Composició de classes
Tema 1: Programació orientada a objectes

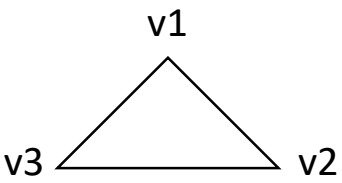
Exercici voluntari pujar nota

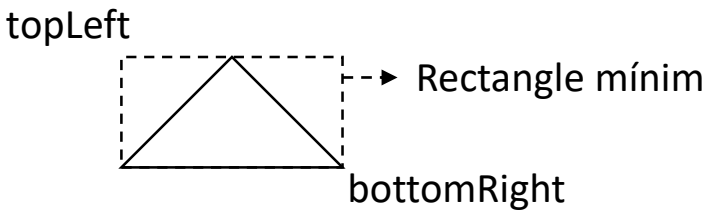
A partir de la definició d'aquesta classe `Punt` volem crear una nova classe `Poligon` que permeti guardar tots els vèrtexs d'un polígon (màxim 30 vèrtexs) i també les coordenades de la cantonada superior esquerra i de la cantonada inferior dreta del rectangle mínim que engloba al polígon

```
class Punt
{
public:
   Punt();
   Punt(float x, float y);
   ~Punt();

   void setX(float x);
   void setY(float y);
   void llegeix();

   float getX() const;
   float getY() const;
   void mostra() const;
   float distancia(const Punt &p) const;
private:
   float m_x, m_y
};
```





Exercici voluntari pujar nota

Definim la classe Poligon aquests atributs:

```
class Poligon
{
private:
   Punt m_vertexs[MAX_VERTEXS];
   int m_nVertexs;
   Punt m_topLeft;
   Punt m_bottomRight;
};
```

Exercici voluntari pujar nota

1. Completar la definició de la classe `Poligon`:
   - Afegir un mètode `afegeixVertex` que rebi com a paràmetre un objecte de la classe `Punt`. S'haurà de comprovar que no es superi el nombre màxim de vèrtexs del polígon. Si el polígon ja té els 30 vèrtexs màxims no es podrà afegir el vèrtex. S'haurà de tenir en compte d'actualitzar (si cal) les coordenades de la cantonada superior esquerra i de la cantonada inferior dreta del rectangle mínim.
   - Afegir mètodes `getTopLeft` i `getBottomRight` per recuperar els punts de la cantonada superior esquerra i de la cantonada inferior dreta del rectangle mínim.
   - Afegir un mètode `calculaPerimetre` que retorni el perímetre del polígon

2. Implementar una funció `interseccioPoligons` que donats dos polígons que es passen com a paràmetre, retorni si els seus rectangles mínims tenen algun punt d'intersecció.

Exercici voluntari pujar nota

Comencem fent la declaració completa de la classe `Poligon`:

- Afegiu a la classe `Poligon` la declaració dels mètodes que s'han indicat abans.
- Especifiqueu quins mètodes i quins paràmetres d'aquests mètodes s'han de declarar com a `const`.
- Especifiqueu la capçalera de la funció `interseccioPoligons` indicant quins paràmetres de la funció s'han de declarar com a `const`.

Exercici: solució

```
class Poligon
{
public:
   Poligon(): m_topLeft(1000, 1000), m_bottomRight(0,0)
      { m_nVertexs = 0; }
   ~Poligon() {}
   void afegeixVertex(const Punt& pt);
   Punt getTopLeft() const { return m_topLeft; }
   Punt getBottomRight() const { return m_bottomRight; }
   float calculaPerimetre() const;.
private:
   Punt m_vertexs[MAX_VERTEXS];
   int m_nVertexs;
   Punt m_topLeft;
   Punt m_bottomRight;
};


bool interseccioPoligons(const Poligon& p1, const Poligon& p2);
```

Exercici: solució

```
void Poligon::afegeixVertex(const Punt& pt)
{
   if (m_nVertexs < MAX_VERTEXS)
   {
      m_vertexs[m_nVertexs] = pt;
      m_nVertexs++;
      if (pt.getX() < m_topLeft.getX())
         m_topLeft.setX(pt.getX());
      else
         if (pt.getX() > m_bottomRight.getX())
            m_bottomRight.setX(pt.getX());
      if (pt.getY() < m_topLeft.getY())
         m_topLeft.setY(pt.getY());
      else
         if (pt.getY() > m_bottomRight.getY())
            m_bottomRight.setY(pt.getY());
   }
}

float Poligon::calculaPerimetre() const
{
   float perimetre = 0;
   for (int i = 0; i < m_nVertexs - 1; i++)
      perimetre += m_vertexs[i].distancia(m_vertexs[i + 1]);
   perimetre += m_vertexs[0].distancia(m_vertexs[m_nVertexs-1]);
   return perimetre;
}
```

Exercici: solució

```
bool interiorRectangle(Punt& pt, Punt& supEsq, Punt& infDreta)
{
   return ((pt.getX() >= supEsq.getX()) && (pt.getX() <= infDreta.getX()) &&
         (pt.getY() >= supEsq.getY()) && (pt.getY() <= infDreta.getY()));
}

bool interseccioPoligons(const Poligon& p1, const Poligon& p2)
{
   Punt supEsq1, infDreta1, supEsq2, infDreta2;
   bool interseccio = false;

   supEsq1 = p1.getTopLeft();
   infDreta1 = p1.getBottomRight();
   supEsq2 = p2.getTopLeft();
   infDreta2 = p2.getBottomRight();

   Punt supDreta1, infEsq1;
   supDreta1.setX(infDreta1.getX());
   supDreta1.setY(supEsq1.getX());
   infEsq1.setX(supEsq1.getX());
   infEsq1.setY(infDreta1.getY());

   interseccio = interiorRectangle(supEsq1, supEsq2, infDreta2);
   interseccio = interseccio || interiorRectangle(supDreta1, supEsq2, infDreta2);
   interseccio = interseccio || interiorRectangle(infEsq1, supEsq2, infDreta2);
   interseccio = interseccio || interiorRectangle(infDreta1, supEsq2, infDreta2);
   return interseccio;
}
```