

Programmation orienté objet

Dr. Ines Ben Tekaya

06 Novembre 2018

Chapitre 3 : Les chaînes de caractères et les tableaux

les méthodes à 7 étapes du chap 1

- Combinez des chaînes en utilisant la concaténation.
- Construire des chaînes dans un programme Java à l'aide de `StringBuilder`.
- Utilisez des tableaux pour stocker et manipuler des collections de données.
- Pratiquez la conception d'algorithme efficace.

- Nous nous servons d'une étude de cas issue de la cryptographie.
 - Introduire quelques notions liées à la cryptographie.
 - Revoir quelques opérations sur les chaînes.
 - Différencier entre les String et les StringBuilder.
 - Utilisez des tableaux.

Etude de cas : la cryptographie

- Partie 1 : Mise en oeuvre du chiffrement César **Cryptage**
- Partie 2 : Briser le chiffre de César **Déchiffrement = décryptage**

Partie 1 : Mise en oeuvre du chiffrement César

chiffrement de César est un algorithme

Une brève histoire de la cryptographie

Achats en ligne : sécurité



- Un peu de sécurité informatique
- Vous voulez acheter quelque chose en ligne

Achats en ligne : sécurité



- Un peu de sécurité informatique
- Vous voulez acheter quelque chose en ligne

Achats en ligne : sécurité



- Un peu de sécurité informatique
- Vous voulez acheter quelque chose en ligne
 - Envoyer les informations de carte de crédit à la boutique en ligne

Achats en ligne : sécurité



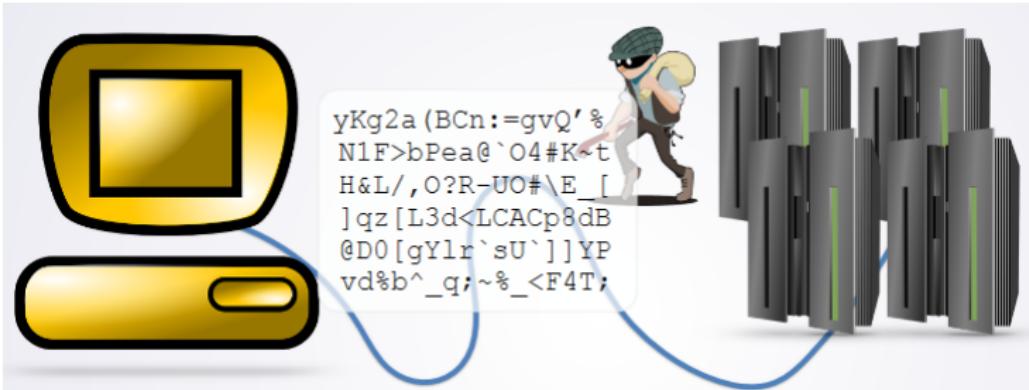
- L'ordinateur chiffre les données avant de les envoyer.
afin de les protéger on chiffrer avant d'envoyer

Achats en ligne : sécurité



- L'ordinateur chiffre les données avant de les envoyer.

Achats en ligne : sécurité



- L'ordinateur chiffre les données avant de les envoyer.
- Le voleur ne peut voir que les données cryptées.
 - Très difficile à déchiffrer

Achats en ligne : sécurité



- L'ordinateur chiffre les données avant de les envoyer.
- Le voleur ne peut voir que les données cryptées.
 - Très difficile à déchiffrer
- Le serveur de réception déchiffre les données.

Cryptographie moderne : https



https://

$$\cancel{n = pq}$$

$$\cancel{gcd(e, (p-1)(q-1)) = 1}$$

$$\cancel{c = m^e \pmod n}$$

$$\cancel{GF(2^8)}$$

La connexion est cryptée et authentifiée avec AES_128_GCM et utilise ECDHE_RSA comme mécanisme d'échange de clé.

- https = sécurisé
- Utilise la cryptographie moderne : RSA + AES
- Math pour ceux-ci : un peu plus que ce que nous aimerais

De l'histoire ancienne à l'époque moderne



- Cryptographie moderne : sécurisée ; mathématiques avancées.
- Cryptographie classique : non sécurisé ; mathématiques simples.

De l'histoire ancienne à l'époque moderne



- Cryptographie moderne : sécurisée ; mathématiques avancées.
- Cryptographie classique : non sécurisé ; mathématiques simples.

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

CFOPQ IBDFLK XQQXZH BXPQ CIXKH

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).

clé du chiffrement = N

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : $N = 3$ (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

F
ABCDEFGHIJKLMNOPQRSTUVWXYZ

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : $N = 23$ (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ

C

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : $N = 23$ (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ

C

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : $N = 3$ (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

I
ABCDEFGHIJKLMNOPQRSTUVWXYZ
C

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : $N = 23$ (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ
CF

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : $N = 23$ (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ

CF

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : $N = 3$ (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ
CF

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : N = 23 (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNPQRSTUVWXYZ
CFO

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : $N = 23$ (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ
CFOPQ IBDFLK

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : $N = 3$ (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

ABCDEFIGHJKLMNOPQRSTUVWXYZ

CFOPQ IBDFLK

A blue arrow points from the word "ATTACK" down to the letter "A" in the Caesar cipher table below.

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : N = 3 (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

A
ABCDEFGHIJKLMNPQRSTUVWXYZ
CFOPQ IBDFLK

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : $N = 23$ (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

A

ABCDEFGHIJKLMNPQRSTUVWXYZ

CFOPQ IBDFLK

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : N = 23 (par exemple : 3 lettres avant)

Aperçu du chiffrement César



FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ
CFOPQ IBDFLK X

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : N = 3 (par exemple : 3 lettres avant)

Aperçu du chiffrement César

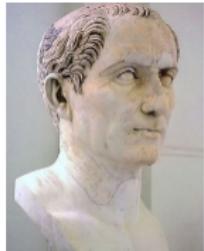


FIRST LEGION ATTACK EAST FLANK

ABCDEFGHIJKLMNOPQRSTUVWXYZ
CFOPQ IBDFLK XQQXZH BXPQ CIXKH

- Nommé pour Jules César.
- Cryptage : substituer la lettre courante avec (lettre + N).
 - César : N = 3 (par exemple : 3 lettres avant)
- Décryptage : chiffrer avec 26 - N

Deux façons de penser



FIRST LEGION ATTACK EAST FLANK
CFOPQ IBDFLK XQQXZH BXPQ CIXKH

- 1^{ère} façon : math sur les lettres.

- Tout est un nombre.
- 'F' - 3 = 'C'
- 'A' - 3 = ?
- Besoin de boucler ...

Deux façons de penser



FIRST LEGION ATTACK EAST FLANK

CFOPQ IBDFLK XQQXZH BXPQ CIXKH

ABCDEFGHIJKLM NOPQRSTUVWXYZ

XYZABCDEFGHIJKLM NOPQRSTUVWXYZ

- 2^{ème} façon : décalage de l'alphabet.
 - Calculer les décalages de chaque lettre au début.
 - Chercher chaque lettre.

Deux façons de penser



FIRST LEGION ATTACK EAST FLANK

CFOPQ IBDFLK XQQXZH BXPQ CIXKH

ABCDEF GHijklmnopqrstuvwxyz

XYZABCDEF GHijklmnopqrstuvwxyz

F

CFOPQ

ABCDEF

XYZABC



- 2^{ème} façon : décalage de l'alphabet.

- Calculer les décalages de chaque lettre au début.
- Chercher chaque lettre.

Deux façons de penser



FIRST LEGION ATTACK EAST FLANK

CFOPQ IBDFLK XQQXZH BXPQ CIXKH

ABCDEFGHIJKLMNOPQRSTUVWXYZ

XYZABCDEFGHIJKLMNOPQRSTUVWXYZSTUVW

- 2^{ème} façon : décalage de l'alphabet.

- Calculer les décalages de chaque lettre au début.
- Chercher chaque lettre.

Deux façons de penser



FIRST LEGION ATTACK EAST FLANK

CFOPQ IBDFLK XQQXZH BXPQ CIXKH

ABCDEFGHIJKLMNOPQRSTUVWXYZ
XYZABCDEFGHIJKLMNOPQRSTUVWXYZ



- 2^{ème} façon : décalage de l'alphabet.
 - Calculer les décalages de chaque lettre au début.
 - Chercher chaque lettre.

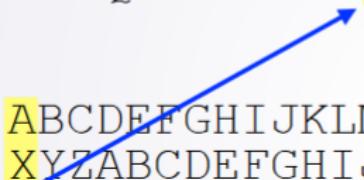
Deux façons de penser



FIRST LEGION ATTACK EAST FLANK

CFOPQ IBDFLK XQQXZH BXPQ CIXKH

ABCDEFGHIJKLMNOPQRSTUVWXYZ
XYZABCDEFGHIJKLMNOPQRSTUVWXYZ



- 2^{ème} façon : décalage de l'alphabet.
 - Calculer les décalages de chaque lettre au début.
 - Chercher chaque lettre.

