Trabajo Práctico: Pilas y Colas

Participantes	Guzmán, Hernán Cichello, Alan
Email	hernan23@gmail.com alancichello@gmail.com

JUZGADOS

Descripción de las funciones desarrolladas:

```
def controlarCriticidad(self, tipo):
```
Función que se utiliza para imprimir si los expedientes, tanto “Urgentes” como “Normales”, entraron en zona crítica, es decir, superaron los 50 expedientes.

```
def recibirExpediente(self, expediente):
```
Función que recibe expedientes, y a partir de la prioridad que tienen, los manda a la cola correspondiente, controlando si los mismos entraron en zona crítica o no.

```
def tieneUrgentes (self):
```
Devuelve si el juzgado tiene expedientes urgentes

```
def tieneNormales (self):
```
Devuelve si el juzgado tiene expedientes normales

```
def primerExpedienteATratar (self):
```
Función que retorna cual es el primer expediente que se debería tratar, consultando en un primer momento por los “urgentes”, y luego por los “normales”

```
def tratarExpediente (self):
```
Función que retornar el expediente a tratar y lo saca del juzgado, consulta primero por los que tienen estado “urgente”, en caso que existan, los saca de la cola, caso contrario prosigue de la misma manera con los que tienen estado “normal”

```
def cantidadDeExpedientesNormales (self):
```
Retorna la cantidad de expedientes “normales” a tratar

```
def cantidadDeExpedientesUrgentes (self):
```
Retorna la cantidad de expedientes “urgentes” a tratar

```
def cantidadTotalDeExpedientes (self):
```
Retorna la cantidad total de expedientes en el juzgado, incluyendo tanto los “normales” como “urgentes”

```
def escritico (self):
```
Retorna si alguna cola de expedientes sobrepasa la cantidad crítica

```
def enJuicioUrgentes(self):
```
Retorna los expedientes “urgentes” que se encuentren en “juicio”, en primer lugar se clonan los expedientes anteriormente mencionados en una cola auxiliar. Mientras la cola auxiliar no este vacía, voy sacando los expedientes y pregunto el estado de los mismos, en caso que se encuentren en “juicio”, los voy sumando.

```
def enJuicioNormales(self):
```
Retorna los expedientes “normales” que se encuentren en “juicio”, en primer lugar se clonan los expedientes anteriormente mencionados en una cola auxiliar. Mientras la

cola auxiliar no este vacía, voy sacando los expedientes y pregunto el estado de los mismos, en caso que se encuentren en “juicio”, los voy sumando.

def enJuicio(self):
Retorna la cantidad total de expedientes que se encuentren en “juicio”, sumando tanto aquellos expedientes “normales”, como “urgentes”

def enInvestigacion(self):
Retorna la cantidad total de expedientes que se encuentren en “investigación”, el cual resulta de sumar aquellos que no se encuentren en estado “enjuicio()”

def buscarExpediente(self, numero):
Función que busca un expediente a partir de un número ingresado, en primer lugar se clonan las cola de “normales” y “urgentes”, se declaran dos variables, una hace referencia al expediente en caso de encontrarla, y la otra un booleano, de si lo encontró o no. Luego, se recorre la cola auxiliar correspondiente a “normales”, y mientras no esté vacía y el valor no sea el buscado la voy recorriendo, en caso de encontrar el número buscado, se asigna el número de expediente, a la variable anteriormente creada. Se repite el procedimiento para la cola en estado “urgente”.

def eliminarExpNormal(self, numero):
Función que elimina los expedientes normales, se clona la misma, y se vacía la original, y se recorre la variable auxiliar clonada, mientras esta no esté vacía, y si el primer expediente que se encuentra en dicha cola no es el buscado, lo envió a la cola original, y en caso de que sea el buscado lo elimino de la cola auxiliar directamente. Se cambia el estado del valor booleano previamente definido, en caso de encontrar el expediente buscado.

def eliminarExpUrgente(self, numero):
Función que elimina los expedientes urgentes, ídem procedimiento de la función anterior (eliminarExpNormal)

def eliminarExpediente(self,nroExp):
Función para eliminar el expediente, que llama a las funciones particulares para cada tipo, las cuales fueron definidas anteriormente, las mismas son: eliminarExpNormal y eliminarExpUrgente

def cambiarDeEstado(self,numero):
Función que cambia el estado de los expedientes y los manda a la cola correspondiente, en primer lugar se clonan ambas colas, es decir, las “normales” y “urgentes”, se vacía la cola original de “urgentes”, y recorro la cola auxiliar de urgentes, y en caso de que el número sea diferente al buscado, lo envió a la cola original, caso contrario, el expediente es el buscado, por lo que le seteo la prioridad “normal”, en caso de no encontrar el expediente buscado, repito el mismo procedimiento para las colas normales, y en caso de encontrar el expediente buscado, se le setea la prioridad “urgente”

