Licence Informatique
(X21I030)
Programmation orienté objet

L2 informatique
2021-2022

Nom et prenom	Groupe
DIALLO Fatoumata	386N
ASSOUD AYHAM	*386N*

Projet noté:*Realisation de logiciel*

Introduction

Notre travail consiste à la réalisation d'un logiciel pour répondre aux besoins d'une entreprise de distribution de repas en milieu urbain.Dans un premier temps il s'agit de l'acquisition de véhicule scooter et vélo;des salariés cyclistes et conducteurs de moto pour effectué des courses.Les meilleurs courses seront retenus pour la livraison des repas.

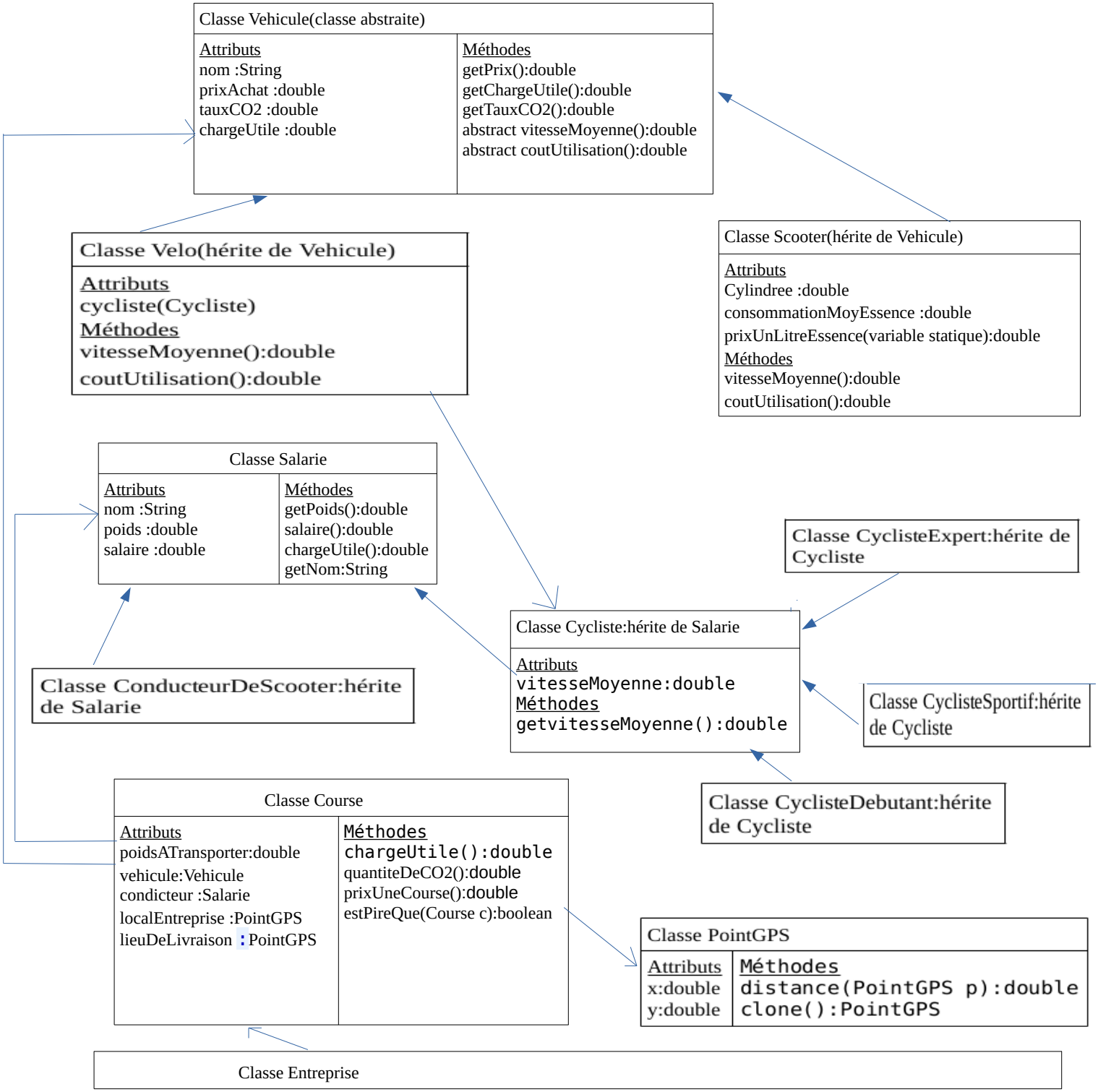
Diagramme de classe

Relation

A ——▶ B : La classe A hérite de la classe B

A ——◇ B:un objet A se compose d'un ou plusieurs objets B.

tous les attributs sont 'private' ; les méthodes et classes sont 'public'.



Attributs	Méthodes
l_course : ArrayList de coure l_coursesOptimales: ArrayList de course l_courseMoinsDeCO2Emi: ArrayList de course	coursePossibles(Course C):Liste de course possibles RepaSeralivrer():boolean courseOptimales():Liste de course optimale courseMoinsDeCO2Emis() :Liste de course dont la quantite de CO2 est minimale

Classe Trajet:extends JComponent implements ActionListener	
Attributs int xLivreur int yLivreur int xEntreprise int yEntreprise int xLivraison int yLivraison	Méthodes public void paint(Graphics g) public void actionPerformed(ActionEvent e)

3) Relation entre les classes

il existe une hiérarchie entre la super Classe ‘Vehicule’ qui représente un véhicule et ses sous classes ‘Velo’ et ‘Scooter’ qui représentent respectivement un objet vélo et un objet Scooter. En effet, ‘Velo’ et ‘Scooter’ ont des caractéristiques communes à tous véhicule notamment le nom,le prix d’achat,la vitesse moyenne et le taux d’émission de CO2 ;un objet Scooter se comporte comme un objet Vehicule avec en plus une cylindrée(en centimètres),une consommation moyenne d’essence Donc ‘Scooter’ hérite de ‘Vehicule’.Un objet ‘Velo’ se comporte