Contents

1 Project Carferry

1.1 Université de Franche-Comté

1.1.1 Alphée GROSDIDIER & Quentin OBERON

Rapport demandé en français

1.2 Présentation de l'applications

1.2.1 Sujet

Il nous était demandé de créer une application tout d'abore sous forme textuelles puis sous forme graphique de géstion de chargement et déchargement d'un ferry avec gestion des tickets.

Tous d'abord nous devions créer une classe camion et voiture pérmettant respectivement de charger et décharger un camion et une voiture. Les deux types de véhicules doivent poivoir avoir un conducteur, une plaque d'immatriculation, un longueur et un poid. Plus spécifiqauement un camion doit avoir le poid de se cargaison et une voiture, le nombre de passager qu'elle transporte.

Le conducteur est définit pas un nom et un prénom ainsi qu'un permis de conduire.

La cale doit pouvoir contenir des camions et des voitures, elle doit permettre des les charger et de les déchargér. Elle a plusieurs particularités. Elle peux contenir seulement une certaine masse définit en Tonnes et a une certeine longueur définit en mètres. De plus on doit répartir équitablement les véhicules pour que le bateu ne penchent pas plus d'un côté que de l'autre. Les véhicules sont chargées en file, c'est a dire que le dernier véhicule chargé doit sortir en dernier et le premier chargé doit sortir en premier.

On a une gestion idépendante du prix des camions et des voitures. Un camion côute pour la traversée un prix de 45€ + 0.1€ * (poid de sa cargaison). Une voiture côute (avec son conducteur) 35€ + 3€ * (nombre de passagers). Un ticket est créer lors de l'embarquement.

Le ticket est composé du nom et prénom du conducteur, de la position du véhicule dans la cale et du tarif du voyage. La position du ticket est noté par la lettre de la rangées et la position dans la rangée. La lettre est soit 'G' pour gauche soit 'D' pour droite. Les tickets sont enregistrées et triées par ordre alphabétique des conducteurs.

Il y auras aussi une gestion des erreurs d'embarquement pour un cale qui ne peux pas accepter le véhicule car il est trop lourd ou parce qu'il est trop long et qu'il n'y a plus de place.

1.2.2 Nos Choix

1.2.2.1 Les choix sur l'interface textuelle

Nous avons tout d'abord construit l'application de façon textuelle. Sachant que nous devions ensuite la porter en mode graphique, nous avons fais une classe regroupant toutes les méthodes que nous aurions besoin pour développer notre interface. Nous somme partis sur l'interface nommée "Boat".

Nous avons ensuite choisis de bien séparer nos modules (nos classes) pour que le project soit lisible et pour que l'implémentation de nouvelles fonctionnalitées soit plus simples. Nous avons donc choisis de creer un module qui se chargerais de la gestions des véhicule dans l'application et un autre du prix du trajet. La liste des ticket du trajet a été laissé dans cette classe pour ne pas s'encombrer de plus de méthodes dans la classe "Wedge".

Par rapport au sujet nous avons prix la liberté de modifier la classe véhicule. Nous n'avons pas ajouter une méthode abstraite dans la classe véhicule pour avoir le prix du véhicule mais nous avons plutôt fais une classe qui se charge de donner le prix. Ce choix a été principalement fait pour donner plus de lisibilité dans le projet en regroupant le meme type d'information au même endroit, ici le prix.

Un autre choix qui esty différent de celui proposé dans le sujet est celui de la classe ticket. Nous avons tout d'abord changer la façon d'enregistrer la position du véhicule dans la cale. Nous avons creer une classe position qui nous permet de transmettre plus facilement la rangée et la position dans la rangée. Puis nous n'enregistrons pas la rangé avec une lettre 'G' ou 'D' mais avec un nombre permettant d'avoir un nombre indéfinis de rangée.

Sur Ticket nous avons aussi pris la liberté d'enregistrer directementy le véhicule et pas les informations relative au conducteur et au véhicule. Nous avons fais ce choix pour éviter la duplications d'information même si, une fois enregistré, le véhicule ne peux plus changer.

On a aussi décider d'enregistrer les véhicule dans des classes "Row" pour nous silplifier les algorithmes au lieu de faire une liste statique de Queue.

Pour les points les futiles, nous avons ajouter une vérifications de certaines érreurs d'enregistrement courrante comme l'ajout de deux fois le même véhicule, le même conducteur pour deux véhicule différent et l'ajout d'un véhicule alors qu'on fais un débarquement.

1.2.2.2 Les choix sur l'interfaces graphique

Nous avons voulus rester au plus proche de ce qui était demandé. Nous n'avons donc pas changer la couleur (ca pique les yeux, désolé) et nous avons éssayé de rester au plus proche de l'interface qui nous étais présenté en image.

