

Ecole polytechnique de Tunisie



Mini-Projet POO

« Initiation à la cryptographie en Java »

Effectué par

HOUES Naim

MECHERGUI Malek

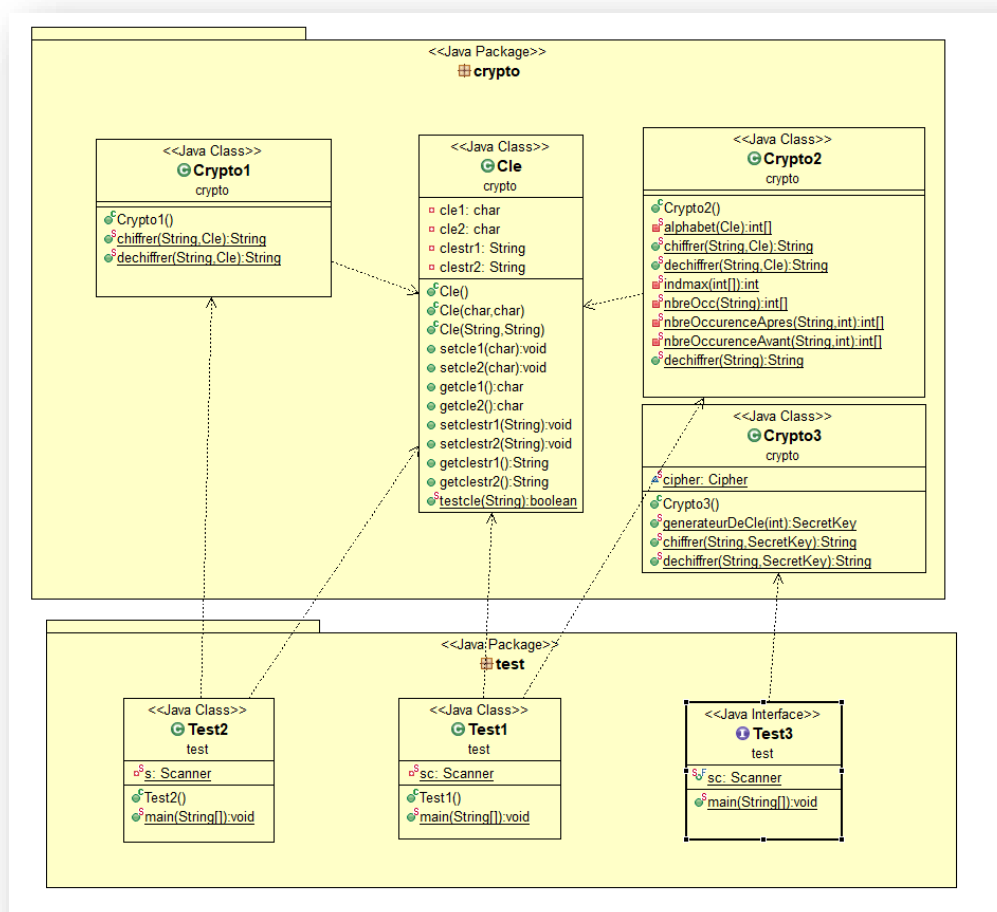
Elèves ingénieurs en 1^{ère} année

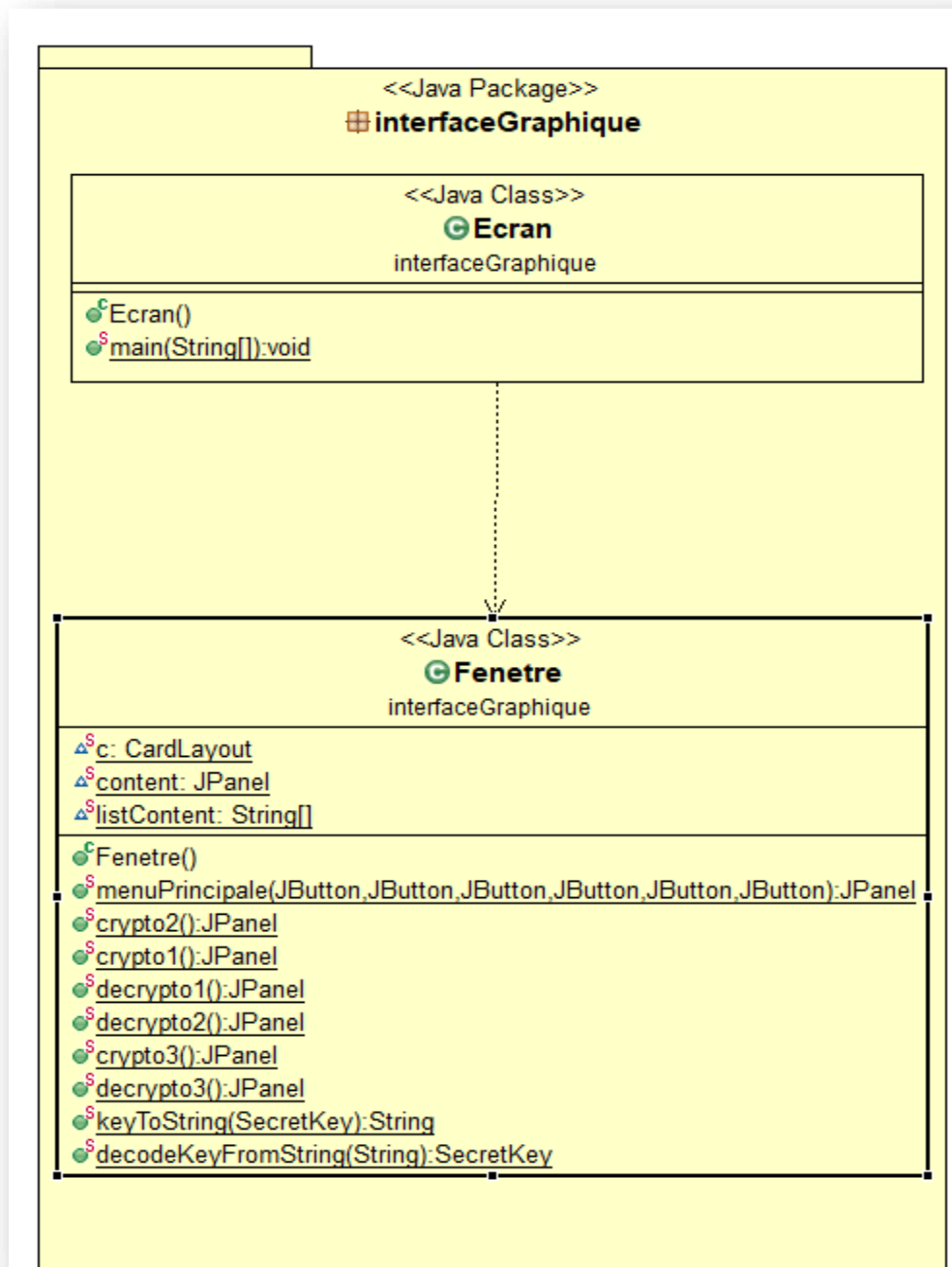
Détails du mini projet

Le projet comporte 3 parties (packages) distinctes :

- Package « **crypto** » qui comporte les classes : **Cle**, **Crypto1**, **Crypto2** et **Crypto3**.
- Package « **test** » qui comporte les classes de testes à effectuer (les main) : **Test1** (qui est le main de **Crypto2**), **Test2** (qui est le main de **Crypto1**) et **Test3** (qui est le main de **Crypto3**).
- Package « **interfacegraphique** » qui comporte les classes qui serviront par la suite à l'interface graphique et au développement de l'application exécutable : **Fenêtre** et **Ecran**.

Digramme UML





Package crypto

1- Class Cle :

Cette classe a comme attributs : Deux clés (cle1 et cle2) de type char ainsi que deux clés (clestr1 et clestr2) de types String pour nous servir par la suite dans les classes Crypto1 et Crypto2.

