

```

/*****
Cercle.h
*****/
#pragma once
#include <math.h>
#include "Point2D.h"
#include "Figure.h"

#define PI 3.14159265358979323846

class Point2D;
class Figure;

class Cercle : public Figure
{
private :
    double rayon;
    Point2D centre;

public:
    Cercle(Point2D leCentre , double leRayon);
    double getPerimetre();
    double getSurface();
};

/*****
Commande.h
*****/
#pragma once
#include <vector>
#include <string>
#include "Figure.h"

using namespace std;

class Figure;

class Commande
{
private:
    bool commandeTerminee;
    double prixMetreDecoupe , prixMetreCarreMatiere ;
    string idCommande;

public:
    Commande(string identifiantCommande , double lePrixMetreDecoupe , double lePrixMetreCarreMatiere);
    string getIdCommande() { return idCommande; }
    void ajouterNouvelleFigure(.....);
    void cloturerCommande();

    double getPrix() ;
};

/*****
Figure.h
*****/
#pragma once

class Figure
{
public:
    virtual double getPerimetre()=0;
    virtual double getSurface()=0;
};

/*****
Point2D.h
*****/
// Cette classe n'est pas à modifier
#pragma once

class Point2D
{
private:
    double x , y ;

public:
    Point2D(double x=0 , double y=0);
    double getX();
    double getY();

```

1 point

0,5 point

```

        void setX(double newX);
        void setY(double newY);
};

/*****
Polygone.h
*****/
#pragma once

#include <vector>
#include "Figure.h"
#include "Point2D.h"

using namespace std;

class Point2D;
class Figure;

#define abs(x) ( (x) >=0 ? (x) : -(x) )

class Polygone : public Figure
{
protected:
    vector<Point2D *> lesSommets;
    bool estFerme;

public:
    Polygone(void);
    static double distance(Point2D p1, Point2D p2);
    void insereUnNouveauSommet(Point2D * leSommet, int position=-1);
    void fermeLePolygone();
    double getPerimetre();
    double getSurface();
};

/*****
Cercle.cpp
*****/
#include "Cercle.h"

Cercle::Cercle(Point2D leCentre, double leRayon)
{
    centre = leCentre;
    rayon = leRayon;
}

double Cercle::getPerimetre()
{
    double perimetre;
    perimetre= (PI*2) * rayon;
    return perimetre;
}

double Cercle::getSurface()
{
    double surface;
    surface = PI * (rayon * rayon);
    return surface;
}

/*****
Commande.cpp
*****/
#include "Commande.h"

Commande::Commande(string identifiantCommande , double lePrixMetreDecoupe , double lePrixMetreCarreMatiere)
{
    .....
}

void Commande::ajouterNouvelleFigure(.....)
{
    .....
}

void Commande::cloturerCommande()
{
    .....
}