Concepts à revoir

- Quelques concepts à revoir avant d'implémenter.
 - Nouvelles manipulations des chaînes de caractères.
 - les boucles qui comptent sur une plage.
 - Utiliser les numéros pour indexer les données.

Les chaînes de caractères : Un sujet familier

- Différent d'avant : construire des chaînes avec `StringBuilder`.
- Auparavant, des morceaux des chaînes existants.

Concaténation des chaînes de caractères

- Concaténation.

- Mot de fantaisie pour "coller ensemble".
- Utilisez l'opérateur + avec un opérande String.

"XYZ" + "ABCDEFGHIJKLMNPQRSTUVWXYZ"

Concaténation des chaînes de caractères

- Concaténation.

- Mot de fantaisie pour "coller ensemble".
- Utilisez l'opérateur + avec un opérande String.

```
"XYZ" + "ABCDEFGHIJKLMNPQRSTUVWXYZ"  
"XYZABCDEFGHIJKLMNPQRSTUVWXYZ"
```

Utile pour le chiffrement César

- Faire un alphabet "réarrangé".
 - Prendre deux parties (substring).
 - Les concaténer ensemble.

Alphabet : ABCDEFGHIJKLMNOPQRSTUVWXYZ

encr : XYZ

String encr = alphabet.substring (23) ;

Utile pour le chiffrement César

- Faire un alphabet "réarrangé".
 - Prendre deux parties (substring).
 - Les concaténer ensemble.

Alphabet : ABCDEFGHIJKLMNOPQRSTUVWXYZ

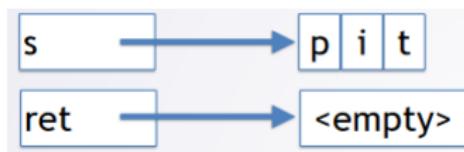
encr : XYZABCDEFGHIJKLMNOPQRSTUVWXYZ

`String encr = alphabet.substring (23);`

`encr = encr + alphabet.substring (0,23);`

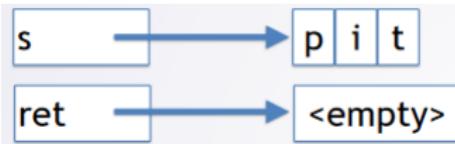
Accéder aux caractères d'une chaîne

- Exemple : Inverser une chaîne de caractères.
- Considérez l'appel inversé du mot ("pit").



Accéder aux caractères d'une chaîne

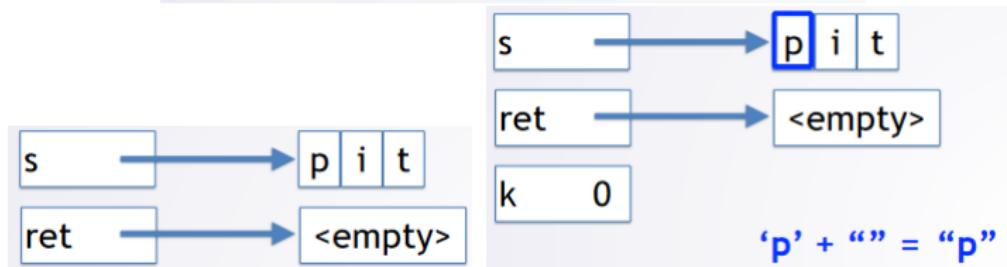
- Exemple : Inverser une chaîne de caractères.
- Considérez l'appel inversé du mot ("pit").



```
public String reverse(String s){  
    String ret = "";  
    for(int k=0; k < s.length(); k += 1){  
        ret = s.charAt(k) + ret;  
    }  
    return ret;  
}
```

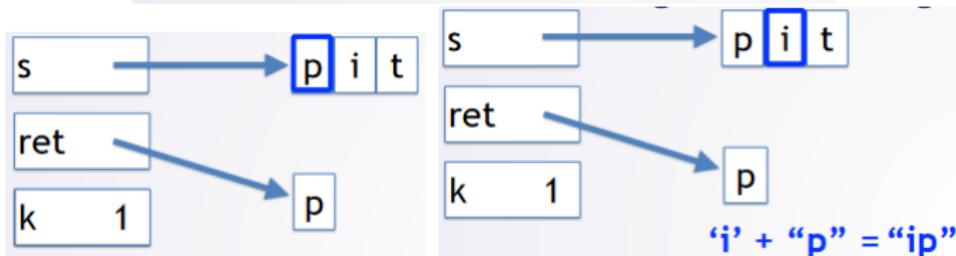
Accéder aux caractères d'une chaîne

```
public String reverse(String s){  
    String ret = "";  
    for(int k=0; k < s.length(); k += 1){  
        ret = s.charAt(k) + ret;  
    }  
    return ret;  
}
```



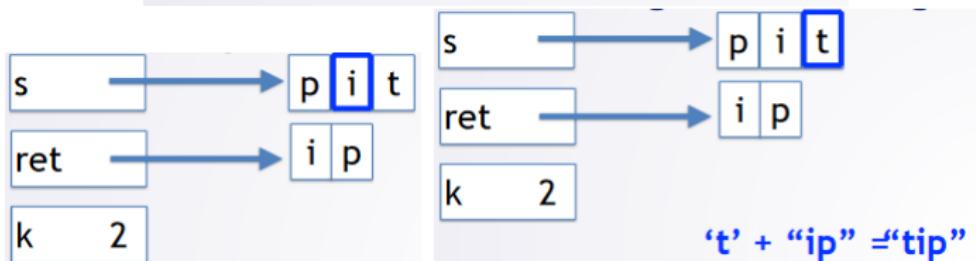
Accéder aux caractères d'une chaîne

```
public String reverse(String s){  
    String ret = "";  
    for(int k=0; k < s.length(); k += 1){  
        ret = s.charAt(k) + ret;  
    }  
    return ret;  
}
```



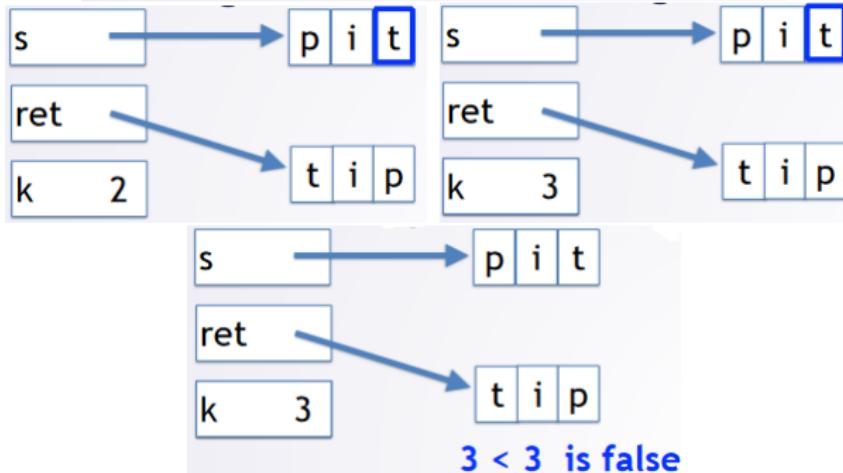
Accéder aux caractères d'une chaîne

```
public String reverse(String s){  
    String ret = "";  
    for(int k=0; k < s.length(); k += 1){  
        ret = s.charAt(k) + ret;  
    }  
    charAt : donne le caractère à l'emplacement préciser  
    return ret;  
}
```



Accéder aux caractères d'une chaîne

```
public String reverse(String s){  
    String ret = "";  
    for(int k=0; k < s.length(); k += 1){  
        ret = s.charAt(k) + ret;  
    }  
    return ret;  
}
```



Chaînes de caractères sont immuables

- Chaînes de caractères sont immuables.
 - On ne peut pas les changer.
 - On ne peut en fabriquer que de nouveaux.

```
String s = "Hello";
```



Chaînes de caractères sont immuables

- Chaînes de caractères sont immuables.
 - On ne peut pas les changer.
 - On ne peut en fabriquer que de nouveaux.

```
String s = "Hello";
```

```
String x = s;
```



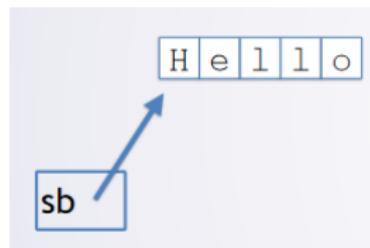
StringBuilder

- Java a StringBuilder dans ce but.
 - Séquence mutable de caractères.
 - `StringBuilder sb = new StringBuilder("Hello");`

