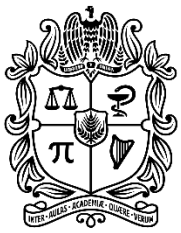


UNIVERSIDAD
NACIONAL
DE COLOMBIA

Actividad 3

programación Orientada a objetos

Grupo 4

Estudiante

Andres Felipe Devia Orrego

Docente

Walter Arboleda

Medellin

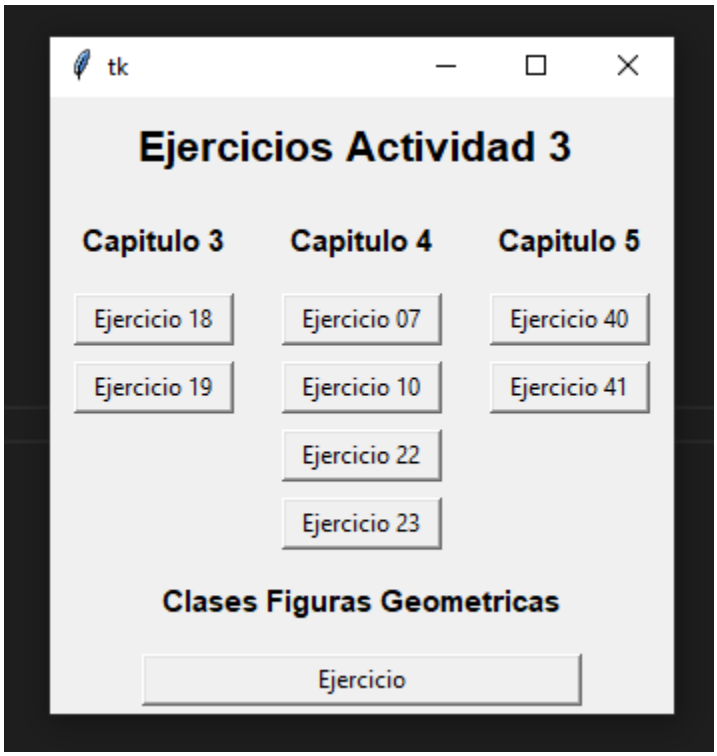
2023

CONTENIDO

INTRODUCCIÓN

Los ejercicios fueron desarrollados en **Python** y divididos en una clase cada ejercicio, se mostrara la interfaz de cada ejercicio, junto con su clase, su diagrama de clase y su resultado.

La interfaz grafica base de la actividad es la siguiente:



Donde cada ejercicio será explicado a continuación:

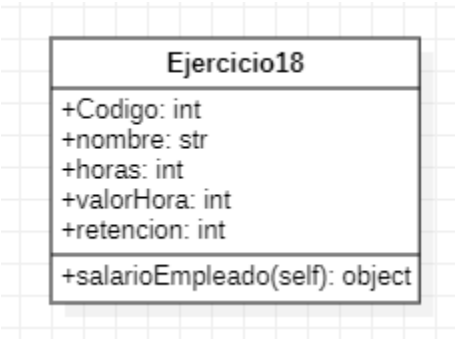
CAPÍTULO 3

Ejercicio 18

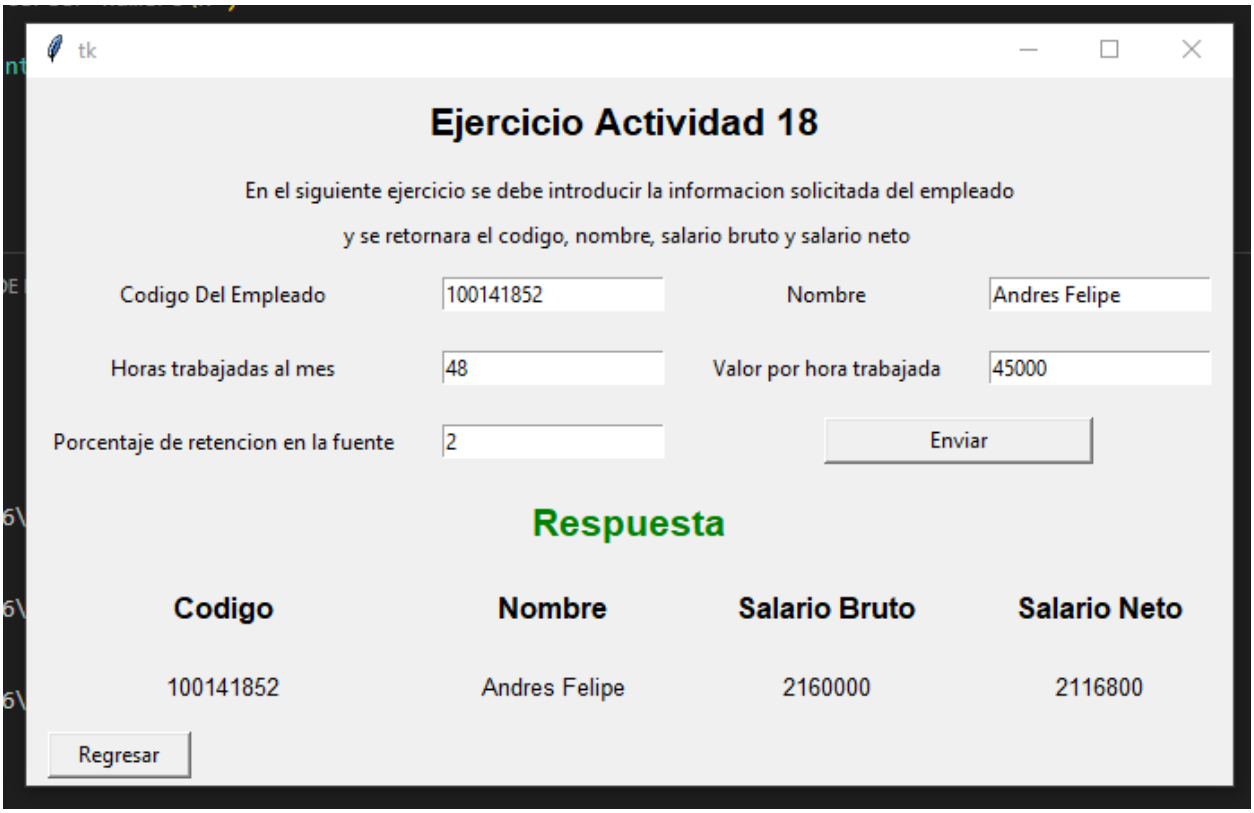
Clase

```
class ejercicio_18():
    def __init__(self,Codigo,nombre,horas,valorHora,retencion):
        self.Codigo=Codigo
        self.nombre=nombre
        self.horas=horas
        self.valorHora=valorHora
        self.retencion=retencion
    def salarioEmpleado(self):
        codigo=self.Codigo
        nombre=self.nombre
        horas=self.horas
        valorHora=self.valorHora
        retencion=self.retencion
        salarioBruto=valorHora*horas
        porcentaje=retencion/100
        salarioNeto=salarioBruto-(salarioBruto*porcentaje)
        return {
            "salarioNeto":salarioNeto,
            "salarioBruto":salarioBruto
        }
```

