La classe java.lang.String

Exercices relatifs aux chaînes de caractères en Java

Thierry BROUARD

23 octobre 2017

1 Exercices dirigés - TD

Les premiers exercices sont à faire sur papier, puis sur machine. La raison est que sur machine on porte moins attention à l'analyse du problème et à la réflexion. De même, il est souvent utile de schématiser, ce qui n'est pas pratique sur machine. On se sert beaucoup de l'écrit pour réfléchir ou aider à la réflexion. Travailler sur machine n'aide donc pas cet aspect pourtant essentiel de l'apprentissage. Cette première série est "encadrée", il y aura donc une correction disponible / présentée / affichée.

Exercice 1. *Écrire une méthode* `affiche(String s)` *qui affiche les différents caractères de* `s`*, séparés les uns des autres par une virgule.*
Exemple : `affiche("UE Info")` *doit afficher* `U,E, ,I,n,f,o`
Vous ferez attention au fait qu'il n'y a pas de virgule après le dernier caractère de `s`*.*

Exercice 2. *Écrire une méthode* `ajouteEtoiles(String s, int n)` *qui retourne une nouvelle chaîne correspondant à* `s` *avec* `n` *étoiles avant et* `n` *étoiles après. Cette chaîne sera ensuite affichée par le programme principal.*
Exemple : `ajouteEtoiles("UE Info",3)` *retourne la chaîne* `***UE Info***`

Exercice 3. *Écrire une méthode* `supprimeCar(String s, char c)` *qui retourne une nouvelle chaîne correspondant à* `s` *dans laquelle tous les caractères égaux à* `c` *auront été supprimés. Cette chaîne sera ensuite affichée par le programme principal.*
Exemple : `supprimeCar("chocolat",'o')` *renvoie la chaîne* `chclat`

Exercice 4. *Dans chacun des cas suivants, dîtes ce que renvoie l'expression indiquée.*

1. *Que vaut* `"Duval".substring(1,3)` *?*
2. *Que vaut* `"Marguerite".indexOf("rg")` *?*
3. *Que vaut* `"multiplication".indexOf("ti",4)` *?*
4. *Que vaut* `"banane".indexOf("an",4)` *?*

Exercice 5. *On suppose que l'on a défini et affecté deux chaînes de caractères* `s1` *et* `s2`*.*

1. *Quel test permet de savoir si elles ont le même contenu ?*
2. *Quel test permet de savoir si elles se situent au même endroit en mémoire ?*

Exercice 6. *On suppose que l'on dispose d'un tableau de chaînes de caractères, nommé* `tab`*, dont chaque élément contient un code article, suivi d'un espace, suivi du rayon correspondant (en un mot) , suivi d'un espace, suivi du nom de l'article (plusieurs mots possibles), par exemple :* `tab[0] = "G1108 bricolage pack tournevis"`

1. *Écrire une méthode* `affiche(String[] t)` *qui affiche tous les éléments de* `t` *en respectant la présentation suivante :*
   `Code : G1108 ; rayon : bricolage ; article : pack tournevis`
2. *Écrire une méthode* `recap(String[] t, String ray)` *qui affiche tous les éléments de* `t` *appartenant au rayon* `ray`*, ainsi que le nombre total d'articles de ce rayon. On respectera la présentation suivante (appel de* `recap(tab, "bricolage")`*) :*
   ```
   Code : G1108 ; article : pack tournevis
   Code : H44362 ; article : perceuse
   Nombre d'articles au rayon bricolage : 2
   ```

2 Exercices autonomes - TP

Dans cette seconde série, vous êtes autonomes sur les exercices. Il n'y a pas de correction disponible comme dans la première série. Cependant vous pouvez toujours faire valider le résultat par l'enseignant, qui peut aussi vous aider et répondre à vos questions en cas de difficulté.

