Project Park Park Go Documentation

**Client Class**

**-numéro: String**

Le numéro unique du client

**-nom: String**

Le nom du client

**-prenom: String**

Le prénom du client

**+creer\_Compte (numéro: String, nom: String, prenom: String): int**

Crée un nouveau compte pour le client.

**+modifier\_Compte (numéro:String, nom:String, prenom: String): int**

Modifie les détails du compte du client.

**+supprimer\_Compte (numéro:String,nom: String, prenom:String): int**

Supprime le compte du client.

**+programmer\_Course(point\_Depart:Lieu, point\_Arriver :Lieu) : int**

Programme un trajet pour le client.

**+getNuméro (): String**

Renvoie le numéro unique du client.

**+getNom (): String**

Renvoie le nom du client

**+getPrenom (): String**

Renvoie le prénom du client.

**+setNuméro (numéro: String): void**

Définit le numéro unique du client.

**+setNom(nom:String): void**

Définit le nom du client.

**+setPrenom (prenom: String): void**

Définit le prénom du client.

**Lieu**

**-nom: string**

Le nom du lieu.

**-longitude: float**

La longitude du lieu.

**-latitude: float**

La latitude du lieu.

**+ajouterLieu(nom: String, longitude: float, latitude:float): int**

Ajoute un nouveau lieu.

**+modifierLieu (nom: String, longitude: float, latitude: float): int**

Modifie les détails d'un lieu.

**+ supprimerLieu(): int**

Supprime un lieu

**+distance(Longitude: float, latitude:float): float**

Calcule la distance jusqu'à un autre lieu.

**+montant (longitude: float, latitude: float): float**

Calcule le montant du trajet jusqu'à un autre lieu.

**+getNom (): String**

Renvoie le nom du lieu.

**+getLongitude(): float**

Renvoie la longitude du lieu.

**+getLatitude(nom: String): float**

Renvoie la latitude du lieu.

**+ setNom (nom :String): void**

Définit le nom du lieu.

**+setLongitude (longitude: float): void**

Définit la longitude du lieu.

**+setLatitude(latitude:float): void**

Définit la latitude du lieu

**Course**

**-idCourse: int**

L'identifiant unique de la course.

**-client: Client**

Le client effectuant la course.

**-vehicule: Vehicule**

Le véhicule assigné à la course.

**-lieuDepart: Lieu**

Le lieu de départ.

**-lieuDarriver: Lieu**

Le lieu d'arrivée.

**-distance: Double**

La distance du trajet.

**-montant: Double**

Le montant de la course.

**-statut: String**

- Le statut de la course.

**+effectuerCourse(): void**

Demarre la course

**+terminerCourse(): void**

Terminer la course

**+programmer\_Course (point\_Depart:Lieu, point\_Arriver: Lieu): int**

- Le statut de la course.

**+getIdCourse(): int**

Renvoie l'identifiant unique de la course.

**+getClient(): Client**

Renvoie le client effectuant la course.

**+getVehicule(): Vehicule**

Renvoie le véhicule assigné à la course.

**+getDistance(): float**

Renvoie la distance du trajet.

**+getMontant (): float**

Renvoie le montant de la course.

**+getLieuDepart(): void**

Renvoie le lieu de départ.

**+getLieuDarriver (): Lieu**

Renvoie le lieu d'arrivée.

**+setIdCourse(idCourse:int) : void**

Définit l'identifiant unique de la course.

**+setClient (client: Client): void**

Définit le client effectuant la course.

**+setVehicule(vehicule:Vehicule): void**

Définit le véhicule assigné à la course.

**+setDistance(distance:float): void**

Définit la distance du trajet.

**+setMontant (montant: float): void**

Définit le montant de la course.

**Vehicule**

**-numéroMatricule: int**

Le numéro matricule unique du véhicule.

**-position: Lieu**

La position actuelle du véhicule.

**-etat: String**

Le statut du véhicule (par exemple, disponible, en service).

**-capacité: int**

La capacité de passagers du véhicule.

**+modifierStatut()**

Modifie le statut du véhicule.

**+getNumeromatricule(): int**

Renvoie le numéro matricule du véhicule.

**+getPosition(): String**

Renvoie la position actuelle du véhicule.

**+getEtat (): String**

Renvoie le statut du véhicule.

**+setNumeromatricule(numéroMatricule: int) : void**

Définit le numéro matricule du véhicule.

**+setPosition(position:String): void**

Définit la position actuelle du véhicule.

**+setEtat(etat:String): void**

Définit le statut du véhicule.

**GestionnaireCourse**

**+enregistrerVehicule()**

Enregistre un nouveau véhicule

**+afficheListeVehicule(): void**

Affiche la liste des véhicules.

**+associerVehicule(): void**

Associe un véhicule à une course.

**+retirerVehicule(): void**

Retire un véhicule du système.

**Relations**

1. *Client 1 — 0.. Course*\*: Un client peut avoir plusieurs courses, mais chaque course est associée à un seul client.
2. **GestionnaireCourse — Course**: Le gestionnaire de course supervise les courses.
3. **Course 1 — 1 Vehicule**: Chaque course est assignée à un véhicule.
4. *Vehicule 0.. — 0.. Lieu*\*: Un véhicule peut avoir plusieurs positions, mais à tout moment, il est associé à un seul lieu.