Diagrama de clase



Interfaz Grafica



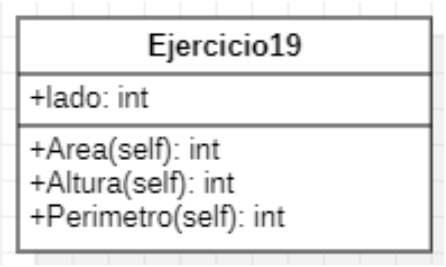
Como se puede observar el código realiza el calculo del salario bruto y realiza el descuento del salario neto.

Ejercicio 19

Clase

```python
class ejercicio_19():
    def __init__(self,lado):
        self.lado=lado
    def Area(self):
        lado=self.lado
        Area=(math.sqrt(3)*(lado*lado))/4
        Area=round(Area,2)
        return Area
    def Altura(self):
        lado=self.lado
        Altura=(math.sqrt(3)*lado)/2
        Altura=round(Altura,2)
        return Altura
    def Perimetro(self):
        lado=self.lado
        Perimetro=3*lado
        Perimetro=round(Perimetro,2)
        return Perimetro
```

Diagrama de clase



Interfaz Grafica

Como se puede observar el código realiza el calculo de la altura, el área, y el perímetro, teniendo como dato de entrada, el lado de un triángulo equilátero.

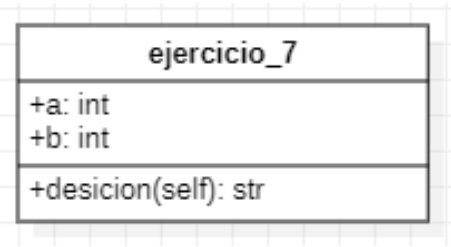
CAPITULO 4

Ejercicio 07

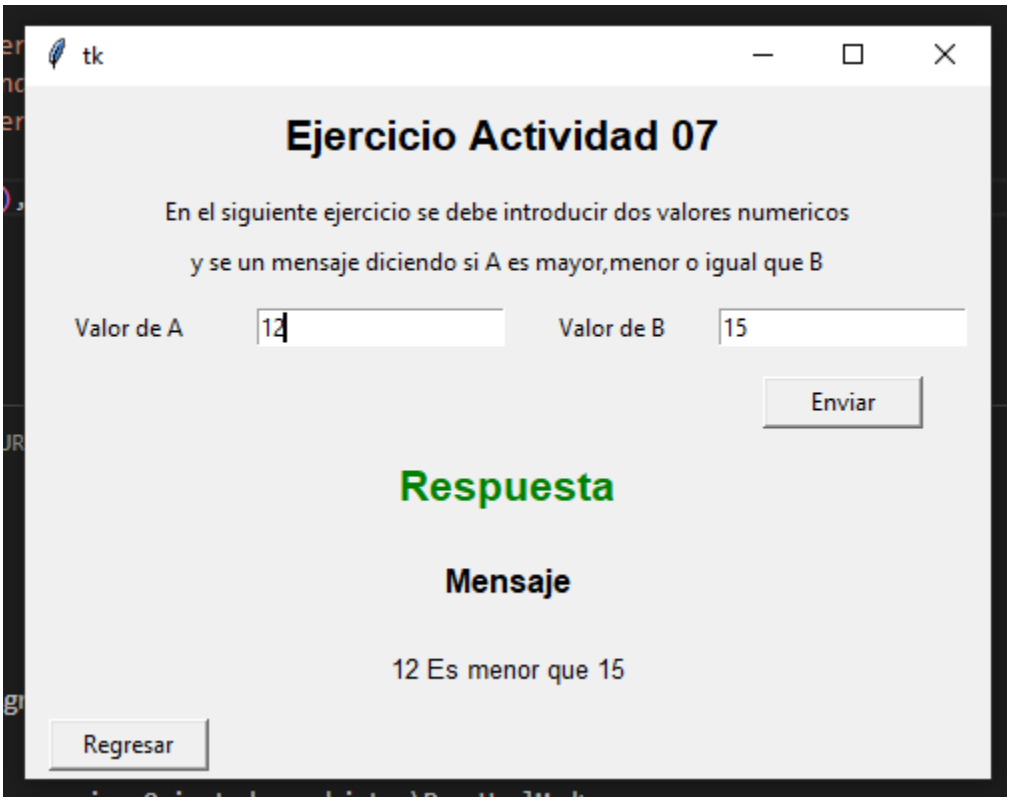
Clase

```python
class ejercicio_7():
    def __init__(self,a,b):
        self.a=a
        self.b=b
    def desicion(self):
        a=self.a
        b=self.b
        mensaje=""
        if a > b:
            mensaje=str(a)+" Es mayor que "+str(b)
        elif a < b:
            mensaje=str(a)+" Es menor que "+str(b)
        else:
            mensaje=str(a)+" Es igual que "+str(b)
        return mensaje
```

