# Práctica 2 Apache Thrift

Para la práctica he hecho dos calculadoras, las he dejado por separado para que no sea tan lioso.

- La primera es la calculadora básica que hace la suma, la resta, la multiplicación y la división.

- La segunda calculadora es una calculadora de fracciones, le metemos dos fracciones y podemos sumarlas, restarlas, multiplicarlas o dividirlas.

## ¿Cómo he creado las calculadoras?

### - calculadora.thrift

Primero he hecho el .thrift de ambas calculadoras cada uno en una carpeta:

- Este es el de la calculadora básica:

```
service Calculadora{
   void ping(),
   i32 suma(1:i32 num1, 2:i32 num2),
   i32 resta(1:i32 num1, 2:i32 num2),
   i32 multiplicacion(1:i32 num1, 2:i32 num2),
   i32 division(1:i32 num1, 2:i32 num2),
}
```

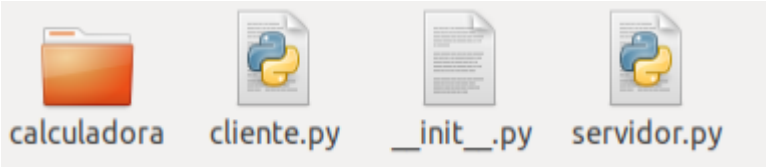
- Este es el de la calculadora de las fracciones:

```
service Calculadora{
   void ping(),
   double suma(1:i32 num1, 2:i32 num2, 3:i32 num3, 4:i32 num4),
   double resta(1:i32 num1, 2:i32 num2, 3:i32 num3, 4:i32 num4),
   double multiplicacion(1:i32 num1, 2:i32 num2, 3:i32 num3, 4:i32 num4),
   double division(1:i32 num1, 2:i32 num2, 3:i32 num3, 4:i32 num4),
}
```

Después de crearlo pongo en la terminal el siguiente comando

```
@clau-virtualbox:~/Escritorio/DSD (copia)/Calculadora_Basica/apa
ca-2-2-codigo/gen-py$ thrift -gen py calculadora.thrift
```

Se me crea una carpeta llamada gen-py donde se crean muchos archivos menos el cliente y el servidor .py, que los tengo que incluir yo:



- **servidor.py**

Aquí es donde declaro que hace cada método es decir como hace la suma la resta etc.

- En la calculadora básica el server me queda tal que así:

```
import glob
import sys

from calculadora import Calculadora

from thrift.transport import TSocket
from thrift.transport import TTransport
from thrift.protocol import TBinaryProtocol
from thrift.server import TServer

import logging

logging.basicConfig(level=logging.DEBUG)


class CalculadoraHandler:
    def __init__(self):
        self.log = {}

    def ping(self):
        print("me han hecho ping()")

    def suma(self, n1, n2):
        print("sumando " + str(n1) + " con " + str(n2))
        return n1 + n2

    def resta(self, n1, n2):
        print("restando " + str(n1) + " con " + str(n2))
        return n1 - n2

    def multiplicacion(self, n1, n2):
        print("multiplicamos " + str(n1) + " con " + str(n2))
        return n1 * n2

    def division(self, n1, n2):
        print("dividimos " + str(n1) + " entre " + str(n2))
        return n1 / n2


if __name__ == "__main__":
    handler = CalculadoraHandler()
    processor = Calculadora.Processor(handler)
    transport = TSocket.TServerSocket(host="127.0.0.1", port=9090)
    tfactory = TTransport.TBufferedTransportFactory()
    pfactory = TBinaryProtocol.TBinaryProtocolFactory()
 server = TServer.TSimpleServer(processor, transport, tfactory, pfactory)

 print("iniciando servidor...")
 server.serve()
 print("fin")
```

- En la calculadora de fracciones el server me queda tal que así:

