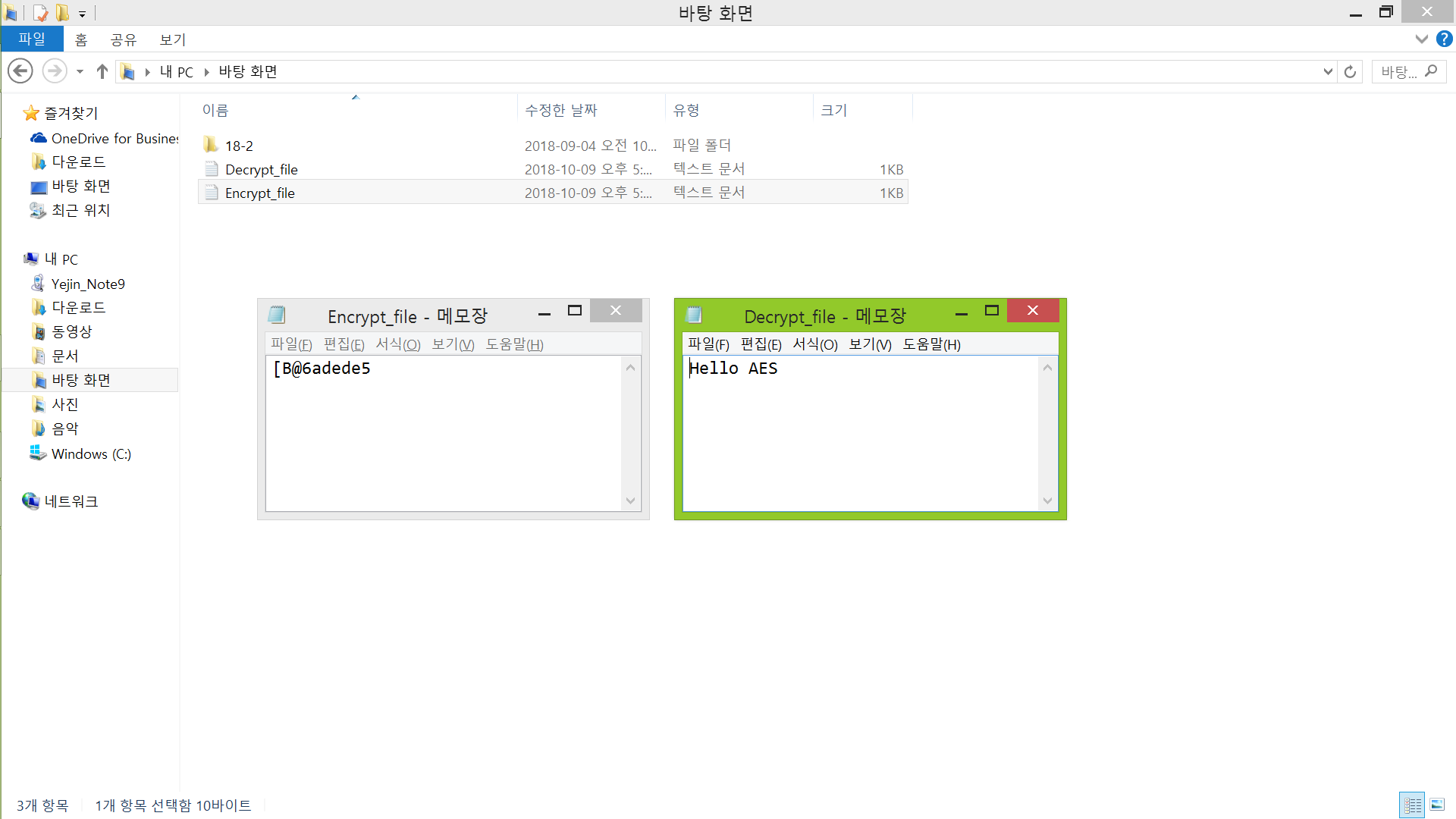
-input.txt에는 Hello AES라는 text가 들어있다. 이는 사용자가 암호화하고자 하는 파일명과 data를 의미한다.



-사용자로부터 평문 파일명 input.txt와 암호에 사용할 비밀키 값 aaaaabbbbbcccccdddddeeeeefffffff을 입력받는다.



지정해둔 경로인 바탕화면에 암호화된 파일 Encrypt\_file과 복호화된 파일 Decrypt\_file이 생성된 것을 확인할 수 있다.



암호화된 파일과 복호화된 파일 내용을 확인하면 다음과 같다. Decrypt\_file에는 암호화했다가 다시 복호화한 내용이 들어있기 때문에, 맨 처음 사용자가 암호화하고자 했던 파일 input.txt에 있는 내용 Hello AES와 동일하다.