Diagrama de clase



Interfaz Grafica



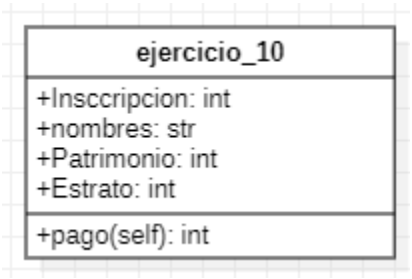
Como se puede observar el código realiza la operación para poder indicar si el valor de A, es menor, igual o mayor, que el valor de B.

Ejercicio 10

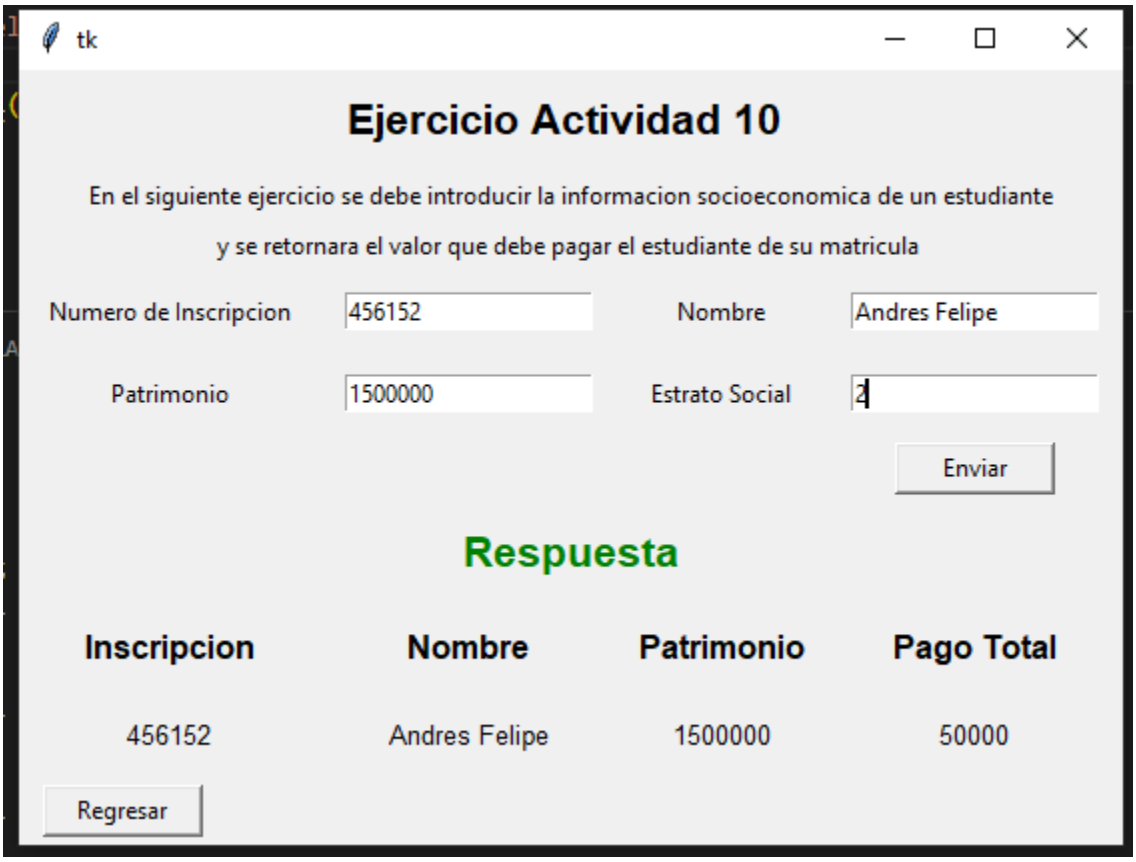
Clase

```python
class ejercicio_10():
    def __init__(self,Inscripcion,nombres,Patrimonio,Estrato,):
        self.Inscripcion=Inscripcion
        self.nombres=nombres
        self.Patrimonio=Patrimonio
        self.Estrato=Estrato
    def pago(self):
        Inscripcion=self.Inscripcion
        nombres=self.nombres
        Patrimonio=self.Patrimonio
        Estrato=self.Estrato

        valor=50000
        if Patrimonio>2000000 and Estrato>3:
            valor=(Patrimonio*0.03)+valor

        return valor
```

