La classe `String` - Les tableaux à une dimension

Une documentation '(en anglais) de la classe `String` : https://www.w3schools.com/java/java_ref_string.asp

Exercice 1

1. Considérons la chaîne suivante :

   ```
   String str1 = "Is James present  ? James is."
   ```

   (a) Quelle est la valeur retournée par `str1.length()` ?

   (b) Que retourne l'instruction `str2.charAt(12)` ?

2. Considérons la chaîne suivante :

   ```
   String str2 = "Is an object or a reference ?".substring(9,12)
   ```

   Quelle est la longueur de la chaîne `str2` ? Quelle est sa valeur ?

3. Considérons les deux chaînes suivantes :

   ```
   String str3 = "Hi";
   String str4 = "Jane";
   ```

   Quelles instructions doit-on écrire pour obtenir le string `"Hi, mom."`.

4. Considérons la chaîne contenant le prénom et le nom d'une personne

   ```
   String str5 = "James Gosling"
   ```

   Ecrire les instructions qui crée une chaîne de caractères ne contenant que les initiales du prénom et du nom séparées par un espace : `"J G"`.

Exercice 2

Considérons les chaînes suivantes :

```
String str1 = "Programmation";
String str2 = "programmation";
String str3 = new String("Programmation");
```

Que retourne les expressions suivantes ?

1. `str1 == str2`
2. `str1 == str1.intern()`
3. `str1 == str3`
4. `str1 == str3.intern()`
5. `str3 == str3.intern()`

Exercice 3

Écrire un programme qui remplace toutes les voyelles d'une chaîne par le caractère `'_'`.
Indication : Pensez à la méthode `replace` de la classe `String`.

Exercice 4

Le **palindrome** est une figure de style désignant un mot ou une phrase dont l'ordre des lettres reste le même qu'on les lise de gauche à droite ou de droite à gauche (les espaces sont ignorés dans la lecture). Des exemples de palindromes

— des mots palindromes : radar, rever

— une phrase palindrome : Esope reste ici et se repose.

Écrire un programme qui vérifie si un mot est un mot

Exercice 5

L'objectif est de déinir une classe `Phrase` permettant d'enregistrer une phrase.
L'interface de la classe contient les méthodes suivantes :

— `String phrase()` qui retourne un `String` avec la phrase enregistée dans l'objet.

— `int caracteres()` qui retourne le nombre total de caractères des mots de la phrase.

— `int mots()` qui retourne le nombre de mots de la phrase.

— `void ajouter(String mot)` qui ajoute un string à la fin de la phrase

— `int compter(char c)` qui compte le nombre d'occurrences d'un caractère dans la phrase.

```
Phrase ph = new Phrase();
System.out.println(ph.longueur()); // affiche 0
ph.ajouter("Programmation").ajouter("orientée").ajouter("objet");
System.out.println(ph.phrase());// affiche programmation orientée objet
System.out.println(ph.caracteres()); // affiche 26
System.out.println(ph.mots()); // affiche 3
System.out.println(ph.compter('e')); // affiche 3
System.out.println(ph.compter('c')); // affiche 0
```

1. Définir la classe `Phrase`.
2. Écrire une classe de démonstration.

Exercice 6

On écrit dans un programme :

```
int[] t1 = new int[] {27,1,2023}, t2 = t1;
t2[1] = 2;
```

Qu'affiche les instructions suivantes ? Expliquer votre réponse.

1. `System.out.println(t2 != t1);`
2. `System.out.println(t1[1]);`

Exercice 7

On crée un tableau contenant les notes d'un étudiant dans un programme à l'aide des instructions

```
double[] notes = new double[] {8.5, 9, 11, 8.5, 13};
```

1. Ecrire un programme qui calcule la moyenne des notes.
2. Ecrire un programme qui compte le nombre de notes au dessus de 10.
3. Ecrire un programme qui détermine la meilleure note.

Exercice 8

Ajouter à la classe `Phrase` une méthode `boolean estPalindrome()` qui retourne `true` si la phrase est une phrase palindrome et `false` sinon.

```
String ph = new Phrase();
ph.ajouter("Esope").ajouter("reste").ajouter("ici").ajouter("et").ajouter("se").ajouter("repose")
System.out.println(ph.estPalindrome()); // affiche true
```

Indication : Pensez à la méthode `String[] split()` de la classe `String`.

Exercice 9

Définir une classe `Etudiant` modélisant un étudiant du cours de POO. Les propriétés de la classe `Etudiant` sont :

— un `String` permettant d'enregister son identité ; Une fois créé, l'état d'un objet de `Etudiant` ne peut plus être modifié.

— un tableau de `double` qui permet d'enregistrer la liste des notes d'un étudiant (absence = 0) ; les notes de l'étudiant peuvent être modifiées à tout instant ; Le nombre de notes enregistrées dans une objet `Etudiant` est une propriété de l'objet.

