

Examen Java – session 1

1. Soit la classe **Point** qui stocke :

- ses coordonnées (x, y, z),
- son nom.

Ecrire cette classe avec ses attributs ainsi que 3 constructeurs : le premier prenant en argument seulement un **String** représentant le nom du point ; le second prenant en argument un **String** pour le nom du point et trois **double** représentant respectivement les coordonnées en x, y et z de ce point ; le troisième prenant un objet de type **Point** en argument. (1.5 pts)

2. Soit la classe **Vecteur** qui stocke :

- ses coordonnées (u, v, w),
- son nom.

Ecrire cette classe avec ses attributs ainsi que 2 constructeurs : le premier prenant en argument seulement un **String** représentant le nom du vecteur ; le second prenant en argument deux objets de type **Point** qui représentent respectivement le point de départ et le point d'arrivée du vecteur. Pour ce second constructeur, le nom du vecteur sera obtenu par concaténation des noms du point de départ et du point d'arrivée. (1 pt)

Ecrire la méthode **norme** qui retourne un **double** correspondant à la norme du vecteur. (0.5 pt)

Rappel : Soit A et B deux points de coordonnées (x_A, y_A, z_A) et (x_B, y_B, z_B) . La norme du vecteur AB est obtenu par :

$$\|\overrightarrow{AB}\| = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}.$$

Ecrire la méthode **produitVectoriel** qui prend en argument un objet de type **Vecteur** et retourne un objet de type **Vecteur** qui est le résultat du calcul du produit vectoriel entre le vecteur courant et le vecteur passé en argument. (0.5 pt)

Rappel : Soit u et v respectivement de coordonnées (u_1, u_2, u_3) et (v_1, v_2, v_3) . Le produit vectoriel est obtenu de la manière suivante :

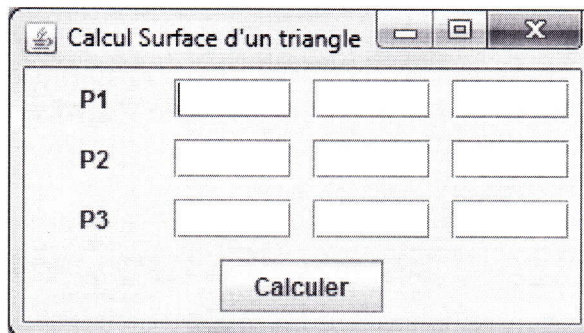
$$u \wedge v = \begin{pmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{pmatrix} = \begin{pmatrix} v_1 u_2 - u_1 v_2 \\ u_1 u_2 - u_1 u_2 \\ u_1 v_2 - v_1 u_2 \end{pmatrix}$$

3. Ecrire une classe **Triangle** qui stocke les 3 sommets a, b et c de ce triangle. Cette classe aura un seul constructeur qui prendra en argument trois objets de type **Point** représentant les sommets du triangle. (1 pt)

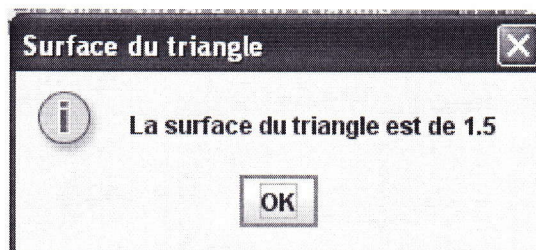
Ecrire la méthode **surface** qui retourne un double correspondant à la surface du triangle. La surface sera arrondie à deux décimales après la virgule. (0.5 pt)

$$S = \frac{1}{2} \|\overrightarrow{AB} \wedge \overrightarrow{AC}\|$$

4. L'interface graphique ci-dessous permet à un utilisateur de calculer la surface d'un triangle après avoir saisi les coordonnées des 3 sommets. Ecrire la classe **InterfaceGraphique** qui permet d'afficher l'interface ci-dessous. (3 pts)



L'appui sur le bouton « Calculer » provoque le calcul de la surface du triangle. Le résultat sera affiché dans une boîte de dialogue sous la forme suivante :



Ecrire le code permettant de gérer l'appui sur le bouton « Calculer » et par conséquent l'affichage de la surface (2 pts).

ANNEXE

Classe Math

Method Summary

static double	<u>floor</u> (double a) Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.
static double	<u>sqrt</u> (double a) Returns the correctly rounded positive square root of a double value.

Examen java

Partie heritage-interface-structure de données

10pts

Exercice 1

Indiquer ce qu'affiche l'exécution du programme suivant.

```
class A
{
    int num;
    String s = "";
    A(int x)
    {
        num = x;
        System.out.println("A");
    }
    void m()
    {
        System.out.println(s + " : " + num);
        f(num);
        num++;
    }
    void f(int n)
    {
        System.out.println("A.f -> " + n);
        num = n + 15;
        s = s + n;
    }
}
```

```
class B extends A
{
}
```

```
B(int y)
{
    super(y + 3);
    System.out.println("B");
}
void m(int x)
{
    System.out.println("B.m -> " + x);
    m();
    m();
}
}

class C extends B
{
    C(int v)
    {
        super(v);
        System.out.println("C");
    }
    void f(int x)
    {
        super.f(-1);
        System.out.println("C.f -> " + x + " ; num = " + num);
        num = num + x;
    }
}

class Test
{
    public static void main(String[] args)
    {
        B b = new B(13);
        b.m(11);
        C c = new C(21);
        c.m(6);
    }
}
```

Exercice 2

On désire réaliser un programme Java pour simuler et / ou gérer un stock de produits. Un produit a les caractéristiques suivantes :

- Il peut être fragile
- Il peut afficher son nom (ici le type de produit)

On dispose de plusieurs types de produit :

- les aliments qui sont fragiles
- les appareils qui ne pas fragiles
- les boîtes qui contiennent un ensemble de produits. Une boîte est fragile si elle contient au moins un produit fragile. Son nom est "boîte" suivi du nom des produits qu'elle contient

Un stock est un ensemble de produit.

Questions :

1. Ecrire l'interface Produit respectant ses caractéristiques
2. Ecrire les classes Aliment et Appareils
3. Ecrire la classe Boîte. Elle recevra dans son constructeur une liste de produit (utiliser l'interface List) qu'elle devra recopier
4. Ecrire la classe Stock en utilisant une des structures de données disponibles dans java.
A l'initialisation le stock est vide. Faire une méthode ajouter et une méthode affiche qui affiche tout les produits du stock
5. Faire une méthode listerFragile qui renvoie une liste de tout les produit fragile du stock