Exercice 1

Quels résultats fournit le programme suivant ?

```
public class Chaine
{ public static void main (String args[])
{ String ch1 = new String();
System.out.println ("A - ch1 =:" + ch1 + ":") ;
String ch2 = "hello" ;
System.out.println ("B - ch2 =:" + ch2 + ":") ;
String ch3 = new String ("bonjour") ;
System.out.println ("C - ch3 =:" + ch3 + ":") ;
String ch4 = new String (ch3) ;
System.out.println ("D - ch4 =:" + ch4 + ":") ;
ch3 = "bonsoir" ;
System.out.println ("E - ch4 =:" + ch4 + ": ch3 =:" + ch3 + ":") ;
ch4 = ch3 ;
ch3 = "au revoir" ;
System.out.println ("F - ch4 =:" + ch4 + ": ch3 =:" + ch3 + ":") ;
}
}
```

Exercice 2

Écrire un programme qui lit une chaîne au clavier et qui en affiche :

- un caractère sur deux (le premier étant affiché),
- le premier et le dernier caractère.

Exercice 3

Écrire un programme qui lit un mot au clavier et qui indique combien de fois sont présentes chacune des voyelles *a*, *e*, *i*, *o*, *u* ou *y*, que celles-ci soient écrites en majuscules ou en minuscules, comme dans cet exemple

```
Donnez un mot : Anticonstitutionnellement
Il comporte
1 fois la lettre a
3 fois la lettre e
3 fois la lettre i
2 fois la lettre o
1 fois la lettre u
0 fois la lettre y
```

Exercice 4

Écrire un programme qui récupère deux entiers sur la "ligne de commande" et qui en affiche la somme en fenêtre console, comme dans cet exemple :

12 + 25 = 37

On vérifiera que les arguments fournis sont formés uniquement de chiffres (sans aucun signe); dans le cas contraire, le programme s'interrompra.

Exercice 5

En cryptographie, le chiffrement par décalage, aussi connu comme le chiffre de César, est une méthode de chiffrement très simple utilisée par Jules César dans ses correspondances secrètes (ce qui explique le nom « chiffre de César »).

Le texte chiffré s'obtient en remplaçant chaque lettre du texte clair original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet. Pour les dernières lettres (dans le cas d'un décalage à droite), on reprend au début. Par exemple avec un décalage de 3 vers la droite, A est remplacé par D, B devient E, et ainsi jusqu'à W qui devient Z, puis X devient A etc. Il s'agit d'une permutation circulaire de l'alphabet. La longueur du décalage, 3 dans l'exemple évoqué, constitue la clé du chiffrement qu'il suffit de transmettre au destinataire — s'il sait déjà qu'il s'agit d'un chiffrement de César — pour que celui-ci puisse déchiffrer le message.

L'objectif de cet exercice est d'écrire une classe (*CaesarCode*) qui permet de crypter et de décrypter des messages codés selon le chiffrement de César. Cette classe devra disposer au minimum des méthodes suivantes :

- **public static String crypter(String message, int clé)** qui devra procéder au chiffrement du message fourni en paramètre
- **public static String decrypter(String message, int clé)** qui devra déchiffrer le message encodé fourni en paramètre.

La classe devra être faite de sorte que pour l'utiliser, on procédera de la sorte :

- java CaesarCode –c messageACrypter –k *clé* pour crypter un message avec la clé spécifiée
- java CaesarCode –d messageADecrypter –k *clé* pour décrypter un message avec la clé spécifiée

NB : L'option « -k » devra être optionnelle et prendra la valeur « 13 » si elle n'est pas spécifiée. L'une des options « -c » ou « -d » doit impérativement être présente pour qu'un cryptage ou un décryptage puisse se faire.
De même, les appels tels que ***java CaesarCode, java CaesarCode -h ou encore java CaesarCode --help*** devront fournir en sortie l'aide du programme qui donnera des instructions sur comment l'utiliser.

Exercice 6

Faites une classe appelée *ConjugaisonIndicatif* qui permet la conjugaison de tout verbe régulier du premier groupe aux temps de l'indicatif que sont le présent, le passé simple, l'imparfait et le futur simple. La classe devra disposer des attributs suivants :

- *verbe*, le verbe à conjuguer
- *pronoms* contiendra la liste des pronoms personnels
- *terminaisonPrésent* contiendra la liste des terminaisons au présent. Il en est de même pour les attributs *terminaisonPassé*, *terminaisonFutur* et *terminaisonImparfait* qui contiendront la liste des terminaisons pour leurs temps respectifs.

Elle devra aussi définir les méthodes suivantes :

- ConjugaisonIndicatif(String verbe), ce constructeur vérifiera (de manière basique) que le verbe donné est bien du 1er groupe avant de l'initialiser.
- Les *getters* et/ou *setters* des attributs qui en ont besoin (bien contrôler les informations renseignées par l'utilisateur avant toute mise à jour)
- public void present()
- public void passe()
- public void futur()
- public void imparfait()

Ecrire une classe TestConjugaisonIndicatif qui permet de tester le bon fonctionnement de votre classe de conjugaison. Le programme de test devra donner à l'utilisateur plusieurs options d'utilisation.

- java TestConjugaisonIndicatif –v verbeAConjuguer : Dans ce cas, le programme donnera en sortie la conjugaison du verbe spécifié dans tous les temps (passé, futur, etc.)
- java TestConjugaisonIndicatif –v verbeAConjuguer –t tempsDeConjugaison : Dans ce cas, le programme donnera en sortie la conjugaison du verbe au temps spécifié. L'option « -t » ne pourra prendre que les valeurs « présent », « passé », « imparfait » et « futur ».
- java TestConjugaisonIndicatif : Dans ce cas, le programme demande à l'utilisateur le verbe qu'il souhaite conjuguer puis le temps de conjugaison. La validation de ces informations devra se faire et si l'utilisateur ne met rien pour le temps, on considère que la conjugaison se fera sur tous les temps.
- java TestConjugaisonIndicatif –h devront fournir en sortie l'aide du programme qui donne les instructions sur l'utilisation du programme. Cette aide devrait aussi être affichée à chaque fois que les options « -v » ou « -t » sont fournies avec une valeur incorrecte.