Tema 1 – Sessió 5
Programació orientada a objecte

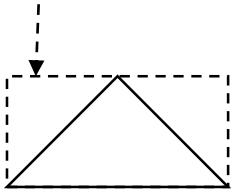
Exercici

A partir de la definició d'aquesta classe `Point` volem crear una nova classe `Poligon` que permeti guardar tots els vèrtexs d'un polígon i també les coordenades de la cantonada superior esquerra i de la cantonada inferior dreta del rectangle mínim que engloba al polígon

```
class Point:
    def __init__(self, x = 0, y = 0):
        self._x = x
        self._y = y

    @property
    def x(self):
        return self._x
    @x.setter
    def x(self, valor):
        self._x = valor

    def __sub__(self,p2):
        return math.sqrt((self.x - p2.x)**2 + (self.y - p2.y)**2)

    def __str__(self):
        return "("+ str(self.x) + ", " + str(self.y) + ")"
```

```
    @property
    def y(self):
        return self._y
    @y.setter
    def y(self, valor):
        self._y = valor
```

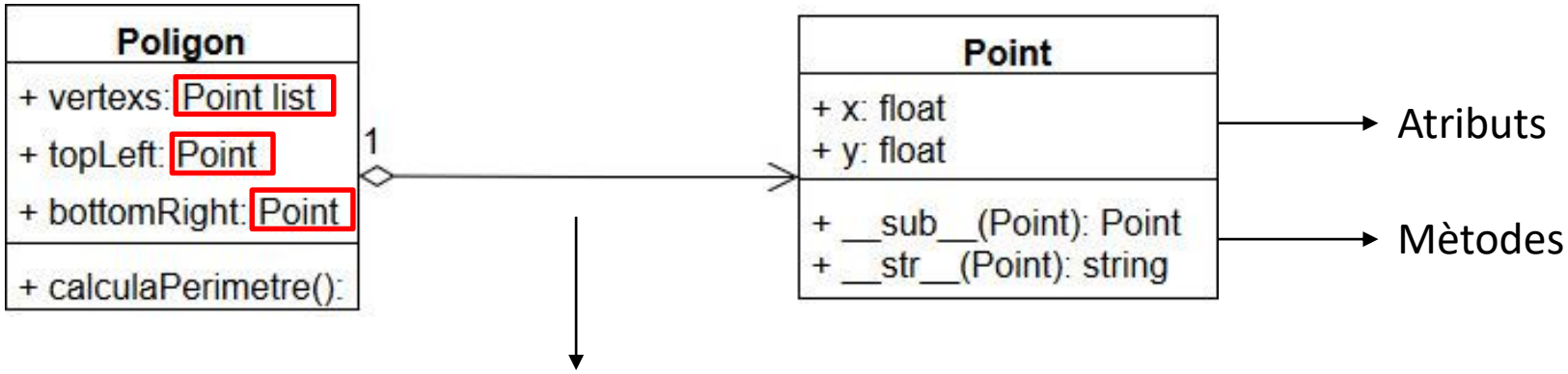
Rectangle mínim



Composició de classes

- Utilització d'una classe ja existent (`Point`) com a atribut dins d'una altra classe (`Poligon`): *un polígon està composat per punts*

Diagrama UML: Representació de l'estructura de les classes



Relació de composició:

- Un polígon està composat per N punts

Composició de classes

```
import punt
```

Importem el mòdul amb la definició de la classe `Point`

```
class Poligon:
    maxim = 1000
    def __init__(self):
        self._vertexs = []
        self._topLeft = punt.Point(Poligon.maxim, Poligon.maxim)
        self._bottomRight = punt.Point()

    def afegeixVertex(self, pt):
        self._vertexs.append(pt)
        if pt.x < self.topLeft.x:
            self.topLeft.x = pt.x
        else:
            if pt.x > self.bottomRight.x:
                self.bottomRight.x = pt.x
        if pt.y < self.topLeft.y:
            self.topLeft.y = pt.y
        else:
            if pt.y > self.bottomRight.y:
                self.bottomRight.y = pt.y
```

Inicialització dels objectes de la classe `Point`

Afegim objectes de la classe `Point` a la llista

- Utilització dels mètodes i propietats de la classe `Point`.
- Només hauríem d'utilitzar mètodes i propietats "*públics*", no els atributs "*privats*".

