Les chaînes de caractères (String)

Les chaînes de caractères (de type String) sont des objets. En tant qu'objets, les chaînes de caractères possèdent des **attributs** et il est possible d'utiliser des **méthodes** pour les manipuler.

Lorsqu'on utilise le nom d'une variable String, on utilise une **référence** à l'objet, c'est-à-dire son adresse en mémoire et non le contenu de la chaîne.

1. Déclaration d'une chaîne

```
String vide   = "";                          // chaîne vide

String monNom = "Claudette Chapleau";

String erreur;                               // chaîne non initialisée

String voyelles = new String("aeiouy");
String voyelles = "aeiouy";                  // déclarations équivalentes

String exemple = "Éclipse";
```

La valeur de **exemple** est `Éclipse`
son nombre de caractères est `7`
la position du premier caractère est `0`
la position du dernier caractère est `6`

É	c	l	i	p	s	e
0	*1*	*2*	*3*	*4*	*5*	*6*

2. Concaténation

L'opérateur **+** permet de concaténer des éléments pour former une nouvelle chaîne de caractères. Si les éléments sont des données de type primitif, ils seront automatiquement convertis en String. Cependant, au moins un des éléments doit être une chaîne de caractères.

```
char etoile = '*';
int numero = 450;
boolean ok = true;
String resultat;
String ville = "Laval";
resultat = ok + " pour " + ville + etoile + ' ' + numero;
```

La valeur de **resultat** est `true pour Laval* 450`

L'opérateur **+=** permet de concaténer des éléments à la fin d'une chaîne de caractères existante.

```
String ville = "Laval";
final String QUEBEC = "Québec";
ville += ", " + QUEBEC;
```

La valeur de **ville** est `Laval, Québec`

3. Longueur d'une chaîne

La méthode **`length()`** retourne un entier donnant la longueur de la chaîne de caractères.

```
String monNom = "Claudette Chapleau";
int nbCar;
nbCar = monNom.length();      // même les espaces comptent !
```

La valeur de **`nbCar`** est **`18`**

```
nbCar = ("Mme " + monNom).length();
```

La valeur de **`nbCar`** est **`22`**

```
String vide = "";
nbCar = vide.length();
```

La valeur de **`nbCar`** est **`0`**

```
String resultat;
nbCar = resultat.length();
```

Cause une erreur car la variable **`resultat`** n'a pas été initialisée.

Si on doit lire quelque chose au clavier et qu'on appuie sur la touche **Entrée** sans rien taper à la console (ou dans un **`showInputDialog`**), la chaîne obtenue est **`""`** et sa longueur est 0.

4. Extraction d'un caractère

La méthode **`charAt(n)`** retourne le caractère à la position **`n`** d'une chaîne de caractères.

```
String monNom = "Claudette Chapleau";
char carac;
carac = monNom.charAt(0);
```

La valeur de **`carac`** est **`'C'`**

```
int position = 2;
carac = monNom.charAt(position);
```

La valeur de **`carac`** est **`'a'`**, soit le caractère à la **3e** position de **`monNom`**

```
carac = monNom.charAt(monNom.length() - 1);
```

La valeur de **`carac`** est **`'u'`**, soit le dernier caractère de **`monNom`**

```
String vide = "";
carac = vide.charAt(0);
```

Cause une erreur - **`StringIndexOutOfBoundsException`** - car on ne peut extraire un caractère d'une chaîne vide.
On aurait une erreur semblable si on essayait d'aller chercher un caractère à une position qui dépasse celle du dernier caractère d'une chaîne.

5. Extraction d'une sous-chaîne

La méthode **`substring(début, fin)`** retourne la sous-chaîne commençant à la position **`début`** et se terminant à la position **`fin - 1`** d'une chaîne de caractères. Si **`fin`** n'est pas indiqué, la sous-chaîne se termine avec la fin de la chaîne de caractères.

```
String monNom = "Claudette Chapleau";
String resultat;
resultat = monNom.substring(0, 5);
```

La valeur de **`resultat`** est `Claud`, soit les 5 premiers caractères de **`monNom`**

```
resultat = monNom.substring(10, 13);
```

La valeur de **`resultat`** est `Cha`

```
resultat = monNom.substring(15);
```

La valeur de **`resultat`** est `eau`, soit les 3 derniers caractères de **`monNom`**

```
resultat = monNom.substring(monNom.length() - 3);
```

