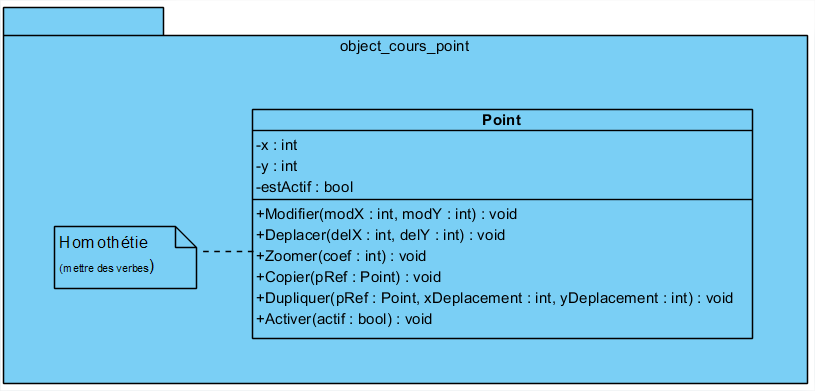
Samuel LITZLER STS 1

**Introduction au cours d’objet en C#**

**1 - Diagramme de classe UML (Unified Modeling Language)**



Attributs (caractéristiques)

Classe

Méthodes

(actions)

A CONNAITRE (visibilité, argument ...)

Methodes :

Classique : actions

Constructeur :

Getter et Setter

Visibilité ( + ou -) :

+ (public) accessible à l’extérieur

- (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# Class 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;

}

}

}

**Class Program (main)**

namespace object\_cours\_point

{

class Program

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

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

{

static void Main(string[] args)

{

Point pRef = new Point(); //instanciation

Point p2 = new Point();

pRef.Zommer(10);

// pRef.Deplacer(1, 1);

p2.Copier(pRef);

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

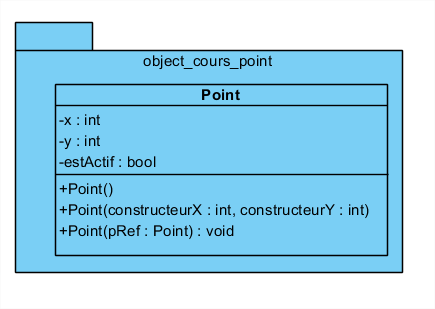
}

}

}

**2 - Constructeur**

**3 types de constructeur : par défaut, par arguments (ou paramètres) et par copie**



public void Activer(bool actif)

{

estActif = actif;

}

// constructeur de base est à 0 , 0

// on peut l'écraser avec des valeurs

public Point()

{

// écrasé

// x = 2;

// y = 4;

}

// si l'on défini un constructeur par argument on doit faire

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

public Point(int x, int y) // ou (int constructeurX, int constructeurY)

{

this.x = x;

this.y = y;

//ou :

// x = constructeurX;

// y = constructeurY;

}

// constructeur copie

public Point(Point pRef)

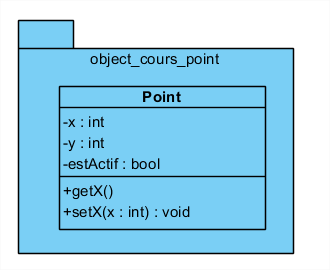
{

x = pRef.x;

y = pRef.y;

}

**3 - Getter & setter**

Getter pour retourner la valeur de l’attribut.

Setter pour changer la valeur de l’attribut.

**Class point**

// setter... dangereux car change le private x

public void setX(int x)

{

this.x = x;

}

// getter retourne la valeur de x

public int getX()

{

return x;

}

**Class program (main) avec constructeur et getter**

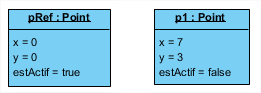
Point pRef = new Point(); //instanciation (0,0)

// 2,3 c'est le constructeur par argument dans la class point

Point p1 = new Point(2,3);

Console.WriteLine(p1.getX());

**4 - Diagramme objet**

Permet de montrer les objets à un instant t.

**5 – Static**

Static est des membres de classe ≠ membre d’instance (si non statique)

Rendre global un attribut static : faire quelque chose de commun à tout le monde

Méthode static : on les utilise de partout (ex : sos(), sin()….)

**6 – Destructeur**

Un seul par objet, il est appelé autant de fois que il y à d’objet.

Utile pour fermer proprement une connexion Bluetooth, réseaux…

**~Point()**