Nom de méthode	Fonctionnalité
append	mettre String, int, char à la fin
insert	insérer String, int, char au milieu
charAt	obtient le caractère à l'index spécifié
setCharAt	change le caractère à l'index spécifié
toString	Récupère la chaîne

Mécanique de StringBuilder

```
StringBuilder sb = new StringBuilder("Hello");
```



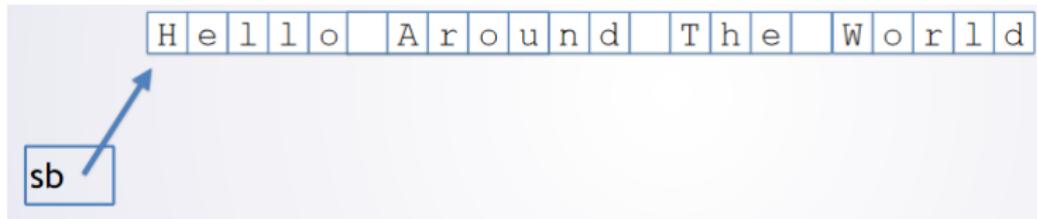
Mécanique de StringBuilder

```
StringBuilder sb = new StringBuilder("Hello") ;  
sb.append("World") ;
```



Mécanique de StringBuilder

```
StringBuilder sb = new StringBuilder("Hello");
    sb.append("World");
    sb.insert(5, " Around The ");
```



Classe Character

- Le type caractère (Character) est primitif.
- Le type caractère utilise des quotes simples.
 - 'a', '1', ' ', ..., mais "a" est String !
'a'=> caractere ≠ « a » => string
- La classe Character a plusieurs méthodes.
 - Character.toLowerCase('G') par exemple

Nom de méthode	Fonctionnalité
isLowerCase(ch)	retourne boolean si ch is 'a', 'b', ...
isDigit(ch)	retourne boolean if ch is '0','1',...,'9'
toLowerCase(ch)	retourne la version minuscule de ch
toUpperCase(ch)	retourne la version majuscule de ch

Développer un algorithme

- Étape 1 : Dérouler un exemple
- Étape 2 : Ecrire ce que vous avez fait
- Étape 3 : Généraliser
- Étape 4 : Tester
- Étape 5 : Transformer en code

Étape 1 : Dérouler un exemple

- Étape 1 : Dérouler un petit exemple

Message	I AM
Key	17
Alphabet	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Shifted Alphabet	RSTUVWXYZABCDEFGHIJKLMNOPQ

Étape 1 : Dérouler un exemple

- Étape 1 : Dérouler un petit exemple

Message	Z	RD
Key	17	
Alphabet		ABCDEFGHIJKLM NOPQRSTUVWXYZ
Shifted Alphabet		RSTUVWXYZ ABCDEFGHIJKLMNOPQ

Étape 2 : Ecrire ce que vous avez fait

Message I AM
Key 17

Alphabet ABCDEFGHIJKLMNOPQRSTUVWXYZ

Shifted Alphabet RSTUVWXYZABCDEFGHIJKLMNOPQ

- 1 Écrire l'alphabet.
- 2 Calculer l'alphabet décalé.

Étape 2 : Ecrire ce que vous avez fait

Message	Z	AM
Key	17	
Alphabet	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	
Shifted Alphabet	R S T U V W X Y Z A B C D E F G H I J K L M N O P Q	

- ③ Chercher à l'index 0 la lettre du message (lettre 0) («l»).
- ④ Chercher 'l' dans l'alphabet.
- ⑤ Trouver la lettre à la même position dans l'alphabet décalé («Z»)
- ⑥ Remplacer le caractère à l'index 0 du message, avec 'Z'.

Étape 2 : Ecrire ce que vous avez fait

Message	Z	AM
Key	17	
Alphabet	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	
Shifted Alphabet	R S T U V W X Y Z A B C D E F G H I J K L M N O P Q	

- ③ Chercher à l'index 0 la lettre du message («l»).
- ④ Chercher 'l' dans l'alphabet.
- ⑤ Trouver la lettre à la même position dans l'alphabet décalé («Z»)
- ⑥ Remplacer le caractère à l'index 0 du message, avec 'Z'.

Étape 2 : Ecrire ce que vous avez fait

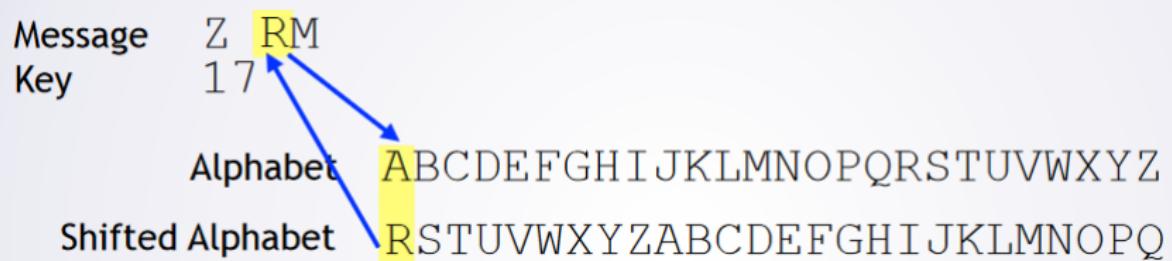
Message Z AM
Key 17

Alphabet ABCDEFGHIJKLMNOPQRSTUVWXYZ

Shifted Alphabet RSTUVWXYZABCDEFGHIJKLMNOPQ

- 7 Regarder la première lettre (à l'index 1) du message (' ').
- 8 Chercher '' dans l'alphabet.
- 9 Non trouvé (on ne change pas le caractère).

Étape 2 : Ecrire ce que vous avez fait



- ⑩ Regarder la lettre à l'index 2 du message ('A').
- ⑪ Chercher 'A' dans l'alphabet.
- ⑫ Trouver la lettre à la même position dans l'alphabet décalé ('R').
- ⑬ Remplacer le 2^{ème} caractère par 'R'.

Étape 2 : Ecrire ce que vous avez fait

Message	Z RD
Key	17
Alphabet	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Shifted Alphabet	RSTUVWXYZABCDEFIGHIJKLMNOPQ

- ⑩ Regarder la lettre à l'index 3 de message ('M').
- ⑪ Chercher 'M' dans l'alphabet.
- ⑫ Trouver la lettre à la même position dans l'alphabet décalé ('D').
- ⑬ Remplacer le 3^{ème} caractère par 'D'.

Étape 2 : Ecrire ce que vous avez fait

- ➊ Faire un `StringBuilder` avec un message (`encrypted`).
- ➋ Écrire l'alphabet.
- ➌ Calculer l'alphabet décalé.
- ➍ Regarder la lettre d'index 0 du message ('I').
- ➎ Chercher 'I' dans l'alphabet.
- ➏ Chercher la lettre correspondante dans la même position dans l'alphabet décalé ('Z').
- ➐ Remplacer le caractère à l'index 0 avec 'Z'.
- ➑ Regarder la 1^{ère} lettre du message (' ').
- ➒ Chercher ' ' dans l'alphabet.
- ➓ Non trouvé (on ne change pas le caractère).
- ➔ Regarder la lettre à l'index 2 du message ('A').
- ➕ Chercher 'A' dans l'alphabet.
- ➖ Trouver la lettre à la même position dans l'alphabet décalé ('R').
- ➗ Remplacer le 2^{ème} caractère par 'R'.
- ➘ Regarder la lettre à l'index 3 du message ('M').
- ➙ Chercher 'M' dans l'alphabet.
- ➚ Trouver la lettre à la même position dans l'alphabet décalé ('D').
- ➛ Remplacer le 3^{ème} caractère par 'D'.

Étape 3 : Trouvez les modèles et généraliser

- ➊ Faire un `StringBuilder` avec un message (`encrypted`).
- ➋ Écrire l'alphabet.
- ➌ Calculer l'alphabet décalé.
- ➍ Regarder la lettre d'index 0 de `encrypted` ('I').
- ➎ Chercher 'I' dans l'alphabet.
- ➏ Chercher la lettre correspondante dans la même position dans l'alphabet décalé ('Z').
- ➐ Remplacer le caractère à l'index 0 de `encrypted` avec 'Z'.
- ➑ Regarder la 1^{ère} lettre du message (' ').
- ➒ Chercher ' ' dans l'alphabet.
- ➓ Non trouvé (on ne change pas le caractère).
- ➊ Regarder la lettre à l'index 2 de `encrypted` ('A').
- ➋ Chercher 'A' dans l'alphabet.
- ➌ Trouver la lettre à la même position dans l'alphabet décalé ('R').
- ➍ Remplacer le 2^{ème} caractère de `encrypted` par 'R'.
- ➎ Regarder la lettre à l'index 3 de `encrypted` ('M').
- ➏ Chercher 'M' dans l'alphabet.
- ➐ Trouver la lettre à la même position dans l'alphabet décalé ('D').
- ➑ Remplacer le 3^{ème} caractère de `encrypted` par 'D'.

