nombre: CARRO

datos

```
string placa;
string color;

string tipo;



string marca;


bool servPublico;



chrono::system_clock::time_point horaEntrada;
```

1 1


nombre:NodoTree
Datos
Carro carro
nodoTree Izquierda
nodoTree Derecha
int altura
1
1
funciones
int calcularAltura();
bool estaBalanceado();

nombre: ArbolBinario

Datos

nodoTree raiz

funciones

```
 nodoTree* insertarAVL(nodoTree*
nodo, Carro* carro);  //
Inserción con balanceo AVL

 nodoTree*
eliminarRecu(nodoTree* nodo,
const std::string& placa);  //
Eliminación con balanceo AVL
 nodoTree*
buscarRecurso(nodoTree* nodo,
const std::string& placa);
 double
calcularGananciasRecursivo(nodoT
ree* nodo, double
pagoPorMinuto);

 void postOrdenRecu(nodoTree*
nodo);

void preOrdenRecu(nodoTree*
nodo);
bool
estaBalanceadoRecu(nodoTree*
nodo);
int altura(nodoTree* nodo);
int getFactorBalance(nodoTree*
nodo);

nodoTree*
rotacionDerecha(nodoTree* y);


nodoTree*
rotacionIzquierda(nodoTree* x);

ArbolBinario();
void insertar(Carro* carro);
void eliminar(const std::string&
placa);
Carro* buscar(const std::string&
placa);

double calcularGanancias(double
pagoPorMinuto);

void postOrden();
void preOrden();
```

1 1

```
bool estaBalanceado();
```

nombre: PARQUEADERO

Datos

string ID

string nombre

int capacidad

double pagoXmin

ArbolBinario arbolCarros

Funciones:

```
Parqueadero(const std::string& _id,
const std::string& _nombre, int
_capacidad, double _pagoXmin);

void agregarCarro(Carro* carro);

 double calcularGanancias();

 double retirar(const std::string& placa);

 void mostrarArbolPreOrden();

void mostrarArbolPostOrden();

 bool estaBalanceado();

string getId() const;
string getNombre() const;

int getCapacidad() const;

 double getPagoPorMinuto() const;
```