Practica 2.2

Apache Thrift[1]

1 Autor: Francisco de Asís Carrasco Conde

Como he hecho la aplicación

Primero rellenamos el .thrift con las funciones adecuadas, en este caso las funciones son las operaciones que va a realizar el server.

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

    list<double> sumaV(1: list<double> v1 , 2: list<double> v2 ),
    list<double> restaV(1: list<double> v1 , 2: list<double> v2 ),
    list<double> multiplicacionV(1: list<double> v1 , 2: list<double> v2 ),

    list<list<double>> sumaM(1: list<list<double>> m1 , 2: list<list<double>> m2 ),
    list<list<double>> restaM(1: list<list<double>> m1 , 2: list<list<double>> m2 ),
    list<list<double>> multiplicacionM(1: list<list<double>> m1 , 2: list<list<double>> m2 )
}
```

Luego ejecutamos en la barra de comandos:

```
# thrift -gen py calculadora.thrift
```

Una vez ejecutado el comando se nos creará una carpeta con unos archivos, estos no los vamos a tocar. Ahora creamos fuera de la carpeta generada, server.py y client.py, que son los archivos que vamos a tocar.

Una vez echo eso, copiamos parte del código de las transparencias para realizar la conexión de manera correcta y el inicio del servidor respectivamente. Despues de esto solo queda rellenar ambos archivos, en el del servidor pondremos las funciones con los resultados que devuelven, tal que:

(Operaciones Basicas)

```
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 (" multiplicando "+ str ( n1 )+ " con "+ str( n2 ))
    return n1 * n2
def division ( self , n1 , n2 ):
    print (" dividiendo "+ str ( n1 )+ " con "+ str( n2 ))
    return n1 / n2
```

(Vectores)

```
def sumaV ( self, v1 , v2 ):
    resul=[v1[0]+v2[0],v1[1]+v2[1],v1[2]+v2[2]]
    return resul
def restaV ( self, v1 , v2 ):
    resul=[v1[0]-v2[0],v1[1]-v2[1],v1[2]-v2[2]]
    return resul
def multiplicacionV ( self, v1 , v2 ):
    a=v1[1]*v2[2]-v1[2]*v2[1]
    b=-v1[0]*v2[2]-v1[2]*v2[0]
    c=v1[0]*v2[1]-v1[1]*v2[0]
    resul=[a,b,c]
    return resul
```

(Matrices)

```
    def sumaM ( self, m1 , m2 ):
        resul=[
            [0,0,0],
            [0,0,0],
            [0,0,0]
        ]
        for i in range(3):
            for j in range(3):
                resul[i][j]=m1[i][j]+m2[i][j]

        return resul
    def restaM ( self, m1 , m2 ):
        resul=[
            [0,0,0],
            [0,0,0],
            [0,0,0]
        ]
        for i in range(3):
            for j in range(3):
                resul[i][j]=m1[i][j]-m2[i][j]

        return resul
    def multiplicacionM ( self, m1 , m2 ):
        resul=[
            [0,0,0],
            [0,0,0],
            [0,0,0]
        ]

        for i in range(3):
            for j in range(3):
                for k in range(3):
                    resul[i][j]+=m1[i][k]*m2[k][j]

        return resul
```

La parte del código cliente quedaría así(solo incluyo la parte rellenada por mi, el resto del código lo obtuve de las diapositivas de la práctica):

(Funciones auxiliares)

```
def toVector(cadena):
        aux=cadena.find(',')
        n1=float(cadena[0:aux])

        cadena=cadena[aux+1:]

        aux=cadena.find(',')
        n2=float(cadena[0:aux])

        cadena=cadena[aux+1:]

        n3=float(cadena)

        resul=[n1,n2,n3]

        return resul

def printVec(vector):
    print(f"{vector[0]} {vector[1]} {vector[2]}")
```

(Acceso a los modos de la calculadora)

```
modo=input("Con que datos piensa operar? Numeros normales : 0 | Vectores : 1 | Matrices : 2  --> ")
if modo!='0' and modo!='1' and modo!='2' :
    raise ValueError("Tipo desconocido")

modo=int(modo)
```

(Operaciones básicas)

```
print("Que operacion desea realizar? + - x /")
signo=input()

if signo!='+' and signo!='-' and signo!='x' and signo!='/':
    raise ValueError("Signo incorrecto")

print("Con que digitos quieres operar?")
num1=input("numero 1: ")
num2=input("numero 2: ")

num1=float(num1)
num2=float(num2)

if signo=='+':
    resultado=client.suma(num1,num2)
elif signo=='-':
    resultado=client.resta(num1,num2)
elif signo=='x':
    resultado=client.multiplicacion(num1,num2)
else:
    if num2==0:
        raise ValueError("No se puede dividir por 0")
    else:
        resultado=client.division(num1,num2)

print(f"{num1}{signo}{num2} = {resultado}")
```

(Vectores)

```
elif modo==1 :
    print("Que operacion desea realizar? + - x")
    signo=input()

    if signo!='+' and signo!='-' and signo!='x':
        raise ValueError("Signo incorrecto")

    print("Con que vectores quieres operar? Inserte los numeros seguidos separados por comas sin espacios")
    chain=input("Tamaño requerido de 3 --> ")

    v1=toVector(chain)

    chain=input("Tamaño requerido de 3 --> ")

    v2=toVector(chain)

    if signo=='+':
        resultado=client.sumaV(v1,v2)
    elif signo=='-':
        resultado=client.restaV(v1,v2)
    else:
        resultado=client.multiplicacionV(v1,v2)

    printVec(v1)
    print(signo)
    printVec(v2)
    print("------- RESULTADO -------")
    printVec(resultado)
```

(Matrices)

```
else:
    print("Que operacion desea realizar? + - x")
    signo=input()

    if signo!='+' and signo!='-' and signo!='x':
        raise ValueError("Signo incorrecto")

    print("Con que matrices quieres operar? Inserte los numeros por filas seguidos separados por comas sin espacios")
    cadena1=input("Primera fila : ")
    cadena2=input("Segunda fila : ")
    cadena3=input("Tercera fila : ")

    m1=[toVector(cadena1),toVector(cadena2),toVector(cadena3)]

    print("----- Segunda matriz ----")

    cadena1=input("Primera fila : ")
    cadena2=input("Segunda fila : ")
    cadena3=input("Tercera fila : ")

    m2=[toVector(cadena1),toVector(cadena2),toVector(cadena3)]

    if signo=='+':
        resultado=client.sumaM(m1,m2)
    elif signo=='-':
        resultado=client.restaM(m1,m2)
    else:
        resultado=client.multiplicacionM(m1,m2)

    for f in m1:
        printVec(f)

    print(signo)

    for f in m2:
        printVec(f)

    print("------- RESULTADO -------")

    for f in resultado:
        printVec(f)
```

Modo de empleo

Ejecutamos primero el servidor y luego el cliente, en la pantalla del cliente se nos preguntará primero que modo queremos usar, después el signo y luego podremos insertar los operandos. Si ponemos cualquier carácter que no sea alguno de los indicados se lanzará un error.

En operaciones básicas:
- No podemos dividir por 0.

En vectores:
- El número de componentes en el vector es fijo (3).

En matrices:
- El tamaño es fijo (3x3)

En todos los modos tenemos que meter los valores por teclado.

Una vez introducimos el signo de operación y los valores a operar, se realiza el cálculo y se cierra el programa.