# Java Codes

## Read and Write files in java:

Read files

\*create a FileInputStream.

FileInputStream input = null;

// try {

\*Create a file that hold the file need be read.

// File file = new File("D:/aaab.txt");

// input = new FileInputStream(file);

//

// } catch (IOException ex) {

// System.out.println("loi");

// }

// try {

\*read each byte in the file

// int noidung;

// while ((noidung = input.read()) != -1) {

// //for(int i = 0 ; i <= input.available(); i++ ){

\*turn bytes into char

// System.out.print((char) noidung);

// }

//

// } catch (IOException ex) {

// System.out.println("loi");

// }

//byte[] a = new byte[input.available()];

//System.out.println(a);

Write into a file:

\*input data which input into the file

Scanner sc = new Scanner(System.in);

String test;

System.out.println("Nhap test: ");

test = sc.nextLine();

System.out.println("Vua moi nhap: "+test);

try{

\*create a FileOutputStream.

FileOutputStream output = new FileOutputStream(("D:/aaa.txt");,true);

\*Turn the string into bytes and write them into the file

output.write(test.getBytes());

System.out.println("da viet");

} catch(IOException ex){

System.out.println("fall"); }

## Asymmetric:

Generate key pair

\*create a KeyPairGenerator

KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");

kpg.initialize(2048, sr);

\*create key pair

KeyPair kp = kpg.genKeyPair();

\*public key

PublicKey pubKey = kp.getPublic();

\*private key

PrivateKey priKey = kp.getPrivate();

Save public and private key

\*Public Key

FileOutputStream fos = new FileOutputStream("D:/file/pubKey.bin");

fos.write(pubKey.getEncoded());

fos.close();

\*Private Key

fos = new FileOutputStream("D:/file/priKey.bin");

fos.write(priKey.getEncoded());

fos.close();

Encryption

public static byte[] asymmetricEncryption(String data, PublicKey publicKey) throws IOException, InvalidKeyException, NoSuchAlgorithmException, NoSuchPaddingException, IllegalBlockSizeException, BadPaddingException {

Cipher cipher = Cipher.getInstance("RSA");

cipher.init(Cipher.ENCRYPT\_MODE, publicKey);

byte[] byteEncrypted = cipher.doFinal(data.getBytes());

return byteEncrypted;

\*Return byte need to be converted to string

}

Convert to string after encryption

String data = "abcde";

byte[] byteEncrypted = asymmetricEncryption(data, publicKey);

String encrypted = Base64.getEncoder().encodeToString(byteEncrypted);

System.out.println("encrpyted:" + encrypted);

Decryption

public static byte[] asymmetricDecrpytion(byte[] byteEncrypted, PrivateKey priKey) throws NoSuchAlgorithmException, InvalidKeyException, NoSuchPaddingException, IllegalBlockSizeException, BadPaddingException {

Cipher cipher = Cipher.getInstance("RSA");

cipher.init(Cipher.DECRYPT\_MODE, priKey);

byte[] byteDecrypted = cipher.doFinal(byteEncrypted);

return byteDecrypted;

\*Return byte need to be converted to string

}

Convert to string after encryption

byte[] byteDecrypted = asymmetricDecrpytion(byteEncrypted, privateKey);

String decrypted = new String(byteDecrypted);

System.out.println("decrypted: " + decrypted);

## Symmetric:

Generate key

public static SecretKey symmetricKeyGeneration() throws IOException, NoSuchAlgorithmException {

KeyGenerator keyGenerator = KeyGenerator.getInstance("DESede"); // generates secrey ket for specified algorithm

keyGenerator.init(168); // set size

SecretKey secretKey = keyGenerator.generateKey(); // put the generated secret key to the specified

return secretKey;

}

Encryption

public static String symmetricEncryption(String data, SecretKey secretKey) throws IOException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException, NoSuchAlgorithmException, NoSuchPaddingException {

Cipher cipher = null;

cipher = Cipher.getInstance("DESede");

cipher.init(cipher.ENCRYPT\_MODE, secretKey);

byte[] byteEncrypted = cipher.doFinal(data.getBytes());

String encrypted = Base64.getEncoder().encodeToString(byteEncrypted);

return encrypted;

}

Decryption

public static String symmetricDecryption(String dataEncrypted, SecretKey secretKey) throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException {

Cipher cipher = null;

cipher = Cipher.getInstance("DESede");

cipher.init(cipher.DECRYPT\_MODE, secretKey);

byte[] asd = dataEncrypted.getBytes();

byte[] dec = Base64.getDecoder().decode(asd);

byte[] dataDecrypted = cipher.doFinal(dec);

String decrypted = new String(dataDecrypted);

return decrypted;

}

## Hashing

public static String Hashing(String data) throws NoSuchAlgorithmException {

String HashedString = null;

MessageDigest md = MessageDigest.getInstance("SHA-512");

byte[] bytes = md.digest(data.getBytes(StandardCharsets.UTF\_8));

StringBuilder sb = new StringBuilder();

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

sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).substring(1));

}

HashedString = sb.toString();

return HashedString;

}