Exercice 7. *Pour cet exercice, le projet contient deux classe. L'une contient le programme principal, l'autre s'appellera* `DemoString` *et contiendra les méthodes décrites ci-dessous. La classe principale créera une instance de* `DemoString` *et l'utilisera de façon à en illustrer le fonctionnement. Procéder question par question : écrire l'appel dans le programme principal, créer la méthode dans la classe* `DemoString` *et vérifier l'appel, écrire le contenu de la méthode et vérifier que cela fonctionne.*

1. *Écrire une méthode* `saisie` *qui demande à l'utilisateur de saisir une chaîne de caractères et qui la renvoie (pas de taille maximale).*
2. *Écrire une méthode* `affiche(String s)` *qui affiche* `s` *selon le modèle :*
   `chaine reçue :` *s*`; longueur totale :` *y*
   où s est le contenu de la variable `s` *et y le nombre de caractères de* `s`*.*
3. *Écrire une méthode* `affiVertical(String s)` *qui affiche toute les lettres de* `s` *les unes en dessous des autres.*

4. *Écrire une méthode* `compteCar(String s, char c)` *qui renvoie le nombre d'occurrences de* `c` *dans* `s`. *Ce nombre peut aller de 0 (*`c` *n'est pas présent dans* `s`*) à un nombre quelconque.*
   Exemple, `compteCar("Honolulu",'o')` *retourne la valeur 2 car il y a deux* o *dans "Honolulu".*
5. *Écrire une méthode* `compteVoy(String s)` *qui renvoie le nombre de voyelles non accentuées (a, e, i, o, u, y) contenues dans* `s`.
   Exemple : `compteVoy("dégraissée")` *retourne 3 (a,i,e).*
6. *Écrire une méthode* `remplace(String s, char a, char b)` *qui retourne le contenu de* `s` *dans lequel tous les caractères identiques au contenu de* `a` *ont été remplacés par le contenu de* `b`.
   Exemple : `remplace("Honolulu", 'o', 'x')` *donne* `Hxnxlulu`
7. *Écrire une méthode* `dans(String e, String s)` *qui teste si une chaîne* `e` *est contenue dans une autre chaîne* `s`. *Pour cela vous utiliserez la méthode* `indexOf()`. *Si* `e` *est contenue dans* `s`, *la méthode retourne* true, *sinon elle retourne* false.

Exercice 8. *Un palindrome reste le même qu'on le lise de gauche à droite ou de droite à gauche (exemples : selles, sagas, radar, rotor, laval sont des palindromes de lettres, 1991 ou 2002 sont des palindromes de chiffres).*

1. *Écrire une méthode* `estPalindrome(String s)` *qui teste si une chaîne* `s` *est un palindrome et qui retourne, selon le cas,* true *ou* false.
2. *Ajouter une méthode* `inverse(String s)` *qui renvoie la chaîne inversée.*
   Exemple : `inverse("rabelais")` *renvoie la chaîne* `sialebar`.
3. *Déduire de la réponse précédente une autre méthode pour la première question.*

Exercice 9. *Le but de l'exercice est de produire une chaîne de façon aléatoire et de rechercher ensuite si dans la chaîne un motif particulier est présent.*

1. *Écrire une méthode, appelée* `genere()` *qui génère aléatoirement*[1] *une chaîne constituée de 10 caractères, où chaque caractère est pris parmi { ' :' , '-' , '(' , ')' }. On peut donc obtenir par exemple la chaîne* `((-:((-(-`.
2. *Écrire une méthode appelée* `gagne()` *qui renvoie* true *si la chaîne générée contient la séquence* `:-)` *et qui renvoie* false *sinon.*
3. *Écrire une méthode, appelée* `joue()`, *qui appelle* `genere()`, *affiche la chaîne obtenue puis le message "Gagné !" si la séquence* `:-)` *est présente dans la chaîne générée, et "Perdu !" sinon.*

Exercice 10. *On s'intéresse à l'amélioration des chaînes de caractères composant un texte. Il arrive par exemple que le texte contienne des espaces surnuméraires entre les mots ce qui ne permet pas ensuite une mise en page homogène dans un traitement de texte.*

1. On rappelle que l'on peut obtenir un nombre réel aléatoire entre 0 inclus et 1 exclu avec `Math.random()`.

1. *Écrire une méthode* `epure(String s)` *qui supprime les caractères espace (blancs) en trop, c.à.d. ceux au début et en fin de chaîne, et ceux entre les mots (on en laisse un à chaque fois).*
   Exemple : `epure("␣␣coucou␣␣␣␣␣le␣␣monde")` *donne* `coucou␣le␣monde`
2. *Écrire une méthode* `nbMots(String s)` *qui renvoie le nombre de mots d'une chaîne : on suppose que chaque signe de ponctuation est collé au mot qui le précède. Les seuls séparateurs possibles entre les mots sont donc un espace ou une apostrophe.*

