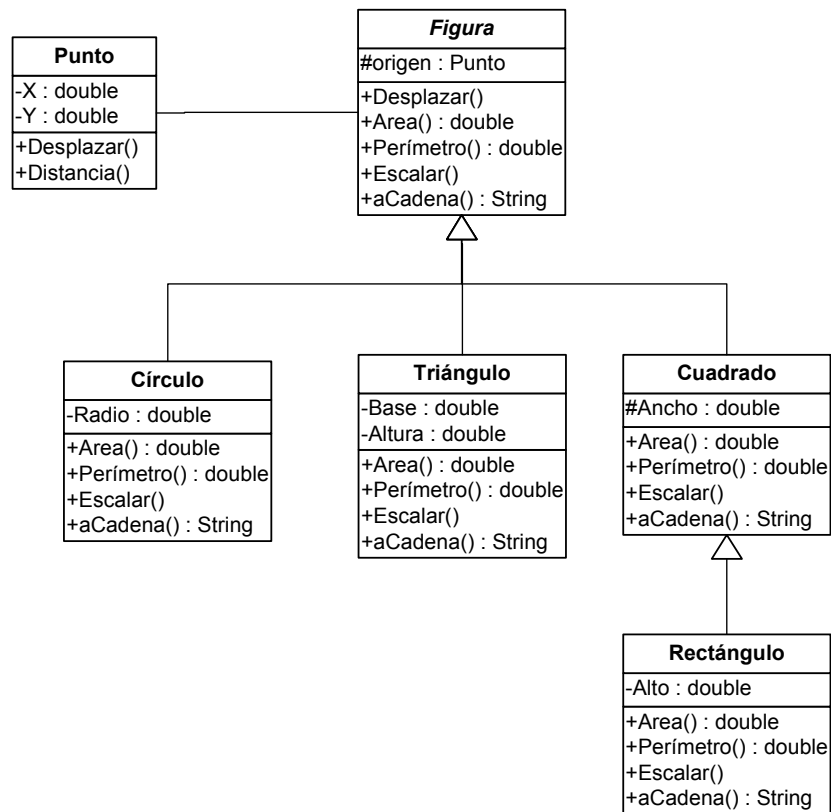


La clase Figura



```

clase Punto
var
    real : x
    real : y

    constructor Punto()
    inicio
        x ← 0
        y ← 0
    fin_constructor

    constructor Punto(real : x,y)
    inicio
        instancia()
        instancia.x ← x
        instancia.y ← y
    fin_constructor

    nada método Desplazar(E real : dx, dy)
    inicio
        x ← x + dx
        y ← y + dy
    fin_método

    real método Distancia(E Punto :p)
    inicio
        devolver (((x - p.x) ** 2 + (y - p.y) ** 2) ** 0.5)
    fin_método

    cadena método aCadena()
    inicio
        devolver ('\ ' & x & '\,' & y & '\')
```

```

    fin_método
fin_clase

abstracta clase Figura
var
    protegido Punto : origen

    constructor Figura()
    inicio
        numFiguras ← numFiguras + 1
    fin_método

    //Desplaza una Figura
    nada método Desplazar(E real : dx, dy)
    inicio
        origen.Desplazar(dx, dy)
    fin_método

    //Calcula el área de una figura
    abstracto real método Area()

    //Calcula el perímetro de una figura
    abstracto real método Perimetro()

    abstracto nada método Escalar(E real : n)
fin_clase

clase Círculo hereda_de Figura
var
    real : radio

    //Crea un círculo nulo en el punto 0,0
    constructor Círculo()
    inicio
        super()
        origen ← nuevo Punto()
        radio ← 0
    fin_constructor

    //Crea un círculo en el punto p con un radio determinado
    constructor Círculo(Punto :p)
    inicio
        instancia()
        origen ← p
        radio = r
    fin_constructor

    //Crea un círculo en el punto 0,0 con un radio determinado
    constructor Círculo(real : r)
    inicio
        instancia(nuevo Punto(), r)
    fin_constructor

    //Crea un círculo nulo en el punto p
    constructor Círculo(Punto : p)
    inicio
        instancia(p, 0)
    fin_constructor

    //Calcula el área de un círculo
    real método Area()
    inicio
        devolver(3.1416 * radio ** 2)

```

fin_método

//Calcula el perímetro de un círculo

real método Perimetro()

inicio

devolver (2 * 3.1416 * radio)

fin_método

//Modifica el radio de un círculo en un porcentaje

nada método Escalar(**E real** : porc)

inicio

 radio ← radio + radio * porc / 100

fin_método

cadena método aCadena()

inicio

devolver (origen.aCadena() & ' ' & radio)

fin_método

fin_clase

//Representa un triángulo isósceles

clase Triángulo **hereda_de** Figura

var

real : base

real : altura

//Crea un triángulo nulo en el punto 0,0

constructor Triángulo()

inicio

super()

 origen ← **nuevo** Punto()

 base ← 0

 altura ← 0

fin_método

//Crea un triángulo en el punto p con una base y altura determinada

constructor Triángulo(Punto : p, **real** : b,a)

inicio

super()

 origen ← p

 base ← b

 altura ← a

fin_constructor

//Crea un triángulo en el punto 0,0 con una base y altura determinada

constructor Triángulo(**real** : b,a)

inicio

instancia (**nuevo** Punto(), b, a)

fin_constructor

//Crea un triángulo nulo en el punto p

constructor Triángulo(Punto : p)

inicio

instancia (p, 0, 0)