aussi comme un objet ‘Vehicule’ avec en plus sa vitesse moyenne qui dépend du Cycliste, Donc Velo hérite de Vehicule. La super classe ‘Vehicule’ est une classe abstraite car elle possède deux méthodes abstraites pour le calcul de la vitesse moyenne et le coût d’utilisation;l’utilisation de méthodes abstraites est dû au fait qu’ on ne connaît pas les spécificités du vehicule donc on ne peut pas effectué le calcul à ce niveau. Par exemple la vitesse moyenne du Velo c’est celle du Cycliste(Donc un Vélo à une variale de Type Cycliste) et la vitesse moyenne du Scooter dépends de sa cylindré; le coût d’utilisation du Velo depends que du prix d’achat du vélo tandis que pour le Scooter dépends du prix d’achat mais aussi du coût de la consommation d’essence sur un kilomètre.
‘ConducteurDeScooter’ qui représente un conducteur de scooter et ‘Cyliste’ qui représente un cycliste sont deux objets qui ont des caractéristiques communes(le nom,le poids,le salaires) et aussi des comportements propre à chaque comme le calcul de la vitesse moyenne. Donc ces deux classes sont dérivés d’une classe mère qu’on a nommé ‘Salarie’ qui représente un salarié.
Comme évoqué précédemment,la vitesse moyenne de l’objet Velo c’est celle de l’objet Cycliste et celle de l’objet Cycliste se détermine par la catégorie du Cycliste: si c’est un expert la vitesse moyenne est de 20 kilomètre par heure,le sportif lui à 15 kilomètre par heure et le débutant à 10 kilomètre par heure.Donc on a les Sous classes ‘CyclistesExpert’ , ’CyclisteSportif’ et ‘CyclisteDebutant’ qui héritent toutes de ‘Cycliste’.
La Classe ‘course’ représente une course qui consiste à faire transporter un certain poids sur un trajet entre le local de l’entreprise et le lieu de livraison par un salarié conduisant un véhicule.Donc la classe course est rattaché à un véhicule qui va permettre le transport de ce poids entre le local de l’entreprise un pointGPS et le lieu de livraison un pointGPS également ;Ce

