TP3

Classes génériques & Objets immuables

Partie I : Classes génériques

Exercice 1 :

1. Écrire une classe générique ***MyFirstGeneric*** définie par trois variables d'instances **X**, **Y** et **Z** d'un même type.
2. Ajouter un constructeur permettant d'initialiser les 3 variables ainsi que des getters et des setters.
3. Ajouter à cette classe une méthode **display ()** affichant la valeur des éléments du ***MyFirstGeneric***.
4. Écrire un programme de test utilisant cette classe générique pour instancier quelques objets de différents types et utiliser les méthodes implémentées.
5. Ecrire une deuxième classe ***MySecondGeneric*** qui hérite de la classe ***MyFirstGeneric*** et qui est définie par un attribut supplémentaire **W** qui doit être numérique.
6. Ajouter un constructeur et la méthode **display()** à la classe ***MySecondGeneric.***
7. Tester la classe ***MySecondGeneric***.

Exercice 2 :

1. Écrire une classe générique ***GenericStack*** qui permet d'implémenter une pile et qui est définie par deux attributs :
    a. un ArrayList pour stocker les valeurs de la pile.
    b. un entier « top » indiquant l'indice du sommet. .
2. Implémenter la méthode **push** qui permet d'empiler un élément de type T ;
3. Implémenter la méthode **pop** qui permet de dépiler un élément de type T ;
4. Implémenter la méthode **sizeStack** qui retourne la taille de la pile.
5. Implémenter une méthode statique générique **addTo** qui permet d'ajouter un élément de type T à un objet de la classe ***GenericStack***.
6. Tester.

Partie II : Objets immuables

Exercice 1 :

1. Créer une classe immuable ***EtudiantImmuable*** avec :
    a. Des attributs : *nom*, *dateNaiss*, *listeModuleNote* (Association d'une note à chaque module) ;
    b. Un Constructeur ;
    c. Des accesseurs / mutateurs.
2. Tester la fonction.

Exercice 2 :

Soit la classe ***VisitCounter*** suivante :

```
public class VisitCounter {
	private int counter;
	public VisitCounter (int c)
	{
		this.counter = c;
	}

	public int getCounter() {
		return counter;
	}

	public void setCounter() {
		this.counter = this.counter + 1 ;
	}
}
```

Etant donné qu'elle est utilisée par plusieurs utilisateurs (objets), nous souhaitons la protéger en ne créant qu'une seule instance afin de conserver l'information stockée dans l'attribut « counter ».

1. Modifier cette classe afin de garantir le respect de cette contrainte.

Exercice 3 : Pour les plus motivés …

1. Ecrire une classe ***JourRDV*** contenant trois entiers : *jj*, *mm*, *aaaa* (pour jour, mois et année respectivement).
2. Ecrire une classe ***RDV*** contenant son horaire (exprimé par un entier représentant le nombre de minutes, par exemple 540min = 9h00) et une description de type String.
3. Créer pour une secrétaire une classe immuable ***Agenda*** qui stocke les informations sur les RDV de ses responsables (String) sous la forme : **< nom du responsable, < jour du rdv, liste ordonnée des RDVs du Jour>>**. Un responsable a donc, pour un jour donné une liste de rendez-vous.
4. Implémenter les fonctionnalités suivantes :
    1. Afficher le contenu de l'agenda.
    2. Ajouter un rdv pour un responsable.
    3. Annuler (supprimer) pour un responsable (un rdv donné, tous ses RDVs du jour j, tous ses RDVs).
    4. Récupérer et afficher pour un responsable (tous ses RDVs du jour j, tous ses RDV, son dernier rdv du jour j).
5. Rajouter une méthode (getAgenda) qui retourne l'attribut agenda de type Map.
6. Tester.