SAE1.01 - Implémentation d'un besoin client

Le carré de Polybe

Cindy Cappelle
cindy.cappelle@univ-lille.fr

Automne 2022

1 Contexte

1.1 Introduction

Le carré de Polybe est une technique de chiffrement par substitution monoalphabétique ancienne, décrite pour la première fois vers 150 av. J.-C. par l'historien grec éponyme.

Figure 1: Polybe

Assez simple de prime abord, elle consiste, au départ, à représenter les lettres de l'alphabet dans un tableau dont chaque ligne est numérotée de bas en haut et chaque colonne est numérotée de gauche à droite.

Pour chiffrer un mot, il suffit de trouver la paire de numéros correspondant à chaque lettre. Il est à noter que l'alphabet comportant 26 lettres, il n'est pas rare que certaines lettres partagent la même case, on retrouve par exemple de nombreux carrés où le V et le W fusionnent, parfois ce sont les I et les J.

Cette méthode présente l'avantage d'être simple et accessible à la compréhension du plus grand nombre puisque peu importe le sens (coder/décoder), le processus est exactement le même. Dans sa forme la plus élémentaire, le carré de Polybe utilise toujours la même clef, pour ne pas dire aucune. En effet, les lettres sont placées dans leur ordre alphabétique et comme il s'agit d'un chiffrement par substitution mono-alphabétique, une simple analyse séquentielle suffit pour décoder un message.

Très utile dans la télégraphie, mais également dans les communications en mer, beaucoup de méthodes reposaient sur ce principe de façon à transmettre un maximum d'informations avec peu d'objets, torches, drapeaux, son. Le morse semble d'ailleurs très clairement s'apparenter à cette méthode.

1.2 Carré de Polybe sous sa forme simple

Le principe est assez simple, il consiste à ordonner les lettres de l'alphabet en ordre alphabétique dans un tableau carré de 5 cases de côté dont chaque ligne et chaque colonne sont numérotées, de haut en bas et de gauche à droite. Considérant que l'alphabet français comporte 26 lettres et que le carré possède seulement 25 cases, par convention les lettres V et W sont fusionnées (on considère que les W sont remplacés par des V) comme dans l'image ci-dessous.

	0	1	2	3	4
0	A	B	C	D	E
1	F	G	H	I	J
2	K	L	M	N	O
3	P	Q	R	S	T
4	U	**V**	X	Y	Z

Figure 2: Carré de Polybe sous la forme la plus simple

Ensuite, pour chiffrer un mot, il faut trouver la paire de chiffres correspondants à chaque lettre. Le premier chiffre est le numéro de la ligne et le second celui de la colonne. Par exemple, le mot " BONJOUR " est ainsi chiffré par le carré de Polybe :

01	24	23	14	24	40	32
B	O	N	J	O	U	R

Pour déchiffrer un mot, il suffit d'effectuer la méthode inverse.

1.3 Variante du carré de Polybe (forme améliorée avec une clé)

Considérant que le carré de Polybe possède plusieurs limitations et qu'il a été créé il y a longtemps, des variantes y ont été apportées dans le but de l'améliorer. En voici une avec une clé. Le moyen le plus simple pour renforcer la méthode de chiffrement est sans aucun doute d'ajouter une clé. Le principe est qu'une fois la clé choisie, il suffit d'ajouter chacune de ses lettres **sans répétition** au début du carré de Polybe. Ensuite, il faut ajouter les **autres** lettres de l'alphabet en ordre alphabétique. Par exemple, le carré de Polybe avec la clé BUTINFORMATIQUE :

	0	1	2	3	4
0	**B**	**U**	**T**	**I**	**N**
1	**F**	**O**	**R**	**M**	**A**
2	**Q**	**E**	C	D	G
3	H	J	K	L	P
4	S	**V**	X	Y	Z

Figure 3: Carré de Polybe avec la clé BUTINFORMATIQUE

Le principe de chiffrement/déchiffrement reste le même.

2 Le travail à réaliser

Après avoir expliqué et analysé le mode de fonctionnement de l'encodage/décodage du carré de Polybe, il s'agit maintenant d'en proposer une simulation en iJava. Le principe est d'entrer éventuellement la clé désirée et

- un message à chiffrer, le programme calcule alors le message chiffré
- ou un message chiffré, le programme calcule alors le message déchiffré.

On suppose que l'analyste de l'entreprise a déjà travaillé sur l'analyse et la décomposition du problème en fonctions. Votre rôle est alors :

- d'implémenter les fonctions demandées ;
- de vérifier que chacune des fonctions demandées produit le résultat attendu sur un certain nombre d'exemples ;
- dans le programme principal, il s'agira de faire un menu permettant de simuler le codage/décodage d'un message saisi par l'utilisateur selon une clé éventuelle choisie par l'utilisateur.

