Correction exercice chapitre 4

Le but de cet exercice est de :

- Vous apprendre à lire l'énoncé d'un problème, de disséquer les informations pertinentes et de concevoir les classes Java qui y découle.
- Revisiter bon nombre de concepts que nous avons étudiés dans le cours : notion d'encapsulation, les attributs, les méthodes, les constructeurs, la méthode main(), l'instanciation, le jeu avec les références d'objets, utiliser Eclipse, etc.

Explication et Solution :

La problématique tourne autour du calcul de l'aire d'un trapèze. Sachant que pour calculer cette aire on a besoin des 3 données : sa grande base, sa petite base et sa hauteur. Ces 3 données constituent donc les attributs de la classe Trapeze que vous allez créer.

Lors de la création de la classe Trapeze, l'énoncé demande de faire en sorte qu'elle respecte le principe de l'encapsulation. Cela signifie que vous allez mettre le modificateur private à tous ses attributs et ajouter des accesseurs (getters/setters) publiques à tous ces attributs.

Dans la méthode calculAire() demandée, on vous laisse le soins de chercher par vous même la formule de calcul de l'aire d'un trapèze et de l'implémenter en Java en respectant le principe d'une méthode qui retourne un résultat appris dans le cours.

Dans la classe principale et notamment dans la méthode main(), vous instanciez la classe Trapeze (objTrap) et invoquez sa méthode calculAire() pour vous rendre compte, en vous servant de l'instruction System.out.println(), qu'il affiche le résultat : **28.8**

Enfin, dans la suite on vous demande de créer une nouvelle instance objTrap2 du trapèze avec le constructeur par défaut. Ce qui signifie en fait qu'il s'agit d'un trapèze vide sans valeurs réels pour ses attributs. En l'état, on ne peut pas calculer son aire, car ses attributs n'ont pas été initialisés par le constructeur.

Pourtant, vous remarquerez qu'après l'opération d'affectation de objTrap dans objTrap2, lorsque vous invoquez sa méthode calculAire() et affichez son résultat, il se fait sans erreur et affiche le même résultat que objTrap : **28.8**

Que constatez vous ?
Réponse : simplement que le résultat des deux méthodes est le même. Ceci parce qu'en affectant objTrap dans objTrap2, la référence de ce dernier est désormais le même que objTrap. Les deux variables en d'autres termes pointent sur la même adresse mémoire. objTrap2 s'approprie donc toutes les données initialement positionnées par objTrap. D'où sans avoir explicitement initialisé des valeurs pour objTra2, on a pu l'utiliser sans problème

NB : la correction de l'exercice se trouve dans le dossier **chapitre4** ci-joint et que nous recopions ici :

```
package geometrie;

public class Trapeze {

	private double grandeBase;
	private double petiteBase;
	private double hauteur;

	public Trapeze() {

	}

	public Trapeze(double grandeBase, double petitBase, double hauteur) {
		this.grandeBase = grandeBase;
		this.petiteBase = petitBase;
		this.hauteur = hauteur;
	}

	public double getGrandeBase() {
		return grandeBase;
	}

	public void setGrandeBase(double grandeBase) {
		this.grandeBase = grandeBase;
	}

	public double getPetiteBase() {
		return petiteBase;
	}

	public void setPetiteBase(double petiteBase) {
		this.petiteBase = petiteBase;
	}

	public double getHauteur() {
		return hauteur;
	}

	public void setHauteur(double hauteur) {
		this.hauteur = hauteur;
	}

	public double calculAire() {
		double aire = ((this.grandeBase + this.petiteBase)*this.hauteur)/2;
		return aire;
	}

}
```

```
package geometrie;

public class ClassePrincipale {

	public static void main(String[] args) {
		Trapeze objTrap = new Trapeze(5, 3, 7.2);
		double aire = objTrap.calculAire();
		System.out.println(aire);

		Trapeze objTrap2 = new Trapeze();
		objTrap2 = objTrap;
		double aire2 = objTrap2.calculAire();
		System.out.println(aire2);
	}

}
```