He tenido que hacer dos funciones a parte que son el mcd y el mcm, el máximo común divisor lo necesito para hacer el mínimo común múltiplo, este último se utiliza para calcular la suma y resta de fracciones.

```
import glob
import sys

from calculadora import Calculadora

from thrift.transport import TSocket
from thrift.transport import TTransport
from thrift.protocol import TBinaryProtocol
from thrift.server import TServer

import logging

logging.basicConfig(level=logging.DEBUG)


class CalculadoraHandler:
    def __init__(self):
        self.log = {}

    def ping(self):
        print("me han hecho ping()")

    def mcd(self, x, y):
        while y != 0:
                temporal = y
                aux = x / y
                y = x - (aux * y)
                x = temporal
        return x

    def mcm(self, x, y):
        return (x * y) / self.mcd(x, y)


    def suma(self, n1, n2, n3, n4):
        print("sumando " + str(n1) + "/" + str(n2) + " con " + str(n3) + "/" + str(n4))
        m1 = self.mcm(n2, n4) / n2 * n1
        m2 = self.mcm(n2, n4) / n4 * n3
        m = m1 + m2
        d = self.mcm(n2, n4)
        resultado = float(m) / float(d)
        print("Fraccion Final: " + str(m) + "/" + str(d))
        return float(resultado)


 def resta(self, n1, n2, n3, n4):
     print("restando " + str(n1) + "/" + str(n2) + " con " + str(n3) + "/" + str(n4))
     m1 = mcm(n2, n4) / n2 * n1
     m2 = mcm(n2, n4) / n4 * n3
     m = m1 - m2
     d = self.mcm(n2, n4)
     resultado = float(m) / float(d)
     print("Fraccion Final: " + str(m) + "/" + str(d))
     return float(resultado)


 def multiplicacion(self, n1, n2, n3, n4):
     print("multiplicamos " + str(n1) + "/" + str(n2) + " con " + str(n3) + "/" + str(n4))
     numeradores = n1 * n3
     denominadores = n2 * n4
     resultado =  float(numeradores) / float(denominadores)
     print("Fraccion Final: " + str(numeradores) + "/" + str(denominadores))
     return float(resultado)
```

```
    def division(self, n1, n2, n3, n4):
        print("dividimos " + str(n1) + "/" + str(n2) + " entre " + str(n3) + "/" + str(n4))
        numeradores = n1 * n4
        denominadores = n2 * n3
        resultado =  float(numeradores) / float(denominadores)
        print("Fraccion Final: " + str(numeradores) + "/" + str(denominadores))
        return float(resultado)


if __name__ == "__main__":
    handler = CalculadoraHandler()
    processor = Calculadora.Processor(handler)
    transport = TSocket.TServerSocket(host="127.0.0.1", port=9090)
    tfactory = TTransport.TBufferedTransportFactory()
    pfactory = TBinaryProtocol.TBinaryProtocolFactory()

    server = TServer.TSimpleServer(processor, transport, tfactory, pfactory)

    print("iniciando servidor...")
    server.serve()
    print("fin")
```

- **cliente.py**

Aquí modificó la forma en la que acepta las entradas del cliente desde teclado y género los procedimientos remotos(add, sub, mul, div)  invocando resultados para el cliente.

- En la calculadora básica el cliente me queda tal que así:

```
from calculadora import Calculadora

from thrift import Thrift
from thrift.transport import TSocket
from thrift.transport import TTransport
from thrift.protocol import TBinaryProtocol

transport = TSocket.TSocket("localhost", 9090)
transport = TTransport.TBufferedTransport(transport)
protocol = TBinaryProtocol.TBinaryProtocol(transport)

client = Calculadora.Client(protocol)

transport.open()

print("hacemos ping al server")
client.ping()

print(" \n*****CALCULADORA BASICA***** \nRecuerda \n 1.Sumar \n 2.Restar \n 3.Multiplicar \n 4.Dividir \n")
print("Ingresa el primer numero")
nu1 = int(input())
print("Ingresa el segundo numero")
nu2 = int(input())
print("Ingresa operador")
op = int(input())

if (op==1):
        resultado = client.suma(nu1, nu2)
        print(" \nSeleccionaste 1.Sumar \n " + str(nu1) + " + " + str(nu2) + " = " + str(resultado))
elif (op==2):
        resultado = client.resta(nu1, nu2)
        print(" \nSeleccionaste 2.Restar \n " + str(nu1) + " - " + str(nu2) + " = " + str(resultado))
elif (op==3):
        resultado = client.multiplicacion(nu1, nu2)
        print(" \nSeleccionaste 3.Multiplicar \n "+ str(nu1) + " * " + str(nu2) + " = " + str(resultado))
elif (op==4):
        resultado = client.division(nu1, nu2)
        print(" \nSeleccionaste 4.Dividir \n " + str(nu1) + " / " + str(nu2) + " = " + str(resultado))

transport.close()
```