véhicule sera conduit par un salarié(cycliste ou conducteur de moto).Donc la classe ‘Course’ est relié à ‘Vehicule’ , ‘Salarie’ et à un un objet ‘PointGPS’ qui sert à la représentation d’un point.
Une course à une charge utile qui est le maximum entre celle du véhicule et celle du conducteur.
Une course est possible si la charge utile est supérieure ou égale au poids à transporter
et si le temps de parcours est d’au plus une heure.La quantité de CO2 émis est calculée en fonction du taux d’émission de CO2 du véhicule et de la distance à parcourir. Le prix d’une course est calculé comme une somme entre le salaire du conducteur et le coût d’utilisation du véhicule sur la distance à parcourir.On peut comparer les courses entre elles,donc la classe course a une méthodes de comparaison de deux courses ;On dit qu’une course est pire qu’une autre si elle a la quantité de CO2 la plus grande ainsi que le prix le plus élevé ; ou si sa quantité de CO2 est supérieur et que les prix sont égaux ; ou si les quantités de CO2 sont égaux et son prix supérieur.
La classe entreprise réalise des course,elle a une liste de course et est donc composé de plusieurs objets ‘Course’.Si cette liste est vide il n’y a pas de courses Possibles(si aucune course n’est possible le repas ne sera pas livré).d’où l’Entreprise a une méthode ‘CoursePossibles’ qui retourne la liste de courses possibles.S’il y’a des courses Possible le gérant de l’entreprise veut toutes les courses qui ne sont pas pires qu’une autre course, appelons-les les courses optimales,ces courses sont determiné avec la méthode de comparaison de l’objet ‘Course’.puis, parmi les courses optimales, il veut celles qui ont une quantité minimale de CO2 émis.
La classe Trajet est une sous classe de ‘JComponent’ et ‘ActionListener’. Cette classe nous permet de faire le graphique:le local de l’entreprise est représenté par un rectangle vert, le lieu de livraison par un rectangle rouge et le salarié qui effectue la livraison par un cercle bleu. Cette classe permet de voir la position du livreur en temps réels.

4)spécification

Classe abstraite:Vehicule
Rôle: la classe Vehicule permet de créer des objets véhicules avec le nom,le prix d’achat ,la quantité de CO2 et la charge utile.
Methodes :

Signature	Description
Vehicule(String nom,double prixAchat,double tauxCO2,double chargeUtile)	Intialisation de ses ttributs.Vehicule Ne peut pas être instancier directement car c’est une classe abstraite
double getPrix()	retourn le prix d’achat du Véhicule en euros
double getChargeUtile()	retourne la charge utile en kilogrammes
double getTauxCO2()	Retourne le taux d’émission de CO2 en gramme par kilomètre
double abstract vitesseMoyenne()	Méthode abstraite qui sera implémentée dans les classes dérrivées de Vehiule
double abstract coutUtilisation()	Méthode abstraite qui sera implémentée dans les classes dérrivées de Vehiule

Classe : Velo
Rôle:la classe Velo est une sous classe de Vehicule,elle permet de créer un objet vélo
Méthodes

Signature	Description

Velo(String nom,double prixAchatVelo,Cycliste c)	Dans le constructeur de Velo il y'aura appel au constructeur de la super classe Vehicule avec le mot clé super(nom,prixAchatVelo,0,0);la charge utile et le taux de CO2 vallent 0.
double VitesseMoyenne()	Implémentation de la méthode abstraite return la vitesse moyenne du Cycliste
double coutUtilisation()	Calcul et retourne le prix divisé par 30000

