

Type énuméré, generics, varargs, boxing

Exercice 1 - Avec style

On souhaite pouvoir écrire des textes de façon marrante sur la console. On définit pour cela la méthode :

```
String stylify(String text,int flags)
```

L'entier flags indique la façon dont le texte doit s'afficher. Avec le texte toto

- 1 STAR affiche t*o*t*o
- 2 REVERSE affiche otot
- 3 REVERSE|STAR affiche o*t*o*t
- 1 Déclarer STAR et REVERSE comme des constantes entières.
- 2 Ecrire la méthode stylify en sachant quelle doit pouvoir accepter d'appliquer plusieurs style en même temps.

Exemple d'utilisation :

```
stylify("toto",REVERSE|STAR);
```

- 3 Quel est l'intérêt de remplacer les constantes par une énumération ?
Transformer les constantes en énumération et modifier la méthode stylify en conséquence.
Note : Vous pouvez regarder la documentation de la classe EnumSet.
- 4 Ajouter une méthode applyStyle dans l'énumération.
Quel est l'intérêt de ce design ?
Changer le code de stylify en conséquence.
- 5 Ajouter un flag BANG qui affiche le texte toto comme ceci t!o!t!o. Partagez le code au maximum !
En cas de BANG/STAR, le STAR devra être traité d'abord (le résultat est alors t!*!o!*!t!*!o).
- 6 Que doit-on modifier pour pouvoir choisir si l'on veut faire un STAR avant ou après un BANG dans le cas d'un STAR/BANG ?
L'astuce consiste à passer en argument de la méthode stylify un ensemble qui conserve l'ordre d'insertion de ses éléments.

Exercice 2 - Minimum

On souhaite pouvoir écrire le code suivant :

```
System.out.println(min(3,12)); // affiche 3  
System.out.println(min(4,2,3)); // affiche 2
```

- 1 Quel est le profil de la méthode min ?
- 2 Comment faire pour que l'appel min() sans argument soit invalide ?
- 3 Écrire le code correspondant.

Exercice 3 - Pile efface

Toutes les classes écrites lors de cette exercice doivent se trouver dans le paquetage `fr.umlv.util.stack`.

1 Écrire la classe `fr.umlv.util.stack.FixedStack` possédant les méthodes :

`boolean isEmpty()` qui indique si la pile est vide.

`push(Object o)` qui empile une valeur

`Object pop()` qui dépile une valeur

On considéra la pile comme possédant une capacité maximale fixée lors de la création de l'objet `FixedStack`.

Rappeler ce qu'est une fuite de mémoire (memory leak) dans le cas général puis en Java.

Vérifier que votre implantation de `pop` ne crée pas de fuite de mémoire.