package com.example.theyan.mapis;  
  
import java.util.ArrayList;  
import java.util.Iterator;  
  
public class Facade {  
  
 private ArrayList<Rutas> rutas;  
 private ArrayList<Reserva> reservas;  
  
  
 public Facade() {  
  
 reservas = new ArrayList<Reserva>();  
 rutas = new ArrayList<Rutas>();  
  
 }  
  
  
 public Rutas crearRuta(String nombre, String salida, int pasajeros, int calles, String conductor,String nombr,long xor,long yor,long xdes,long ydes,long distanci,long tiemp) throws Exception {  
 if (comprobarRuta(nombre) != null) {  
 throw new Exception("existe una ruta llamada así");  
 }  
 Rutas r = new Rutas();  
 r.setNombre(nombre);  
 r.setSalida(salida);  
 r.setPasajeros(pasajeros);  
 r.setConductor(conductor);  
 for (int i = 0; i < calles; i++) {  
 String nombrec = nombr;  
 long coordenadaXorigen = xor;  
 long coordenadaYorigen = yor;  
 long coordenadaXdestino = xdes;  
 long coordenadaYdestino = ydes;  
 long distancia =distanci;  
 long tiempo = tiemp;  
 Componente c = crearCalle(coordenadaXorigen, coordenadaYorigen, coordenadaXdestino, coordenadaYdestino, nombrec, distancia, tiempo);  
 r.add(c);  
 }  
 rutas.add(r);  
 return r;  
 }  
  
 public String listarRutas(String conductor) {  
 String r = "nombre | conductor | hora de salida | n de pasajeros \n";  
 Rutas ru;  
 for (Componente c : rutas) {  
 if (c instanceof Rutas) {  
 ru = (Rutas) c;  
 if (ru.getConductor().equals(conductor)) {  
 r = r + ru.getNombre() + " | " + ru.getConductor() + " | " + ru.getSalida() + " | " + ru.getPasajeros() + "\n";  
 }  
 }  
 }  
  
 return r;  
 }  
  
 public boolean eliminarRuta(String nombre) throws Exception {  
 Iterator it = rutas.iterator();  
 Componente c;  
 Rutas r;  
 while (it.hasNext()) {  
 c = (Componente) it.next();  
 if (c instanceof Rutas) {  
 r = (Rutas) c;  
 if (r.getNombre().equals(nombre)) {  
 rutas.remove(r);  
 return true;  
 }  
 }  
 }  
  
 throw new Exception("no se pudo eliminar la ruta,porque no fue encontrada");  
 }  
  
 public boolean modificarRuta(String bnombre, String nsalida, int npasajeros) throws Exception {  
 if (npasajeros < 0) {  
 throw new Exception("esta ruta no dispone de pasajeros");  
 }  
 Rutas rut;  
 for (Componente r : rutas) {  
 if (r instanceof Rutas) {  
 rut = (Rutas) r;  
 if (rut.getNombre().equals(bnombre)) {  
 rut.setSalida(nsalida);  
 rut.setPasajeros(npasajeros);  
 rutas.set(rutas.indexOf(r), (Rutas) (Componente) rut);  
 return true;  
 }  
 }  
 }  
  
 throw new Exception("no se ha podido modificar la ruta");  
 }  
  
 public void crearReserva(String hora, String ruta, String pasajero) throws Exception {  
 if (comprobarRuta(ruta) == null) {  
 throw new Exception("no existe esa ruta");  
 }  
 Rutas r = (Rutas) comprobarRuta(ruta);  
 modificarRuta(ruta, r.getSalida(), (r.getPasajeros() - 1));  
 reservas.add(new Reserva(pasajero, ruta, hora));  
 }  
  
 public boolean modificarReserva(String bruta, String nruta, String hora, String pasajero) throws Exception {  
 Reserva re;  
 for (Reserva r : reservas) {  
 re = r;  
 if (r.getRuta().equals(bruta) && r.getPasajero().equals(pasajero)) {  
 re.setHora(hora);  
 re.setRuta(nruta);  
 reservas.set(reservas.indexOf(r), re);  
 return true;  
 }  
 }  
  
 throw new Exception("no fue posible modificar la ruta");  
 }  
  
 public boolean eliminarReserva(String ruta, String pasajero) throws Exception {  
 for (Reserva r : reservas) {  
 if (r.getPasajero().equals(pasajero) && r.getRuta().equals(ruta)) {  
 Rutas ru = (Rutas) comprobarRuta(ruta);  
 modificarRuta(ruta, ru.getSalida(), (ru.getPasajeros() + 1));  
 reservas.remove(r);  
 return true;  
 }  
 }  
  
 throw new Exception("no se encontro una reserva para eliminar");  
 }  
  
 public String verReservas(String pasajero) {  
 String r = "ruta | hora\n";  
 for (Reserva re : reservas) {  
 if (re.getPasajero().equals(pasajero)) {  
 r = r + re.getRuta() + " | " + re.getHora() + "\n";  
 }  
 }  
 return r;  
 }  
  
  
 private Componente crearCalle(Long coordenadaXorigen, Long coordenadaYorigen, Long coordenadaXdestino, Long coordenadaYdestino, String nombre, Long distancia, Long tiempo) {  
 return new Calle(coordenadaXorigen, coordenadaYorigen, coordenadaXdestino, coordenadaYdestino, nombre, distancia, tiempo);  
 }  
  
 private Rutas comprobarRuta(String nombre) {  
 for (Rutas c : rutas) {  
 if (c.getNombre().equals(nombre)) {  
 return c;  
 }  
 }  
 return null;  
 }  
}