Classe : Scooter

Rôle:la classe Scooter est une sous classe de Vehicule,elle permet de créer un objet Scooter

Méthodes

Signature	Description
Scooter(String nom,double prixAchatScooter,double chargeUtile,double cylindree,double consommationMoyEssence)	Dans le constructeur de Scooter il y'aura appel au constructeur de la super classe Vehicule avec le mot clé super(nom,prixAchatScooter,cylindree/4, chargeUtile) et on va initialisé ses attributs
double VitesseMoyenne()	Implémentation de la méthode abstraite calcul la vitesse moyenne du Scooter comme étant 30+(cylindre/4) la et returne
double coutUtilisation()	Implémentation de la méthode abstraite calcul la consommation d'essence sur un kilomètre=(consommationMoyEssence/100)*prixUnLitreEssence); ensuite le coût d'utilisation est calculé par :(le prix /20000)+ coutConsommationEssenceSurUnkm et ensuite elle retourne le résultat

Classe : Salarie

Rôle:la classe Salarie est une super classe de la classe Cycliste et ConducteurDeScooter,elle permet de créer un objet salarié

Méthodes

Signature	Description
Salarie(String nom,double poids,double salaire)	Initialise les attributs
String getNom()	Retourne le nom du salarié
getPoids()	Retourne le poids du salarié
getSalaire()	Retourne le salaire
chargeUtile()	Calcul la charge utile comme étant le poids divisé 8 et retourne le résultat

Classe : ConducteurDeScooter

Rôle:la classe ConducteurDeScooter est une sous classe de la classe Salarie,elle permet de créer un objet ConducteurDeScooter

Méthodes

Signature	Description
ConducteurDeScooter(String nom,double poids,double salaire)	Appel du constructeur de la super classe Salarie avec le mot clé super(nom,poids,salaire)

Classe : Cycliste
Rôle:la classe Cycliste est une sous classe de la classe Salarie,elle permet de créer un objet cyliste
Méthodes

Signature	Description
Cycliste(String nom,double poids,double salaire,double vitesseMoyenne)	Appel du constructeur de la super classe Salarie avec le mot clé super(nom,poids,salaire),l'attribut vitesseMoyenne sera initialisé avec vitesseMoyenne en parametre
double getVitesseMoyenne()	Retourne la vitesse moyenne

Classe : CyclisteExpert
Rôle:la classe CyclisteExpert est une sous classe de la classe Cycliste ,elle permet de créer un objet CyclisteExpert
Méthodes

Signature	Description
CyclisteExpert(String nom,double poids,double salaire)	Appel du constructeur de Salarie super(nom,poids,salaire,20). On met la vitesse moyenne à 20

Classe : CyclisteSportif
Rôle:la classe CyclisteSportif est une sous classe de la classe Cycliste ,elle permet de créer un objet CyclisteSportif
Méthodes

Signature	Description
CyclisteExpert(String nom,double poids,double salaire)	Appel du constructeur de Salarie super(nom,poids,salaire,15). On met la vitesse moyenne à 15

Classe : CyclisteDebutant
Rôle:la classe CyclisteDebutant est une sous classe de la classe Cycliste ,elle permet de créer un objet CyclisteDebutant
Méthodes

Signature	Description
CyclisteExpert(String nom,double poids,double salaire)	Appel du constructeur de Salarie super(nom,poids,salaire,10). On met la vitesse moyenne à 10

Classe : Course

Rôle:la classe Course est une classe qui permet de créer un objet Course pour la livraison d'un repas.Cette méthode a comme attribut Vehicule (vehicule),Salarie (conducteur),PointGPS (localEntreprise) et PointGPS (lieuDeLivraison).C'est une classe dont le constructeur peut lever une exception si la charge utile est inférieur au poids à transporter et si le temps de parcours est plus d'une heure.