Étape 3 : Trouvez les modèles et généraliser

- ① Faire un `StringBuilder` avec un message (`encrypted`).
- ② Écrire l'alphabet.
- ③ Calculer l'alphabet décalé.

Initialisation

Étape 3 : Trouvez les modèles et généraliser

- 3 Regarder la lettre d'index 0 de encrypted ('I').
 - 4 Chercher 'I' dans l'alphabet.
 - 5 Chercher la lettre correspondante dans la même position dans l'alphabet décalé ('Z').
 - 6 Remplacer le caractère à l'index 0 de encrypted avec 'Z'.
-
- 7 Regarder la 1^{ère} lettre du message (' ').
 - 8 Chercher ' ' dans l'alphabet.
 - 9 Non trouvé (on ne change pas le caractère).

Étape 3 : Trouvez les modèles et généraliser

- 10 Regarder la lettre à l'index 2 de encrypted ('A').
 - 11 Chercher 'A' dans l'alphabet.
 - 12 Trouver la lettre à la même position dans l'alphabet décalé ('R').
 - 13 Remplacer le 2^{ème} caractère de encrypted par 'R'.
-
- 14 Regarder la lettre à l'index 3 de encrypted ('M').
 - 15 Chercher 'M' dans l'alphabet.
 - 16 Trouver la lettre à la même position dans l'alphabet décalé ('D').
 - 17 Remplacer le 3^{ème} caractère de encrypted par 'D'.

Étape 3 : Trouvez les modèles et généraliser

- 3 Regarder la lettre d'index 0 du message ('I').
- 4 Chercher 'I' dans l'alphabet.
- 5 Chercher la lettre correspondante dans la même position dans l'alphabet décalé ('Z').
- 6 Remplacer le caractère à l'index 0 de encrypted avec 'Z'.
- 7 Regarder la 1^{ère} lettre du message (' ').
- 8 Chercher ' ' dans l'alphabet.
- 9 Non trouvé (on ne change pas le caractère).
- 10 Regarder la lettre à l'index 2 du message ('A').
- 11 Chercher 'A' dans l'alphabet.
- 12 Trouver la lettre à la même position dans l'alphabet décalé ('R').
- 13 Remplacer le 2^{ème} caractère de encrypted par 'R'.
- 14 Regarder la lettre à l'index 3 du message ('M').
- 15 Chercher 'M' dans l'alphabet.
- 16 Trouver la lettre à la même position dans l'alphabet décalé ('D').
- 17 Remplacer le 3^{ème} caractère de encrypted par 'D'.

Étape 3 : Trouvez les modèles et généraliser

- ③ Regarder la lettre d'index 0 de encrypted ('I') .
- ④ Chercher ' I ' dans l'alphabet.
- ⑤ Chercher la lettre correspondante dans la même position dans l'alphabet décalé ('Z') .
- ⑥ Remplacer le caractère à l'index 0 de encrypted avec 'Z' .

Étape 3 : Trouvez les modèles et généraliser

- ➊ Faire un StringBuilder avec un message (encrypted).
- ➋ Écrire l'alphabet.
- ➌ Calculer l'alphabet décalé.
- ➍ Comptez de 0 à ≤ 3 , (appelez ça i).
 - ➎ Regardez le $i^{\text{ème}}$ caractère de encrypted (appelez-le currChar).
 - ➏ Trouvez l'index de currChar dans l'alphabet (appelez-le idx).
 - ➐ Si currChar est dans l'alphabet
 - ➑ Récupère le caractère de l'index idx de l'alphabet décalé (newChar).
 - ➒ Remplace le $i^{\text{ème}}$ caractère de encrypted par newChar.
 - ➓ Sinon : ne rien faire.

Étape 3 : Trouvez les modèles et généraliser

- ➊ Faire un StringBuilder avec un message (encrypted).
- ➋ Écrire l'alphabet.
- ➌ Calculer l'alphabet décalé. ← **Nécessite une réflexion, mais déjà vu comment.**
- ➍ Comptez de 0 à ≤ 3 , (appelez ça i).
 - ➎ Regardez le $i^{\text{ème}}$ caractère de encrypted (appelez-le currChar).
 - ➏ Trouvez l'index de currChar dans l'alphabet (appelez-le idx).
 - ➐ Si currChar est dans l'alphabet
 - ➑ Récupère le caractère de l'index idx de l'alphabet décalé (newChar).
 - ➒ Remplace le $i^{\text{ème}}$ caractère de encrypted par newChar.
 - ➑ Sinon : ne rien faire.

Étape 3 : Trouvez les modèles et généraliser

- ➊ Faire un StringBuilder avec un message (encrypted).
- ➋ Écrire l'alphabet.
- ➌ Calculer l'alphabet décalé.
- ➍ Comptez de **0** à $<= 3$, (appelez ça i). **Toujours commencer par 0 ?**
 - ➎ Regardez le $i^{\text{ème}}$ caractère de encrypted (appelez-le currChar).
 - ➏ Trouvez l'index de currChar dans l'alphabet (appelez-le idx).
 - ➐ Si currChar est dans l'alphabet
 - ➑ Récupère le caractère de l'index idx de l'alphabet décalé (newChar).
 - ➒ Remplace le $i^{\text{ème}}$ caractère de encrypted par newChar.
 - ➑ Sinon : ne rien faire.

Étape 3 : Trouvez les modèles et généraliser

- ➊ Faire un StringBuilder avec un message (encrypted).
- ➋ Écrire l'alphabet.
- ➌ Calculer l'alphabet décalé.
- ➍ Comptez de 0 à $<= 3$, (appelez ça i). **Toujours se terminer par 3 ?**
 - ➎ Regardez le $i^{\text{ème}}$ caractère de encrypted (appelez-le currChar).
 - ➏ Trouvez l'index de currChar dans l'alphabet (appelez-le idx).
 - ➐ Si currChar est dans l'alphabet
 - ➑ Récupère le caractère de l'index idx de l'alphabet décalé (newChar).
 - ➒ Remplace le $i^{\text{ème}}$ caractère de encrypted par newChar.
 - ➑ Sinon : ne rien faire.

Étape 3 : Trouvez les modèles et généraliser

- ➊ Faire un StringBuilder avec un message (`encrypted`).
- ➋ Écrire l'alphabet.
- ➌ Calculer l'alphabet décalé.
Non : c'est la longueur du message `encrypted`
- ➍ Comptez de 0 à `<longeur de encrypted`, (appelez ça `i`).
 - ➎ Regardez le $i^{\text{ème}}$ caractère de `encrypted` (appelez-le `currChar`).
 - ➏ Trouvez l'index de `currChar` dans l'alphabet (appelez-le `idx`).
 - ➐ Si `currChar` est dans l'alphabet
 - ➑ Récupère le caractère de l'index `idx` de l'alphabet décalé (`newChar`).
 - ➒ Remplace le $i^{\text{ème}}$ caractère de `encrypted` par `newChar`.
 - ➑ Sinon : ne rien faire.

Étape 4 : Tester

- ① Faire un StringBuilder avec un message (encrypted).
- ② Écrire l'alphabet.
- ③ Calculer l'alphabet décalé.

Message : A BAT

KEY 19 Nous n'avons pas précisé quoi donner comme réponse !

- ④ Comptez de 0 à <longeur de encrypted, (appelez ça i).
 - a Regardez le $i^{\text{ème}}$ caractère de encrypted (appelez-le currChar).
 - b Trouvez l'index de currChar dans l'alphabet (appelez-le idx).
 - c Si currChar est dans l'alphabet
 - i Récupère le caractère de l'index idx de l'alphabet décalé (newChar).
 - ii Remplace le $i^{\text{ème}}$ caractère de encrypted par newChar.
 - d Sinon : ne rien faire.

Étape 4 : Tester

- ➊ Faire un StringBuilder avec un message (encrypted).
- ➋ Écrire l'alphabet.
- ➌ Calculer l'alphabet décalé.
- ➍ Comptez de 0 à <longeur de encrypted, (appelez ça i).
 - ➎ Regardez le $i^{\text{ème}}$ caractère de crypté (appelez-le currChar).
 - ➏ Trouvez l'index de currChar dans l'alphabet (appelez-le idx).
 - ➐ Si currChar est dans l'alphabet
 - ➑ Récupère le caractère de l'index idx de l'alphabet décalé (newChar).
 - ➒ Remplace le $i^{\text{ème}}$ caractère de encrypted par newChar.
 - ➓ Sinon : ne rien faire.
- ➕ Votre réponse est la chaîne à l'intérieur de encrypted

Étape 5 : Transformer en code

Live coding !