Exercice 11. *L'écriture d'expressions incluant des parenthèses nécessite d'être vérifiée avant d'être traitée automatiquement.*

1. *Écrire une méthode* `verifie(String s)` *qui renvoie un booléen et qui vérifie qu'il y a le même nombre de parenthèses ouvrantes et fermantes dans l'expression* `s`.
2. *Modifier la méthode précédente pour vérifier de plus qu'une parenthèse fermante n'est pas là sans correspondre à une ouvrante.*
   Exemple : `( x-3 )(( y+5 ))` *est possible mais pas* `( x-3 ))(( y+5 )`

Exercice 12. *On considère un tableau de chaînes, vide mais déjà alloué. Ce tableau est destiné à stocker des mots de passe. Chacun d'entre eux est une chaîne d'au moins 8 caractères et contenant au moins un des caractères spéciaux suivants :* `&, @, #, $, %, ^ , +, =, *`

1. *Écrire une méthode* `saisiePwd()` *qui réalise la saisie d'un mot de passe et qui retourne le mot de passe saisi. On redemande la saisie du mot de passe jusqu'à ce qu'il soit valide. On utilisera par exemple une chaîne* `caracSpec` *contenant tous les caractères spéciaux ci-dessus afin de contrôler la validité du mot de passe.*
2. *Écrire une méthode* `saisieTab(String[] tab)` *qui réalise la saisie de tous les mots de passe du tableau* `tab`. *La méthode retourne le tableau saisi.*
3. *Écrire une méthode* `taillePlusCourt(String[] tab)` *qui retourne la longueur du mot de passe le plus court contenu dans le tableau* `tab`.
4. *Écrire une méthode* `taillePlusLong(String[] tab)` *qui retourne la longueur du mot de passe le plus long du tableau* `tab`.
5. *Écrire une méthode* `tailleMoyenne(String[] tab)` *qui retourne la longueur moyenne des chaînes du tableau* `tab`.
6. *Écrire une méthode* `nbCaracSpec(String s)` *qui compte le nombre de caractères spéciaux de la chaîne* `s`.

Exercice 13. *On considère un tableau de chaînes de caractères où chaque élément est une chaîne de la forme :* `Prénom : âge` *où (âge) est un entier. Écrire une méthode* `calculeMoyenne(String [] tab)` *qui retourne la moyenne des âges contenus dans chaque chaîne du tableau.*
Pour cela on utilisera `Integer.valueOf (String s)` *qui renvoie l'entier correspondant à la chaîne* `s` *ou bien* `Integer.parseInt (String s)`.

Exercice 14. *On considère une chaîne de caractères contenant des mesures d'ailes de drosophiles (une variété de mouche) en cm, exemple :* `String ch = "1.93cm;2.2cm;1.65cm;1.76cm;2.07cm;1.8cm;"`
Écrire une méthode `afficheMinMax(String s)` *qui affiche le minimum et le maximum des longueurs d'ailes. Le principe est de décomposer la chaîne afin d'isoler les parties correspondant à des nombres, puis à transformer chaque chaîne correspondant à une longueur en nombre pour ensuite en extraire les valeurs extrêmes.*

Exercice 15. *On considère un tableau de chaînes de caractères vide mais déjà alloué. On souhaite trier ce tableau de sorte à ce que les chaînes de caractères apparaissent dans l'ordre alphabétique selon leur rang dans le tableau (la chaîne la plus "près" du 'a' au début).*

1. *Écrire une méthode* `saisie(String [] tab)` *qui permet la saisie de toutes les chaînes du tableau* `tab`.
2. *Ajouter une méthode* `trie(String [] tab)` *qui trie le tableau par la méthode du tri sélection. Le principe est de parcourir le tableau, depuis le rang 0, à la recherche de l'élément le plus petit, qui sera mis au rang 0. On recommence alors sur la partie du tableau qui n'a pas encore été trié : on recherche depuis le rang 1, et on placera le plus petit élément trouvé au rang 1.*
3. *Ajouter une méthode* `affiche(String [] tab)` *qui affiche le contenu du tableau* `tab` *pour vérifier que le tableau a bien été trié.*

Exercice 16. *On va demander à la machine de conjuguer les verbes réguliers du 1er et du 2e groupe. Pour cela on utilise :*

