ESTRUCTURA DE DATOS Y ALGORITMOS
OBLIGATORIO

Foto estudiante A
Nombre estudiante A
Numero estudiante A

Foto estudiante B
Nombre estudiante B
Numero estudiante B

Grupo
Nombre del docente
Fecha

Table of Contents

1. Interfaz del Sistema: Pre y Post condiciones

//**Pre:** El parámetro cantidadCiudades debe ser un entero valido superior a cero
//**Post:** Se genera el sistema de emergencia junto con el mapa de ciudades

TipoRet crearSistemaDeEmergencias(int cantidadCiudades);

//**Pre:** Debe haber un sistema de emergencias creado con anterioridad
//**Post:** Se destruye el sistema de emergencia, liberando la memoria (se ponen en null todas las variables para que el garbaje collector las libere)

TipoRet destruirSistemaEmergencias();

//**Pre:**
a)El parámetro ambulanciaId debe ser un string valido,
b) El parámetro ciudadID debe ser un entero valido y debe existir previamente una ciudad agregada en la lista de ciudades que tenga ese ID.
c) La ambulancia a registrar no debe existir previamente en la lista

//**Post:** Se registra una ambulancia en la ciudad de destino y en la lista de ambulancias
TipoRet registrarAmbulancia(String ambulanciaId, int ciudadID);

//**Pre:**
a) El parámetro ambulanciaID debe ser un string valido
b) La ambulancia a eliminar debe existir previamnte en la lista

//**Post:**
Se quita la ambulancia de la lista apropiada
TipoRet eliminarAmbulancia(String ambulanciaID);

//**Pre:**
a) El parámetro ambulanciaID debe ser un string valido
b) La ambulancia debe existir en la lista

//**Post:**
a) EL atributo 'estado' de la ambulancia seleccionada se pone en 'deshabilitada'

TipoRet deshabilitarAmbulancia(String ambulanciaId);

//**Pre:**
a) El parámetro ambulanciaID debe ser un string valido
b) La ambulancia debe existir en la lista

//**Post:**

a) EL atributo 'estado' de la ambulancia seleccionada se pone en 'habilitada'
TipoRet habilitarAmbulancia(String ambulanciaID);

//**Pre:**
a) El parámetro ambulanciaID debe ser un string valido
b) La ambulancia debe existir en la lista

//**Post:**
a) Devuelve un objeto ambulancia de ID = parametro pasado

Ambulancia buscarAmbulancia(String ambulanciaID);

//**Pre:**
a) Deben haber ambulancias registradas en la lista

//**Post:**
a) Se imprime en consola el reporte de ambulancias
TipoRet informeAmbulancia();

//**Pre:**
a) Deben haber ciudades agregadas a la lista
b) El parametro ciudadID debe ser un enero valido mayor a cero

//**Post:**
a) Se imprime en consola la lista de ambulancias (id) que la ciudad seleccionada tenga.
b) En caso de no existir ambulancias asociadas se imprime un mensaje diciendo que no existen.
TipoRet informeAmbulancia(int ciudadID);

//**Pre:**
a) ambulanciaID debe ser una cadena valida.
b) ciudadID debe ser un entero valido mayor que cero.
c) El id de la ambulancia a mover debe corresponder con una de las registradas en la lista.
d) El id de la ciudad debe corresponder a una ciudad registrada en la lista de ciudades.
//Post:
a) La ambulancia seleccionada se registra en la ciudad seleccionada cambiando su atributo ciudad
TipoRet cambiarUbicacion(String ambulanciaID, int ciudadID);

//**Pre:**
a) El parametro ciuadadNombre debe ser una cadena valida
b) EL nombre de la ciudad no debe existir previamente en la lista

//**Post:**

a) Se agrega la ciudad a la lista de ciudades.

TipoRet agregarCiudad(String ciudadNombre);

//**Pre:**
a) La lista de ciudades debe estar poblada con al menos una ciudad
//**Post:**
a) Se imprime en consola el nombre de cada ciudad en la lista
TipoRet listarCiudades();

//**Pre:**
a) El parametro ciudadOrigen debe ser un entero valido mayor a cero y corresponder a una ciudad registrada previamente
b) El parametro ciudadDestino debe ser un entero valido mayor a cero y corresponder a una ciudad registrada previamente
c) El parametro minutosViaje debe ser un entero valido mayor a cero

//**Post:**
a) Se agrega en el mapa de rutas el registro para la la nueva ruta
TipoRet agregarRuta(int ciudadOrigen, int ciudadDestino, int minutosViaje);

//**Pre:**
a) El parametro ciudadOrigen debe ser un entero valido mayor a cero y corresponder a una ciudad registrada previamente
b) El parametro ciudadDestino debe ser un entero valido mayor a cero y corresponder a una ciudad registrada previamente
c) El parametro minutosViaje debe ser un entero valido mayor a cero
//**Post:**
a) Se modifica en el mapa de rutas el registro para la ruta seleccionada
TipoRet modificarDemora(int ciudadOrigen, int ciudadDestino, int minutosViaje);