Diagrama de clase



Interfaz Grafica



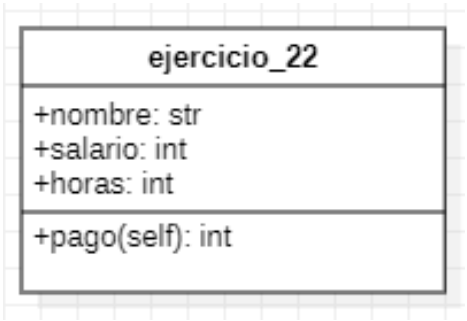
Como se puede observar el código realiza la operación para calcular el pago total de la matricula de un estudiante

Ejercicio 22

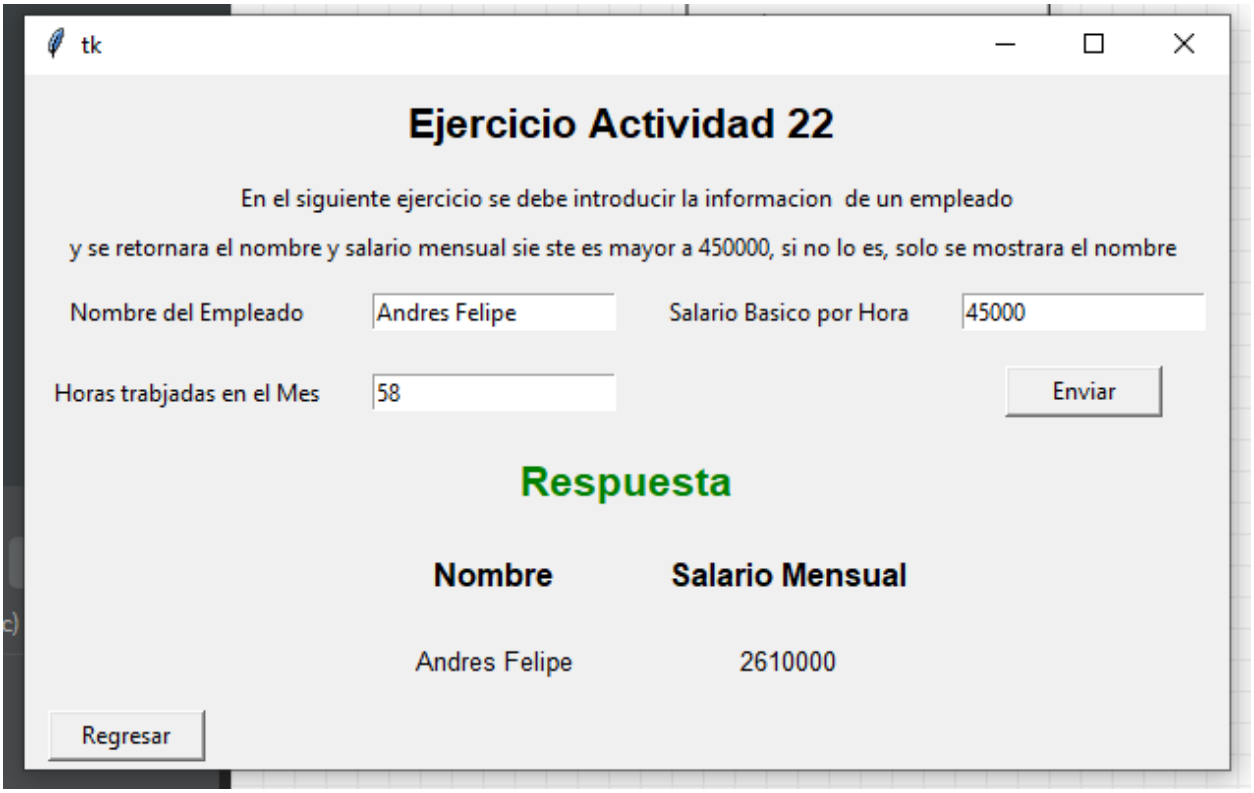
Clase

```
77  class ejercicio_22():
78      def __init__(self,nombre,salario,horas):
79          self.nombre=nombre
80          self.salario=salario
81          self.horas=horas
82      def pago(self):
83          mensual=int(self.salario)*int(self.horas)
84          if mensual>450000 :
85              return mensual
```

Diagrama de clase



Interfaz Grafica



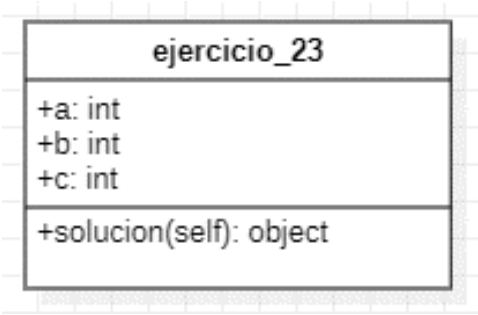
Como se puede observar el código realiza la operación para calcular el salario mensual de un empleado.