fin_constructor

//Calcula el área de un triángulo

real método Area()

inicio

devolver (base * altura / 2)

fin_método

//Calcula el perímetro de un triángulo

```

real método Perimetro()
inicio
    devolver((altura ** 2 + (base / 2) ** 2) ** 0.5) * 2 + base)
fin_método

//Modifica la base y la altura de un triángulo en un porcentaje determinado
nada método Escalar(E real : porc)
inicio
    base ← base * porc / 100
    altura ← altura * porc / 100
fin_método

cadena método aCadena()
inicio
    devolver(origen.aCadena() & ' ' & base & ' ' & altura)
fin_método

fin_clase

clase Cuadrado hereda_de Figura
var
    protegido real : ancho

//Crea un Cuadrado nulo en el punto 0,0
constructor Cuadrado()
inicio
    super()
    origen ← nuevo Punto()
    ancho ← 0
fin_constructor

//Crea un Cuadrado en el punto p con un ancho determinado
constructor Cuadrado(Punto : p, real : ancho)
inicio
    instancia()
    origen ← p
    instancia.ancho ← ancho
fin_constructor

//Crea un Cuadrado en el punto 0,0 con un ancho determinado
constructor Cuadrado(real : ancho)
inicio
    instancia(nuevo Punto(), ancho)
fin_constructor

//Crea un Cuadrado nulo en el punto p
constructor Cuadrado(Punto : p)
inicio
    instancia(p, 0)
fin_constructor

//Calcula el área de un Cuadrado
real método Area()
inicio
    devolver (ancho * ancho)
fin_método

//Calcula el perímetro de un Cuadrado
real método Perimetro()
inicio
    devolver(ancho * 4)
fin_método

//Modifica el ancho y el alto de un en un porcentaje determinado
nada método Escalar(E real : porc)

```

```

inicio
    ancho ← ancho + ancho * porc / 100
fin_método

cadena método aCadena()
inicio
    devolver(origen.aCadena() & ' ' & ancho)
fin_método

fin_clase

clase Rectángulo hereda_de Cuadrado
var
    real : alto

    //Crea un rectángulo nulo en el punto 0,0
    constructor Rectángulo()
    inicio
        super()
        alto ← 0
    fin_constructor

    //Crea un rectángulo en el punto p con un ancho y alto determinados
    constructor Rectángulo(Punto : p, real n:, alto)
    inicio
        super (p, ancho)
        instancia.alto = alto
    fin_constructor

    //Crea un rectángulo en el punto 0,0 con un ancho y alto determinados
    constructor Rectángulo(real : ancho, alto)
    inicio
        instancia(nuevo Punto(), ancho, alto)
    fin_constructor

    //Crea un rectángulo nulo en el punto p
    constructor Rectángulo(Punto : p)
    inicio
        instancia (p, 0, 0)
    fin_constructor

    //Calcula el área de un rectángulo
    real método Area()
    inicio
        devolver (ancho * alto)
    fin_método

    //Calcula el perímetro de un rectángulo
    real método Perimetro()
    inicio
        devolver(ancho * 2 + alto * 2)
    fin_método

    //Modifica el ancho y el alto de un rectángulo según un porcentaje
    nada método Escalar(real : porc)
    inicio
        super.Escalar(porc)
        alto ← alto + alto * porc / 100
    fin_método

    //Modifica el ancho y el alto de un rectángulo
    nada método Escalar(real : nuevoAncho,nuevoAlto)
    inicio
        ancho ← nuevoAncho
        alto ← nuevoAlto

```

```

fin_método

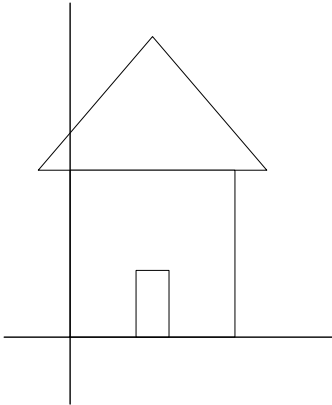
cadena método aCadena()
inicio
    devolver(super.aCadena() & ' ' & alto)
fin_método

```

```
fin_clase
```

Dibujar una casa

El siguiente algoritmo realiza el dibujo simple de una casa con un cuadrado, un rectángulo y un triángulo, lo desplaza, escribe las características de cada figura y calcula el área total ocupada por cada una de las figuras.



```

algoritmo DibujarCasa
var
    array[1..3] de Figura : casa
    entero : i
    real : area
inicio
    //Crea el cuadrado en el punto (0,0) con una ancho de 5
    casa[1] ← nuevo Cuadrado(5.0)
    //Crea la puerta en el punto 0,2 con un ancho de 1 y un alto de 2
    casa[2] ← nuevo Rectángulo(nuevo Punto(0.0,2.0),1.0,2.0)
    //Crea el tejado en el punto -1,5 con una base de 7 y una altura de 4
    casa[3] ← nuevo Triángulo(nuevo Punto(-1,5),7.0,2.0)

    //Se desplaza la casa
    desde i ← 0 hasta 3 hacer
        casa[i].Desplazar(0.0,2.0)
    fin_desde

    //Escribir las figuras(ejemplo de polimorfismo)
    area ← 0
    desde i ← 0 hasta 3 hacer
        escribir(casa[i].aCadena())
    fin_desde

    //¿Qué area ocupan todas las figuras? (ejemplo de polimorfismo)
    area ← 0
    desde i ← 0 hasta 3 hacer
        area ← area + casa[i].Area()
    fin_desde
    escribir(total)

fin

```