//**Pre:**
a) El parametro ciudadID debe ser un entero valido mayor a cero
b) Debe existir una ciudad en la lista con el ID pasado
//**Post:**
a) Se imprime en consola la ambulancia mas cercana en minutos de viaje
TipoRet ambulanciaMasCercana(int ciudadID);

//**Pre:**
a) El parametro ciudadOrigen debe ser un entero valido mayor a cero y corresponder a una ciudad registrada previamente
b) El parametro ciudadDestino debe ser un entero valido mayor a cero y corresponder a una ciudad registrada previamente
//**Post:**
a) Imprime en consola la ruta mas rapida entre las dos ciudades seleccionadas

TipoRet rutaMasRapida(int ciudadOrigen, int ciudadDestino);

//Pre:
//Post:
TipoRet informeCiudades();

//Pre:
//Post:
TipoRet ciudadesEnRadio(int ciudadID, int duracionViaje);

//**Pre:**
a) El parametro ambulanciaID debe ser un stringa valido y corresponder a una ambulancia previamente registrada
b) El parametro cedula debe ser una cadena valida.
c) El parametro nombre debe ser una cadena valida.
//**Post:**
a) Se agrega en la lista de choferes de la ambulancia al conductor seleccionado.

TipoRet registrarChofer(String ambulanciaID, String nombre, String cedula);

//**Pre:**
a) El parametro ambulanciaID debe ser un stringa valido y corresponder a una ambulancia previamente registrada
b) El parametro cedula debe ser una cadena valida.
c) El chofer de cedula seleccionada debe estar en la lista de la ambulancia
//**Post:**
a) Se elimina de la lista de choferes al conductor seleccionado
TipoRet eliminarChofer(String ambulanciaID, String cedula);

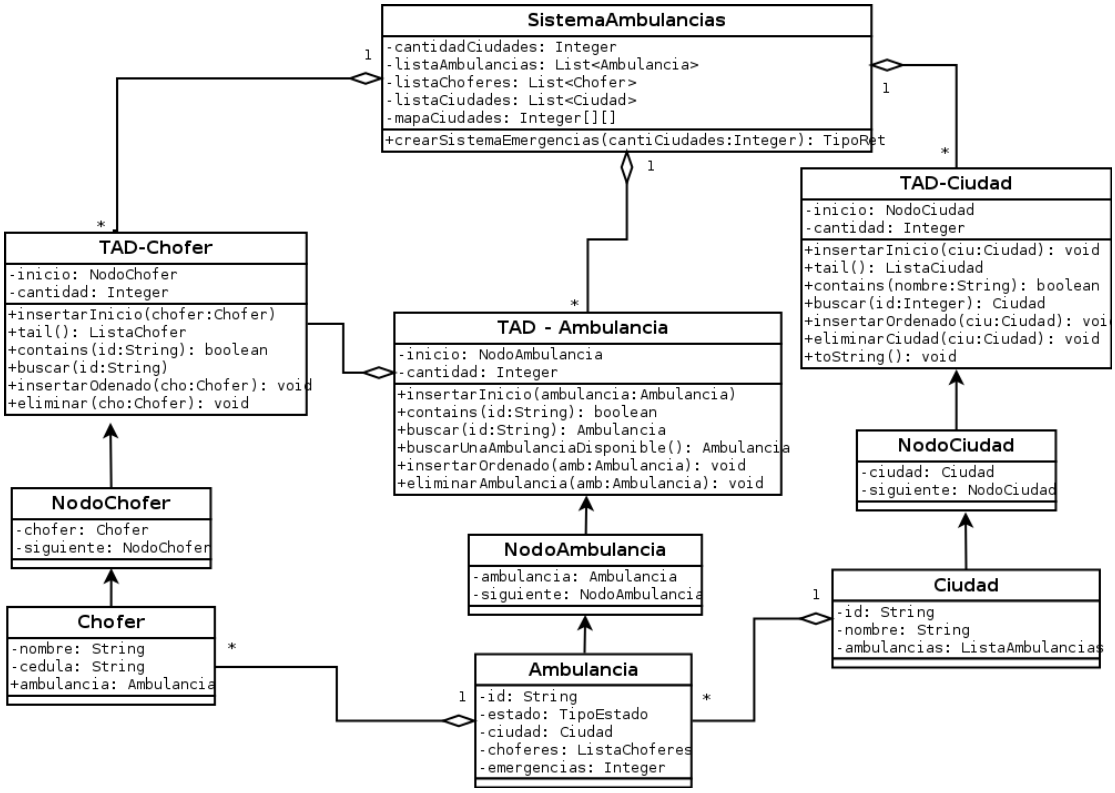
//**Pre:**
a) El parametro ambulanciaID debe ser un stringa valido y corresponder a una ambulancia previamente registrada

//**Post:**
a) Se imprime por consola el nombre de cada uno de los choferes que este en la lista de la ambulancia.