La valeur de **`resultat`** est `eau`, soit les 3 derniers caractères de **`monNom`**
Cette dernière façon de faire est plus logique car on n'a pas besoin de calculer la position de départ !

```
String titre = "Mme";
resultat = titre + ' ' + "Claudette Chapleau".substring(10);
```

La valeur de **`resultat`** est `Mme Chapleau`

6. "Nettoyage" d'une chaîne

La méthode **`trim()`** retourne une chaîne de caractères composée des caractères de la chaîne initiale moins les espaces du début et de la fin.

```
String texte = "   Cours 101          ";
String resultat;
resultat = texte.trim();
```

La valeur de **`resultat`** est `Cours 101`

7. Position d'un élément dans une chaîne

Pour chercher la position d'un caractère dans une chaîne, on pourrait toujours faire une boucle pour parcourir la chaîne en comparant le caractère cherché avec chaque caractère de la chaîne. Si le caractère ne se trouve pas dans la chaîne, on s'attend d'obtenir la position **-1**.

```
String monAdresse = "9155 St-Hubert, Mtl, Qc";
int position = -1; // on initialise à -1 ce qui signifie pas trouvé
char recherche = 't';

// boucle pour parcourir la chaîne
//    on arrête le parcours si la chaîne est terminée
//    ou qu'on a trouvé le caractère cherché
for (int i = 0; i < monAdresse.length() && position == -1; i++)
   if (monAdresse.charAt(i) == recherche)
      position = i;
```

La valeur de **position** est **6**, soit la position du premier **t**

La méthode **indexOf(élémentCherché, début)** retourne un entier indiquant la position de la première occurrence de **élémentCherché** dans la chaîne de caractères, de gauche à droite, à partir de la position **début**. Si **élémentCherché** n'est pas trouvé, la valeur **-1** est retournée. Si **début** n'est pas indiqué, on commence la recherche au début de la chaîne. (**élémentCherché** peut être un caractère (type **char**) ou une autre chaîne de caractères)

Si on reprend l'exemple précédent en utilisant la méthode **indexOf**, on simplifie le code de beaucoup :

```
String monAdresse = "9155 St-Hubert, Mtl, Qc";
int position; // pas besoin de l'initialiser !
char recherche = 't';
position = monAdresse.indexOf(recherche);
```

La valeur de **position** est **6**, soit la position du premier **t**

Si on recherche une chaîne de caractères :

```
String monAdresse = "9155 St-Hubert, Mtl, Qc";
int position;
String recherche = "Mtl";
position = monAdresse.indexOf(recherche);
```

La valeur de **position** est **16**, soit la position du premier **Mtl**

```
String monNom = "Claudette Chapleau";
int position;
position = monNom.indexOf('C');
```

La valeur de **position** est **0**, soit la position du premier **C**

```
position = monNom.indexOf('C', 3);
```

La valeur de **position** est **10**, soit la position du premier **C** après le 4e caractère

```
position = monNom.indexOf("di");
```

La valeur de **position** est **-1**, car cette chaîne n'existe pas dans **monNom**

```
String prenom;
String famille;
prenom = monNom.substring(0, monNom.indexOf(" "));
famille = monNom.substring(monNom.indexOf(" ") + 1);
```

La valeur de **prenom** est **Claudette**
La valeur de **famille** est **Chapleau**

Pour ne pas faire deux fois l'appel à la méthode **indexOf**, on pourrait conserver le résultat et s'en servir deux fois. C'est aussi plus facile à lire !

```
int posEspace = monNom.indexOf(" ");
prenom = monNom.substring(0, posEspace);
famille = monNom.substring(posEspace + 1);
```

La méthode **lastIndexOf(élémentCherché, début)** retourne un entier indiquant la position de la première occurrence de **élémentCherché** dans la chaîne de caractères, de droite à gauche, à partir de la position **début**. Si **élémentCherché** n'est pas trouvé, la valeur **-1** est retournée. Si **début** n'est pas indiqué, on commence la recherche **à la fin** de la chaîne.

```
String exemple = "ABC DEF ghi jkl MNO";
int posDernierEspace = exemple.lastIndexOf(" ");
int posAvantDernier = exemple.lastIndexOf(" ", posDernierEspace - 1);
```

La valeur de **posDernierEspace** est **15**
La valeur de **posAvantDernier** est **11**