La classe comporte également 3 constructeurs :

- **Cle ()** : sans paramètres, elle initialise cle1 et cle2 à ' ?'.
- **Cle (char c1,char c2)** : prend en paramètre deux char, elle initialise cle1 à c1 et cle2 à c2.
- **Cle (String c1,String c2)** : prend en paramètre deux String, elle initialise clestr1 à c1 et clestr2 à c2.

La classe comporte 3 méthodes de classe qui sont publiques :

- **void setcle1 (char c1)** : affecte la valeur c1 à Cle1.
- **void setcle2 (char c2)** : affecte la valeur c2 à Cle2.
- **char getcle1 ()** : retourne la cle1.
- **char getcle2 ()** : retourne la cle2.
- **void setclestr1 (String c1)** : affecte la valeur c1 à Clestr1.
- **void setclestr2 (String c2)** : affecte la valeur c2 à Clestr2.
- **String getclestr1 ()** : retourne la clestr1.
- **String getclestr2 ()** : retourne la clestr2.
- **boolean testcle (String ch)** : en plus d'être publique, cette méthode est statique car on en aura besoin dans le test de Crypto2 (Test1). Donc cette méthode est applicable en appelant la classe elle-même. Au début elle génère un tableau contenant les caractères de la chaîne donnée en paramètre. Le tableau doit être constitué que de lettres alphabétiques (on a vérifié ceci à l'aide du code ASCII). A l'aide du tableau on a vérifié aussi que la chaîne ne contient pas de la même lettre deux fois. A l'issue de ces tests on valide la chaîne pour l'affecter à un objet clestr.

2- Class Crypto1 :

Cette classe ne possède pas d'attributs. Néanmoins, elle possède 2 méthodes publiques et statiques.

- **String chiffrer (String text, Cle cle)**: elle prend en paramètres un texte et un objet cle de type Cle avec lequel la fonction va chiffrer (ou crypter) le texte. Cette méthode utilise les clés de types char.

- **String déchiffrer (String text, Cle cle):** elle prend en paramètres un texte chiffré et un objet cle de type Cle avec lequel la fonction va déchiffrer (ou décrypter) le texte. Cette méthode utilise les clés de types char.

Pour ces 2 méthodes on remplace une lettre par la lettre qui lui est associée par la clé et ceci en manipulant des entiers (i.e. les codes ASCII des lettres en MAJUSCULE).

On a pris en considération les cas particuliers :

Exemple :

Si on demande de remplacer 'O' par la lettre 'Z' dans le mot « ZOOM », un traitement classique donnera le mot chiffré « ZZZM » et il y aura par la suite un problème de déchiffrement i.e. si on veut obtenir le mot initial on aura un « OOOM », pour cela notre programme remplacera également la lettre 'Z' par la lettre 'O' et il fera un traitement inverse en décryptage.

3- Class crypto2 :

Cette classe ne possède pas d'attributs. Mais, elle possède 8 méthodes statiques.

- **int[] alphabet (Cle cle) :** Il s'agit d'une méthode publique qui servira énormément dans le chiffrement et le déchiffrement.
Cette méthode prend comme paramètre un objet de type cle qui se compose de clestr1 et clestr2.
A l'aide de ces dernières on construira un tableau qui aura comme indices le code ASCII des lettres de l'alphabet en majuscule (et -65) et le contenu du tableau (i.e. tableau [indice]) contient la lettre qui la correspond et ceci selon clestr2. Si la lettre n'est pas changée avec la clé le contenu du tableau est son indice (i.e. tableau [indice]=indice) puisque le tableau sera initialisé ainsi.

Exemple :

Si ma clé est composé de clestr1= 'ABKZ' et de clestr2= 'LPOR'.