1. *un tableau* `sujet` *qui les pronoms (je, tu, il...)*
2. *un tableau* `fin_er` *qui contient les terminaisons du présent, de l'imparfait et du futur pour les verbes du 1er groupe*
3. *un tableau* `fin_ir` *qui contient les terminaisons du présent, de l'imparfait et du futur pour les verbes du 2e groupe*

Pour faciliter la lecture on pourra définir des constantes représentant les indices : par ex. une constante JE avec `const int JE = 0`*, valant 0, permettant d'écrire* `sujet[JE]` *pour accéder au pronom associé ou bien une constante* `PRES` *pour désigner le présent...*

1. *Écrire une méthode, appelée* `conjugue(String verbe)` *qui retourne un tableau String[][], de paramètre une chaîne verbe, qui construit le tableau des conjugaisons au présent, à l'imparfait, et au futur de verbe passé en paramètre. La méthode identifie si le verbe se termine en ER ou en IR et, après avoir supprimé cette terminaison, rempli le tableau résultat par les conjugaison complètes.*
   Exemple, l'appel de `conjugue("manger")` *renvoie un tableau (qu'on va*

noter `res`*) tel que* `res[0][0]` *contient* `je mange`, `res[0][1]` *contient* `je mangeais`, `res[0][2]` *contient* `je mangerai`.

2. *Écrire une méthode, appelée* `affiche(String[][] tab)` *qui affiche les conjugaisons, temps par temps.*

Exercice 17. *Le but du jeu du Pendu est de retrouver un mot caché en moins de 10 essais, en proposant à chaque fois une lettre. Au départ on affiche la 1ère et la dernière lettre du mot, on met entre les deux le nombre d'étoiles correspondant au nombre de lettres manquantes. Un utilisateur saisira le mot caché, on "cachera" le mot saisi en affichant 50 lignes vides, un autre essaiera de trouver le mot en proposant des lettres à l'ordinateur. Si la lettre se trouve dans le mot caché, on l'affiche autant de fois qu'elle apparaît dans le mot, on laisse les étoiles sur ce qui n'a pas été trouvé. Si la lettre n'est pas dans le mot, on affiche le dessin du pendu, au stade où il est.*

On affiche le dessin à chaque fois que l'utilisateur se trompe de lettre. Lorsque le dessin est complet, le jeu s'arrête et on affiche "Perdu !". Si l'utilisateur saisit une lettre se trouvant à l'intérieur du mot, on remplace toutes les étoiles correspondant à la lettre (par la lettre) et on affiche le mot avec les étoiles correspondant au nouveau stade de recherche.

1. *Écrire la fonction* `affichePendu(int n)` *qui affiche le dessin du pendu correspondant au nombre d'essais* `n` *déjà effectués : on commence par le tiret du bas, puis les 4 traits verticaux, puis les 5 tirets du haut, puis le trait oblique de la potence, puis la corde, puis la tête, puis le corps, puis les deux bras, puis chaque jambe, ce qui fait bien 10 essais en tout.*
   Tester en appelant `affichePendu(3)`, `affichePendu(7)`, *...*
2. *Ajouter une fonction* `initMot(String mot)` *qui retourne une chaîne contenant le mot où tous les caractères sauf le 1er et le dernier ont été remplacés par* `*`.
   Tester en demandant à l'utilisateur de saisir une chaîne et en appelant `initMot()` *ensuite pour traiter la saisie.*
3. *Ajouter une fonction* `remplaceCar(String mot, String ch, char c)` *qui retourne sous la forme d'une chaîne la chaîne correspondant à* `ch` *dans laquelle on met* `c` *partout où* `c` *apparaît dans* `mot`. *Exemple :* `remplaceCar("Taratatta", "T*r*****a", 'a')` *renvoie* `Tara*a**a`
4. *Ajouter une fonction* `joue(String mot)` *qui réalise le jeu du pendu...*

Exercice 18. *La méthode* split() *permet de découper une chaîne selon un délimiteur. Tester les formes suivantes et vérifier que le comportement est correct ou expliquer pourquoi (à l'aide d'internet, de la documentation en ligne, etc)*

1. `System.out.println("02-47-36-70-14".split("-").length);`
2. `System.out.println("02-47.36.70.14".split(".").length);`.

3. `System.out.println("02-47.36.70.14".split(".-").length);`.
4. `System.out.println("02 47 36 70 14".split(" ").length);`.
5. `System.out.println(" 02 47 36 70 14 ".split(" ").length);`.