* 짧은 소견

AES는 비밀키 하나로 데이터를 암호화하고 복호화 하는 대칭키 방식이다. 그 중 강력한 암호화 방식에 속하는 AES 256은 미국에서만 사용 가능하기 때문에 프로그램을 구현하기 전 JAVA의 security 문서에 추가로 policy.jar문서 2개를 넣어주었다.

AES는 블록암호화 방식을 사용하므로 패딩이 필요하고, 블록암호화 모드 중 CBC 방식을 사용하는데, 이는 각 블록이 이전 블록과 XOR연산을 거치며 암호화되는 방식을 말한다. 따라서 최초의 입력 IV이 필요하다. 이는 코드

Cipher aes\_Cipher;

aes\_Cipher = Cipher.*getInstance*("AES/CBC/PKCS5Padding");

에서 확인 가능하다.

비밀키 값은 사용자가 원하는 것을 설정하고자 하였으므로

String user\_key = "";

Scanner user\_input = **new** Scanner(System.***in***);

user\_key = user\_input.nextLine();

Key aes\_key = **new** SecretKeySpec(user\_key.getBytes(),"AES");

와 같이 구현하였다.

2.

* 소스코드

//컴퓨터공학부 20143079 오예진

**package** sha;

**import** java.security.MessageDigest;

**import** java.util.Random;

**public** **class** ShaMain {

**public** **static** **void** main(String[] args) {

String CHAR\_LIST = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890"; //random한 string을 만들기위한 array

Random random = **new** Random();

StringBuffer Random\_string = **new** StringBuffer();

String to\_sha; //sha256을 수행하기 위한 string

String finalresult\_sha=""; //sha256을 수행한 결과

String allzero = "0000"; //앞의 16bit가 모두 0인지 비교하기 위함.

String for\_compare=""; //결과string에서 앞의 16bit만큼을 잘라 결과값이 모두 0인지 확인

**int** Num\_of\_Zero\_Sha = 0; //결과에 맞는 String의 개수

**for**(**int** i=0; i<100000; i++) { //원하는 string을 찾기 위해 for loop사용.

Random\_string.setLength(0); //random string 초기화

**for**(**int** j=0; j<20; j++){ //임의의 string 길이는 20

**int** number = random.nextInt(62); //CHAR\_LIST 원소개수 62개

**char** ch = CHAR\_LIST.charAt(number);

Random\_string.append(ch);

}

to\_sha = Random\_string.toString(); //string으로 형변환하여 sha256으로 암호화

finalresult\_sha = *sha256*(to\_sha); //sha256을 이용해서 생성된 결과값

for\_compare = finalresult\_sha.substring(0,4); //앞의 16bit를 자른다

**if**(for\_compare.equals(allzero)) { //앞의 16자리가 모두 0이라면 원래 sha256결과값256bit과 해당string을 출력

System.***out***.println(*sha256*(to\_sha));

System.***out***.println(to\_sha);

Num\_of\_Zero\_Sha++;

**if**(Num\_of\_Zero\_Sha>1)

**break**; //조건에 맞는 3개의 string을 찾았으므로 여기까지 하고 break

}

}

}

**public** **static** String sha256(String to\_sha) { //sha256함수

String result\_sha = "";

**try** {

MessageDigest M\_digest = MessageDigest.*getInstance*("SHA-256"); //SHA-256 알고리즘

M\_digest.update(to\_sha.getBytes()); //messagedigest 클래스에 있는 method

//호출할 때마다 digest값 갱신

**byte** byteData[] = M\_digest.digest();

StringBuffer string\_buffer = **new** StringBuffer();

**for** (**int** i = 0; i < byteData.length; i++) {

string\_buffer.append(Integer.*toString*((byteData[i] & 0xff) + 0x100, 16).substring(1));

}

result\_sha = string\_buffer.toString();

