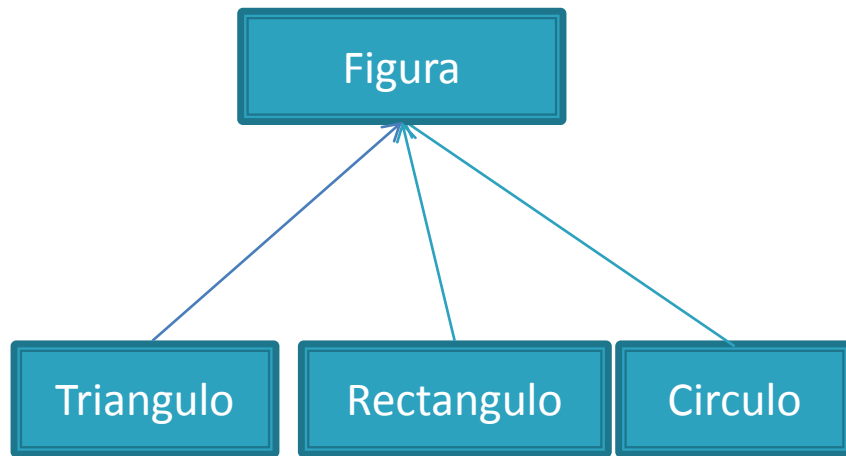


Smalltalk

Temas: Herencia–Polimorfismo–Colecciones

Ejercicio: Figuras Geométricas

- ▶ Jerarquía de figuras geométricas
- ▶ Se desea calcular el perímetro y superficie de cada figura.

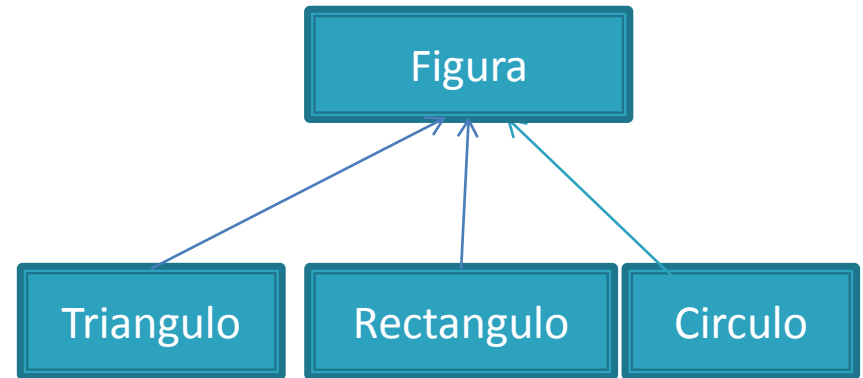


- ▶ Agregar un método que informe si el rectángulo es un rectángulo o un cuadrado.
- ▶ Agregar un método que informe el tipo de triangulo(equilatero, isósceles, escaleno).

(Autor: Ing. Germán Romaní)

Clase Figura

Object subclass: #Figura
instanceVariableNames: "
classVariableNames: "
poolDictionaries: "
category: 'Figuras'
"Metodos a Redefinir"
area
perimetro



Clase Triangulo

Ayuda:

- ▶ Formula para calcular el área de un triangulo en base a sus lados:

$$\text{Área} = \frac{1}{4} \sqrt{(a + b - c)(a - b + c)(-a + b + c)(a + b + c)}$$

Clase Circulo

Figura subclass: #Circulo

instanceVariableNames: 'radio'

classVariableNames: "

poolDictionaries: "

category: 'Figuras'

“Metodos Propios”

radio: unRadio

"Establece el Radio de un circulo"

radio: =unRadio.

radio

"Indica cual es el radio de un circulo"

^radio.

“Redefinicion de Metodos Heredados”

area

"Metodo que redefine el area de una figura para que calcule la de un circulo"

| calculo |

calculo:= Float pi * (self radio raisedTo: 2).