8. Conversion minuscules/majuscules

La méthode **`toUpperCase()`** retourne une chaîne dont toutes les lettres minuscules ont été converties en majuscules. La méthode **`toLowerCase()`** retourne une chaîne dont toutes les lettres majuscules ont été converties en minuscules. Tous les autres caractères sont inchangés.

```
String monNom = "Claudette Chapleau";
String majuscules = monNom.toUpperCase();
```

La valeur de **`majuscules`** est **`CLAUDETTE CHAPLEAU`**

```
String minuscules = monNom.toLowerCase();
```

La valeur de **`minuscules`** est **`claudette chapleau`**

```
String resultat;
resultat = monNom.substring(0, monNom.indexOf(" ")) +
           monNom.substring(monNom.indexOf(" ")).toUpperCase();
```

La valeur de **`resultat`** est **`Claudette CHAPLEAU`**

```
String saisie;
char reponse;
saisie = JOptionPane.showInputDialog("Entrez votre réponse (Oui/Non) :");
reponse = saisie.toLowerCase().charAt(0);
```

Si l'usager a tapé **`Non`**, la valeur de **`reponse`** est **`'n'`**

9. Comparaison de chaînes

La méthode **`equals(autreChaîne)`** retourne **`true`** si le contenu de la chaîne est égal au contenu de l'**`autreChaîne`**. La méthode **`equalsIgnoreCase(autreChaîne)`** fait la même chose mais ne s'occupe pas de la casse (majuscules/minuscules).

```
String maVille = "Laval";
boolean pareil;

pareil = maVille.equals("Laval");
```

Attention, si on écrit **`"Laval "`**, les deux chaînes ne seront pas égales car elles ne sont pas de la même longueur !

La valeur de **`pareil`** est **`true`**

```
pareil = maVille.equals("LAVAL");
```

La valeur de **`pareil`** est **`false`**

```
pareil = maVille.equalsIgnoreCase("LAVAL");
```

La valeur de **`pareil`** est **`true`**

```
pareil = maVille == "Laval";
```

La valeur de **`pareil`** est **`false`** car l'objet **`maVille`** n'est pas à la même adresse-mémoire que l'objet **`"Laval"`**

10. Conversions

Conversion d'une chaîne de caractères en donnée d'un autre type

Les méthodes suivantes permettent d'obtenir une valeur à partir d'une chaîne de caractères déclarée `String texte`.

type obtenu	exemples
`int`	`ordinaire = Integer.parseInt(texte);`
`byte`	`petit = Byte.parseByte(texte);`
`short`	`moyen = Short.parseShort(texte);`
`long`	`grand = Long.parseLong(texte);`
`double`	`ventes = Double.parseDouble(texte);`
`float`	`taux = Float.parseFloat(texte);`
`boolean`	`ok = Boolean.getBoolean("true");`

Conversion d'une donnée d'un autre type en chaîne de caractères

Les méthodes suivantes permettent d'obtenir une chaîne de caractères `texte` à partir d'un autre type de donnée.

texte obtenu	exemples
`"123"`	`texte = String.valueOf(123);` La donnée fournie en paramètre est convertie en `String` si son type est un des types primitifs.
`"15.8"`	`texte = "" + 15.8;` Toute donnée concaténée à une chaîne vide est convertie en `String` si son type est un des types primitifs.

`"1234.50$"`	`DecimalFormat argent = new DecimalFormat("0.00$");`
	`texte = argent.format(1234.5);`
	L'utilisation d'un `DecimalFormat` permet d'obtenir une chaîne de caractères à partir d'une valeur numérique.

`"145.75"`	`Double valeur = new Double(145.75);`
	`texte = valeur.toString();`
	Pour tout objet, l'appel de la méthode `toString` retourne une chaîne de caractères contenant la valeur de l'objet si la méthode `toString` est définie dans la classe de l'objet.

`"Bidon@119c082"`	`Bidon bid = new Bidon();`
	`texte = bid.toString();`
	Comme la méthode toString n'est pas définie pour la classe `Bidon`, la valeur retournée est composée du nom de la classe et de l'adresse de l'objet.

Les caractères (char)

La classe **`Character`** fournit des méthodes pour manipuler des caractères (type **`char`**).