Les fonctions à développer sont listées ci-après ainsi que les étapes du programme principal (ce fichier à compléter est fourni sur Moodle).

```
class SAEpolybe_NOM1_NOM2_NOM3 extends Program {

    final int LARGEUR = 5;  // taille du carré (5x5 dans notre cas)

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

    // La fonction String initialiserCarre() retourne une chaine de caractères contenant le carré de
Polybe (version de base, sans clé, c'est-à-dire la chaine "ABCDEFGHIJKLMNOPQRSTUVXYZ", le V et le W sont
"fusionnés" en V)
    String initialiserCarreSimple(){
        return "";
    }

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

    // La fonction void afficherCarre(String carre) affiche le carré de Polybe carrz passé en paramètre
comme illustré dans l'exemple ci-après.
    // Par exemple :  si le carré passé en paramètre est :  "ABCDEFGHIJKLMNOPQRSTUVXYZ", la fonction
devra afficher :
    // |0 1 2 3 4
    // ------------
    // 0|A B C D E
    // 1|F G H I J
    // 2|K L M N O
    // 3|P Q R S T
    // 4|U V X Y Z

    // NB : On considère dans cette fonction que le carré passé en paramètre est valide (càd constitué
de 25 lettres distinctes en majuscule, le W se substituant en V)

    void afficherCarre(String carre){
    }

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

    // La fonction String coderLettre (String carre, char lettre) retourne une chaîne de 2 caractères
(2 entiers entre 0 inclus et LARGEUR exclus) contenant l'encodage du caractère lettre passé en paramètre
en considérant le carré de Polybe carre également passé en paramètre.
    // Par exemple :
    // si on considère le carré de Polybe sans clé (càd le carré ABCDE représenté par la chaine "ABCDEFGHI
:
    // FGHIJ
    // KLMNO
    // PQRST
    // UVWYZ
    // 'A' est codé "00"
    // 'B' est codé "01"
    // 'F' est codé "10"
    // 'V' est codé "41"
    // 'W' est codé "41"
    // 'Z' est codé "44"
    // 'P' est codé "30"

    // si on considère le carré de Polybe donné par la chaine "AZERTYUIOPQSDFGHJKLMXCVBN" :
```

```
    // 'A' est codé "00"
    // 'B' est codé "43"
    // 'Z' est codé "01"

    // NB : On considère dans cette fonction que le carré passé en paramètre est valide (càd constitué
de 25 lettres distinctes en majuscule, le W se substituant en V)

    // Indication :  pensez à la division entière et au modulo

    String coderLettre(String carre, char lettre){
        return "";
    }

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

    // La fonction String coderMessage (String carre, String message) retourne une chaîne de caractères
contenant l'encodage de la chaîne de caractères message passé en paramètre avec le carré de Polybe carre
donné en paramètre.
    // Chaque paire d'entiers (compris entre 0 et 4) correspondant à chaque lettre sera séparée de la
suivante par un espace.
    // Pensez à utiliser la fonction coderLettre.
    // Par exemple, si le carré considéré est celui sans clé ("ABCDEFGHIJKLMNOPQRSTUVXYZ") et le message
à coder est "BONJOUR" alors le résultat attendu est "01 24 23 14 24 40 32 "

    // NB : On considère dans cette fonction que le carré passé en paramètre est valide (càd constitué
de 25 lettres distinctes en majuscule, le W se substituant en V)
    // NB : On considère dans cette fonction que le message passé en paramètre est valide (càd constitué
uniquement des 26 lettres de l'alphabet en majuscule)

    String coderMessage(String carre, String message){
        return "";
    }
    ////////////////////////////////////////////////////////////////////////

    // La fonction String decoderMessage (String carre, String messageCode) retourne une chaîne de caractè
contenant le décodage de la chaîne de caractère messageCode avec le carré de Polybe carre donné en paramètr
    // Par exemple, si carre = "ABCDEFGHIJKLMNOPQRSTUVXYZ" et messageCode = "01 24 23 14 24 40 32 "
alors le résultat attendu est "BONJOUR"

    // NB : On considère dans cette fonction que le carré passé en paramètre est valide (càd constitué
de 25 lettres distinctes en majuscule, le W se substituant en V)
    // NB : On considère dans cette fonction que le message codé passé en paramètre est valide (càd
constitué de paires d'entiers compris entre 0 et LARGEUR-1 inclus et séparées par un espace)

    String decoderMessage(String carre, String messageCode){
        return "";
    }

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

    // La fonction boolean estPresent(String mot, char lettre) retourne True si le caractère lettre
est dans mot, faux sinon.
    // Par exemple :
    // si mot = "BONJOUR" et lettre = 'B' alors le résultat de la fonction est True
    // si mot = "BONJOUR" et lettre = 'R' alors le résultat de la fonction est True
    // si mot = "BONJOUR" et lettre = 'M' alors le résultat de la fonction est False

    boolean estPresent(String mot, char lettre){