Le tableau avant changement :

0 (A)	1 (B)	2 (C)	...	10 (K)	...	25 (Z)
0 (A)	1 (B)	2 (C)	...	10 (K)	...	25 (Z)

Le tableau après changement:

0 (A)	1 (B)	2 (C)	...	10 (K)	...	25 (Z)
11 (L)	15 (P)	2 (C)	...	14 (O)	...	17 (R)

- **String chiffrer (String text, Cle cle).**
- **String déchiffrer (String text, Cle cle).**

Il s'agit de méthodes publiques et statiques.

Elles font 2 tâches inverses (crypter et décrypter).

Au début, ces 2 méthodes génèrent un tableau d'alphabet (à l'aide de la méthode alphabet ci-dessus) au moyen des clés données en paramètres.

Elles parcourent le texte crypté ou décrypté caractère par caractère tout en cherchant à chaque l'indice de la lettre qui lui correspond.

Une fois trouvé, l'indice est incrémenté de 65 et il est concaténé à une chaîne qui va être retourné par l'une des méthodes.

Au cas où le caractère n'est alphabétique il sera directement concaténer.

- **int indmax (int [] t) :** une méthode privée à la classe qui à partir d'un tableau d'entiers retournera l'indice du maximum.
- **int[] nbreOcc (String msg) :** une méthode privée qui à partir d'un texte retournera un tableau qui aura comme indice le code ASCII de l'alphabet en majuscule -65 (i.e. des indices de 0 à 25) et le contenu du tableau est le nombre d'occurrence de chaque lettre dans le msg donné en paramètre.

- **int[] nbreOccurenceApres (String msg,int ordreAlphabétique)** : une méthode privée qui à partir d'un ordre alphabétique donné en paramètre elle retourne un tableau contenant l'occurrence des lettres qui viennent après.
- **int[] nbreOccurenceAvant (String msg, int ordreAlphabétique)** : même esprit que la méthode précédente elle retourne un tableau qui contient l'occurrence des lettres qui viennent avant.
- **String dechiffrer (String text)** : cette méthode retourne à partir d'un texte crypté le texte initial le plus probable et ceci en se basant sur l'analyse fréquentielle des lettres de l'alphabet française (*** Cette partie sera expliquée en détails ultérieurement).

4- Class Crypto3 :

Cette classe comporte un attribut de classe qui est « cipher » de la classe java Cipher (javax.crypto.Cipher). Le terme Cipher est un terme standard pour les algorithmes de cryptographie.

Cette classe manipule des méthodes et classes fournis par java permettant de crypter et de décrypter les données. On a utilisé :

- La classe **KeyGenerator (javax.crypto.KeyGenerator)** sert à générer des clés aléatoires de chiffrement.
- La classe **SecretKey (javax.crypto.SecretKey)**.
- La classe **Java Cipher (javax.crypto.Cipher)** représente l'algorithme de chiffrement.

N.B : Pour chiffrer ou pour déchiffrer les données on a besoin d'une clé. Le cryptage de type « AES » utilise des clés symétriques c.-à-d. « AES » utilise la même clé pour le chiffrement et le déchiffrement.

La classe possède aussi 3 méthodes statiques et publiques.

- **SecretKey generateurDeCle (int niveauDeSecurite)** : cette méthode sert à générer une clé de type SecretKey (javax.crypto.SecretKey) à partir de la classe KeyGenerator.

D'abord, la méthode crée un objet keyGenerator de type KeyGenerator et appelle la méthode getInstance() en lui passant en paramètres 'AES' et

ceci parce qu'on veut utiliser un algorithme de chiffrement symétrique (Advanced Encryption Standard - AES). Puis, elle initialise keyGenerator en appelant la méthode `init()`. La méthode `init()` prend en paramètres la taille en bits des clés à générer[128,192,256] qui n'est autre que le niveau de sécurité. Après avoir initialisé le keyGenerator, notre méthode l'utilise pour générer des clés en appelant la méthode `generateKey()`.

- **String chiffrer (String TextaCrypte, SecretKeycle)** : cette méthode est publique statique, elle sert à chiffrer un texte à partir d'une clé donnée en paramètre de type `SecretKey`.

