Département Réseaux et Télécommunications
2eme Année R&T Matière : Informatique 4 (POO)

Nom :
Prénom :
Groupe :

Corrigé type examen S4

Exercice 1 (5 pts)

1. Créer la classe Article caractérisée par les attributs : ***Référence***, ***Désignation***, ***PrixHT***, ***TauxTVA***.
   Ces attributs doivent seulement être accessibles par le biais des accesseurs (get / set) en lecture/écriture mis en œuvre par les propriétés.
2. Ajouter les constructeurs suivants :
- Un constructeur par défaut
- Un constructeur initialisant tous les attributs.
3. Implémentez la méthode **CalculerPrixTTC()** ;
   Cette méthode doit calculer le prix TTC d'un article qui équivaut à : *PrixHT + (PrixHT*TauxTVA/100* et retournera la valeur calculée.
4. Ajouter la méthode **AfficherArticle()** qui affiche les informations de l'article.
5. Créer un programme de test où il faut créer des objets (en utilisant les différents constructeurs) et leur calculer le prix TTC.

```
#include <iostream>
using namespace std;
class Article{
 private:
    int ref;          ……..0.25 pts
    string designation;          ……..0.25 pts
    float prixHT;          ……..0.25 pts
    float tauxTVA;          ……..0.25 pts
 public:
    Article(){    ……..0.5 pts
        ref=0; designation="";
        prixHT=0; tauxTVA=0;
    }
    Article(int r, string d,float p,float t ){          ……..0.25 pts
        ref=r; designation=d;
        prixHT=p; tauxTVA=t;
    }
    int getRef(){ return ref;}                ……..0.25 pts
    string getDesignation(){ return designation;}          ……..0.25 pts
    float getPrixHT(){ return prixHT;}          ……..0.25 pts
    float getTauxTVA(){ return tauxTVA;}    ……..0.25 pts
    void setRef(int r){ ref=r;}  ……..0.25 pts
    void setDesigantion(string d){ designation=d;}    ……..0.25 pts
    void setPrixHT(float p){ prixHT=p;}                ……..0.25 pts
    void setTauxTVA(float t){ tauxTVA=t;}              ……..0.25 pts
    double calculerPrixTTC(){                          ……..0.25 pts
        return prixHT + (prixHT*tauxTVA/100);
    }
    void AfficherArticle(){                            ……..0.25 pts
        cout<< ref<<" "<<designation<<"  "<<prixHT<<"  "<<tauxTVA<<endl;
    }
};
int main()
{
   Article a(111, " sucre", 100, 0.5);              ……..0.5 pts
    a.AfficherArticle();                            ……..0.5 pts
   return 0;
}
```

Exercice 2 (6 pts)

1) Créez une classe Compte modélisant un compte en banque. La classe possède deux attributs **somme** et **taux** correspondant à la somme placée sur le compte et au taux d'intérêt. Ces deux attributs doivent être affectés par le constructeur.

2) Ajoutez une méthode **affiche()** qui affiche la somme et le taux comme ci-dessous :

Compte | somme : 2000DA | taux : 2%

3) Ajoutez une méthode **depot(x)** qui ajoute x à la somme sur le compte.

4) Ajoutez une méthode **retrait(x)** qui enlève x à la somme sur le compte. Elle pourra renvoyer un erreur si la somme sur le compte devient négative et annuler alors l'opération.

5) Ajoutez une méthode **interets()** qui calcule les intérêts perçus en un an et les ajoute à la somme placée. Pour rappel les intérêts se calculent avec la formule interets = taux x somme / 100.

```
class Compte {
private:
   double somme; ……..1 pts
   float taux;
public:
   Compte (double s, float t) { ……..1 pts
      somme=s; taux = t;
   }
void afficher(){……..1 pts
      cout<<"Compte| somme : "<<somme<< "  DA | taux :  "<<taux<<" %"<<endl;
   }

   void depot(double  s) { ……..1 pts
      somme =somme+s;
   }
   void retrait(double  s) {……..1 pts
      if (somme>=s) somme=somme-s;
      else
          cout<<"compte insuffisant"<<endl;
   }

   double interrets() {return ( taux * somme)/100;} ……..1 pts
};
```

Exercice 2 (9 pts)

Voici le diagramme de la classe Point Cette classe permet de représenter et de manipuler un point du plan. Le but sera de manipuler des formes géométriques basées sur cette classe.

Donner le code de cette classe par découpage en module c'est-*à-dire* son code de définition (**point.h**) et son implémentation (**point.cpp**).

- la méthode « **distance** » de la classe **Point** calcule la distance entre deux points : c'est-à-dire entre le point courant et un point passé en paramètre à la méthode.

Point
x :double et y: double +**Point**(x: double, y: double) +**distance**(pt: Point): double +**add**(pt: Point): Point +**afficher**(): void +**getX**(): double +**getY**(): double +**setX**(x: double): void +**setY**(y: double): void

- La méthode « **add** » ajoute l'abscisse et l'ordonnée du point courant avec celles du point passé en paramètre

Point.h

```
#include <iostream>
#include <math.h>
using namespace std;
class Point{
    private:
        double x;        ……..0.5 pts
        double y;        ……..0.5 pts
    public:
        Point(double x1, double y1);        ……..0.5 pts
        void setX(double x1);               ……..0.5 pts
        void setY(double y1);               ……..0.5 pts
        double getX();                      ……..0.5 pts
        double getY();                      ……..0.5 pts
        double distance(Point pt);          ……..0.5 pts
        void add(Point pt);                 ……..0.5 pts
        void afficher();                    ……..0.5 pts
```

Point.cpp

```
#include <iostream>
#include <math.h>
#include "Point.h"
using namespace std;

Point::Point(double x1, double y1){      ……..0.5 pts
            x=x1; y=y1;
        }
void Point::setX(double x1){       ……..0.5 pts
            x=x1;
        }
void Point::setY(double y1){       ……..0.5 pts
            y=y1;
        }
double Point::getX(){       ……..0.5 pts
            return x;
        }
double Point::getY(){       ……..0.5 pts
            return y;
        }
double Point::distance(Point pt){          ……..0.5 pts
            double d;
            d= pow(pt.getX()-x,2)+pow(pt.getY()-y, 2); d=sqrt(d);
            return d;
        }
void Point::add(Point pt){          ……..0.5 pts
            x= pt.getX()+x;
            y=pt.getY()+y;
        }
void Point::afficher(){             ……..0.5 pts
            cout<<"p("<<x<<" ,"<<y<<")"<<endl;
        }
```