- En la calculadora de fracciones el cliente me queda tal que así:

```
from calculadora import Calculadora

from thrift import Thrift
from thrift.transport import TSocket
from thrift.transport import TTransport
from thrift.protocol import TBinaryProtocol

transport = TSocket.TSocket("localhost", 9090)
transport = TTransport.TBufferedTransport(transport)
protocol = TBinaryProtocol.TBinaryProtocol(transport)

client = Calculadora.Client(protocol)

transport.open()

print("hacemos ping al server")
client.ping()

print(" \n*****CALCULADORA FRACCIONES***** \nRecuerda \n 1.Sumar \n 2.Restar \n 3.Multiplicar \n 4.Dividir \n")
print("Ingresa el numerador de la primera fraccion")
nu1 = float(input())
print("Ingresa el denominador de la primera fraccion")
nu2 = float(input())
print("Ingresa el numerador de la segunda fraccion")
nu3 = float(input())
print("Ingresa el denominador de la segunda fraccion")
nu4 = float(input())
print("Ingresa operador")
op = int(input())

if (op==1):
        resultado = client.suma(nu1, nu2, nu3, nu4)
        print(" \nSeleccionaste 1.Sumar \n " + str(nu1) + "/" + str(nu2) + " + " + str(nu3) + "/" + str(nu4) + " = " + str(resultado))
elif (op==2):
        resultado = client.resta(nu1, nu2, nu3, nu4)
        print(" \nSeleccionaste 2.Restar \n " + str(nu1) + "/" + str(nu2) + " - " + str(nu3) + "/" + str(nu4) + " = " + str(resultado))
elif (op==3):
        resultado = client.multiplicacion(nu1, nu2, nu3, nu4)
        print(" \nSeleccionaste 3.Multiplicar \n "+ str(nu1) + "/" + str(nu2) + " * " + str(nu3) + "/" + str(nu4) + " = " + str(resultado))
elif (op==4):
        resultado = client.division(nu1, nu2, nu3, nu4)
        print(" \nSeleccionaste 4.Dividir \n " + str(nu1) + "/" + str(nu2) + " / " + str(nu3) + "/" + str(nu4) + " = " + str(resultado))

transport.close()
```

## - Funcionamiento

Para comprobar cómo funcionan hacemos el siguiente comando en ambas lo mismo:

- Lanzamos el servidor, esto en la básica:

```
ractica-2-2-codigo/gen-py$ python servidor.py
iniciando servidor...
me han hecho ping()
```

- Lanzamos el cliente, en la básica:

```
ractica-2-2-codigo/gen-py$ python cliente.py
hacemos ping al server

*****CALCULADORA BASICA*****
Recuerda
 1.Sumar
 2.Restar
 3.Multiplicar
 4.Dividir

Ingresa el primer numero
```

- Y ahora pruebo una multiplicación por ejemplo:

```
Ingresa el primer numero
3
Ingresa el segundo numero
5
Ingresa operador
3

Seleccionaste 3.Multiplicar
 3 * 5 = 15
```

```
multiplicamos 3 con 5
```

Ahora probamos en la de fracciones:

- Primero lanzamos el servidor:

```
-py$ python servidor.py
iniciando servidor...
me han hecho ping()
```

- Segundo lanzamos el cliente

```
python cliente.py
hacemos ping al server

*****CALCULADORA FRACCIONES*****
Recuerda
 1.Sumar
 2.Restar
 3.Multiplicar
 4.Dividir

Ingresa el numerador de la primera fraccion
```

- Y a continuación hacemos una suma por ejemplo:

```
Ingresa el numerador de la primera fraccion
1
Ingresa el denominador de la primera fraccion
2
Ingresa el numerador de la segunda fraccion
1
Ingresa el denominador de la segunda fraccion
3
Ingresa operador
1

Seleccionaste 1.Sumar
 1.0/2.0 + 1.0/3.0 = 0.833333333333
```

```
sumando 1/2 con 1/3
Fraccion Final: 5/6
```

## Nota:

Al enviar los archivos fuera del zip de cada calculadora los he tenido que renombrar para identificandolos con calculadora_basica y calculadora_fracciones.
Si se cogen esos seguramente no funcionen ya que tienen nombre distinto a los que he usado para hacer las calculadoras..