Ejercicio 23

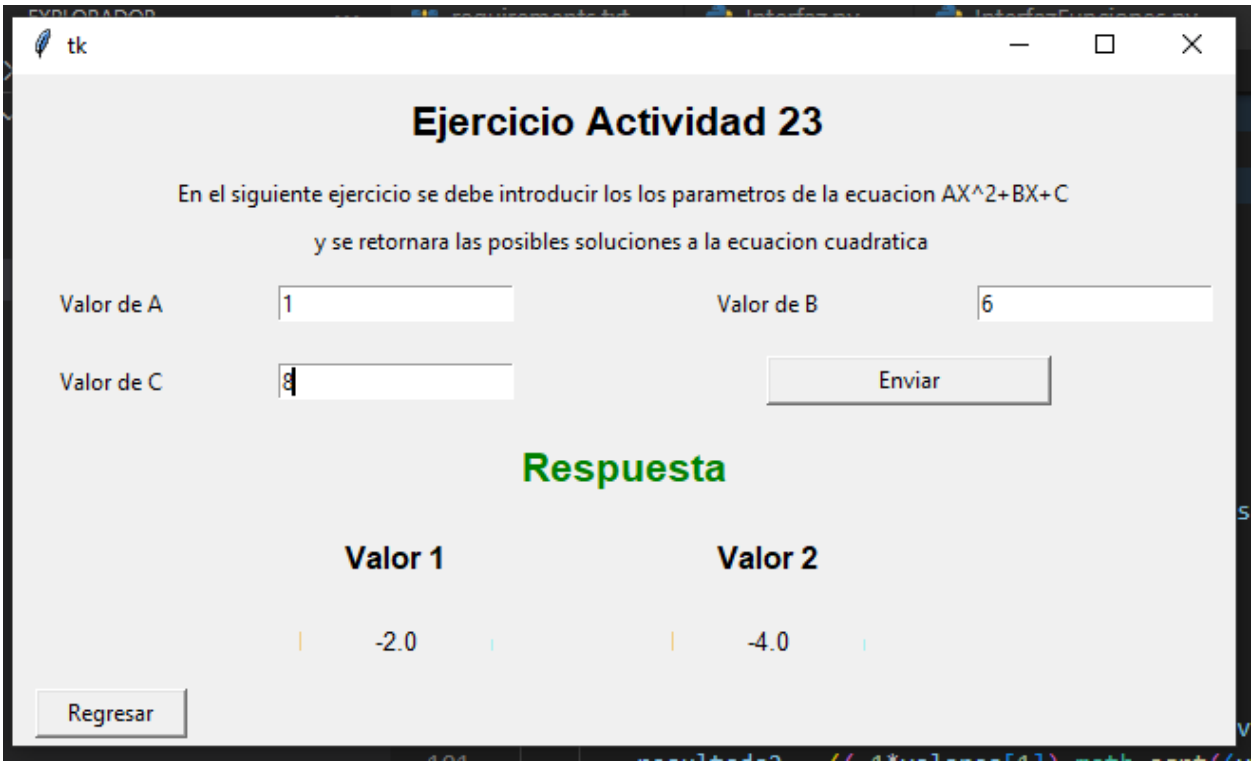
Clase

```
87  class ejercicio_23():
88      def __init__(self,a,b,c):
89          self.a = a
90          self.b = b
91          self.c = c
92      def solucion(self):
93          valores = [self.a,self.b,self.c]
94          disc=(valores[1]**2)-(4*valores[0]*valores[2])
95          if disc<0:
96              return{
97              "valor1":"No tiene Solucion",
98              "valor2":"No tiene Solucion"
99          }
100         resultado1 = ((-1*valores[1])+math.sqrt((valores[1]**2)-(4*valores[0]*valores[2])))/(2*valores[0])
101         resultado2 = ((-1*valores[1])-math.sqrt((valores[1]**2)-(4*valores[0]*valores[2])))/(2*valores[0])
102         return{
103             "valor1":resultado1,
104             "valor2":resultado2
105         }
106
```

Diagrama de clase



Interfaz Grafica



Como se puede observar el código realiza las operaciones necesarias para hallar los valores de la ecuación cuadratica