Tribunales

def establecerJuzgado(self, piso, oficina, Juzgado):
Función que agregar un juzgado a una oficina del edificio

def cantOficinasOcupadas(self, piso):
Función que retorna la cantidad de oficinas ocupadas

def oficinasOcupadas(self, piso, ocupadas):
Función que retorna las oficinas ocupadas, en caso de que un piso no este vació, se incrementa el contador.

def cantidadDeJuzgadosCriticos(self,piso):
Función que retorna la cantidad de juzgados en situación crítica en el piso que recibe por parámetro, se reciben la cantidad de juzgado críticos, y a dicha función se le pasa el array del piso.

def juzgadoMenosRecargado(self):
Función que recorre todo el edificio, con el objetivo de encontrar el juzgado con menos expedientes, recorre piso por piso, y pregunta si la cantidad de expedientes urgentes que tiene dicho piso es menor que la variable previamente inicializada en 100, caso afirmativo, guarda el piso y oficina como aquellos que tienen menos expedientes, y retorna los mismos.

def buscaJuez(self, juez):
Función que busca el juzgado de un juez, se recorre el edificio, los pisos, y en caso de encontrar un piso cuya oficina tenga el nombre del juez, es que encontró el juez buscado, retorna piso y oficina.

def mesaDeEntradas(self, pilaDeExpedientes, juez):
Función que envía una pila de expedientes y el juez a la mesa de entrada, se consulta el piso y oficina del juez que pase por parámetro, adicionalmente, consulto por el juzgado menos ocupado. Recorro la pila de expedientes, y mientras no esté vacía, voy consultando si el juzgado del juez de guardia no está en estado crítico, en caso de no estar en estado crítico, agrego el expediente al juzgado, caso contrario, lo agrego al juzgado con menos expedientes.

def moverExpediente(self, numero, juezOrigen, juezDestino):
Función que mueve los expedientes desde un juezOrigen a un juezDestino, en primer lugar consulto el piso y oficina de ambos jueces, se busca el expediente del juzgado, una vez encontrado se elimina y se envía al juzgado del juezDestino.

Expediente

def __init__(self, numero, fuero = Fuero.Civil, prioridad = Prioridad.Urgente, estado = Estado.Investigacion):
Constructor del expediente

def __repr__(self):
retorna información de un expediente

def getPrioridad(self):
Asigno la prioridad que va a tener el expediente “Urgente” o “Normal”

def getEstado(self):
Devuelve el estado del expediente, “Juicio” o “Investigación”

def setEstado(self,nuevoEstado):
Asigno un estado al expediente

def setPrioridad(self,nuevaPrioridad):
Asigno una prioridad al expediente

def getNumero(self):
Retorna el número de expediente

Colas

```
def __init__(self):
Inicializa una cola nueva

def vaciar(self):
Deja vacía una cola

def encolar(self, elemento):
Agrega un nuevo elemento a la cola

def __repr__(self):
Representa información

def desencolar(self):
Saca el primer elemento de una cola

def top(self):
Primer elemento de una cola

def estaVacia(self):
Devuelve true o false si la cola está vacía

def clonar(self):
Clona una cola determinada

def len(self):
Devuelve la longitud de una cola
```

Pilas

```
def __init__(self):
Inicializa una pila

def empty(self):
Deja una pila vacía

def apilar(self, elemento):
Agrega un elemento a la pila

def __repr__(self):
Devuelve información de una pila

def desapilar(self):
Mientras pila no esté vacía, saca elementos de la misma

def top(self):
Mientras la pila no esté vacía, obtengo el dato del primer elemento de la pila

def isEmpty(self):
Pregunta si la pila está vacía

def clonar(self):
Clona una pila

def len(self):
Devuelve la longitud de una pila
```

TipoEstado

`class Estado(str, Enum):`
Vamos a tener el estado de los expedientes, para este ejercicio en particular, pueden estar en “Juicio”, o “Investigación”. Se implementa en forma de Enum para que solo tome los valores antes descriptos.

TipoFuero

`class Fuero(str, Enum):`
En este caso vamos a definir el tipo de fuero, para este ejercicio en particular, definimos: Civil, Penal, Laboral, Familia, Comercial. Es una clase Enum para que el fuero solo pueda tomar los valores previamente cargados.

TipoPrioridad

`class Prioridad(str, Enum):`
Se define la prioridad de un expediente, ya sea “Normal”, o “Urgente” en forma de Enum para que la prioridad solo pueda tomar los valores previamente cargados.