La méthode initialise l'attribut de classe « cipher » en mode de chiffrement (`ENCRYPT_MODE`) avec la méthode `init()`. Puis pour chiffrer le texte, elle utilise la méthode `doFinal()` .

- **String déchiffrer (String textcrypte ,SecretKey cle)** : cette méthode est publique statique, elle sert à déchiffrer un texte à partir d'une clé donnée en paramètre de type `SecretKey`. La méthode initialise l'attribut de classe « cipher » en mode de déchiffrement (`DECRYPT_MODE`) avec la méthode `init()` . Puis pour déchiffrer le texte, elle utilise la méthode `doFinal()` .

N.B :

La méthode `init()` prend deux paramètres:

- Le mode de fonctionnement chiffrement / déchiffrement.
- La clé de chiffrement / déchiffrement.

Les méthodes de la classe Cipher de chiffrement et de déchiffrement n'utilisent que des données stockées dans un tableau de byte, pour cela il faut convertir le texte de type String à un tableau de byte (avec la méthode `getBytes()`).

Le résultat de la méthode `doFinal()` est un tableau de byte (le texte chiffré ou déchiffré) donc pour convertir le texte en String on utilise la méthode `encodeToString()` de la classe `Base64.Encoder`.

Package test

1- Class Test1 :

Cette classe est le « **main** » de Crypto2, elle sert à saisir les 2 clés chaines en faisant des testes de saisie avec « do ... while » puisque ces 2 clés doivent êtres composés de lettres alphabétiques, doivent être d'une longueur minimale de 2 lettres et doivent être de même taille. Elle contrôle la saisie du texte à crypter ou à décrypter qui ne doit pas être vide.

2- Class Test2 :

Cette classe est le « **main** » de Crypto1, elle sert à saisir les 2 clés caractères en faisant des testes de saisie avec « do ... while » puisque ces 2 clés doivent êtres des lettres alphabétiques et ne doivent pas dépasser une lettre (la longueur=1).

3- Class Test3 :

Cette classe est le « **main** » de Crypto3, elle sert à saisir le texte à chiffrer ou à déchiffrer ainsi que le niveau de sécurité et ceci avec contrôle de saisie sur les 2. En fait, le texte ne doit pas être vide et les niveaux de sécurité ne sont que 3 (0,1 ou 2). Cette classe sert également à calculer la durée de cryptage ou de décryptage en nanosecondes moyennant la méthode **nanoTime** qui marque le début et la fin de cryptage ou de décryptage.

Package interfacegraphique

Pour concrétiser notre travail, une interface graphique qui va embellir et mettre en valeur le projet s'impose.

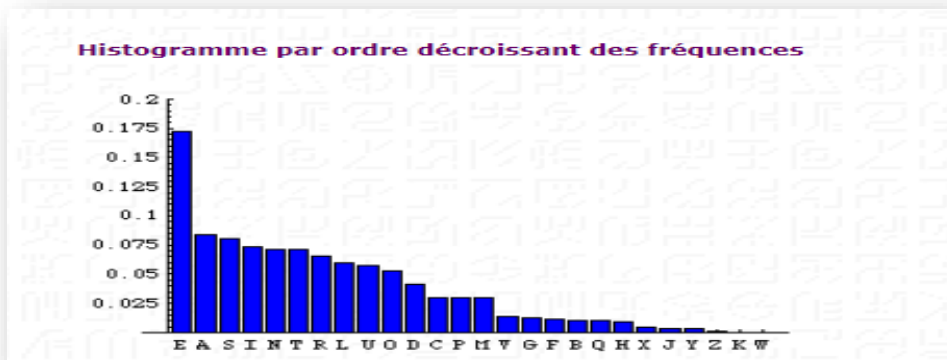
1- Class Fenetre :

Cette classe implémente l'interface graphique de l'application java.

2- Class Ecran :

Cette classe est le « **main** » de classe fenêtre.