L'accès à ces deux propriétés doit protégé et restreint aux seuls objets `Etudiant`. L'interface des objets `Etudiant` contient les méthodes suivantes :

setNote(int i,double note) : modifie la note du i-ème contrôle

getNote(int i) : retourne la note du i-ème contrôle

moyenne() : retourne la moyenne des notes ;

1. Définir la classe `Etudiant`.
2. Écrire une classe de démonstration.

Exercice 10

L'objectif est de définir une classe modélisant une liste de chansons.

1. Définir une classe `Chanson` contient les informations suivants : le titre de la chanson et sa durée en secondes.
2. Définir une classe `ListeChansons` permettant de créer des objets pouvant stocker des objets `Chanson`. La définition de la classe est telle que le programme ci-dessous

```
ListeChansons liste = new ListeChansons(5);

liste.ajouter("In the Flesh ?",199);
liste.ajouter("The Thin Ice",149);
liste.ajouter("Another Brick in the Wall, Part I",190);
liste.ajouter("The Happiest Days of Our Lives",111);
liste.ajouter("Another Brick in the Wall, Part II",239);
liste.ajouter("Mother",336);

liste.afficher();
```

provoque l'affichage ci-dessous :

```
Impossible d'ajouter Mother : liste pleine !
1. In the Flesh ? 3:19
2. The Thin Ice 2:29
3. Another Brick in the Wall, Part I 3:10
4. The Happiest Days of Our Lives 1:51
5. Another Brick in the Wall, Part II 3:59
5 chansons, durée : 14:48
```

Exercice 11

L'objectif est de définir deux classes `Vol` et `Passager` modélisant l'enregistrement de passagers sur un vol d'une compagnie aérienne. Les seules informations concernant un passager sont son `identite` (`String`). Le programme ci-dessous

```
Vol vol = new Vol(10); // vol avec 10 places proposées

vol.enregistrer(3,new Passager("Client 1"));
vol.enregistrer(6,new Passager("Client 2"));

vol.afficher();
```

Si le vol est plein, la méthode `enregistrer` affiche un message dans la console indiquant que le vol est plein.

```
2 places réservées sur 10
Place 3 : Client 1
Place 6 : Client 2
```

1. Définir la classe `Passager`.
2. Définir la classe `Vol`.
3. Écrire une classe de démonstration.

Exercice 12

L'objectif est d'écrire deux classes `De` et `Lancer` simulant le lancer de plusieurs dés cubiques (les six faces sont numérotées de 1 à 6).

```
// on lance trois dés
Lancer lancer = new Lancer(3);
System.out.println("Les résultats des trois lancers sont :");
for(De de : lancer.resultats()) de.afficher();
```

Le programme ci-dessus produit l'affichage :

```
Les résultats des trois lancers sont :
 -----
| O O |
|     |
| O O |
 -----
 -----
|   O |
|  O  |
| O   |
 -----
 -----
| O O |
|  O  |
| O O |
 -----
```

Pour simuler le lancer d'un dé, on pourra utiliser les instructions suivante :

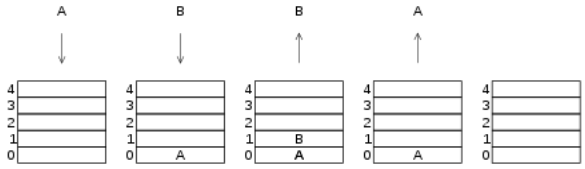
```
import java.util.Random;

Random random = new Random();
for(int i = 0;i < 5;i++) System.out.println(random.nextInt(5) + 1);
```

La méthode `nextInt(int n)` génère un nombre aléatoire entre 0 et $n-1$.

1. Définir la classe `De`.
2. Définir la classe `Lancer`.
3. Écrire une classe de démonstration.

Exercice 13

Définir une classe `Pile` modélisant une pile d'entiers. Une pile est une structure de donnée fondée sur le principe "Premier arrivé, dernier sorti" ce qui veut dire que le premier élément ajouté à la pile sera le dernier retiré.



Une pile est caractérisée par sa capacité, c'est-à-dire le nombre maximal d'entiers que peut contenir la pile. Les opérations possibles sur une pile sont :

— empiler un entier au sommet de la pile si la pile n'est pas pleine

— dépiler l'entier au sommet si la pile n'est pas vide

Dans le cas où il n'est pas possible d'empiler ou de dépiler un entier de la pile, un message est affiché dans la console indiquant que l'opération n'est pas possible. Il doit aussi être possible de savoir si la pile est vide ou si la pile est pleine.

1. Définir une classe `Pile` ayant les caratéristiques et les opérations décrites ci-dessus et écrire une classe de démonstration.
2. Écrire une classe de démonstration.