Aclaración de las funciones mas complejas

Función para buscar los juzgados críticos

El estado crítico es seteado por una cantidad para cada juzgado. Recibe por parámetro el número del piso en el que buscará los juzgados en estado crítico.

```
def cantidadDeJuzgadosCriticos(self,piso):
llamo a la función cantOficinasOcupadas para que me devuelva la cantidad de oficinas
        ocupadas = self.cantOficinasOcupadas(self.edificio[piso])
con el número de oficinas completo devuelvo los juzgados de ese piso
        oficOcupadas = self.oficinasOcupadas(self.edificio[piso], ocupadas)
llamo a la función buscar críticos y le paso los juzgados que me devolvió la anterior
función
        cantidad = self.buscarCriticos(oficOcupadas)
        return cantidad
```

```
FUNCION RECURSIVA

Recibe los juzgados del piso

def buscarCriticos(self, piso):

        juzgado = piso[0]
Si la cantidad de juzgados en el  array es 1
        if(len(piso)==1):
            'Consulto si el juzgado es crítico y retorno 1
            if juzgado.esCritico():
                return 1
            else:
                return 0
        Sino es el último juzgado
       else:
            Consulto si el juzgado es critico
            if juzgado.esCritico():
                Si es crítico retorno 1 y vuelvo a llamar a la función pero sin la
posición 0
                return 1 + self.buscarCriticos(piso[1:])
            else:
                Sino es crítico llamo a la función sin la posición 0
                return self.buscarCriticos(piso[1:])


FUNCION QUE DEVUELVE LA CANTIDAD DE OFICINAS EN EL PISO INDICADO, PISO ES UN ARRAY DE
JUZGADOS
def cantOficinasOcupadas(self, piso):
        contador = 0
        for i in range(0, len(piso)):
              Si es distinta de vacío le sumo uno
            if (piso[i]) != None:
                contador +=1

        return contador



FUNCION QUE DEVUELVE LOS JUZGADOS QUE ESTAN EN EL PISO INDICADO, PISO ES UN ARRAY DE
JUZGADOS Y OCUPADAS LA CANTIDAD DE OFICINAS OCUPADAS
def oficinasOcupadas(self, piso, ocupadas):
         Declaro un array vacío de juzgados
        oficOcupadas = np.empty((ocupadas), Juzgado)
        contador = 0
        Recorro el piso de juzgados y si es distinto de vacío lo guardo en el array
creado
        for i in range(0, len(piso)):
            if (piso[i]) != None:
                oficOcupadas[contador] = piso[i]
                contador += 1
        return oficOcupadas
```

Función Mesa de Entrada:

Esta función recibe un Juez de turno y una pila de expedientes. El problema es que si es juzgado de turno se encuentra en estado critico se deben pasar los expedientes al juzgado menos cargado.

```
'''ENVIO A LA MESA DE ENTRADA UNA PILA DE EXPEDIENTES Y EL JUEZ DE GUARDIA'''
    def mesaDeEntradas(self, pilaDeExpedientes, juez):
        '''CONSULTO EL PISO Y OFICINA DEL JUEZ '''
        piso, oficina = self.buscaJuez(juez)
        '''CONSULTO EL JUZGADO MENOS OCUPADO POR SI FUERA NECESARIO'''
        pisoMenosOcupado , oficinaMenosOcupada = self.juzgadoMenosRecargado()
        '''RECORRO LA PILA DE EXPEDIENTES HASTA QUE ESTE VACIA'''
        while not pilaDeExpedientes.isEmpty():
            '''CONSULTO SI EL JUZGADO DEL JUEZ DE GUARDIA NO ESTA EN ESTADO
CRITICO'''
            if(not self.edificio[piso, oficina].esCritico()):
                '''SINO ESTA EN ESTADO CRITICO AGREGO EL EXPEDIENTE AL JUZGADO'''
                self.edificio[piso,
oficina].recibirExpediente(pilaDeExpedientes.desapilar())

            else:
                '''SI ESTA EN ESTADO CRITICO LO AGREGO AL JUZGADO CON MENOS
EXPEDIENTES'''
                self.edificio[pisoMenosOcupado,
oficinaMenosOcupada].recibirExpediente(pilaDeExpedientes.desapilar())
```

Función Mover Expediente:

Esta función recibe un Juez de origen, un Juez de destino y un numero de expediente que se desea mover. Si el expediente se encuetra se pasa a el juzgado de destino, pero si el expediente no esta en el juzgado se da un aviso.

```
def moverExpediente(self, numero, juezOrigen, juezDestino):
        '''CONSULTO EL PISO Y OFICINA DEL JUEZ DE ORIGEN'''
        pisoOrgien, oficinaOrigen = self.buscaJuez(juezOrigen)
        '''CONSULTO EL PISO Y OFICINA DEL JUEZ DE DESTINO'''
        pisoDest, oficinaDest = self.buscaJuez(juezDestino)
        '''BUSCO EL EXPEDIENTE DEL JUZGADO'''
        exp = self.edificio[pisoOrgien, oficinaOrigen].buscarExpediente(numero)
        '''SI EL EXPEDIENTE ES ENCONTRADO'''
        if(exp != None):
            '''ELIMINO EL EXPEDIENTE EN EL JUZGADO DE ORIGEN'''
            self.edificio[pisoOrgien, oficinaDest].eliminarExpediente(numero)
            '''ENVIO EL EXPEDIENTE A EL JUZGADO DEL JUEZ DE DESTINO'''
            self.edificio[pisoDest, oficinaDest].recibirExpediente(exp)
            print("El Expediente se movio correctamente al juzgado del doctor : " +
juezDestino)
       '''SINO SE ENCONTRO EL EXPEDIENTE CON EL NUMERO INDICADO'''
       else:
            print("El Expediente no se encuentra en el juzgado indicado")
```