Partie 2 : Briser le chiffre de César

Introduction

Partie 2 : Briser le chiffre de César

- Vous avez implémenté l'algorithme de chiffrement de César.
 - Forme de base du cryptage, concepts utiles.
 - Nous utilisons une clé pour chiffrer, comment déchiffrer ?
- Le destinataire prévu saura la clé.
 - Crypter avec 7, décrypter 19
- Qu'en est-il "cracking" ?
 - Un voleur ou un pirate "trouve" la clé.
 - Peut-on utiliser l'algorithme naïf (force brute) ?

Et si un humain aide ?

- Supposons que nous interceptons ce message.
Lujyfwapvu huk zljbypaf hyl mbukhtluahs whyaz vm avkhf'z Pualyula.
- Que dit ce message ?
 - Si nous savions la clé à chiffrer, nous pourrions déchiffrer.
 - Combien y a-t-il de clés possibles ? Essayez tous !
- Esquisse de cette approche
- Ayez le code à chiffrer, appelez-le avec chaque clé
- Brute Force : chiffrement rapide, espace clé faible

Décryptage humain ou oculaire

- Déverrouiller ou déchiffrer un message chiffré.
 - Quand on n'a pas la clé



Décryptage humain ou oculaire

- Déverrouiller ou déchiffrer un message chiffré.
 - Quand on n'a pas la clé
 - Mais nous avons le code pour le chiffrement de César !

```
public void eyeballDecrypt(String encrypted){  
    CaesarCipher cipher = new CaesarCipher();  
    for(int k=0; k < 26; k++){  
        String s = cipher.encrypt(encrypted,k);  
        System.out.println(k+"\t"+s);  
    }  
}
```

Décryptage humain ou oculaire

- Déverrouiller ou déchiffrer un message chiffré.
 - Quand on n'a pas la clé
 - Mais nous avons le code pour le chiffrement de César !

```
public void eyeballDecrypt(String encrypted){  
    CaesarCipher cipher = new CaesarCipher();  
    for(int k=0; k < 26; k++){  
        String s = cipher.encrypt(encrypted,k);  
        System.out.println(k+"\t"+s);  
    }  
}
```

Décryptage humain ou oculaire

- Déverrouiller ou déchiffrer un message chiffré.
 - Quand on n'a pas la clé
 - Mais nous avons le code pour le chiffrement de César !

```
public void eyeballDecrypt(String encrypted){  
    CaesarCipher cipher = new CaesarCipher();  
    for(int k=0; k < 26; k++){  
        String s = cipher.encrypt(encrypted,k);  
        System.out.println(k+"\t"+s);  
    }  
}
```

Qu'est-ce que le message crypté ?

Lujyfwapvu huk zljbypaf hyl mbukhtluahs whyaz vm avkhf'z Pualyula.

0 Lujyfwapvu huk zljbypaf hyl mbukhtluahs whyaz vm avkhf'z Pualyula.
1 Mvkzgxbqwv ivl amkczqbg izm ncvliumvbit xizba wn bwlig'a Qvbmzvmb.
2 Nwlahycrxw jwm bnldarch jan odwmnjvnwcju yjacob xo cxmjh'b Rwcnaawnc.
3 Oxmbizdsyx kxn comebsdi kbo pexnkwoxdkv zkbdc yp dynki'c Sxdobxod.
4 Pyncjaetzy lyo dpnfctej lcp qfylxpwyelw olced zq ezoij'd Tyeprcype.
5 Qzdkbfauz mzp eqogdufk mdq rgzpmqzfmx bmdfe ar faprnk'e Uzfqdzqf.
6 Rapelcgvba naq frphevgl ner shaqnzragny cnegf bs gbqln'g Vagrearg.
7 Sbqfmduhwcb obr ghsqifwhm ofs tibroasbhoz dofhg ct hcrom'g Wbhsfbsh.
8 Tcrgneixdc pcs htrjgxin pgt ujcspbtcipa eppgh idspn'h Xcitgcti.
9 Udshofjyed qdt iuskhijo qhu vktqscudjqb fhqji ev jetqo'i Ydjuhdju.
10 Vetipgkze reu jvtlizkp riv wleurdvekrc grikj fw kfurp'j Zekvievk.
11 Wfujqhlagf sfv kwunjalq sjw xmfvsewfslsds hsjlk gx lgvsq'k Aflwjfwl.
12 Xgvkrmbhg tgw lxvnkbmr tkx yngwtfxgntme itkmnl hy mnwtr'l Bgmxkgxm.
13 Yhwljsncih uhx mywolcn5 uly zohxugyhnu5 julnm iz nixus'm Chnylhyn.
14 Zixmtkodji viy nxzpmdot vnz apiyvhziowv kvmon ja ojyvt'n Diozmizo.
15 Ajynulpekj wjz oyqnepu wna bqqzwiajpwh lwmpo kb pkzwu'o Ejpanjap.
16 Bkzovmqflk xka pbzrofqv xob crkaxjbkqxi mnoqp lc qlaxv'p Fkqbokbq.
17 Clapwnrgml ylb qcaspgrw ypc dslbykclryj nyprq md rmbwy'q Glrplcr.
18 Dmbqxoshnm zmc rdbtqhsx zqd etmczldmszk ozqsr ne snczx'r Hmsdqmds.
19 Encryption and security are fundamental parts of today's Internet.
20 Fodszqujpo boe tfdvjsuz bsf gvoebnfoubm qbsut pg uepbz't Jouflsofu.
21 Gpetarvkap cpf ugewtkva ctg hwpcogpvcn rctvu qh vqfcu' Kpvgtpgv.
22 Hqfubswlrq dgg vhfxulwb duh ixqgdpqhwdo sduuu ri wrgdb'v Lqwhuqhw.
23 Irgvctxmsr erh wigyvmxc evi jyrheqirxep tevxw sj xshec'w Mrxivrix.
24 Jshwduynts fsi xjhzwnyd fwj kzsiffrjsyfq ufwyx tk ytifd'x Nsyjwsjy.
25 Ktixevzout gtj ykiaxoze gnxk latjgsktzgr vgxyz ul zuge'y Otzkxtkz.



Qu'est-ce que le message crypté ?

Lujyfwapvu huk zljbypaf hyl mbukhtluahs whyaz vm avkhf'z Pualyula.

- 0 Lujyfwapvu huk zljbypaf hyl mbukhtluahs whyaz vm avkhf'z Pualyula.
- 1 Mvkzgxbqwv ivl amkczqbg izm ncvlium'bit xizba wn bwlig'a Qvbmzvmb.
- 2 Nwlahycrxw jwm bnldarch jan odwnjvnwcju yjacob xo cxmjh'b Rwcnaawnc.
- 3 Oxmbizdsyx kxnm comebsdi kbo pexnkwoxdkv zkbdc yp dynki'c Sxdobxod.
- 4 Pyncjaetzy lyo dpnfctej lcp qfylxpyelw olced zq ezoij'd Tyepcype.
- 5 Qzdkbfauz mzp eqogdufk mdq rgzpmqzfmx bmdfe ar fapmk'e Uzfqdzqf.
- 6 Rapelcgvba naq frphevg'l ner shaqnzragny cnefp bs gbqln'p Vagrearg.
- 7 Sbqfmdhweb obr gqsiwhm ofs tibroasbhoz dofha ct hcrom'g Wbhsfbsh.
- 8 Tcrgneixdc pcs htrjgxin pgt ujcspbtcipa eppih idspn'h Xcitgcti.
- 9 Udshofjyed qdt iuskhijo ghu vktqscudjqb fhqji ev jetqo'i Ydjuhdju.
- 10 Vetiqgkze reu tvtizkpb riv wleurdvekrc grikj fw kfurp'j Zekvievk.
- 11 Wfujqhlafg sfv kwumjalq sjw xmfvsewflsd hsjlk gx lgvsq'k Aflwjfwl.
- 12 Xgvkrinbhg tgw lxvnkbmr tkx yngwtfxgme itkmly hy mhwt'r'l Bgmxkgxm.
- 13 Yhwljsncih uhx mywolcnx uly zohxugyhnuf julnm iz nixus'm Chnylhyn.
- 14 Zixmtkodji viy nzxpmot vnz apiyvhziouv kvmon ja ojyvt'n Diozmizo.
- 15 Ajyjnulpekj wjz oayqnepu wna baojzwiajpwh lwmpo kb pkzwu'o Ejpanjap.
- 16 Bkzovmoflk xka pbzrofqv xob crkaxjbkxai mxaop lc qlaxv'p Fkabokbq.
- 17 Clapwnrgml ylb qcaspgrw_ycp dslbykclryj nyprq md rmbyw'q Glrcplcr.
- 18 Dmbqxoshhnm zmc rdbtqhsx zqd etmczldmszk ozqr ne snczx'r Hmsdqmds.
- 19 Encryption and security are fundamental parts of today's Internet.
- 20 Fodsqujpo boe tfdvsjuz bsf gvoebnoubm qbsut pg uebz't Jouflsofu.
- 21 Gpetarvkap cpf ugewtkva ctg hwptfcogpvcn rctvu qh vqfcu'k Kpvgtpgv.
- 22 Hqfubswlrq dgg vhfxulwb duh ixqgdpqwdo suuw ri wrgdb'v Lqwhuqhw.
- 23 Irgvctxmsr erh wigyvmxc evi jyrheqirxep tevxw sj xshec'w Mrxivrix.
- 24 Jshwduynts fsi xjhzwmyd fwj kzsiffrjsyfq ufwyx tk ytifd'x Nsyjwsjj.
- 25 Ktixevzout gtj ykiazoze gnxk latjgsktzgr vxxy ul zuje'y Otzkxtkz.



