Correction Deuxième Séance d'Exercice

Ex1

- Ecrire un code qui appelle la construction de trois objets de type Facture.

```
public class Facture{

int nombreUnitesVendues;
static int nombreFacturesCrees;

Facture()
        {
        nombreUnitesVendues++;
        nombreFacturesCrees++;
        }

public static void main(String[] arfs)
        {
        Facture f1=new Facture();
        Facture f2=new Facture();
        Facture f3=new Facture();
        }
}
```

- Lors de l'exécution du code précédent, combien d'entiers sont stockés en mémoire ? Quelles sont les valeurs de ces trois entiers après la construction des trois objets de type Facture ?

4 entiers sont stockés en mémoire ici :

f1.nombreUnitesVendues
f2.nombreUnitesVendues
f3.nombreUnitesVendues
Facture.nombreFacturesCrees

Pour chaque variable nombreUnitesVendues : elle est d'abord à 0 au début du constructeur, puis elle augmente de 1.

Donc, à la fin de l'exécution :

f1.nombreUnitesVendues : 1
f2.nombreUnitesVendues : 1
f3.nombreUnitesVendues : 1

Pour la variable Facture.nombreFactureCrees, elle est initialisée à 0 au début de l'exécution. Elle est augmentée de 1 à chaque appel du constructeur Facture. Donc à la fin :

Facture.nombreFacturesCrees : 3

Soit le code :

```
public class Facture{

int nombreUnitesVendues;
static int nombreFacturesCrees;

Facture()
        {
        nombreUnitesVendues++;
        nombreFacturesCrees++;
        }

public static void test()
        {
        System.out.println(nombreFacturesCrees);
        System.out.println(nombreUnitesVendues);
        }

}
```

- Ce code se compile t-il ?

Non, l'erreur pointe l'instruction :

System.out.println(nombreUnitesVendues);

qui est dans test(). Cette fonction test() est static, elle ne correspond à aucun objet, elle ne peut donc pas manipuler d'attribut d'objet, seulement des attributs static.

- Pouvez vous envisager de créer une classe telle que toutes les occurrences créées pour cette classe soient collectionnées ?

Une solution consiste à créer un attribut de classe qui est une collection d'objets :

```
import java.util.LinkedList;
public class A{
        public static LinkedList<A> listesEltsA=new LinkedList<A>();

        public A()
        {
        listesEltsA.add(this);
        }
}
```

Ex2

Soit la fonction suivante :

```
public double ecartType(double[] serie)
        {
        double moyenne=0;
```

```
        for(int i=0;i<serie.length;i++)
        moyenne=moyenne+serie[i];
moyenne=moyenne/serie.length;
double ecartType=0;
        for(int j=0;j<serie.length;j++)
        {
        double temp=Math.pow(serie[j]-moyenne,2);
        ecartType+=temp;
        }
ecartType=Math.pow(ecartType/serie.length,0.5);
System.out.println(ecartType+" "+temp);
return ecartType;
}
```

- Quelle(s) erreur(s) renvoie cette fonction à la compilation ? Pourquoi ?

Toute variable est locale au bloc d'instructions dans laquelle elle est déclarée.

Un bloc d'instructions est le code contenu entre deux accolades. Les instructions d'une boucle, d'une fonction, d'une instruction conditionnelle constituent un bloc d'instructions etc.

Ici avec **`double temp=Math.pow(serie[j]-moyenne,2);`** , la variable temp est déclarée locale à la boucle for : les ressources liées sont libérées après l'exécution de la boucle, d'où l'instruction

`System.out.println(ecartType+" "+temp);`

ne peut pas être exécutée hors de la boucle où la variable temp existe.