Ecole National Polytechnique ENP **Spécialité : DSIA**	**2°Année du 2°Cycle** **Module : Cryptographie (CRYPTO)** **Année 2021-2022**

TP 03: L'AES

Objectif :

Créer un programme qui permet de crypter et de décrypter un fichier en utilisant l'Algo AES.

Advanced Encryption Standard (AES)

Advanced Encryption Standard ou **AES** (litt. « norme de chiffrement avancé »), est un algorithme de chiffrement symétrique;, Il est actuellement le plus utilisé et le plus sûr.
L'algorithme prend en entrée un bloc de 128 bits (16 octets), la clé fait 128, 192 ou 256 bits
Pour une clé de 128, 192 ou 256, AES nécessite respectivement 10, 12 ou 14 tours.

Les flux d'entrée/sortie

Une entrée/sortie en Java consiste en un échange de données entre le programme et une autre source, par exemple la mémoire, un fichier, le programme lui-même… Pour réaliser cela, Java emploie ce qu'on appelle un *stream* (qui signifie « flux »). Celui-ci joue le rôle de médiateur entre la source des données et sa destination.
Toute opération sur les entrées/sorties doit suivre le schéma suivant :
ouverture, lecture, fermeture du flux.
Java décompose les objets traitant des flux en deux catégories :
les objets travaillant avec des flux d'entrée (in), pour la lecture de flux ;
les objets travaillant avec des flux de sortie (out), pour l'écriture de flux.

Utilisation de java.io

L'objet File

L'objet File permet de créer un objet File de n'importe quel type
-le fichier existe sur l'Ordi,
-sinon il faut le créer

```
public static void main(String[] args) {
File f = new File("test.txt");// Création de l'objet File en paramètre le chemin du ficher+ nom
f. createNewFile();// créer un fichier vide
System.out.println("Chemin du fichier : " +f.getPath());
System.out.println("Nom du fichier : " + f.getName());
System.out.println("Est-ce qu'il existe ? " + f.exists());
System.out.println("Est-ce un répertoire ? " + f.isDirectory());
System.out.println("Est-ce un fichier ? " + f.isFile());
}
```

```
import java.io.File;
```

Vous pouvez aussi effacer le fichier grâce la méthode delete(), créer des répertoires avec la méthode mkdir()

Les objets FileInputStream et FileOutputStream

C'est par le biais des objets FileInputStream et FileOutputStream que nous allons pouvoir :

- lire dans un fichier ;
- écrire dans un fichier.

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
```

Ces classes héritent des classes abstraites InputStream et OutputStream, présentes dans le package java.io. Les classes héritant d'InputStream sont destinées à la lecture et les classes héritant d'OutputStream se chargent de l'écriture !

La lecture :

On récupère le flut (flut d'octets) d'un fichier(fichier text) qui existe sur le bureau (**sérialisation**) sous la forme d'un objet de type FileInputStream

```
FileInputStream fis = new FileInputStream("C:\\Users\\zaki\\Desktop\\fichier.txt");
byte [] octet= new byte[1];// un tableau de 1 octet pour lire à chaque fois 1 octet du flut
while(fis.read(octet)>=0){//si read retourne -1 donc on a terminé le flut
        System.out.println(octet[0]);// on obtient comme resultat le code ascii
        String binaryString = Integer.toBinaryString(octet[0]);
        System.out.println(binaryString);
    }
```

l'écriture :

On écrit sur un fichier qui existe sur le bureau, sinon on crée un nouveau fichier(contenu):

```
String text = "zaki nygma"; // une chaine de caractère
byte[] byte_code = text.getBytes(); //pour récupérer les bytes (sérialisation)
//on crée un objet de type FileOutputStream qui permet d'écrire dans un fichier(paramètre)
FileOutputStream fos = new FileOutputStream("C:\\Users\\zaki\\Desktop\\text.txt");
// FileOutputStream fos = new FileOutputStream(file); file est de type File
fos.write(byte_code); //on converte les bytes vers le fichier original : désérialisation
fos.close();
```

Remarque:

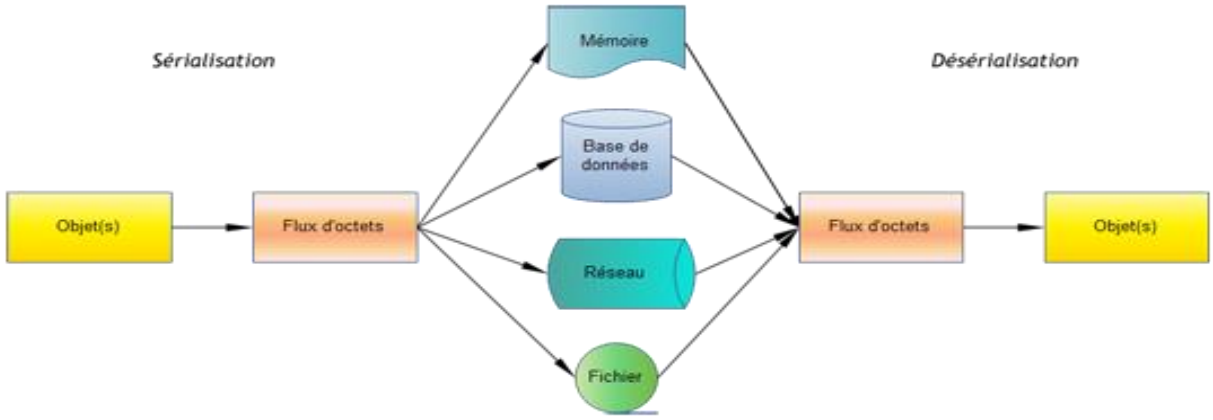
Pour un fichier de type File, pour récupérer son flut d'octets il faut passer par son chemin (path)