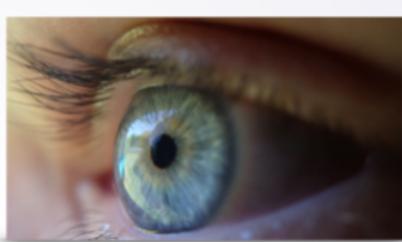
Qu'est-ce que le message crypté ?

Lujyfwapvu huk zljbypaf hyl mbukhtluahs whyaz vm avkhf'z Pualyula.

0 Lujyfwapvu huk zljbypaf hyl mbukhtluahs whyaz vm avkhf'z Pualyula.
1 Mvkzgxqbw ivl amkczqbq izm ncvliumbit xizba wn bwlig'a Qvbmrwmb.
2 Nwlahyrcrxw jwm bnldarch jan odwnjnvncju yjacb xo cxmjhb' Rwcnaawc.
3 Oxmbizdsyx kxn comebsdi kbo pexnkwoxdv zkbdc yp dynki'c Sxdoxbod.
4 Pyncjaetzy lyo dpnfciej lcp qfyolxpyew alced zq ezolj'd Tycpcye.
5 Qzodkbfuaz mzp egodufk mdq rgzpmqazfmx bmdfe ar fapmk'e Uzfqdzqf.
6 Rapelcvba naq frphevg'l ner shaqnzragny cnefg bs gbqnl'f Vagrearg.
7 Sqbfmdhwcb obr gsqfwhm ofs tibroashbz dofhg ct hcrom'g Whhsfbsh.
8 Tcrgeinxdc pcs htrjgxjn pgt ujcspbtcepa epghl da idspn'h Xcitgcti.
9 Udshofjyed qdt iuskyho qhu vkdtpcudjgb fhqhi ev jetqo'i Ydjuhduj.
10 Vetipgkzfe reu jvtlizkp riv wleurdvekrc grikj fw kfurp'j Zekvievk.
11 Wfujqhlagf sfv kwunjalq sjw xmfvseflsd hsjlk gx lgvsq'k AfLwjfwl.
12 Xgvkrimbhg tgw lxvnkbmr tkx yngwtxgmcite itkmly mhwt'r Bgmxkgxm.
13 Yhwljsnjcih uxh mywolcns ulz zohxugyhnq julnm iz nixus'm Chnlyhny.
14 Zixmtkodji viy nxzpmot vnz apiyvhziogv kvmon ja ojyvt'n Diozmizo.
15 Ajynulpekj wjz oayqnepu wna bajzwiqjpwh lwmpo kb pkzwu'o Ejpanjap.
16 Bkzovmqlfk xka pbzrnofqv xob crkaxjbkxqi mwoap lc qlaxvp' Fkqbokbq.
17 Clapwrmqml ylb acasparw ypc dslbykclryi nypqa md rmbyw'q Gircplcr.

18 Dmboxoshnm zmc rdbtqhsx zqd etmczldmszk ozqsr ne snczx'r Hmsdqmds.

19 Encryption and security are fundamental parts of today's Internet.
20 Fodszaqjpo boe tfdvsjuz bsf gvoebnfoubm qbsut pg upebz't Jouflsofu.
21 Gpetarvkap cpf ugewartva ctg hwpcfcoppgvn rctvu qh vqfc'a'u Kpvgtgv.
22 Hqfbuswlrq dag vhfxulwb duh ixqddphqwd sduuw ri wrqdb'v Lqwhuqhw.
23 Irgvctxmsr erh wigyvmmc evi jyrheqirxep tevxw sj xshec'w Mrxivrix.
24 Jshwduynts fsi xjhzwmyd fwj kzsfirjsyfq ufwyx tk ytifd'x Nsyjwsjy.
25 Ktixevezout gtj ykiaxoze gnxk latjgsktzgr vgxzy ul zujge'y Otzkxtkz.



Qu'est-ce que le message crypté ?

Lujyfwapvu huk zljbypaf hyl mbukhtluahs whyaz vm avkhf'z Pualyula.

0 Lujyfwapvu huk zljbypaf hyl mbukhtluahs whyaz vm avkhf'z Pualyula.
1 Mvkzgxbaw ivl amkczqbg izm ncvliumbit xizba wn bwlig'a Qvbmzvmb.
2 Nwlahycrxw jwm bnldarch jan odwnjvnwcija yjacb xo cxmjh'b Rwcnavnc.
3 Oxmbizdsyx kxn comesbsdi kbo pexnkwoxdzk zkbdc yp dynki'c Sxdobxod.
4 Pyncjaetzy lyo dpnfctej lcp qfyolxpyelw alced zq ezolj'd Tycpcye.
5 Qzodkbfuaz mzp eqogdufk mdq rgzpmqzfmx bmdfe ar fapmk'e Uzfdzqf.
6 Rapelcgvba naq frphevgl ner shaqnzragny cnegf bs gbqnlf'f Vagrearg.
7 Sbqfmndhwcb obr gsqifwhm ofs tibroasbhz dofhg ct hrcrom'g Wbhsfbsh.
8 Tcrgeinxdc pcs htrjgxin pgt ujcspbtcipa epgih du idspn'g Xcitgcti.
9 Udshofqyed qdt iuskhkyo qhu vkdtdqcdubjh fahji qd jetqoi' Ydjuhduj.
10 Vetipgkzfe reu jvtlizkp riv wleurdvekrc grikj fw kfurp'j Zekvievk.
11 Wfujahlagf sfv kwunjalq sjw xmfvseflsd hsjlk gx lgvsq'k Aflwjfwl.
12 Xgvkimbhg tgw lxvnkbmr tkx yngwtxgmte itkmil hy mnwtr'l Bgmxkgxm.
13 Yhwlsjncih uxh mywolcns uly zhuxghyhnuf julnm iz nixus'm Chnylhyn.
14 Zixmtkodji viy nxzpmot vnz apiyvhziogv kvmon ja ojyvt'n Diozmizo.
15 Ajynulpekj wzj oayqnepu wna bqjzwiajpwh lwmpo kb pkzsu'o Ejpanjap.
16 Bkzovmqlk xka pbzrofqv xob crkaxjbkqxi mkoap lc qlaxv'p Fkqbkobq.
17 Clapwnrgml ylb qcaspgrw ypc dslbykclryj nyprq md rmbyw'q Glrcplcr.
18 Dmbqxoshnm zmc rdbtahsxz zad etmczldmszk ozqsr ne snczx'r Hmsdqmids.
19 Encryption and security are fundamental parts of today's Internet.
20 Fodszuqjpo boe tfdvjsuz bst gvoebnfoubm qbsut pg upebz't Joufsou.
21 Gpetarvkap cfp ugetkvka ctg hwpcfcgpcvn rctvu qh vafca'u Kpvgtpgv.
22 Hqfubswlrq dag vhfxulwb duh ixqgdphqwd sduvw ri wrgdb'v Lqwhughw.
23 Irgvctxmsr erh wigymvxc evi jyrheqirxep tevxw sj xshec'w Mrxivrix.
24 Jshwduynts fsi xjhzwnyd fwj kzsifrsyfq ulwyx tk ytifd'x Nsyjwsjy.
25 Ktixevezout gtj ykiazoze gxx latjgsktzr vxz y ul zujge'y Otzkxtkz.

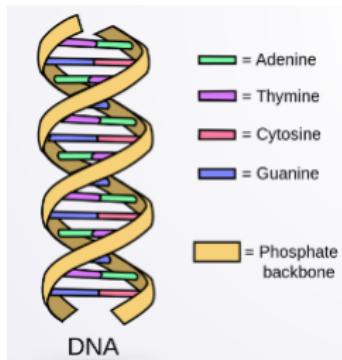


**solution 1: fair tester tt les fois(decriptage humain
solution 2 : baser sur l'occurense des lettre**

Partie 2 : Briser le chiffre de César Tableaux (Arrays)

Vue d'ensemble du problème

- En tant que scientifique en calcul du génome, vous :
 - Compter les occurrences de c, g, a, t dans l'ADN.
 - Partie de la recherche de régions codant des protéines.