Exercici

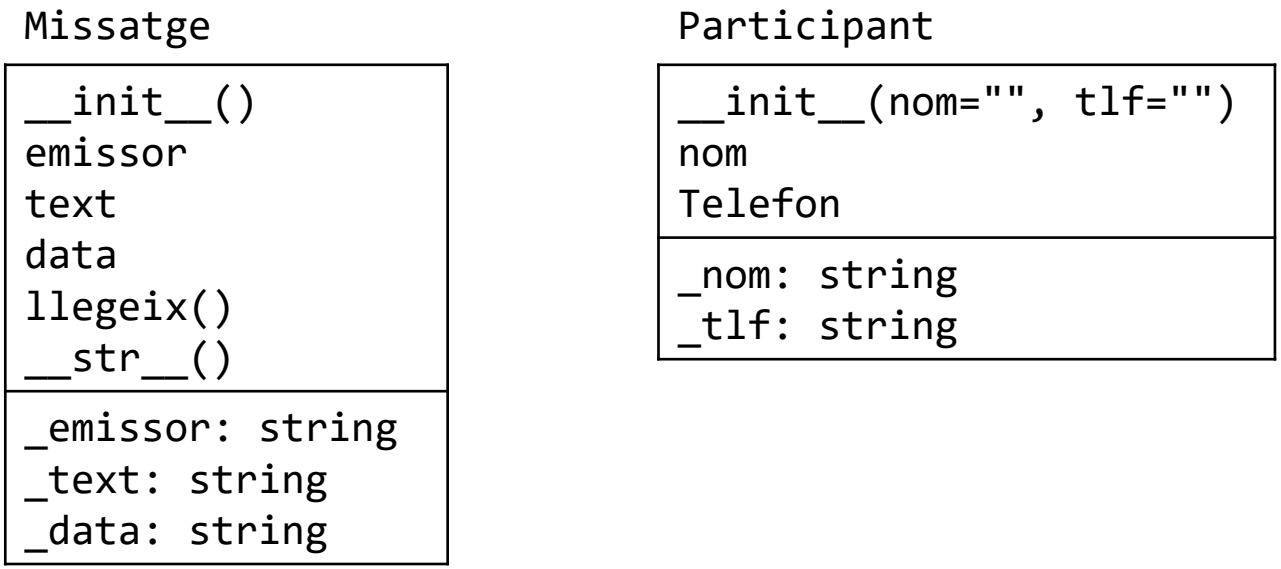
Completar la definició de la classe `Poligon`:

- Afegir propietats per recuperar els punts de la cantonada superior esquerra i de la cantonada inferior dreta del rectangle mínim
- Afegir un mètode que retorni el perímetre del polígon

Fer un programa que llegeixi tots els vèrtexs d'un polígon i després calculi i mostri per pantalla el perímetre del polígon i mostri per pantalla els punts de la cantonada superior esquerra i de la cantonada inferior dreta del rectangle mínim.

Exercici

Suposem que tenim la classe `Missatge` que hem definit en un exercici anterior i que tenim també una classe `Participant` que guarda el nom i el telèfon d'una persona que participa en un grup de conversa en una aplicació de missatgeria de l'estil de WhatsApp.



Exercici

Utilitzant aquestes dues classes volem crear la classe `GrupConversa`, que permeti guardar la llista de tots els participants en el grup i la llista de tots els missatges que s'han enviat al grup.

Declareu i implementeu la classe `GrupConversa` amb tots els atributs que facin falta per gestionar el grup segons el que s'ha explicat fins ara i amb els mètodes següents:

- Un constructor que rebi com a paràmetre una llista amb tuples `(nom, telefon)` que corresponen a les dades dels participants del grup i que s'ha d'utilitzar per inicialitzar la llista de participants. La llista de missatges inicialment estarà sempre buïda.
- Un mètode per buscar si un participant està donat d'alta al grup, que a partir del nom d'un participant, retorni cert si el participant està dins del grup i fals si no ho està.
- Un mètode per afegir un missatge al grup que rebi com a paràmetres el nom de l'emissor, el text del missatge i la data d'enviament i afegeixi el missatge a la llista de missatges enviats. Abans haurà de comprovar que l'emissor realment forma part de la llista de participants al grup. Per això es podrà fer servir el mètode anterior. Si l'emissor no és participant s'ha de generar una excepció utilitzant assercions.
- Un mètode rebi com a paràmetre el nom d'una persona i mostri per pantalla tots els missatges que ha enviat aquesta persona.

Implementeu un programa principal que declari un objecte de la classe `GrupConversa` i l'inicialitzi amb una llista de participants. Després ha de llegir i guardar un conjunt inicial de missatges. Finalment, ha de demanar el nom d'una persona i mirar si es correspon amb algun dels participants al grup. Si és un dels participants al grup ha de mostrar tots els missatges que hagi enviat al grup.