3,5/20 : Beaucoup trop incomplet

OK.cpp


```
/**********************
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(double leRayon, Point2D leCentre);
        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;
        double prix;


public:
        vector<Figure*>lesFigure;
        Commande(string identifiantCommande , double lePrixMetreDecoupe , double lePrixMetreCarreMatiere);
        string getIdCommande() { return idCommande; }
        void ajouterNouvelleFigure(Figure *laFigure);
        void cloturerCommande();

        double getPrix() ;

};



/**********************
Figure.h
**********************/
#pragma once
#include <iostream>
#include "Polygone.h"

class Figure
{
private:
        double Perimetre, Surface ;

public:
        virtual void getPerimetre();
        virtual void getSurface();

};

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

1 point

Méthodes virtuelles pures ! ne pas oublier =0

```
class Point2D
{
private:
        double x , y ;

public:
        Point2D(double x=0 , double y=0);
        double getX();
        double getY();
        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 Point2D
{
protected:
        vector<Point2D *> lesSommets;
        bool estFerme;

public:
        Polygone(void);
        static void distance();
        void insereUnNouveauSommet();
        void fermeLePolygone();
        double getPerimetre();
        double getSurface();

};



/***********************
Cercle.cpp
***********************/
#include <Math.h>
#include <iostream>
#include "Point2D.h"
#include "Figure.h"
#include "Cercle.h"



Cercle(double leRayon , Point2D leCentre)
{

};
double getPerimetre()
{

};

double Cercle::getSurface()
{

};

double Cercle::getPerimetre()
{

};


/***********************
Commande.cpp
***********************/
#include "Commande.h"
#include "Figure.h"
#include "Polygone.h"
```

0,25 point

```
#include "Figure.cpp"


using namespace std;

Commande::Commande(string identifiantCommande, double lePrixMetreDecoupe, double lePrixMetreCarreMatiere)
{
        this->idCommande = identifiantCommande;
        this->prixMetreDecoupe = lePrixMetreDecoupe;
        this->prixMetreCarreMatiere = lePrixMetreCarreMatiere;
};

string Commande::getIdCommande()
{
        return idCommande;
};

void Commande::ajouterNouvelleFigure(Figure *laFigure)
{
        this->lesFigure = lesFigure;
};

void Commande::cloturerCommande()
{
        this->commandeTerminee;
};

double Commande::getPrix()
{
        for (prix; prix >= 0; prix++)
        {
                prix = (getPerimetre * prixMetreDecoupe) + (getSurface * prixMetreCarreMatiere);

        }
};




/**********************
Figure.cpp
**********************/
#include <iostream>
#include <iostream>
#include "Polygone.h"
#include "Cercle.h"
#include "Commande.h"
#include "Point2D.h"
#include "Figure.h"

using namespace std;



void Figure::getPerimetre()
{

};
void Figure::getSurface()
{

};



/**********************
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
```

this->commandeTerminee=false;

0,5 points

```
	// 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 } };









	// 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 <<"\nCout de la commande : " << ... <<" = " << ... <<" euros" << endl;




	_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)
{

};

 void Polygone::distance()
{

 };

 void Polygone::insereUnNouveauSommet()
 {

 };


void Polygone::fermeLePolygone()
{

};

double Polygone::getPerimetre()
{

}

double Polygone::getSurface()
{


};
```