        return true;
    }

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

    // La fonction String initialiserCarreAvecCle(String cle) retourne une chaine de caractères contenant
le carré de Polybe amélioré en considérant la clé passée en paramètre.
    // Pensez à utiliser la fonction estPresent
    // Par exemple, si cle = "BONJOUR" alors le résultat attendu est :  "BONJURACDEFGHIKLMPQSTVXYZ"
    // Par exemple, si cle = "BUTINFORMATIQUE" alors le résultat attendu est :  "BUTINFORMAQECDGHJKLPSVXYZ
```

```
    // NB : On considère dans cette fonction que la clé passée en paramètre est valide (càd constituée
uniquement de lettres de l'alphabet en majuscule, le W se substituera en V)

    String initialiserCarreAvecCle(String cle){
        return "";
    }

    ////////////////////////////////////////////////////////////////////////
    // LES FONCTIONS QUI SUIVENT SONT DES FONCTIONS DE VERIFICATION DE SAISIE
    ////////////////////////////////////////////////////////////////////////

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

    // La fonction boolean estLettreMajuscule(char c) vérifie le caractère passé en paramètre est une
lettre de l'alphabet en majuscule
    // Par exemple :
    // si c='b', la fonction retourne false
    // si c='B', la fonction retourne true

    boolean estLettreMajuscule(char c){
        return true;
    }

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

    // La fonction estCleValide vérifie que la clé passée en paramètre est valide (càd constituée uniqueme
de lettres de l'alphabet en majuscule)
    // Par exemple :
    // si cle="BUTINFORMATIQUE", la fonction retourne true
    // si cle="BUTINF ORMATIQUE", la fonction retourne false
    // si cle="BUTINFORMATIQUE!", la fonction retourne false
    // si cle="ButInformatique", la fonction retourne false

    boolean estCleValide(String cle){
        return true;
    }

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

    // La fonction estChiffreOK vérifie que le chiffre passé en paramètre est valide (càd est un entier
compris entre 0 et LARGEUR-1)
    // Par exemple :
    // si messageCode=""01 24 23 14 24 40 32 ", la fonction retourne true
    // si messageCode=""01 24 23 14 24 40 32", la fonction retourne false
    // si messageCode=""01 24 23 14 24 40 3", la fonction retourne false
    // si messageCode=""01 25 23 14 24 40 32 ", la fonction retourne false
    // si messageCode=""01242314244032", la fonction retourne false

    boolean estChiffreOK(int chiffre){
        return true;
    }

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

    // La fonction estMessageCodeValide vérifie que le message codé passé en paramètre est valide (càd
constituée uniquement de paires d'entiers compris entre 0 et LARGEUR-1 et que chaque paire est séparée
de la suivante par un espace, et un espace final)
    // Par exemple :
    // si messageCode=""01 24 23 14 24 40 32 ", la fonction retourne true
    // si messageCode=""01 24 23 14 24 40 32", la fonction retourne false
    // si messageCode=""01 24 23 14 24 40 3", la fonction retourne false
    // si messageCode=""01 25 23 14 24 40 32 ", la fonction retourne false
    // si messageCode=""01242314244032", la fonction retourne false

    boolean estMessageCodeValide(String messageCode){
```

```
        return true;
    }

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

    // La fonction estMessageValide vérifie que le message passé en paramètre est valide (càd constitué uniquement de lettres de l'alphabet en majuscule)

    boolean estMessageValide(String message){
        return true;
    }

    ////////////////////////////////////////////////////////////////////////
    // PROGRAMME PRINCIPAL
    ////////////////////////////////////////////////////////////////////////

    // Ecrire un programme principal qui :
    // 1.  affiche un message d'introduction à l'utilisateur
    // 2.  affiche un message à l'utilisateur demandant s'il veut utiliser une clé ?
    // 3.  lit la réponse de l'utilisateur
    // 4.  si l'utilisateur a répondu oui, demande et lit la clé souhaitée
    // 5.  affiche le carré de Polybe (générique (càd sans clé) ou avec clé selon la réponse précédente de l'utilisateur)
    // 6.  tant que la réponse n'est pas 0, affiche un menu et demande à l'utilisateur de saisir un entier (0 ou 1 ou 2 ou 3) pour :
    // 0.  QUITTER
    // 1.  CODER UN MESSAGE
    // 2.  DECODER UN MESSAGE
    // 3.  MODIFIER LE MODE AVEC/SANS CLE
    // puis agit en conséquence.
    // NB : si et tant qu'une saisie de l'utilisateur n'est pas correcte, il faut la redemander (que ce soit pour la clé, le message à coder, le message à décoder ou le choix dans le menu)

    void algorithm(){
        println("SAE1.01 - Le carré de Polybe");
    }
    ////////////////////////////////////////////////////////////////////////
}
```