Conversion minuscules/majuscules :

char obtenu	exemples
`'K'`	`maj = Character.toUpperCase('k');` La donnée fournie en paramètre est convertie en majuscule. Si ce n'est pas une lettre minuscule, alors elle retournera inchangée.
`'w'`	`min = Character.toLowerCase('W');` La donnée fournie en paramètre est convertie en majuscule. Si ce n'est pas une lettre minuscule, alors elle retournera inchangée.

Tests sur un caractère :

boolean obtenu	exemples
`true`	`resultat = Character.isLetter('k');`
`false`	`resultat = Character.isLowerCase('W');`
`true`	`resultat = Character.isUpperCase('W');`
`false`	`resultat = Character.isLetterOrDigit('*');`
`true`	`resultat = Character.isDigit('4');`
`true`	`resultat = Character.isSpaceChar(' ');`

Exemples combinés :

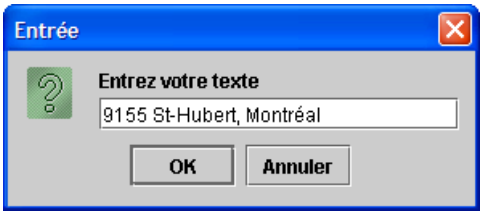
exemples
`numero = Integer.parseInt(texte.substring(0, 4));` si `texte` vaut `"12345678"`, la valeur de `numero` sera `1234`
`morceau = texte.substring(3, 6).toLowerCase();` si `texte` vaut `"AAbbCCddEEff"`, la valeur de `morceau` sera `"bcc"`
`if (Character.toUpperCase(texte.charAt(0)) == 'Y')` donnera la valeur `true` si le premier caractère de `texte` est `'Y'` ou `'y'`
`if (Double.parseDouble(texte.substring(8)) < 12.5)` donnera la valeur `true` si `texte` vaut `"1.0-2.0-3.0"`
`morceau = (String.valueOf(123) + 'x').toUpperCase();` la valeur de `morceau` sera `"123X"`
`dernier = texte.charAt(texte.length() - 1);` si `texte` vaut `"AbcdEfghIjklMnop"`, la valeur de `dernier` sera `'p'`

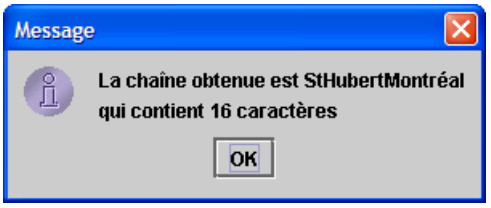
`reponse = Character.toUpperCase(JOptionPane.showInputDialog(`
`"Entrez O ou N").charAt(0));`
place dans `reponse` la conversion en majuscule du premier caractère saisi

Exemple 1 : on veut lire une chaîne de caractères et créer une nouvelle chaîne formée uniquement des lettres de la chaîne lue. On veut afficher cette nouvelle chaîne ainsi que sa longueur.

Programme en java :

```
import javax.swing.*;
public class Exemple1 {

    public static void main(String[] args) {

        // déclaration des variables
        String texte,
               resultat = "";
        char carac;

        // lecture du texte
        texte = JOptionPane.showInputDialog("Entrez votre texte");

        // parcours du texte pour extraire seulement les lettres
        for (int i = 0; i < texte.length(); i++)
        {
            carac = texte.charAt(i);
            if (Character.isLetter(carac))
                resultat += carac;
        }

        // affichage des résultats
        JOptionPane.showMessageDialog(null,
            "La chaîne obtenue est " + resultat +
            "\nqui contient " + resultat.length() + " caractères");
        System.exit(0);
    }
}
```



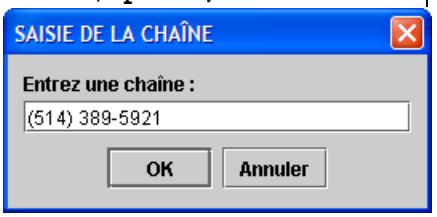


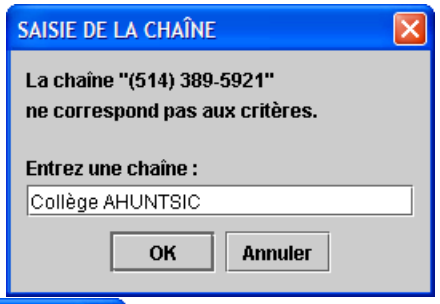
Exemple 2 : on veut valider une chaîne de caractères