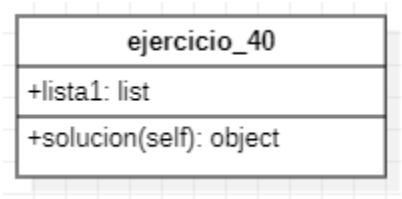
CAPITULO 5

Ejercicio 40

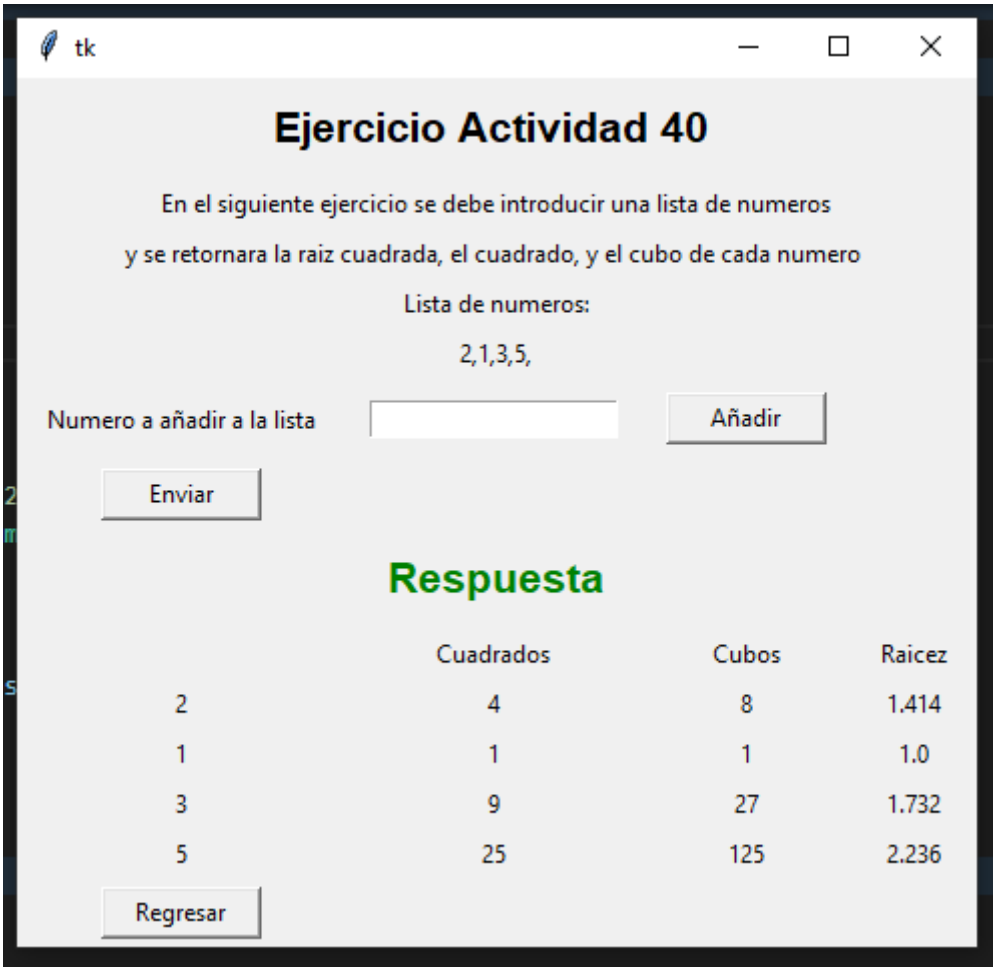
Clase

```
107  class ejercicio_40():
108      def __init__(self,lista1):
109          self.lista1 = lista1
110      def solucion(self):
111          lista1=self.lista1
112          cuadrados=[]
113          raicez=[]
114          cubos=[]
115          for i in lista1:
116              cuadrados.append(i**2)
117              raicez.append(round(math.sqrt(i),3))
118              cubos.append(i**3)
119          print(cubos)
120          return{
121              "cuadrados":cuadrados,
122              "raicez":raicez,
123              "cubos":cubos
124          }
125
```

Diagrama de clase



Interfaz Grafica



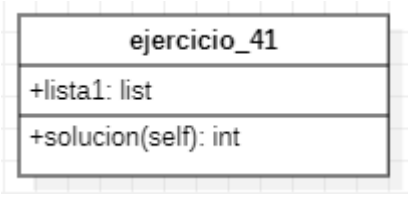
Como se puede observar el código realiza los cuadrados, cubos, y raíces de la lista de números que introduzcamos.

Ejercicio 41

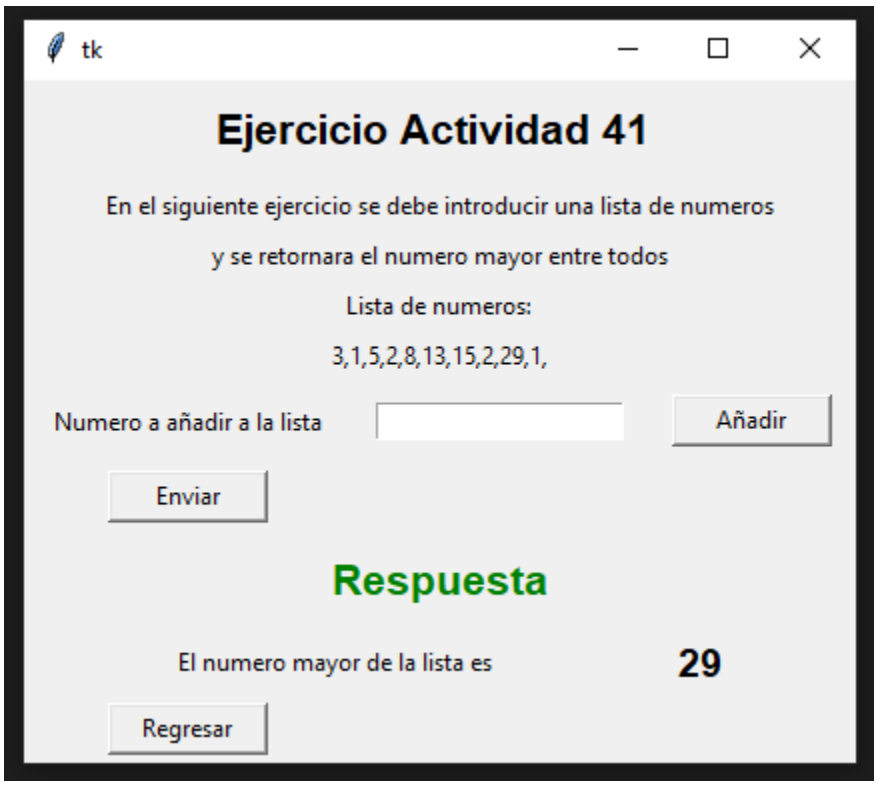
Clase

```
126  class ejercicio_41():
127      def __init__(self,lista1):
128          self.lista1 = lista1
129      def solucion(self):
130          lista1=self.lista1
131          mayor=0
132          for i in lista1:
133              if i>mayor:
134                  mayor=i
135          return mayor
136
```

Diagrama de clase



Interfaz Grafica



Como se puede observar el código acumula una lista de números y retorna el mayor de todos estos.

