TP – La classe Pile
Pile d'entiers – Pile générique

Le but de l'exercice est de développer une classe pour représenter une pile d'entiers, donc une structure de données. La **capacité** de la pile, c'est-à-dire le nombre maximum d'entiers qu'elle pourra contenir, sera fixée à la création de la pile. Par défaut, la capacité sera égale à 10. Remarquons qu'il ne faut pas confondre la notion de capacité avec celle de taille. La **taille** d'une pile est le nombre d'éléments qu'elle contient à un instant donnée, alors que la capacité est le nombre maximum d'éléments qu'elle pourra contenir.

Les opérations à prévoir dans la classe ***Pile*** sont les suivantes :

- un **constructeur par défaut.** La pile ainsi créée sera vide.
- un **constructeur avec en argument la capacité de la pile** à créer. Bien sûr, au moment de sa création, une pile sera vide.
- ***estVide*** pour déterminer si la pile courante est **vide**
- ***estPleine*** pour déterminer si la pile courante est **pleine**
- ***empiler*** pour empiler l'entier argument
- ***sommet*** pour renvoyer la valeur du ***sommet*** de la pile
- ***dépiler*** pour dépiler l'élément sommet de la pile. Cette opération ne renverra pas de résultat.
- ***toString*** qui renverra sous la forme d'une chaîne de caractères, **le contenu de la pile**. Par exemple, si la pile contient 3, 6 et 9, 9 étant le sommet, la méthode renverra la chaîne
  "[sommet = 9 | 6 | 3 |]".
- ***memeCapacite***, une opération qui détermine si **deux piles ont la même capacité**.
- ***equals***, une opération qui détermine si **2 piles sont identiques**, c'est-à-dire ont la même capacité et le même contenu. Dans la mesure du possible, on s'efforcera d'écrire cette opération de manière à ce qu'elle puisse être redéfinie dans une future classe dérivée de la classe *Pile*.

Les situations anormales seront gérées en levant et en propageant l'une des exceptions prédéfinies ***IllegalArgumentException*** ou ***IllegalStateException.***

Partie 1 - Questions

1) Ecrire la classe *Pile* conformément aux spécifications ci-dessus.

2) Ecrire un premier programme de test de la classe *Pile*. Il n'est pas demandé de tester dans ce programme toutes les opérations de la classe ***Pile***, ni toutes les situations possibles. Les instructions pourront être écrites dans une fonction ***main***. Les exceptions seront gérées localement dans la fonction

main, par l'affichage d'un message général indiquant qu'une erreur s'est produite. On réalisera les étapes suivantes (vous pouvez les ajouter progressivement) :

1) créer une pile de capacité 5
2) vérifier que la pile ainsi créée est vide
3) empiler 3 entiers quelconques dans cette pile
4) vérifier que la pile n'est pas vide
5) afficher à l'écran la valeur située au sommet de la pile
6) afficher à l'écran le contenu de la pile
7) dépiler l'élément sommet
8) afficher à nouveau le contenu de la pile
9) empiler 3 entiers quelconques dans la pile
10) vérifier qu'elle est pleine
11) empiler un élément dans la pile (cette opération doit normalement provoquer une exception).

3) Le programme précédent ne permet pas de tester les erreurs liées à une capacité invalide, ou une pile vide. Le but de cette question est de le compléter pour provoquer ces deux erreurs et vérifier que l'exception adéquate a bien été propagée vers le programme appelant. Ajouter des instructions à la suite des précédentes pour :

1) créer une pile avec une capacité invalide
2) dépiler alors que la pile est vide
3) utiliser l'opération ***sommet*** sur une pile vide

Pour chacune de ces situations, on affichera un message expliquant la cause de l'erreur.

4) Une classe de test plus complète est disponible sur Moodle : ***TestPileEntierComplet***. Récupérer cette classe et exécuter le programme de test pour vérifier votre code.

Partie 2

La classe développée dans la partie précédente permet de gérer une pile d'entiers. On souhaiterait, dans différentes applications, utiliser des piles d'entiers, mais aussi des piles de chaînes de caractères, ou bien des piles contenant des matrices ou encore des numéros de téléphone, instances de la classe *Telephone*. Pour éviter de multiplier les implémentations de la classe *Pile*, il faut bien sûr, développer une classe ***Pile Générique***. Celle-ci pourra être instanciée avec différents types, tels que entier, *String*, *Matrice* ou *Telephone*, par exemple.

Les fonctionnalités de cette classe générique seront identiques à celles de la classe *Pile* d'entiers. On retrouvera donc les mêmes opérations.

Indications

1) Il faudra déclarer, dans la classe, un attribut :

```
/**  Tableau contenant les éléments de la pile */
private T[] element;
```

2) Dans les constructeurs, il ne sera pas possible de créer un tableau dont les éléments seront de type *T*, *T* étant le paramètre de généricité. En effet, vous constaterez que l'instruction :

```
element =  new T[capacite];
```

provoque une erreur à la compilation, le type *T* étant inconnu. Il faut donc allouer un tableau d'éléments de type *Object* et convertir celui-ci en un tableau d'éléments de type *T* :

```
element =  (T[]) new Object[capacite];
```

3) Il faudra penser à adapter les méthodes *toString*, et *equals*. En effet, les éléments de la pile n'étant pas des entiers, on ne pourra plus utiliser l'opérateur `==` pour les comparer.

Questions

1) Ecrire la classe *Pile Générique* conformément aux spécifications ci-dessus.

2) Tester la classe ainsi obtenue.
On écrira 3 programmes de test avec 3 types d'éléments différents.

a) Dans un premier temps, vous pourrez adapter le programme de test qui avait été donné avec le TP sur la classe pile d'entiers (récupéré sur Moodle). Vous obtiendrez ainsi un programme de test pour une **instanciation sur des entiers.**

b) Sur le même modèle que ce programme, vous pourrez développer un programme pour tester la classe afin d'obtenir **une pile d'éléments de type *String***.

c) Pour le troisième programme de test, vous pourrez récupérer sur Moodle une classe *Telephone* (attention, il ne s'agit pas de la classe *Telephone* fournie lors du premier TP, mais d'une nouvelle version de celle-ci). Votre programme de test devra permettre de **tester une pile contenant des numéros de téléphone**, instances de la classe *Telephone***.**