^calculo.

perimetro

"Redefine el perimetro de un circulo, en este caso la longitud de la circunsferencia"

| perimetro |

perimetro := 2 * Float pi * radio.

^perimetro.

Clase Rectangulo

Figura subclass: #Rectangulo

instanceVariableNames: 'ladoMenor ladoMayor'

classVariableNames: "

poolDictionaries: "

category: 'Figuras'

“Metodos Propios”

ladoMayor: unLado

ladoMayor:= unLado.

ladoMenor: unLado

ladoMenor:= unLado.

ladoMenor

^ladoMenor.

ladoMayor

^ladoMayor.

tipoRectagunlo

"Determina si es un cuadrado o no"

(ladoMenor = ladoMayor)

ifTrue: [^'El Rectangulo es un cuadrado'].

ifFalse: [^'El Rectangulo tiene lados diferentes'].

^'Rectangulo'.

“Metodos Heredados”

area

^ladoMenor * ladoMayor.

perimetro

| calculo|

calculo := 2 * ladoMenor + 2 * ladoMayor.

^calculo.

Clase Triángulo

Figura subclass: #Triangulo

instanceVariableNames: 'base catetoA catetoB'

classVariableNames: "

poolDictionaries: "

category: 'Figuras'

"Metodos Propios"

base

^base.

catetoA

^catetoA.

catetoB

^catetoB.

base: laBase

base:= laBase.

catetoA: unCateto

catetoA:= unCateto.

catetoB: unCateto

catetoB:= unCateto.

Clase Triángulo(2)

tipoTriangulo

"Determina si un triangulo es equilatero, iscoceles, o escaleno"

(base = catetoA and: base = catetoB)

ifTrue: [^'Triangulo Equilatero'.]

ifFalse: [(base ~= catetoA and: base ~= catetoB and: catetoA ~= catetoB) ifTrue: [^'Triangulo escaleno'.]

ifFalse: [^'Triangulo Isoceles'.]

]

“Redefinicion de Metodos Heredados”

area

| t calculo|

t := (catetoA + catetoB - base) * (catetoA - catetoB + base) *
(catetoA negated + catetoB + base) * (catetoA + catetoB + base).

calculo:= 1 / 4 * (t sqrt).

^calculo.

perimetro

^base + catetoA + catetoB.

$$\text{Área} = \frac{1}{4} \sqrt{(a + b - c)(a - b + c)(-a + b + c)(a + b + c)}$$

Ejercicio: Gráfico

- ▶ En base al ejercicio anterior de Figuras Geométricas modificarlo de acuerdo a la siguiente funcionalidad:
- ▶ Clase Gráfico: contiene una colección de Figuras.
- ▶ Métodos que se requieren en Gráfico:
 - Métodos de inicialización, acceso y modificadores.
 - Figura con mayor superficie.
 - Cantidad de Figuras que tienen un área mayor que su perímetro.
 - Promedio de las superficies de las Figuras.
 - Generar una nueva colección con las figuras que tengan un perímetro mayor que un valor dado.

Clase Gráfico

Object subclass: #Grafico

instanceVariableNames: 'figuras'

classVariableNames: "

poolDictionaries: "

category: 'Figuras '

“Métodos de inicializacion”

initialize

"Inicializa la instancia de Grafico"

figuras:= OrderedCollection new.

“Métodos de acceso”

figuras

"Retorna los alumnos del curso "

^figuras.

“ Metodos modificadores”

figuras: unasFiguras

"Asigna unasFiguras al Grafico"

figuras:= unasFiguras.

Clase Gráfico(2)

"Metodos de control"

addFigura: unaFigura

"Asigna unaFigura a la coleccion de Figuras"

figuras add:unaFigura.

promedio

"Retorna el promedio de las superficies de las Figuras"

|prom ac|

ac:=0.

figuras do: [:unaFigura | ac:= ac+ unaFigura area].

(figuras size ~= 0)

ifTrue: [prom:= ((ac asFloat) /figuras size) asFloat.]

ifFalse:[prom:=0.]