Méthode de déchiffrement sans clé (Crypto2) ***



A partir de l'analyse fréquentielle des lettres dans un texte de français nous avons tiré que la lettre alphabétique la plus fréquente dans n'importe quel texte en langue française est la lettre 'E' avec un énorme écart par rapport aux autres lettres alphabétiques. Nous avons trouvé aussi que 'A' est la 2^{ème} la plus fréquente lettre alphabétique française avec des proportions très proches de 'S' ce qui génère des erreurs au niveau de décodage du texte.

Pour cette raison et pour éviter au maximum les fautes de décodage, nous avons travaillé sur les « **Bigrammes** ».

Fréquences des bigrammes

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	31	242	392	208	48	135	232	37	1255	32	7	663	350	1378	17	412	44	905	409	613	599	301	2	6	59	12
B	158	2	1	2	130	1	2	0	132	4	10	181	1	1	146	1	3	187	29	16	44	3	0	0	4	0
C	312	0	73	19	765	2	2	411	209	3	5	124	5	1	677	11	7	100	14	142	132	2	0	0	11	0
D	427	1	8	24	2408	2	5	25	378	3	0	14	21	5	231	4	6	134	64	3	406	4	1	0	5	0
E	616	176	917	998	782	258	209	67	179	96	8	1382	1056	2121	136	699	190	1514	3318	1307	761	258	11	125	15	60
F	181	1	1	8	180	118	1	1	190	0	0	43	1	1	213	1	2	106	12	1	61	0	0	0	1	0
G	135	1	10	9	408	4	63	3	69	6	4	74	10	103	47	5	1	197	12	23	81	1	0	0	2	0
H	267	5	4	1	285	0	0	0	149	3	0	3	4	17	107	0	3	18	5	0	42	0	1	0	7	0
I	176	85	203	172	1030	114	115	6	49	14	0	798	181	797	524	75	215	400	897	1243	11	190	1	40	0	4
J	76	0	0	0	100	0	0	0	2	0	0	0	0	0	91	0	0	0	0	0	42	0	0	0	2	0
K	8	0	0	0	6	0	3	0	6	0	0	0	10	3	9	0	0	5	1	0	0	0	0	0	3	0
L	1270	14	22	58	2368	25	14	39	512	4	1	647	18	41	281	69	47	16	126	42	369	14	0	0	15	1
M	510	152	11	11	1099	0	1	1	302	0	0	7	243	4	334	201	2	10	10	8	52	1	0	0	3	0
N	405	30	438	785	985	124	222	24	316	17	7	89	68	249	303	130	82	55	846	1694	114	109	0	1	19	20
O	6	83	88	101	46	32	115	7	452	14	3	184	391	1646	8	175	19	491	126	109	1086	28	9	4	62	4
P	671	1	3	21	441	5	1	136	119	0	0	377	2	4	505	125	1	363	31	65	140	1	0	0	1	0
Q	2	0	3	0	1	0	0	1	0	0	0	1	3	0	0	1	0	1	0	0	975	0	0	0	0	0
R	896	53	168	302	1885	46	96	5	583	11	3	292	181	88	520	82	51	176	386	445	183	77	1	1	21	5
S	809	85	306	735	1377	151	73	83	565	36	0	453	192	107	521	496	191	137	702	578	343	92	1	6	30	10
T	881	25	166	515	1484	52	19	64	984	28	3	331	70	40	363	268	96	668	404	269	270	41	4	6	18	3
U	168	87	165	162	781	40	83	4	534	41	3	302	128	516	19	184	15	980	591	469	14	177	1	264	8	4
V	277	0	1	0	502	0	0	0	288	0	0	1	0	0	167	0	0	81	0	0	11	0	0	0	0	0
W	11	1	1	0	3	0	0	2	8	0	0	0	0	0	3	0	1	0	4	0	0	0	0	0	2	0
X	35	14	37	36	68	8	7	5	57	0	0	21	15	3	7	56	11	3	15	35	2	18	0	4	0	0
Y	63	0	7	7	59	3	4	0	0	0	0	13	8	5	15	14	0	10	75	9	2	4	0	0	0	0
Z	8	0	2	6	49	3	1	0	1	1	0	11	4	2	15	4	1	0	3	1	0	7	4	0	0	2

Explication

N.B : Il faut prendre en considération qu'on manipule des tableaux avec des indices qui sont les codes ASCII des lettres de l'alphabet en majuscule et -65 et le contenu de ces tableaux ce sont les nombres d'occurrence.