Méthodes

Signature	Description
Course(PointGPS localEntreprise,PointGPS lieuDeLivraison,double poidsATransporter,Salarie conducteur,Vehicule vehicule)	Le constructeur de Course peut lever une exception ,initialisation des attributs
Double getPoidsATransporter()	return poidsATransporter
PointGPS getLieuDeLivraison()	return lieuDeLivraison
double chargeUtile()	Si la charge utile du vehicule est supérieur à celle du conducteur on retourne la charge utile du vehicule sinon on retourne la charge utile du conducteur
double quantiteDeCO2()	On effectue le calcul puis retourne (vehicule.getTauxCO2()*(this.localEntreprise.distance(this.lieuDeLivraison)*2))
boolean estPireQue(Course c)	Retourne vrai si this est pire que la course en parametre
double prixUneCourse()	On effectue le calcul puis retourne conducteur.getSalaire()+ (vehicule.coutUtilisation()/ (this.localEntreprise.distance(this.lieuDeLivraison)*2))

Classe : PointGPS

Rôle:la classe PointGPS permet la représentatation d'un point avec les coordonnés x et y

Méthodes

signature	Decription
PoiPointGPS(int x, int y)	Initialise les attributs x et y
int getX()	Retourne la valeur de x
int getY()	Retourne la valeur de y
double distance(PointGPS p)	Calcul et retourne la distace entre deux pointGPS

Classe :Entreprise

Rôle:la classeEntreprise permet la représentatation d'un objet Entreprise ,cette contient une liste de course,c'est dans cette classe que nous allons déterminé les courses optimales et celles qui émettent moins de CO2

Méthodes

signature	Description
Entreprise()	Initialise la liste de la course à une liste vide
ArrayList<Course> coursePossibles(Course c)	Permet d'ajouter c à notre liste et rétourne la liste courses possibles
boolean RepaSeraLivrer()	Retourne vrai si la liste de course possible n'est pas vide,pour savoir si le repas sera livré ou pas
ArrayList<Course> l_courseMoinsDeCO2Emis()	Retourne la liste émettent moins de CO2
ArrayList<Course> coursesOptimales()	Retourne liste des course optimales
void afficherNoMCoursierOptimales()	Retourne les noms des salarié de course optimale
Void afficherNoMCoursierPossibles()	Retourne les noms des salarié de course possible
void afficherNoMCoursierMoinsDeCO2()	Retourne les noms des salarié de course qui emmettent le minimum de CO2

Classe :Trajet

Rôle:la classe Trajet permet la représentation d'un Trajet de course entre le local de l'entreprise et lieu de livraison.

Méthodes

signature	Description
Trajet(Entreprise entreprise)	Le constructeur Trajet initialise ses donnés avec celles de la course optimale
public void paint(Graphics g)	Permet de faire le graphique
public void actionPerformed(ActionEvent e)	Permet le déplacement du salarié en temps réel

5)Jeu de teste avec 3 salariés

Nom	Poids	Salaire	Fonction
Jade	70	8	CyclisteDebutant
Alfred	80	9	ConducteurDeScooter
Anna	67	8	CyclisteExpert

Le nom des coursiers possibles sont:`Jade,Alfred,Anna.`

```
Le nom du coursier dont la course est optimale est:Anna.
les noms de Coursiers qui emmettent moins de quantité de CO2 sont: Jade et Anna.
```

Conclusion

Ce projet s'est révélé très enrichissant dans la mesure où il a consisté en une approche concrète du concepteur de logiciel.En effet,la prise d'initiative, le respect des délais et le travail en équipe seront des aspects essentiels de notre futur métier.

Notre solution est extensible dans la mesure où d’autres objets peuvent être intégré dans le programme par exemple une voiture qui représente un nouveau type de véhicule.

Le principe d’encapsulation est respecté car on n’a pas accès direct aux données d’un objet ainsi on ne peut pas les modifiés,on les manipules via ses méthodes.