CLASES FIGURAS GEOMÉTRICAS

Para este conjunto de ejercicios mostrar su interfaz de manera individual, pero su diagrama se representara a continuación:

Interfaz General

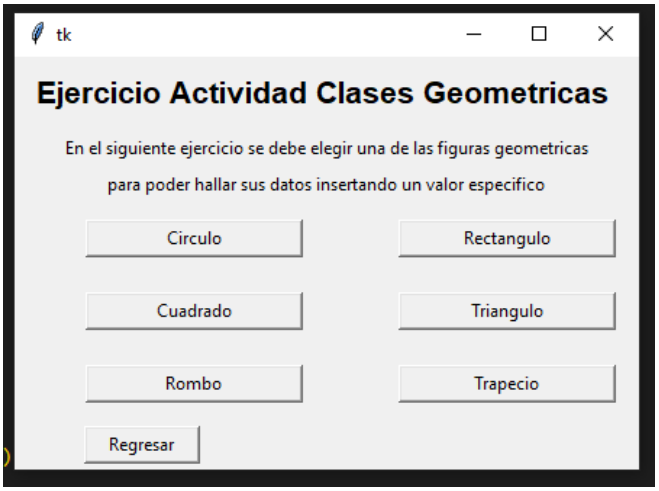
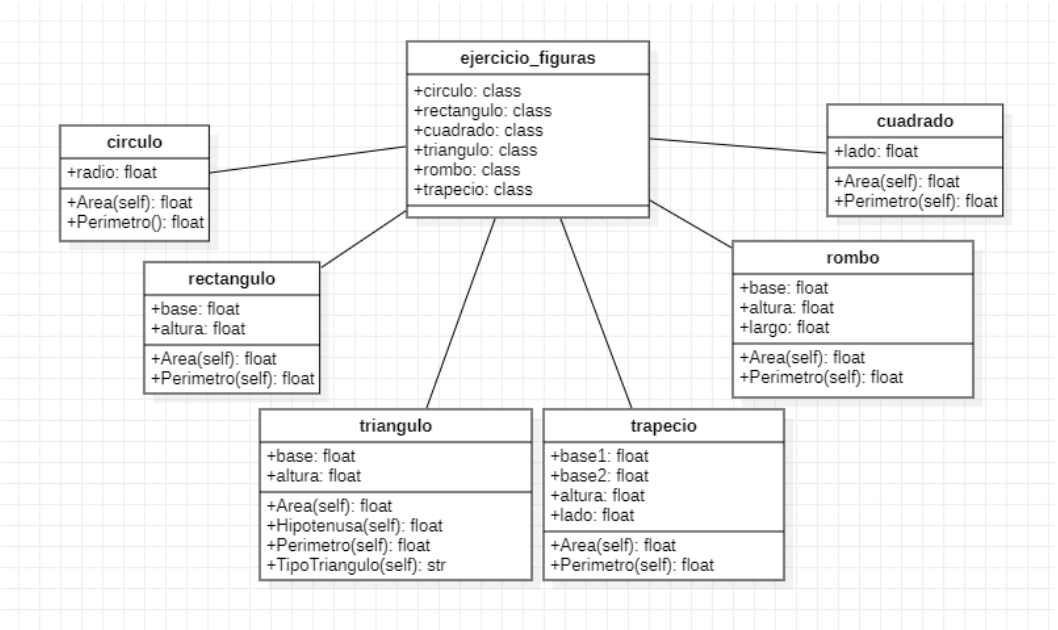
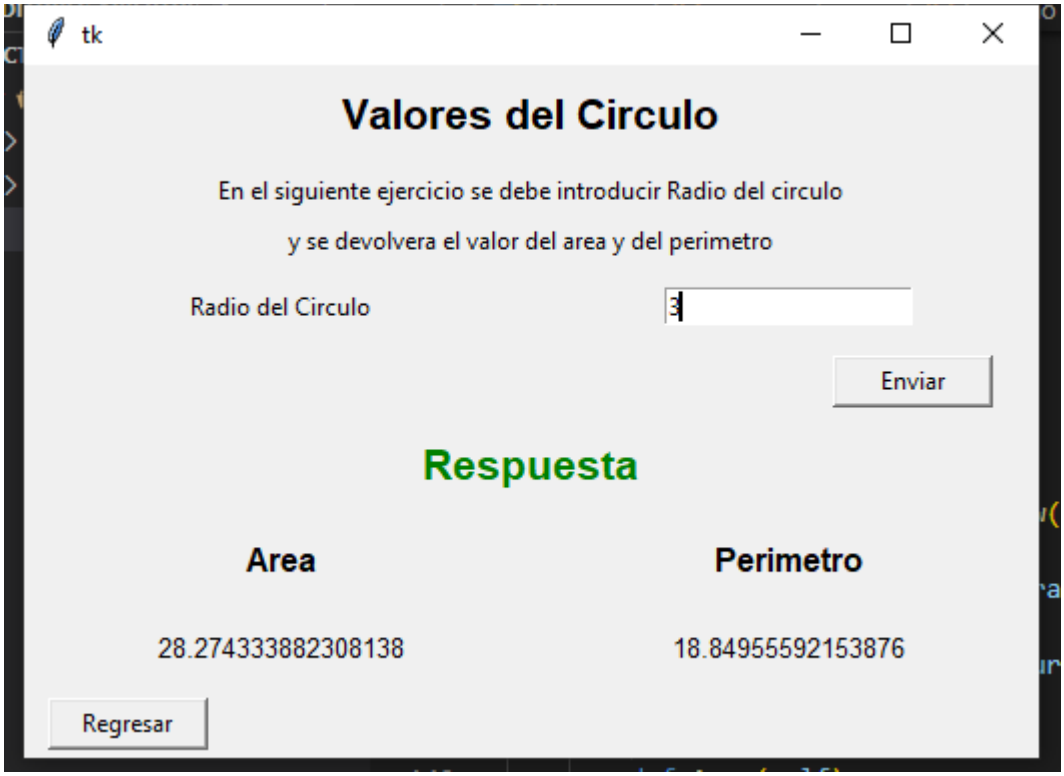