Tout d'abord, le programme cherche la lettre la plus fréquente dans le texte à déchiffrer et il remplace la lettre par 'E'.

On cherche par la suite la 2^{ème} lettre la plus fréquente dans le texte qui peut être remplacé soit par 'A' soit par 'S', à priori on dit que c'est 'A'.

On cherche la lettre la plus fréquemment lié à 'E' qui est sera remplacé soit par 'S' soit par 'N' (moins que 'S'), à priori on dit que c'est 'S' (i.e. on a 'ES' plus que 'EN').

Ici on aura 2 cas :

- S'il s'agit de la même lettre qu'on veut remplacer (i.e. même indice) par 'S' et par 'A' à la fois c.-à-d. on a trouvé que la 2^{ème} lettre la plus fréquente dans le texte est elle-même la lettre qui est le plus lié à 'E'. Ici on remplacera la 2^{ème} lettre la plus occurrente dans le texte par 'S' et la 3^{ème} lettre la plus occurrente dans le texte par 'A'. On n'oubliera pas qu'on a cherché la lettre qui est le plus lié à 'E' et on la remplace par 'N'.
- S'il ne s'agit pas de la même lettre qu'on veut remplacer (i.e. pas le même indice). Ici on remplacera la 2^{ème} lettre la plus occurrente dans le texte par 'A' et la 3^{ème} lettre la plus occurrente dans le texte par 'S' et connaissant la lettre qui est le plus lié à 'E' on la remplace par 'N'.

En se basant sur le fait que la lettre qui se trouve le plus lié avant 'A' est 'L' (i.e. 'LA'), on peut déterminer la lettre (i.e. indice) qui va être remplacé par 'L'.

En se basant sur le fait que les lettres 'E', 'O' et 'A' se trouvent le plus lié avant 'N' et puisque les lettres 'E' et 'A' sont déjà bien placées on peut trouver facilement la lettre qui sera remplacer par le 'O'.

En se basant sur le fait que les lettres 'T', 'E' et 'S' se trouvent le plus lié après 'N' et puisque les lettres 'E' et 'S' sont déjà bien placées on peut trouver facilement la lettre qui sera remplacer par le 'T'.

Il faut mentionner qu'on a créé des clés au fur et à mesure du décryptage afin d'utiliser la méthode déchiffrer déjà existante avec clé chaîne (dans la même classe Crypto2).

En conclusion, cette démarche nous a permis d'obtenir 7 lettres bien placées.

Application exécutable

Grâce à l'interface graphique, nous avons extrait un fichier .jar et nous l'avons transformé en une application exécutable .exe.



1- Le menu principal :

Le menu principal permet à l'utilisateur le choix de cryptage ou de décryptage avec un niveau.

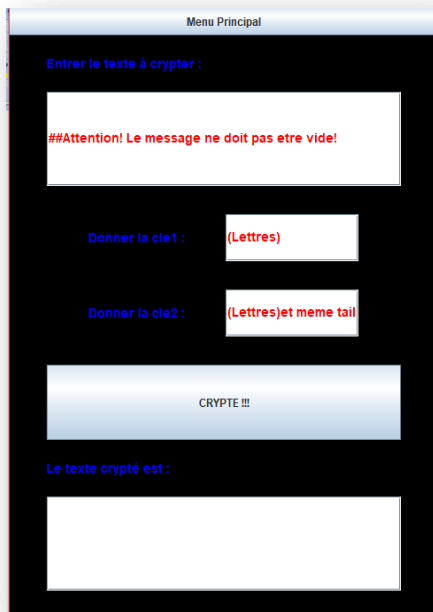
Il y a 3 boutons pour les 3 niveaux de cryptage.