```
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
```

```
File file = new File("C:\\Users\\zaki\\Desktop\\file.txt");
System.out.println(file.getPath());
System.out.println(file.getAbsolutePath());
Path path = Paths.get(file.toString());
System.out.println(path);
byte[] data = Files.readAllBytes(path);// on récupère les bytes du fichier file.txt
FileOutputStream fileoutputsstream = new FileOutputStream("C:/Users/zaki/Desktop/file2.txt");
fileoutputsstream.write(data); //on copy le contenur du file.text dans un autre fichier file2.txt
```

La sérialisation

La sérialisation est un procédé qui permet de rendre un objet persistant pour stockage ou échange et vice versa. Cet objet est mis sous une forme sous laquelle il pourra être reconstitué à l'identique. Ainsi il pourra être stocké sur un disque dur ou transmis au travers d'un réseau.



Les objets ObjectInputStream et ObjectOutputStream

La classe **ObjectInputStream** permet de désérialiser une grappe d'objets à partir d'un flux binaire (un stream).
La classe **ObjectInputStream** peut être utilisée pour lire des objets précédemment écrits par **ObjectOutputStream**.

```
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
```

Exemple:

```
String text="hello world";
File file= new File("C:\\Users\\zaki\\Desktop\\test.txt");
file.createNewFile();
FileOutputStream fos= new FileOutputStream(file);
ObjectOutputStream oos= new ObjectOutputStream(fos);
oos.writeObject(text);
FileInputStream fis = new FileInputStream("C:\\Users\\zaki\\Desktop\\test.txt");
ObjectInputStream ois = new ObjectInputStream(fis);
String text_input=(String)ois.readObject();
System.out.println(text_input);
```

Cryptographie en JAVA

Deux bibliothèques fournies depuis Java 1.4 permettent la mise en œuvre de la cryptographie :

JCA (Java Cryptography Achitecture) : qui définit l'architecture générale du framework et les fonctionnalités cryptographiques de base (fonctions de hachage, signatures numériques, clés, certificats, ...) : **java.security**

JCE (Java Cryptography Extension) : qui fournit des fonctionnalités cryptographiques de haut niveau (chiffrement/déchiffrement avec algorithmes symétriques/asymétriques, authentification de messages (HMAC), ...) : **javax.crypto**

Les classes les plus utilisés sont les suivantes :

```
import java.security.Key; // pour créer une clé
import javax.crypto.Cipher;//pour l'instanciation d'un cryptosystème exemple : AES
import javax.crypto.KeyGenerator;//pour la génération des clés
```

1-la génération des clés :

```
KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
keyGenerator.init(128);//une clé de 128 bits
Key key = keyGenerator.generateKey();// la génération
```

2-Instantiation du cryptosystem

```
Cipher c = Cipher.getInstance("AES");// une instance du cryptosystème AES
```

3-Cryptage

```
c.init(Cipher.ENCRYPT_MODE, key);//initialisation pour le cryptage
String text="hello world";
byte [] text_bytes= text.getBytes();
byte [] cryptogramme_bytes;
cryptogramme_bytes = c.doFinal(text_bytes);
```

4-Decryptage

```
Cipher c = Cipher.getInstance("AES");
c.init(Cipher.DECRYPT_MODE, key);//initialisation pour le décryptage
text_bytes= c.doFinal(cryptogramme_bytes);
```

Enoncé :

Créer un programme qui permet de crypter et de décrypter un message en utilisant l'AES :

-Commencer par créer un message de type string à crypter

-Sauvegarder le message à transmettre dans un fichier (message.txt)

-Crypter le message en utilisant l'AES :

-La clé doit être sauvegardée dans un fichier (cle.key)

-Ecrire une fonction qui permet la génération et la sauvegarde de la clé

-Ecrire une fonction pour le cryptage et une autre pour le décryptage

-Sauvegarder le cryptogramme dans un autre fichier (cryptogramme.txt)

-Décrypter le cryptogramme pour avoir le message en clair en utilisant le fichier cryptogramme.txt

Remarque :

-Convertir un tableau de byte en un String

String s = new String(bytes, StandardCharsets.UTF_8);

```
import java.nio.charset.StandardCharsets;
```

-Utiliser le bloc try-catch pour capturer toute éventuelle erreur

```
try {
        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
        keyGenerator.init(128);
        Key key = keyGenerator.generateKey();
    } catch (Exception ex) {
        ex.printStackTrace();
    }
```