Diagrama de Clases



Circulo

```
138     class circulo():
139         def __init__(self,radio):
140             self.radio = radio
141         def Area(self):
142             return math.pi*math.pow(self.radio,2)
143         def Perimetro(self):
144             return 2*math.pi*self.radio
```

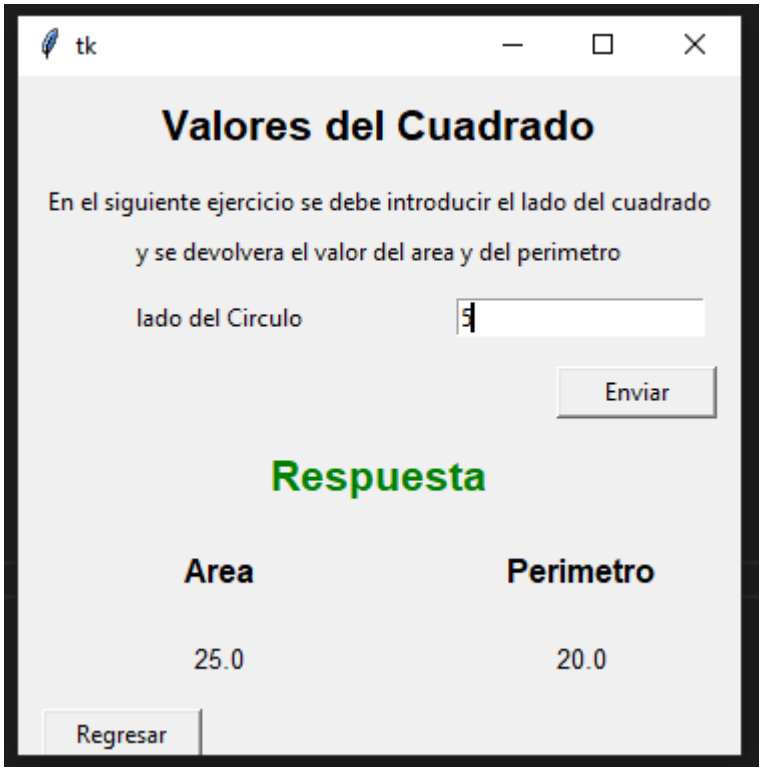


Rectángulo

```
145     class rectangulo():
146         def __init__(self,base,altura):
147             self.base = base
148             self.altura = altura
149         def Area(self):
150             return self.base*self.altura
151         def Perimetro(self):
152             return (2*self.base)+(2*self.altura)
```

Cuadrado

```
153     class cuadrado():
154         def __init__(self,lado):
155             self.lado = lado
156         def Area(self):
157             return self.lado**2
158         def Perimetro(self):
159             return 4*self.lado
```



Triangulo

```python
class triangulo():
    def __init__(self,base,altura):
        self.base = base
        self.altura = altura
    def Area(self):
        return (self.altura*self.base)/2
    def Hipotenusa(self):
        return math.sqrt((self.altura**2)+(self.base**2))
    def Perimetro(self):
        return self.base+self.altura+self.Hipotenusa()
    def TipoTriangulo(self):
        base=self.base
        altura=self.altura
        hipotenusa=self.Hipotenusa()
        if base==altura and base==hipotenusa and altura==hipotenusa:
            return "Equilatero"
        elif base!=altura and base!=hipotenusa and altura!=hipotenusa:
            return "Escaleno"
        else:
            return "Isoceles"
```

Rombo

```
180     class rombo():
181         def __init__(self,lado,ancho,largo):
182             self.lado = lado
183             self.ancho = ancho
184             self.largo = largo
185         def Area(self):
186             return (self.largo*self.ancho)/2
187         def Perimetro(self):
188             return 4*self.lado
```

Trapecio

```
189     class trapecio():
190         def __init__(self,lado,altura,base1,base2):
191             self.altura = altura
192             self.lado = lado
193             self.base1 = base1
194             self.base2 = base2
195         def Area(self):
196             return ((self.base1+self.base2)/2)*self.altura
197         def Perimetro(self):
198             return self.lado+self.lado+self.base1+self.base2
```