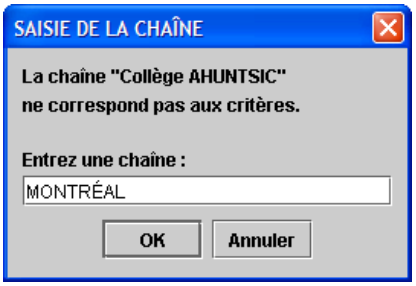
Programme en java :

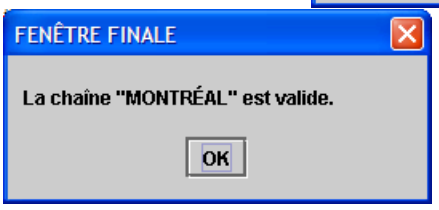
```
/*
 * Auteur : Claudette Chapleau
 * Date   : septembre 2006
 * Ce programme doit lire une chaîne et la valider
 *    - la chaîne doit avoir de 5 à 15 caractères
 *    - la chaîne ne doit comporter que des lettres majuscules
 * Si la chaîne est invalide, on doit afficher un message
 * et demander une nouvelle chaîne
 */
import javax.swing.*;
public class ValidationChaine {
    public static void main(String[] args) {
        final int LONGUEUR_MIN = 5,
                  LONGUEUR_MAX = 15;

        String chaine,
               messageErreur = "";
        char carac;
        boolean valide;

        do
        {
            chaine = JOptionPane.showInputDialog(null,
                        messageErreur + "Entrez une chaîne :",
                        "SAISIE DE LA CHAÎNE", JOptionPane.PLAIN_MESSAGE);

            // vérifier la longueur de la chaîne
            valide = chaine.length() >= LONGUEUR_MIN &&
                     chaine.length() <= LONGUEUR_MAX;

            // vérifier si la chaîne ne comporte que des majuscules
            for (int pos = 0; pos < chaine.length() && valide; pos++)
            {
                carac = chaine.charAt(pos);
                if (!Character.isUpperCase(carac))
                   valide = false;
            }

            // message pour la prochaine saisie
            messageErreur = "La chaîne \"" + chaine +
                "\"\nne correspond pas aux critères.\n\n";
        } while (!valide);

        // afficher la chaîne valide
        JOptionPane.showMessageDialog(null,
                "La chaîne \"" + chaine +
                "\" est valide.",
                "FENÊTRE FINALE",
                JOptionPane.PLAIN_MESSAGE);
        System.exit(0);
    }
}
```

Lorsque **valide** est faux, on arrête le **for**
- soit que la longueur de la chaîne est invalide
ou
- aussitôt qu'on a trouvé un caractère qui n'est pas une lettre majuscule





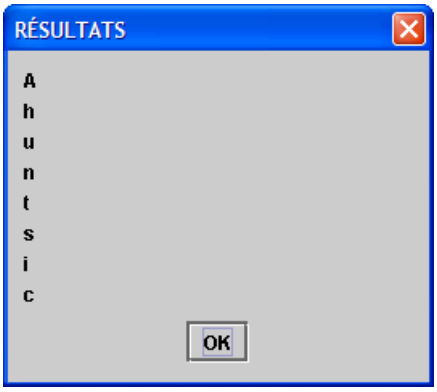




Exercice 1 : Complétez le programme qui permet de lire une chaîne de caractères et d'afficher les caractères de la chaîne lue dans une boîte de dialogue, un par ligne

Programme en java :

