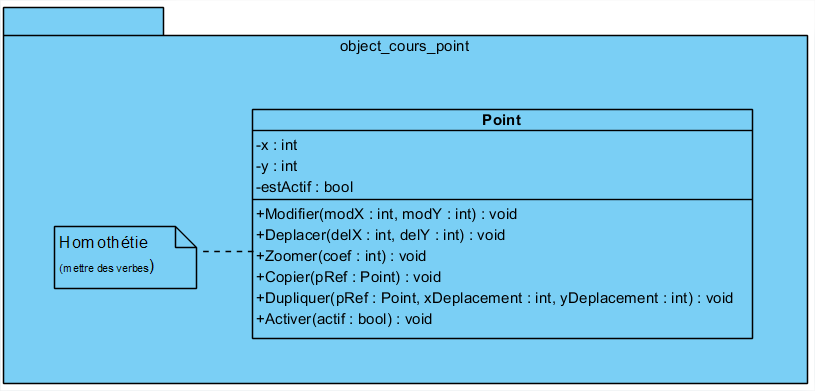
Diagramme de classe UML (Unified Modeling Language)

Attributs (caractéristiques)



Classe

Méthodes

(actions)

Visibilité ( + ou -) :

+ (public) accessible à l’intérieur

- (private) inaccessible à l’extérieur

(sens d’écriture) :

Visibilité nomVariable : type

Visibilité nomMéthode(argument ou rien) : typeDeRetour

Les arguments : nomArgument1 : type, nomArgument2 : type, …

**Code en C# Classe Point**

namespace object\_cours\_point

{

class Point

{

private int x, y;

private bool estActif;

public void Modifier(int modX, int modY)

{

//donne la valeur de modX à x et modY à y

x = modX;

y = modY;

}

public void Deplacer(int delX, int delY)

{

x = x + delX;

y = y + delY;

}

//homothétie

public void Zoomer(int coef)

{

// agrandir les cordonnées avec le coefficient donné

x \*= coef;

y \*= coef;

}

public void Copier(Point pRef)

{

//copier les coordonnées

x = pRef.x;

y = pRef.y;

}

public void Dupliquer(Point pRef, int xDeplacement, int yDeplacement)

{

//copier un point et le déplacer de combien de pixel on veut

x = pRef.x + xDeplacement;

y = pRef.y + yDeplacement;

}

public void Activer(bool actif)

{

estActif = actif;

}

// constructeur de base est à 0 , 0

// on peut l'écraser avec des valeurs

public Point()

{

// ecrasé

// x = 2;

// y = 4;

}

// si l'on défini un nouveau constructeur on doit faire

// "public Point(){}" sinon le constructeur de base est écrasé

public Point(int constructeurX, int constructeurY)

{

x = constructeurX;

y = constructeurY;

}

}

}

**Classe Program (main)**

namespace object\_cours\_point

{

class Program

{

static void Main(string[] args)

{

Point pRef = new Point(); //instanciation

Point p2 = new Point();

pRef.Zommer(10);

New Point() instancie la classe : devient un objet (en mémoire, sans nom)

P1 est une référence d’objet instanciation de la classe Point

// pRef.Deplacer(1, 1);

p2.Copier(pRef);

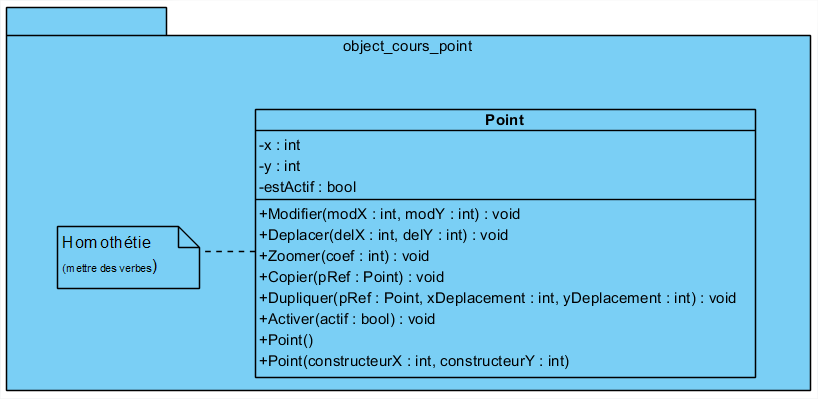
// p2.Dupliquer(pRef, 2, 3);

}

}

}

**2 - Constructeur**



namespace object\_cours\_point

{

class Point

{

// constructeur de base est à 0 , 0

// on peut l'écraser avec des valeurs

public Point()

{

// ecrasé

// x = 2;

// y = 4;

}

// si l'on défini un nouveau constructeur on doit faire

// "public Point(){}" sinon le constructeur de base est écrasé

public Point(int constructeurX, int constructeurY)

{

x = constructeurX;

y = constructeurY;

}

}

}