Il y a 3 boutons pour les 3 niveaux de décryptage.



2- Cryptage :

Les 2 premiers niveaux de cryptage crypte des textes avec des clés. Nous avons écrit dans les boxes des messages d'alertes volontairement en couleur rouge pour attirer l'attention de celui qui va saisir. Nous avons aussi ajouté un bouton pour crypter et un bouton pour revenir au menu principal ainsi que des boxes pour afficher les réponses ou les erreurs.



Menu Principal

Entrer le texte à crypter :

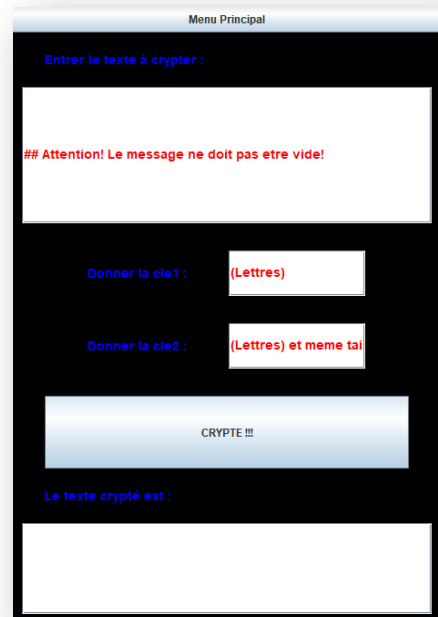
##Attention! Le message ne doit pas etre vide!

Donner la cle1 : (Lettres)

Donner la cle2 : (Lettres)et meme tail

CRYPTE !!!

Le texte crypté est :



Menu Principal

Entrer le texte à crypter :

Attention! Le message ne doit pas etre vide!

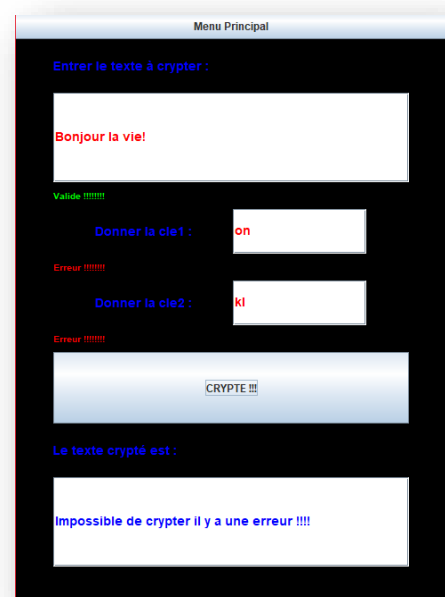
Donner la cle1 : (Lettres)

Donner la cle2 : (Lettres) et meme tai

CRYPTE !!!

Le texte crypté est :

En cas d'un problème de saisie des données (clé ou texte) il y a des messages de validation (en verre) ou d'erreur (en rouge) sous les boxes. Il y a aussi un message pour indiquer que le texte donné ne peut pas être décrypté et ceci dans le boxe des réponses.



Menu Principal

Entrer le texte à crypter :

Bonjour la vie!

Valide !!!!!

Donner la cle1 : on

Erreur !!!!!

Donner la cle2 : kl


Erreur !!!!!

CRYPTE !!!

Le texte crypté est :

Impossible de crypter il y a une erreur !!!!

Le niveau 3 est un cryptage « AES » donc il comporte 3 niveaux de sécurité qu'on peut sélectionner.



The screenshot shows a web application window titled "Menu Principale". It features a dark background with white and blue text. At the top, it says "Entrer le texte à crypter :". Below this is a white input field containing the red text "##Attention! Le message ne doit pas etre vide!". Underneath the input field is a label "Choisir le niveau de securite de la cle :" followed by a dropdown menu currently set to "Niveau 1". A large blue button with the text "CRYPTER !!!" is positioned below the dropdown. At the bottom, there is a label "Le texte crypté est :" followed by a white output field.