}**catch**(Exception e){

**throw** **new** RuntimeException(e);

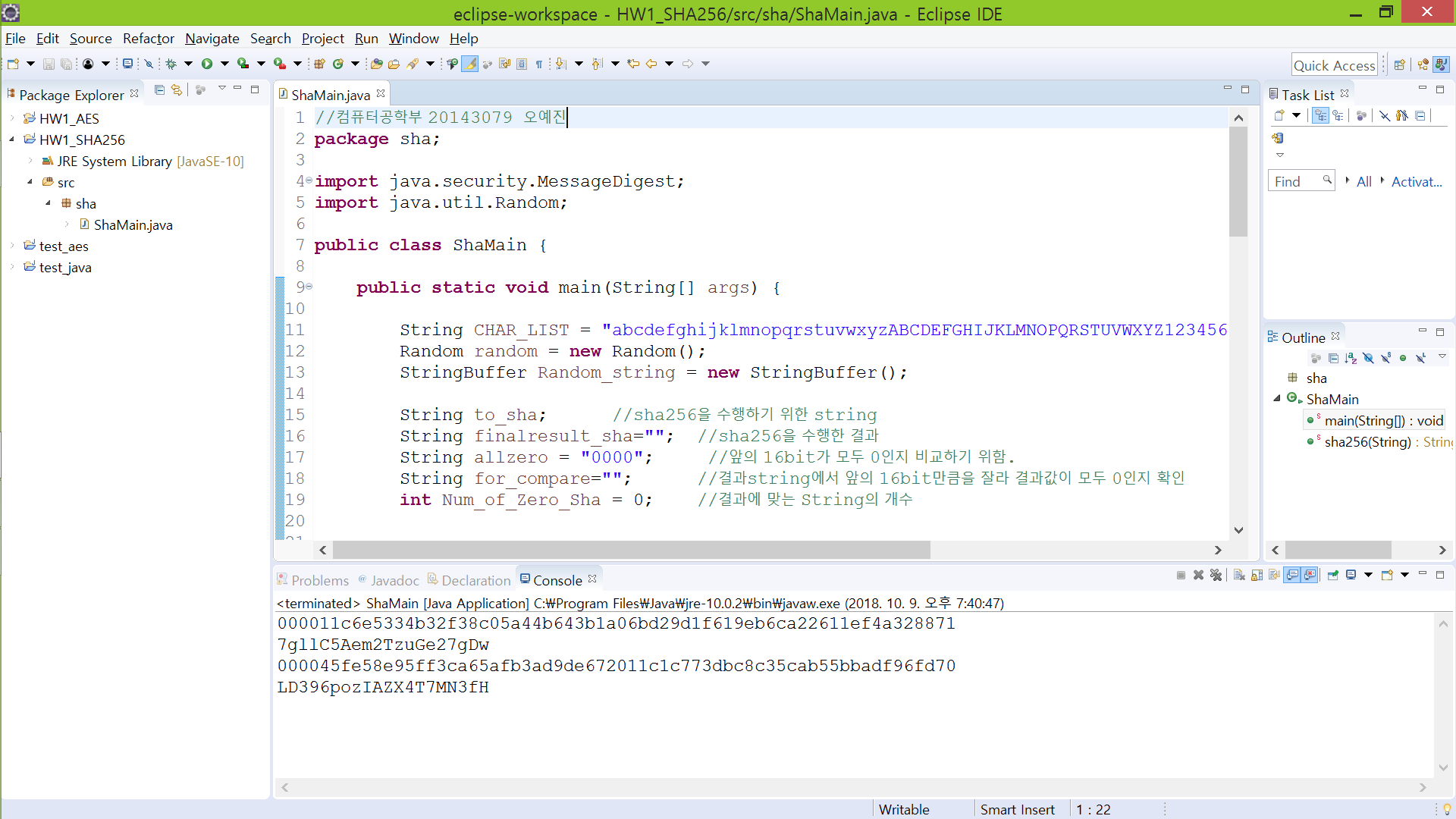
}

**return** result\_sha;

}

}

* 프로그램 시연



String과 그에 해당하는 Sha256 암호 결과값을 한 개 이상 확인할 수 있다. 콘솔 창에서 결과값 앞의 Char=”0000”을 통해16bit가 0인 것을 확인 할 수 있다.

* 짧은 소견

SHA256은 256bit의 출력길이를 가지며 복호화를 할 수 없으므로 별도의 키가 필요하지 않다. 키가 없기 때문에 SHA256을 통해서는 암호화의 결과값이 항상 같다.

MessageDigest.getInstance(X)에서 X에 사용하려는 알고리즘을 작성하면, 라이브러리에 원하는 알고리즘이 모두 구현 되어있기 때문에 따로 코드를 구현하지 않아도 된다. SHA256을 사용하기로 했기 때문에

MessageDigest.*getInstance*("SHA-256")

와 같이 구현하였다.

Random한 String을 만든 후 sha256으로 암호화를 100000번 시킨다. 이 때 Random String은

String CHAR\_LIST = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890"; //random한 string을 만들기위한 array

Random random = **new** Random();

StringBuffer Random\_string = **new** StringBuffer();

Random\_string.setLength(0); //random string 초기화

**for**(**int** j=0; j<16; j++){ //string 길이는 17

**int** number = random.nextInt(62); //CHAR\_LIST 원소개수 62개

**char** ch = CHAR\_LIST.charAt(number);

Random\_string.append(ch);

}

와 같이 구현하였다. Random String을 만들기 위해 array를 만들고, 그 배열안에서 랜덤하게 골라 String을 만드는 방식이다. 후에 Sha256으로 암호화 한 후, 결과값의 앞 16bit를 잘라내서 미리 만들어둔 “0000”과 비교한 후, 두 String이 일치하면 콘솔창에 해당하는 원본 String과 sha256암호화 결과값을 나타낸다.