Remarque importante :
Attention, dans cette SAE, l'usage des tableaux est INTERDIT.
Le carré de Polybe sera représenté par une chaine de caractères.
Par exemple, le carré de Polybe avec la clé BUTINFORMATIQUE (cf fig.3) sera représenté par la chaine de caractère : "BUTINFORMAQECDGHJKLPSVXYZ".
A partir de la position d'une lettre dans la chaine, il s'agit alors de retrouver sa position dans un carré 5x5.
(**indication** : pensez à la division entière et au modulo % !)

3 Calendrier des rendus intermédiaires et livrables

NB : Travail à réaliser en **trinôme**

- constitution des **trinômes**
  - au plus tard le **vendredi 30 septembre 2022 à 11h59** : renseignez la constitution de votre trinôme en complétant le fichier suivant : `https://nextcloud.univ-lille.fr/index.php/s/fHj5WpYHGrgPb8p`
  - **Attention, les trinômes doivent être constitués au sein d'un même groupe**.
- dépôts intermédiaires hebdomadaires : il s'agit de déposer sur Moodle (page de la SAE1.01) l'état d'avancement de votre code c'est-à-dire un fichier nommé **SAEpolybe_NOM1_NOM2_NOM3.java**
  - dépot intermédiaire 1 :
    - date limite : **lundi 10 octobre à 8h**

    - a minima les fonctions suivantes sont attendues : initialiserCarreSimple, afficherCarre
  - dépot intermédiaire 2 :
    - date limite : **lundi 17 octobre à 8h**
    - a minima les fonctions suivantes sont attendues : *initialiserCarreSimple, afficherCarre*, coderLettre, coderMessage, decoderMessage, estPresent, initialiserCarreAvecCle
  - dépot intermédiaire 3 :
    - date limite : **lundi 24 octobre à 8h**
    - toutes les fonctions sont attendues : *initialiserCarreSimple, afficherCarre, coderLettre, coderMessage, decoderMessage, estPresent, initialiserCarreAvecCle*, et les fonctions de vérification de saisie : estLettreMajuscule, estCleValide, estChiffreOK, estMessageCodeValide, estMessageValide

- rendu final : dépôt sur Moodle d'une archive nommée **SAEpolybe_NOM1_NOM2_NOM3.zip**
  - date limite : **lundi 14 novembre à 10h**
  - l'archive devra comprendre :
    - le code source complet, documenté et commenté (**toutes les fonctions et le programme principal**),
    - un document (.pdf) illustrant tous les tests que vous avez effectués pour vérifier vos fonctions, c'est-à-dire les exemples sur lesquels les fonctions ont été testées et les résultats obtenus. Cela peut par exemple être sous la forme d'un tableau récapitulatif par fonction comme ci-dessous :

Nom de la fonction : lettreEnNombre		
Signature de la fonction : int lettreEnNombre (char lettre)		
Description de la fonction : fonction qui retourne un entier correspondant à la position d'une lettre donnée dans l'alphabet (A=0, B=1, C=2, ...)		
#param1 : lettre (char)	#resultat attendu : int	#resultat obtenu : int
'A'	0	0
'B'	1	1
'Z'	25	25

Figure 4: Vérification d'une fonction sur un exemple

    - un fichier **readme.txt** (ou **readme.md**) indiquant le trinôme, comment compiler et exécuter le fichier, toute autre information jugée utile,
    - une (ou des) capture(s) d'écran ou illustrations(s) illustrant votre projet (en vue de la construction de votre e-portfolio),
    - une courte vidéo de la démonstration de votre projet (en vue de la construction de votre e-portfolio).

- **contrôle TP** : les compétences que vous avez développées pour répondre au cahier des charges du présent sujet (simulation d'un codeur/décodeur du carré de Polybe), seront à réemployer pour l'implémentation d'une autre demande "client"
  - à réaliser **seul**
  - date : jeudi 24 Novembre à 13h30 (date à confirmer)
  - durée : 1h30