#include "Linea.h"

Linea::Linea(std::string nombre, bool esTransporte) {

this->nombre = nombre; // Inicialización del nombre de la línea

this->esTransporte = esTransporte; // Inicialización del indicador de transporte

this->cabezaEstaciones = nullptr; // Inicialización del puntero a la primera estación como nulo

this->siguienteLinea = nullptr; // Inicialización del puntero a la siguiente línea como nulo

this->numEstaciones = 0; // Inicialización del número de estaciones como cero

}

Linea::~Linea() {

Estacion\* temp = cabezaEstaciones;

while (temp != nullptr) {

Estacion\* siguiente = temp->obtenerSiguienteEstacion(); // Accediendo a través del método público

delete temp;

temp = siguiente;

}

}

std::string Linea::getNombre() const {

return nombre; // Devuelve el nombre de la línea

}

int Linea::getNumEstaciones() const {

return numEstaciones; // Devuelve el número de estaciones en la línea

}

bool Linea::getEsTransporte() const {

return esTransporte; // Devuelve el indicador de transporte de la línea

}

void Linea::agregarEstacion(Estacion\* estacion, int posicion) {

if (posicion < 0 || posicion > numEstaciones) {

return; // No se puede insertar en una posición inválida

}

if (cabezaEstaciones == nullptr) {

cabezaEstaciones = estacion; // Si la lista está vacía, la estación se convierte en la cabeza

numEstaciones = 1;

return;

}

if (posicion == 0) {

estacion->enlazarSiguienteEstacion(cabezaEstaciones);

cabezaEstaciones->enlazarAnteriorEstacion(estacion);

cabezaEstaciones = estacion; // La estación se inserta al principio

} else {

Estacion\* temp = cabezaEstaciones;

for (int i = 0; i < posicion - 1; i++) {

temp = temp->obtenerSiguienteEstacion();

}

estacion->enlazarSiguienteEstacion(temp->obtenerSiguienteEstacion());

estacion->enlazarAnteriorEstacion(temp);

temp->obtenerSiguienteEstacion()->enlazarAnteriorEstacion(estacion);

temp->enlazarSiguienteEstacion(estacion); // La estación se inserta en la posición especificada

}

numEstaciones++;

}

bool Linea::eliminarEstacion(std::string nombreEstacion) {

if (cabezaEstaciones == nullptr) {

return false; // No hay estaciones en la línea

}

Estacion\* temp = cabezaEstaciones;

while (temp != nullptr) {

if (temp->getNombre() == nombreEstacion) {

// Eliminar la estación temp

if (temp == cabezaEstaciones) {

cabezaEstaciones = temp->getSiguienteEstacion();

} else {

temp->getAnteriorEstacion()->enlazarSiguienteEstacion(temp->getSiguienteEstacion());

if (temp->getSiguienteEstacion() != nullptr) {

temp->getSiguienteEstacion()->enlazarAnteriorEstacion(temp->getAnteriorEstacion());

}

}

delete temp;

numEstaciones--;

return true; // La estación se eliminó con éxito

}

temp = temp->getSiguienteEstacion();

}

return false; // No se encontró la estación en la línea

}

bool Linea::existeEstacion(std::string nombreEstacion) const {

Estacion\* temp = cabezaEstaciones;

while (temp != nullptr) {

if (temp->getNombre() == nombreEstacion) {

return true; // La estación existe en la línea

}

temp = temp->getSiguienteEstacion();

}

return false; // La estación no existe en la línea

}

Linea\* Linea::getSiguienteLinea() const {

return siguienteLinea; // Devuelve la siguiente línea

}

Estacion\* Linea::getCabezaEstaciones() const {

return cabezaEstaciones; // Devuelve la cabeza de estaciones de la línea

}

void Linea::enlazarSiguienteLinea(Linea\* linea) {

this->siguienteLinea = linea; // Establece el enlace a la siguiente línea

}