TipoRet informeChoferes(String ambulanciaID);

2. Solución escogida

2.1 Diagrama de la estructura de datos



.

2.2 Justificación

TAD - ListaAmbulancias	
Estructura	Se utiliza una lista ordenada, con inicio y un atributo de cantidad (int).
Justificación	El atributo 'cantidad' se utiliza para no tener que calcular cada vez el largo de la lista. Se utiliza insertarOrdenado debido a que si bien es mas costoso al inicio , luego para recuperar la información es mucho menos costoso

TAD - ListaChofer	
Estructura	Se utiliza una lista ordenada, con inicio y un atributo de cantidad (int).
Justificación	El atributo 'cantidad' se utiliza para no tener que calcular cada vez el largo de la lista. Se utiliza insertarOrdenado debido a que si bien es mas costoso al inicio , luego para recuperar la información es mucho menos costoso.

TAD - ListaCiudad	
Estructura	Se utiliza una lista ordenada, con inicio y un atributo de cantidad (int).
Justificación	El atributo 'cantidad' se utiliza para no tener que calcular cada vez el largo de la lista. Se utiliza insertarOrdenado debido a que si bien es mas costoso al inicio , luego para recuperar la información es mucho menos costoso.

Matriz Mapa	
Estructura	Es una matriz de enteros [ ][ ] que representa el mapa de rutas entre las ciudades, se genera al momento de crear el sistema en el procedimiento de carga.
Justificación	Todas las operaciones requeridas se pueden hacer sobre esa matriz por lo que no se necesitaba generar una clase aparte para representar el elemento.

3. Pruebas Ejecutadas

Se generaron los paquetes 'ambulancias', 'ciudades', 'mapas' y 'sistema de emergencias' en el directorio TestPackages donde se encuentran todas las pruebas Junit para cada uno de los métodos descritos en la letra del obligatorio.

Listado de pruebas:

public void testBuscarAmbulanciaConAmbulanciaIdExistente()

public void testBuscarAmbulanciaConAmbulanciaIdNOExistente()

public void testBuscarUltimaAmbulanciaAgregada()

public void testBuscarPrimeraAmbulanciaAgregada()

public void testBuscarAmbulanciaConUnaSolaAmbulanciaEnLaLista()

public void testBuscarAmbulanciaConLaListaVacia()

public void testInformeAmbulancias()

public void testInformeAmbulanciasSinAmbulanciasRegistradas()

public void testInformeAmbulanciasPorCiudad()

public void testInformeAmbulanciasPorCiudadNoExiste()

public void testAgregarUnaAmbulancia()

public void testAgregarUnaAmbulanciaExistente()

public void testAgregarUnaAmbulanciaCiudadNoexiste()

public void testAgregarDiezAmbulancia()

public void testCambiarUbicacionAMbulancia()

public void testCambiarUbicacionAMbulancianciaNoExistente()

public void testCambiarUbicacionAMbulanciaCiudadNoExistente()

public void testDesHabilitarUnaAmbulanciaExistente()

public void testDesHabilitarUnaAmbulanciaNoExistente()

public void testDesHabilitarUnaAmbulanciaYaNODisponible()

public void testDesHabilitarUnaAmbulanciaAsignadaAunVIaje()

public void testEliminarUnaAmbulancia()

public void testEliminarUnaAmbulanciaNoExistente()

public void testEliminarUnaAmbulanciaAsignadaViaje()

public void testHabilitarUnaAmbulanciaDeshabilidatada()

public void testHabilitarUnaAmbulanciaYahabilidatada()

public void testHabilitarUnaAmbulanciaInexistente()

public void testRegistrarCiudadNOExistente()

public void testRegistrarCiudadExistente()

public void testRegistrarCiudadSinCapacidadenSistema()

public void testLIstarCiudades()

public void testLIstarCiudadesMapaVacio()

public void testAgregarRutaAlMapa()

public void testAgregarRutaAlMapaCiudadOrigenInexistente()

public void testAgregarRutaAlMapaCiudadDestinoInexistente()

public void testAgregarRutaAlMapaMInutoscero()

public void testModificarrRutaAlMapaMInutoscero()

public void testModificarDemoraAlMapaCiudadOrigenNoExiste()

public void testModificarDemoraDestinoNOExiste()

public void testModificarDemoraAlMapa()

public void testBuscarAmbulanciaMasCercana()

public void testBuscarRutaMasRapida()

public void testRegistrarCiudadNOExistente()