

Generics, varargs, boxing

Exercice 1 - 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 2 - Pile efface

Toutes les classes écrites lors de cet 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 considère 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 `pop` ne crée pas de fuite de mémoire.

Exercice 3 - Pile suite

- 1 Écrire une autre classe de pile, `fr.umlv.util.stack.LinkedStack`, possédant les méthodes `isEmpty`, `push` et `pop` et utilisant une `java.util.LinkedList` comme structure de stockage sous-jacente.
- 2 Indiquer dans quels cas, il est mieux d'utiliser une `FixedStack` ou une `LinkedStack`. Comment doit-on faire s'il l'on veut pouvoir dynamiquement choisir entre les deux implantations pour exécuter un même code ? Modifier votre code et écrire un `main` de test.
- 3 Générer l'ensemble des classes créées.

Exercice 4 - Minimum (plus)

On souhaiterait écrire une nouvelle méthode `min` qui puisse être aussi utilisée avec des chaînes de caractères ou tout autre objet comparable.

```
System.out.println(min(3,12)); // affiche 3
System.out.println(min("trois","douze")); // affiche douze
```

- 1 Quelle est la signature de la méthode min ?
- 2 Écrire le code correspondant