^prom.

mayorSuperficie

"Retorna la superficie de la Figura mas grande"

|may|

may:=0.

figuras do: [:unaFigura |

(unaFigura area>may) ifTrue: [may:= unaFigura area]

]

^may.

Clase Gráfico(3)

cantAreaPerimetro

"Retorna la cant. de la Figuras con área mas grande que su perímetro"

|ar per cont|

cont:=0.

figuras do: [:unaFigura | ar := unaFigura area. per := unaFigura
perimetro.

(ar > per) ifTrue: [cont:= cont + 1]
].

^cont.

generarMayorPerimetro: perimetro

"Genera una nueva colección con los elementos mayores a un perímetro
dado"

|gen|

gen := OrderedCollection new

figuras do: [:unaFigura |

(unaFigura perimetro > perimetro) ifTrue: [gen add:unaFigura]
].

^gen.

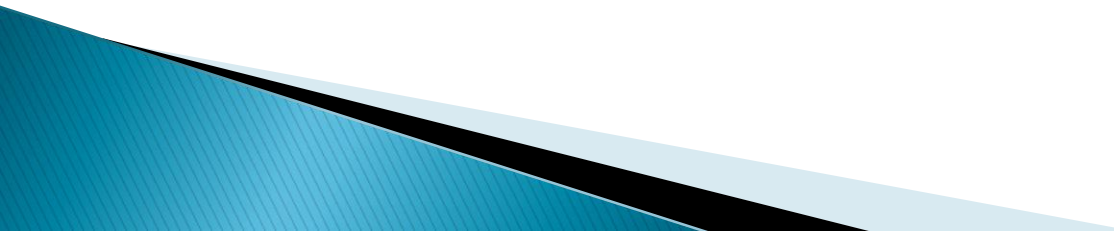
Ejercicio Adicional

- ▶ En la categoría anterior cree una clase Aplicación que le permita automatizar las operaciones de creación de carga de datos.

Ejercicio Adicional

- ▶ En base al ejercicio de las Figuras:
 - Cree la clase Triangulo Rectángulo
 - Permitir que los Triángulos Rectángulos contengan una colección de triángulos semejantes.

Tomando como criterio:

- Son semejantes si tienen 2 catetos proporcionales.
 - Son semejantes si tienen un cateto y la hipotenusa proporcionales.
 - ▶ Permitir agregar un triangulo en la colección solo si es semejante.
 - ▶ Informar cuantos triángulos semejantes son más grandes que el triángulo actual.
- 

Clase Triángulo Rectángulo

Triangulo subclass: #TrianguloRectangulo
instanceVariableNames: 'hipotenusa semejantes'
classVariableNames: "
poolDictionaries: "
category: 'Figuras'
"Metodos Propios"

hipotenusa
^hipotenusa.

initialize
super initialize
semejantes := OrderedCollection new.

hipotenusa: lahipotenusa
hipotenusa:= lahipotenusa.

agregarTringuloRectangulo: trianguloR
|prop propR res|
res:= ''.
prop:= (self catetoA / self catetoB) asFloat.
propR:= (tringuloR catetoA / trianguloR catetoB) asFloat.
(prop=propR) ifTrue:[res:='semejante agregado'].

prop:= (self catetoA / self hipotenusa) asFloat.
propR:= (tringuloR catetoA / trianguloR hipotenusa) asFloat.
(prop=propR) ifTrue:[res:='semejante agregado'].

prop:= (self catetoB / self hipotenusa) asFloat.
propR:= (tringuloR catetoB / trianguloR hipotenusa) asFloat.
(prop=propR) ifTrue:[res:='semejante agregado'].

(res = '') ifTrue: [^'no es semejante']
ifFalse:[semejantes add: trianguloR. ^res]

Clase Triángulo Rectángulo(2)

semejantesMayores

|cont|

cont:=0.

semejantes do[:x|(self area < x area)

ifTrue:[cont:=cont+1.]

]

^cont.

Otra forma:

semejantesMayores

^(semejantes select: [:x |self area < x area]) size.