```
'''
Created on 08/12/2013

@author: Flor
'''

import csv

from grafo import Grafo
from esquina import Esquina
from juguete import Juguete
from fabrica import Fabrica

class Oficina(object):
    '''
    classdocs
    '''


    def __init__(self, mapa, fabricas, juguetes, idEsquinaUbicacion, maximoPeso):
        '''
        Constructor
        '''
        self.grafo = Grafo()
        self.agenda = dict()
        self.maximoPeso = int(maximoPeso)
        self.idUbicacion = idEsquinaUbicacion
        cargarEsquinasYCalles(self.grafo,mapa)
        cargarFabricas(self.grafo, self.agenda, fabricas)
        cargarJuguetes(self.grafo, self.agenda, juguetes)


    '''
    Obtiene las fabricas que tendra que pasar en navidad
    Devuelve una tupla  = (cantidad de fabricas, lista de fabricas)
    '''
    def seleccionarFabricas(self):
        lista = sorted(self.agenda, lambda key,key2: getFabrica(self.grafo, self.agenda, key).compare
(getFabrica(self.grafo, self.agenda, key2)))
        #lista.reverse() # porque estaba al reves el comparador
        optimos = list() # lista donde guardo las fabricas optimas

        ultimoIngresado = lista.pop()
        optimos.append(ultimoIngresado)
        while lista.__len__() > 0:
            #idFabrica = min(lista, key = lambda x: grafo.obtenerVertice(agenda[x]).getDato
().getFabrica(x).getHorarioEntradaCondicional(getFabrica(ultimoIngresado).getHorarioSalida()))
            idFabrica = lista.pop()
            fabrica = getFabrica(self.grafo, self.agenda,idFabrica)

            if fabrica.getHorarioEntrada() >= getFabrica(self.grafo,
self.agenda,ultimoIngresado).getHorarioSalida():
                optimos.append(idFabrica)
                ultimoIngresado = idFabrica
        contador = 0
        resultado = list()
        for op in optimos:
            contador += 1
            resultado.append(getFabrica(self.grafo, self.agenda, op))
        return (contador, resultado)

    '''
    Obtiene la cantidad de sonrisas por fabrica
    Parametro (idFabrica)
    Devuelve una tupla = (resultado de comando ( 0 = OK, -1 = Error ), (cantidad de sonrisas de la
fabrica, lista de juguetes))
    '''
    def valuarJuguetes(self, idFabrica):
        if (self.agenda.has_key(idFabrica)):
            return (0,getFabrica(self.grafo, self.agenda, idFabrica).getSonrisasMaximas
(self.maximoPeso))
        return (-1, None)
```

```
    '''
    Obtiene la cantidad de sonrisas en total que obtedra recorriendo las fabricas optimas para recorrer
    Devuelve una tupla = (resultado de comando (0 = OK), cantidad de sonrisas totales)
    '''
    def valuarJugutesTotal(self):
        camino = self.seleccionarFabricas()
        suma = 0
        for fabrica in camino[1]:
            suma += self.valuarJuguetes(fabrica.getId())[1][0]
        return (0, suma)

    '''
    Obtiene el camino mas corto a una fabrica
    idFabrica id de la fabrica a llegar
    Devuelve una tupla = (resultado de comando( 0 = OK, -1 = No existe la fabrica ), lista de las
esquinas a pasar para llegar a la fabrica)
    '''
    def getCaminoOptimoFabrica(self, idFabrica):
        if (self.agenda.has_key(idFabrica)):
            resultadoCamino = self.grafo.getCaminoOptimo(self.idUbicacion, self.agenda[idFabrica])
            return (0, int(resultadoCamino[0]), self.getEsquinas(resultadoCamino[1]))
        return (-1, None)

    def getEsquinas(self, lista):
        resultado = list()
        for idEsquina in lista:
            resultado.append(self.grafo.obtenerVertice(idEsquina).getDato())
        return resultado

# Carga archivo de juguetes
def cargarJuguetes(grafo, agenda, nombreArchivoJuguetes):
    f = open(nombreArchivoJuguetes,'rb')
    # f = open('juguetes.csv','rb')
    reader = csv.reader(f)
    for idFabrica,idJuguete,valor,peso in reader:
        juguete = Juguete(idJuguete, valor, peso)
        getFabrica(grafo, agenda, idFabrica).addJuguete(juguete)

# Carga archivo de fabricas
def cargarFabricas(grafo, agenda, nombreArchivoFabricas):
    f = open(nombreArchivoFabricas,'rb')
    # f = open('fabricas.csv','rb')
    reader = csv.reader(f)
    for idFabrica,esquina,entrada,salida in reader:
        fabrica = Fabrica(idFabrica,entrada,salida)
        grafo.obtenerVertice(esquina).getDato().addFabrica(fabrica)
        agenda[idFabrica] = esquina

# Carga archivo de mapa
def cargarEsquinasYCalles(grafo, nombreArchivoMapa):
    f = open(nombreArchivoMapa,'rU')
    # f = open('mapa.csv','rb')
    reader = csv.reader(f)
    cantidadEsquinas = 0
    # index indica la fila en la que esta. Si es 0, esta arriba de todo, corresponde a la
    # cantidad de esquinas. Voy avanzando y mientras no supere esa cantidad, agrego esquinas
    for index,row in enumerate(reader):
        if (index == 0):
            cantidadEsquinas = row[0]
        else:
            if (int(index) - 1 < int(cantidadEsquinas)):
                esquina = Esquina(row[0],row[1],row[2],row[3],row[4])
                grafo.agregarVertice(esquina)
            elif (int(index) -1 > int(cantidadEsquinas)):
                grafo.agregarArista(row[1],row[2],getPeso(grafo, row[1],row[2]))

# Toma datos de las esquinas 1 y 2 y calcula la distancia entre una y otra
def getPeso(grafo, id1, id2):
    esquina1 = grafo.obtenerVertice(id1).getDato()
    esquina2 = grafo.obtenerVertice(id2).getDato()
    return esquina1.calcularDistancia(esquina2)
```

```python
# Toma la fabrica con cierto idF
def getFabrica(grafo, agenda, idF):
    return grafo.obtenerVertice(agenda[idF]).getDato().getFabrica(idF)
```