Date : Novembre 2020
Enseignant : Marie-Laure Nivet
Diplôme : Licence 3 SFA Informatique
Nom de l'UE : Programmation orientée objet, langage Java
Type de document : Énoncé de TD sur les interfaces

Interfaces

Exercice 1 : Interface et implémentation (version sans exception)

Question 1 : Ecrire le code d'une interface `AStack` contenant les méthodes classiques des Piles :

- `isEmpty` qui teste si la pile est vide ;
- `push` qui ajoute un objet au sommet de la pile
- `peek` qui retourne l'objet du sommet de la pile, sans toutes fois l'enlever de la pile, si la pile est vide, `peek` retourne `null`
- `pop` qui retourne et retire l'objet qui est au sommet de la pile, si la pile est vide, `pop` retourne `null`.

Question 2 : On vous demande de définir une classe `ConcreteStack` qui implémente cette interface. Vous pouvez pour stocker les éléments de la pile, utiliser soit un tableau, soit un tableau dynamique ou une liste. Enumérez les différences (points positifs et négatifs) qui découleront de ce choix ?
Question 3 : Ecrivez le code de votre classe `ConcreteStackArray` basée sur un stockage dans un tableau statique.
Question 4 : Testez cette classe dans un main en créant une nouvelle pile, empilez quelques éléments de votre choix, dépilez… (bref écrivez au moins une fois un appel à toutes les méthodes que vous avez écrites précédemment).
Question 5 : Ecrivez le code d'une classe `ConcreteStackList` basée sur un stockage dans une ArrayList.
Question 6 : Testez cette nouvelle classe dans votre main en créant une nouvelle pile, empilez quelques éléments de votre choix, dépilez… (bref écrivez au moins une fois un appel à toutes les méthodes que vous avez écrites précédemment).

Exercice 2 : Clonage

Rappel concernant le clonage :

- Cloneable est une interface drapeau appartenant au package java.lang
- La classe Object racine de l'arbre d'héritage des classes Java (package java.lang) possède une méthode d'instance nommée clone, dont l'accès est limité aux classes filles ainsi qu'aux classes du même package. Cette méthode ne prend aucun paramètre et retourne un Object. Elle est par ailleurs susceptible de propager une exception de type CloneNotSuppertedException dans le cas ou le clonage de l'objet est interdit.

Soit les codes de classes suivantes :

```
public class Mere{
      private int i;
      private UneClasse unObjet;    //Clonable
```

```
	public Mere(int unI, UneClasse unObj){
		i = unI; unObjet = unObj;
	}
}

public class Fille extends Mere{
	private Object unObjetSensible;
	public Fille(int unI, UneClasse unObj, Object unObjSens){
		super(unI,unObj);
		unObjetSensible = unObjSens;
	}
}
```

On vous demande de faire les modifications et ajouts nécessaires :

1. à la classe `Mere` pour que les objets issus de cette classe puissent être clonés en profondeur.
2. à la classe `Fille` pour que les objets issus de cette classe ne puissent pas être clonés.