Vue d'ensemble du problème

- En tant que scientifique en calcul du génome, vous :
 - Compter les occurrences de c, g, a, t dans l'ADN.
 - Partie de la recherche de régions codant des protéines.
- En tant que programmeur s'initiant au cryptage
 - Compter les occurrences de a, b, ..., x, y, z
 - Vous devez casser le chiffre de César



Vue d'ensemble du problème



- Apprendre un nouveau concept de programmation :
 - Tableaux : collection homogène.
 - Concept de programmation très important.

Compter le contenu en ADN

solution 2

- Quatre caractères différents : 'c', 'g', 't', 'a'
 - Compter les occurrences de chacun.

```
public void dnaFingerprint(String s){  
    int cc = 0, cg = 0, ca = 0, ct = 0;  
  
    for(int k=0; k < s.length(); k++){  
        char ch = s.charAt(k);  
        if (ch == 'c'){  
            cc +=1 ;  
        }  
        else if (ch == 'g'){  
            cg += 1;  
        }  
        else if (ch == 'a'){  
            ca += 1;  
        }  
        else if (ch == 't'){  
            ct += 1;  
        }  
    }  
}
```

Compter le contenu en ADN

- Quatre caractères différents : 'c', 'g', 't', 'a'
 - Compter les occurrences de chacun.
- Difficile à faire évoluer vers 'a', 'b', 'c', ..., 'y', 'z'
 - Pas conceptuellement difficile, mais fragile face au changement

```
for(int k=0; k < s.length(); k++){  
    char ch = s.charAt(k);  
    if (ch == 'c') {  
        cc +=1 ;  
    }  
    else if (ch == 'g') {  
        cg += 1;  
    }  
}
```

Compter le contenu en ADN

- Quatre caractères différents : 'c', 'g', 't', 'a'
 - Compter les occurrences de chacun.
- Difficile à faire évoluer vers 'a', 'b', 'c', ..., 'y', 'z'
 - Pas conceptuellement difficile, mais fragile face au changement
- Qu'en est-il des résultats d'impression ? Mise à l'échelle ?

```
System.out.println("number of c's = "+cc);
System.out.println("number of g's = "+cg);
System.out.println("number of a's = "+ca);
System.out.println("number of t's = "+ct);
```

Tableau en tant que collection indexée

- Pour rompre le chiffrement de César, il faut compter les lettres.
 - Le caractère le plus fréquent dans le texte anglais est : 'e'.
 - En général, compter et collecter des informations importantes.
- Nous avons StorageResource pour collecter les chaînes de caractères
 - Utile, mais limité, nous développerons l'idée plus tard.
- Nous avons besoin d'une collection indexée.
 - Comme des chaînes, mais stocke n'importe quel type, pas seulement des caractères



Concepts pour les tableaux

```
String s = ".....";  
for(int k=0; k < s.length(); k++) {  
    char ch = s.charAt(k);  
}  
  
int[] a = new int[256];  
for(int k=0; k < a.length; k++) {  
    int val = a[k];  
}
```

- Définir un tableau, similaire à String, utilisez [].

Concepts pour les tableaux

```
String s = ".....";  
for(int k=0; k < s.length(); k++) {  
    char ch = s.charAt(k);  traiter un tableau on met pas .charAt  
}
```

```
int[] a = new int[256];  
for(int k=0; k < a.length; k++) {  
    int val = a[k];  
}
```

- Définir un tableau, similaire à String, utilisez [].
 - À la place d'utiliser s.charAt(k) utiliser a[k]

Concepts pour les tableaux

```
String s = ".....";
for(int k=0; k < s.length(); k++) {
    char ch = s.charAt(k);
}
```

```
int[] a = new int[256];
for(int k=0; k < a.length; k++) {
    int val = a[k];
}
```

tableau on ne met pas de parenthèse

- Définir un tableau, similaire à String, utilisez [].
 - À la place d'utiliser s.charAt(k) utiliser a[k]
 - À la place d'utiliser s.length() utiliser a.length.

Tableaux en action : comptez 26 caractères

- counters [0] est le nombre d'occurrences 'a'.
 - counters[k] est le nombre d'occurrences de la kème lettre (Z = 25)

[] tableau

```
public void textFingerPrint(String s){  
    String alpha = "abcdefghijklmnopqrstuvwxyz";  
    int[] counters = new int[26];  
    for(int k=0; k < s.length(); k++){  
        char ch = s.charAt(k);  
        int index = alpha.indexOf(Character.toLowerCase(ch));  
        if (index != -1){  
            counters[index] += 1;  
        }  
    }  
    for(int k=0; k < counters.length; k++){  
        System.out.println(alpha.charAt(k)+"\t"+counters[k]);  
    }  
}
```

Tableaux en action : comptez 26 caractères

- counters [0] est le nombre d'occurrences 'a'.
 - counters[k] est le nombre d'occurrences de la kème lettre (Z = 25)

```
public void textFingerPrint(String s){  
    int[] counters = new int[26]  
  
    char ch = s.charAt(k);  
    int index = alpha.indexOf(Character.toLowerCase(ch));  
    if (index != -1){  
        counters[index] += 1;  
    }  
    for(int k=0; k < counters.length; k++){  
        System.out.println(alpha.charAt(k)+"\t"+counters[k]);  
    }  
}
```

Tableaux en action : comptez 26 caractères

- counters [0] est le nombre d'occurrences 'a'.
 - counters[k] est le nombre d'occurrences de la kème lettre (Z = 25)

```
public void textFingerPrint(String s){  
    String alpha = "abcdefghijklmnopqrstuvwxyz";  
    int[] counters = new int[26];  
    for(int k=0; k < s.length(); k++){  
        int index = alpha.indexOf(Character.toLowerCase(ch));  
        if (index != -1){  
            counters[index] += 1;  
        }  
        for(int k=0; k < counters.length; k++){  
            System.out.println(alpha.charAt(k)+"\t"+counters[k]);  
        }  
    }  
}
```

Tableaux en action : comptez 26 caractères

- counters [0] est le nombre d'occurrences 'a'.
 - counters[k] est le nombre d'occurrences de la kème lettre (Z = 25)

```
public void textFingerPrint(String s){  
    String alpha = "abcdefghijklmnopqrstuvwxyz";  
    int[] counters = new int[26];  
    for(int k=0; k < s.length(); k++){  
        char ch = s.charAt(k);  
        int index = alpha.indexOf(Character.toLowerCase(ch));  
        if (index != -1){  
            counters[index] += 1;  
        }  
    }  
    for(int k=0; k < counters.length; k++){  
        System.out.println(alpha.charAt(k)+"\t"+counters[k]);  
    }  
}
```

Tableau récapitulatif

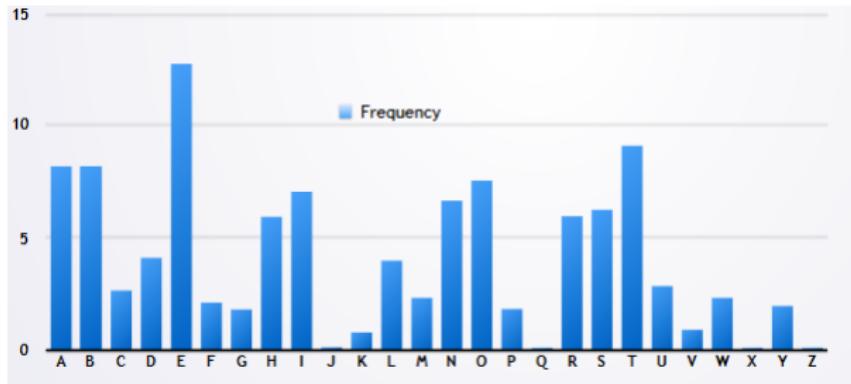
- Collection indexée d'éléments ou de valeurs.
 - `int [] x // pas de stockage, juste un type`
 - `int [] x = new int[12] // Zéro`
 - `String [] s = new String[12]; //null`

Tableau récapitulatif

- Collection indexée d'éléments ou de valeurs.
 - `int [] x` // pas de stockage, juste un type
 - `int [] x = new int[12]` // Zéro
 - `String [] s = new String[12];` //null
- Lire et écrire via des index :
 - `s[3] = "Hello"` ; `x[2] = x[3] + 4;`
- Stockage alloué, alors ne change pas
 - `.length` est une valeur, pas une méthode.
 - modifier des éléments se fait via des appels de méthodes.

Décrypter sans globe oculaire/humain

- S'appuyer sur les fréquences des lettres en anglais.
- Utilisez des fréquences dans d'autres langues au besoin.
- Nous trouverons le caractère occurrant le maximum de fois.
- Supposons qu'il s'agisse d'un 'e', trouvez le changement !