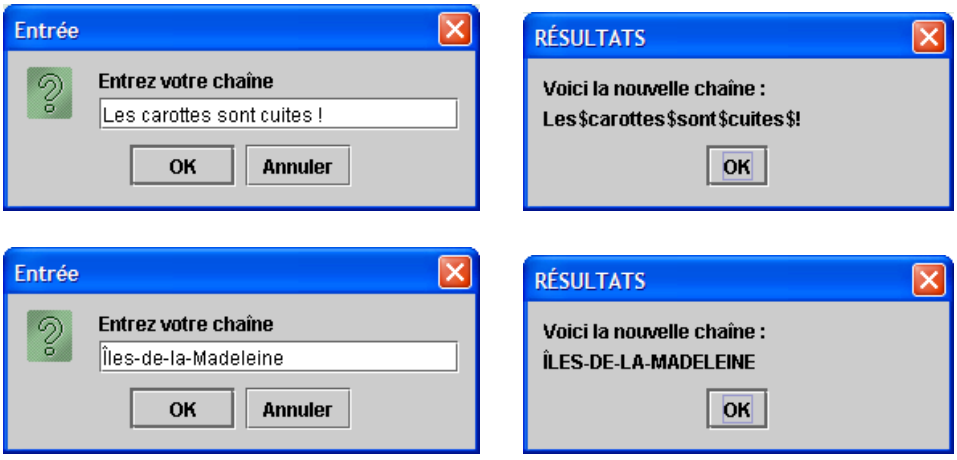
```
import javax.swing.*;
public class ExerciceChaine1
{
    public static void main(String[] args)
    {
        String chaine,
               resultat = "";

        // lecture de la chaîne



        // parcours de la chaîne pour mettre les caractères un par ligne




        // affichage des résultats
        JOptionPane.showMessageDialog(null,
            resultat, "RÉSULTATS",
            JOptionPane.PLAIN_MESSAGE);
        System.exit(0);
    }
}
```



Exercice 2 : Complétez le programme qui permet de lire une chaîne de caractères et de remplacer tous les espaces par des $. Si la phrase ne possède aucun espace, on doit plutôt convertir la chaîne lue en majuscules.

Programme en java :

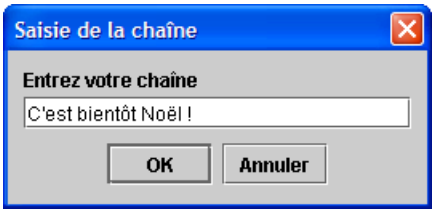
```
import javax.swing.*;
public class ExerciceChaine2
{
    public static void main(String[] args)
    {
        final char DOLLAR = '$';
        String chaine,
               resultat = "";
        char carac;
        boolean trouveEspace = false;

        // lecture de la chaîne


        // parcours de la chaîne pour remplacer les espaces









        // traitement si trouvé aucun espace



        // affichage des résultats
        JOptionPane.showMessageDialog(null,
            "Voici la nouvelle chaîne :\n" + resultat, "RÉSULTATS",
            JOptionPane.PLAIN_MESSAGE);
        System.exit(0);
    }
}
```

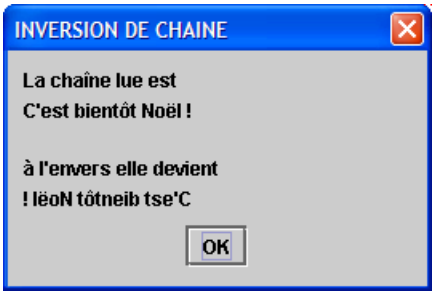


Exercice 3 : Complétez le programme qui permet de lire une chaîne de caractères et de l'afficher à l'envers.

Programme en java :

```
import javax.swing.*;
public class ExerciceChaine3 {

    public static void main(String[] args) {
        // déclaration des variables nécessaires
        String chaineLue,
               envers = "";

        // lecture de la chaîne
        chaineLue =




        // inversion de la chaîne









        // affichage des résultats
        JOptionPane.showMessageDialog(null,
               "La chaîne lue est\n" + chaineLue +
               "\n\nà l'envers elle devient\n" + envers,
               "INVERSION DE CHAINE", JOptionPane.PLAIN_MESSAGE);

        // fin de l'exécution
        System.exit(0);
    }
}
```





Exercice 4 : Complétez le programme qui permet de lire une chaîne de caractères et de l'afficher en soulignant les consonnes par =.

Programme en java :

```
import javax.swing.*;
import java.awt.Font;
public class ExerciceChaine4 {

    public static void main(String[] args) {
        // déclaration des variables nécessaires
        final String CONSONNES = "BCDFGHKJLMNPQRSTVWXZ";
        String chaine,
               soulignement = "";
        char carac;

        JTextArea sortie = new JTextArea();
        sortie.setFont(new Font("Courier", Font.BOLD, 15));

        // lecture de la chaîne
        chaine =




        // parcours pour souligner les consonnes

















        // affichage des résultats dans un JTextArea
        sortie.setText(chaine + "\n" + soulignement);
        JOptionPane.showMessageDialog(null, sortie,
               "SOULIGNEMENT DES CONSONNES", JOptionPane.PLAIN_MESSAGE);

        // fin de l'exécution
        System.exit(0);
    }
}
```