```

1 point

1 point

```

}

double Commande::getPrix()
{
    .....
}

/*****
main.cpp
*****/
#include <iostream>
#include <conio.h>
#include "Polygone.h"
#include "Cercle.h"
/*#include "Commande.h"*/

using namespace std ;           // espace de nommage standard

int main()
{
    // Testez la classe Cercle

    Cercle* toto;
    toto = new Cercle(Point2D(3, 5), 6.2);
    cout << "Le perimetre est:" << toto->getPerimetre() << endl;
    cout << "La surface est:" << toto->getSurface() << endl;

    // Testez la classe Polygone avec la figure de test du sujet
    double Coordonnees[6][2]={ { 1 , 1 } , { 3 , 5 } , { 5 , 7 } , { 5 , 1 } , { 3 , 3 } , { 3 , 1 } };

    Polygone* totol;
    totol = new Polygone;
    totol->insereUnNouveauSommet(new Point2D(5,3));
    totol->insereUnNouveauSommet(new Point2D(7, 5));
    totol->insereUnNouveauSommet(new Point2D(1, 5));
    totol->insereUnNouveauSommet(new Point2D(3, 3));
    totol->insereUnNouveauSommet(new Point2D(1, 3));
    totol->insereUnNouveauSommet(new Point2D(1, 1));

    cout << "Le perimetre est:" << totol->getPerimetre() << endl;
    cout << "La surface est:" << totol->getSurface() << endl;

/*
    // Sapin de Noel et boules
    double CoordonneesSapin[15][2]={ { 2 , 2 } , { 5 , 4 } , { 3 , 4 } , { 5 , 6 } , { 4 , 6 } , { 6 , 8 } ,
    { 8 , 6 } , { 7 , 6 } ,
    { 9 , 4 } , { 7 , 4 } , { 10 , 2 } , { 6
    .5 , 2 } , { 6.5 , 1 } , { 5.5 , 1 } , { 5.5 , 2 } };

    double CoordonneesCentreCercles[6][2]={ { 2.5 , 3.5 } , { 3.5 , 5.5 } , { 4.5 , 7.5 } , { 7.5 , 7.5 }
    , { 8.5 , 5.5 } , { 9.5 , 3.5 } };
    int i;

    // Création du polygone sapin
    ...

    cout <<"superficie du sapin = " << .... << " " ;
    cout <<"Perimetre du sapin = " << .... << endl;

    // Création des 6 cercles
    ...
        cout <<"superficie du cercle " << i <<" = " <<... << " " ;
        cout <<"Perimetre du cercle " << i <<" = " << ...<< endl;
    ...

    // Création de la commande du Père Noel
    ...

    // Ajout des figures (le sapin et les 6 cercles) à la commande
    ...

    // Affichage du prix de cette commande
    cout <<"\n Cout de la commande : " << ... <<" = " << ... <<" euros" << endl;

```

Test de la classe  
Cercle : 0,5 point

Je vous avais préparé  
un tableau des  
coordonnées pour  
éviter de les resaisir

pg principal de test du  
polygone de la figure :  
1 point

```

    */

    _getch();        // on attend l'appui sur une touche
    return 0 ;       // fin du programme
}

/*****
Point2D.cpp
*****/
// Cette classe n'est pas à modifier
#include "Point2D.h"

Point2D::Point2D(double x , double y)
{
    this->x = x;
    this->y = y;
}

double Point2D:: getX()
{ return x ;}

double Point2D::getY()
{
    return y;
}

void Point2D::setX(double newX)
{
    x = newX;
}

void Point2D::setY(double newY)
{
    y = newY;
}

/*****
Polygone.cpp
*****/
#include <math.h>
#include "Polygone.h"

Polygone::Polygone(void)
{
    estFerme = false;
}

double Polygone::distance(Point2D p1, Point2D p2)
{
    double distance;
    distance = sqrt((p2.getX() - p1.getX())* ((p2.getX() - p1.getX()) + (p2.getY() - p1.getY())* (p2.getY()
- p1.getY())));
    return distance;
}

void Polygone::insereUnNouveauSommet(Point2D * leSommet, int position=-1)
{
    if (position == -1)
    {
        lesSommets.push_back(leSommet);
    }
    else
    {
        lesSommets.insert(lesSommets.begin()+ position, leSommet);
    }
}

void Polygone::fermeLePolygone()
{
    lesSommets.insert(lesSommets.begin(), 0);
    estFerme = true;
}

double Polygone::getPerimetre()
{
    int i;
    double perimetre;
    for (i = 0; i < lesSommets.size()-1; i++)
    {
        perimetre = perimetre + Polygone::distance(*lesSommets[i], *lesSommets[i + 1]);
    }
}

```

1 point

```
    }  
    return perimetre;  
}  
  
double Polygone::getSurface()  
{  
    int i;  
    double surface;  
    for (i = 0; i < lesSommets.size() - 1; i++)  
    {  
        surface=surface+1/2(  
    }  
    return surface;  
}
```