Compter les mots communs

- Données en ligne pour les mots les plus courants dans l'anglais.
- <https://github.com/first20hours/google-10000-english>

Compter les occurrences

Hi, do you want a lollipop today?
I own many good flavors,
but banana is outstanding.

- Code pour scanner le texte et incrémenter le compteur.

<code>{0,0}</code>	<code>0 1 2 3 4 5 6 7 8.....14</code>	<code>23 24 25</code>
<code>"A B C D E F G H I.....O</code>	<code>X Y Z"</code>	

Compter les occurrences

Hi, do you want a lollipop today?
I own many good flavors,
but banana is outstanding.

- Code pour scanner le texte et incrémenter le compteur.

```
{0,0,0,0,0,0,0,1,0,0,0,0,0,...,0,0,0}
```

```
0 1 2 3 4 5 6 7 8.....14 23 24 25
```

```
"A B C D E F G H I..... O X Y Z"
```

Compter les occurrences

Hi, do you want a lollipop today?
I own many good flavors,
but banana is outstanding.

- Code pour scanner le texte et incrémenter le compteur.

```
{0,0,0,0,0,0,0,1,1,0,0,0,0...,0,0,0}  
0 1 2 3 4 5 6 7 8.....14      23 24 25  
"A B C D E F G H I..... O      X Y Z"
```

Compter les occurrences

```
Hi, do you want a lollipop today?  
I own many good flavors,  
but banana is outstanding.
```

- Code pour scanner le texte et incrémenter le compteur.

```
{0,0,0,0,0,0,0,1,1,0,0,0,0...,0,0,0}  
0 1 2 3 4 5 6 7 8.....14 23 24 25  
"A B C D E F G H I..... O X Y Z"
```

Compter les occurrences

Hi, **do** you want a lollipop today?
I own many good flavors,
but banana is outstanding.

- Code pour scanner le texte et incrémenter le compteur.

```
{0,0,0,1,0,0,0,1,1,0,0,0,0,0...,0,0,0}  
0 1 2 3 4 5 6 7 8.....14 23 24 25  
"A B C D E F G H I..... O X Y Z"
```

Compter les occurrences

Hi, do you want a lollipop today?
I own many good flavors,
but banana is outstanding.

- Code pour scanner le texte et incrémenter le compteur.

```
{0,0,0,1,0,0,0,1,1,0,0,1,0,0...,0,0,0}  
0 1 2 3 4 5 6 7 8.....14 23 24 25  
"A B C D E F G H I..... O X Y Z "
```

Compter les occurrences

Hi, do you want a lollipop today?
I own many good flavors,
but banana is outstanding.

- Code pour scanner le texte et incrémenter le compteur.

```
{0,0,0,1,0,0,0,1,1,0,0,1,0,0...,0,0,0}  
0 1 2 3 4 5 6 7 8.....14      23 24 25  
"A B C D E F G H I..... O      X Y Z "
```

Compter les occurrences

Hi, do you want a lollipop today?
I own many good flavors,
but banana is outstanding.

- Code pour scanner le texte et incrémenter le compteur.

```
{0,0,0,1,0,0,0,1,1,0,0,1,0,0...,0,1,0}  
0 1 2 3 4 5 6 7 8.....14 23 24 25  
"A B C D E F G H I..... O X Y Z"
```

Compter les occurrences

Hi, do you want a lollipop today?
I own many good flavors,
but banana is outstanding.

- Code pour scanner le texte et incrémenter le compteur.

```
{0,0,0,1,0,0,0,1,1,0,0,2,0,0...,0,1,0}  
0 1 2 3 4 5 6 7 8.....14 23 24 25  
"A B C D E F G H I..... O X Y Z"
```

Compter les occurrences

Hi, do you want a lollipop today?
I own many good flavors,
but banana is outstanding.

- Code pour scanner le texte et incrémenter le compteur.

methode approximative

{9,2,0,4,0,1,2,1,5,0,0,4,1,7,10,2,0,1,3,5,3,1,2,0,3,0}

0 1 2 3 4 5 6 7 8.....14 23 24 25

"A B C D E F G H I..... O X Y Z "

Compter les occurrences avec le code

```
String alph = "abcdefghijklmnopqrstuvwxyz";
int[] counts = new int[26];
for(int k=0; k < message.length(); k++){
    char ch = Character.toLowerCase(message.charAt(k));
    int dex = alph.indexOf(ch);
    if (dex != -1){
        counts[dex] += 1;
    }
}
```

Compter les occurrences avec le code

```
String alph = "abcdefghijklmnopqrstuvwxyz";
int[] counts = new int[26];
for(int k=0; k < message.length(); k++){
    char ch = Character.toLowerCase(message.charAt(k));
    int dex = alph.indexOf(ch);
    if (dex != -1){
        counts[dex] += 1;
    }
}
```

- Accès au message, tester si le caractère est alphabétique.

Compter les occurrences avec le code

```
String alph = "abcdefghijklmnopqrstuvwxyz";
int[] counts = new int[26];
for(int k=0; k < message.length(); k++){
    char ch = Character.toLowerCase(message.charAt(k));
    int dex = alph.indexOf(ch);
    if (dex != -1){
        counts[dex] += 1;
    }
}
```

- Accès au message, tester si le caractère est alphabétique.
 - Trouver l'emplacement dans
"abcdefghijklmnopqrstuvwxyz"

Compter les occurrences avec le code

```
String alph = "abcdefghijklmnopqrstuvwxyz";
int[] counts = new int[26];
for(int k=0; k < message.length(); k++){
    char ch = Character.toLowerCase(message.charAt(k));
    int dex = alph.indexOf(ch);
    if (dex != -1){
        counts[dex] += 1;
    }
}
```

- Accès au message, tester si le caractère est alphabétique.
 - Trouver l'emplacement dans "abcdefghijklmnopqrstuvwxyz".
 - Utiliser index pour incrémenter l'un des 26 compteurs.

De l'algorithme au code

```
public String decrypt(String encrypted) {
    CaesarCipher cc = new CaesarCipher();
    int[] freqs = countLetters(encrypted);
    int maxDex = maxIndex(freqs);
    int dkey = maxDex - 4;
    if (maxDex < 4) {
        dkey = 26 - (4-maxDex);
    }
    return cc.encrypt(encrypted, 26-dkey);
}
```

- Comptez les 26 fréquences de 'a' - 'z', 'A' - 'Z'.
 - Trouver la plus grande valeur, supposons que c'est 'e'

De l'algorithme au code

```
public String decrypt(String encrypted) {
    CaesarCipher cc = new CaesarCipher();
    int[] freqs = countLetters(encrypted);
    int maxDex = maxIndex(freqs);
    int dkey = maxDex - 4;
    if (maxDex < 4) {
        dkey = 26 - (4-maxDex);
    }
    return cc.encrypt(encrypted, 26-dkey);
}
```

- Comptez les 26 fréquences de 'a' - 'z', 'A' - 'Z'.
 - Trouver la plus grande valeur, supposons que c'est 'e'
 - Trouver la distance de 'e' qui a l'indice 4

De l'algorithme au code

```
public String decrypt(String encrypted) {
    CaesarCipher cc = new CaesarCipher();
    int[] freqs = countLetters(encrypted);
    int maxDex = maxIndex(freqs);
    int dkey = maxDex - 4;
    if (maxDex < 4) {
        dkey = 26 - (4-maxDex);
    }
    return cc.encrypt(encrypted, 26-dkey);
}
```

- Comptez les 26 fréquences de 'a' - 'z', 'A' - 'Z'.
 - Trouver la plus grande valeur, supposons que c'est 'e'
 - Trouver la distance de 'e' qui a l'indice 4

De l'algorithme au code

```
public String decrypt(String encrypted) {  
    CaesarCipher cc = new CaesarCipher();  
    int[] freqs = countLetters(encrypted);  
    int maxDex = maxIndex(freqs);  
    int dkey = maxDex - 4;  
    if (maxDex < 4) {  
        dkey = 26 - (4-maxDex);  
    }  
    return cc.encrypt(encrypted, 26-dkey);  
}
```

- Comptez les 26 fréquences de 'a' - 'z', 'A' - 'Z'.
 - Trouver la plus grande valeur, supposons que c'est 'e'
 - Trouver la distance de 'e' qui a l'indice 4
 - Utilisez 26-distance pour décrypter en utilisant encrypt !

- Le tableau freqs a une relation entre index et valeur, freqs[8] est combien de fois 'i' se produit.
 - Lorsque vous recherchez une valeur maximale, renvoyez index,
 - index est utilisé pour trouver la distance entre 'e' ou 4 comme décalage.

```
public int maxIndex(int[] vals){  
    int maxDex = 0;  
    for(int k=0; k < vals.length; k++){  
        if (vals[k] > vals[maxDex]) {  
            maxDex = k;  
        }  
    }  
    return maxDex;  
}
```

Live coding !

Télécharger CaesarBreaker