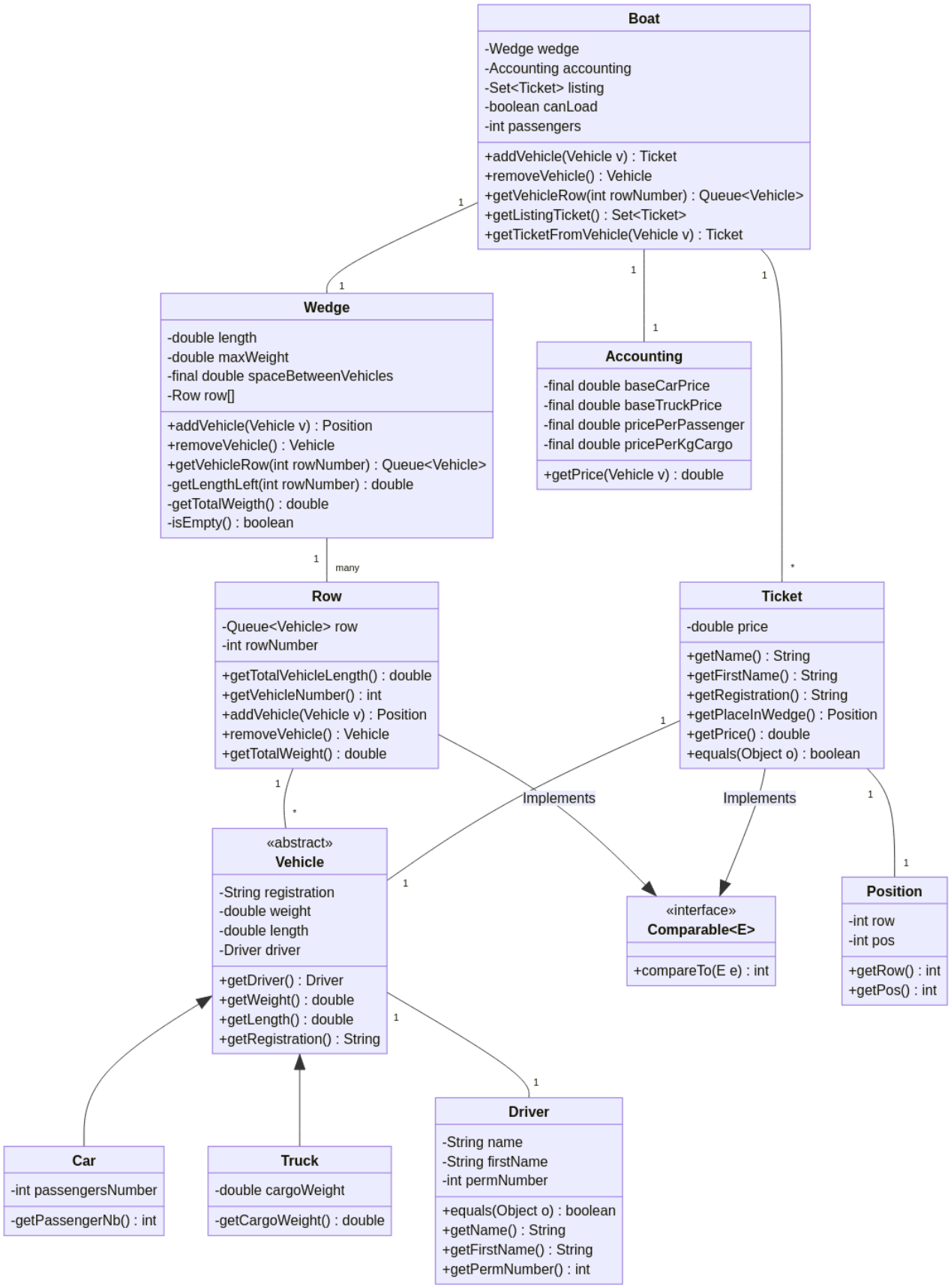
Nous avons cependant fait quelques modifications sur l'affichage des fenêtres. Nous pouvons a la fois ouvrir la fenêtre de la cale et ajouter ou supprimer des véhicules. L'interface de la cale se met automatiquement a jour suite a l'ajout ou la suppression d'un véhicule. Et il y a un affichage de tous les messages d'érreur suite a un ajout impossible du véhicule.

Nous avons aussi fais quelques changement sur la fenêtre d'ajout d'un véhicule. Il y a un formattage automatique des nombres pour le poid du véhicule, le nombre de passagers et le poid de la cargaison. Pour le reste des champs de saisis ils sont seulement vérifiées lors de la validation du formulaire.

Ce sont les seules libertées que nous avons prises par rapport au sujet.

1.3 Conception de l'application

1.3.1 Diagramme



1.3.2 Boat

C'est la classe principale de l'application en mode console. Elle centralise toutes les opérations de l'application et permet la communication entre la cale du bateau (wedge), la billetrie et les prix (Accounting). On a preferer gerer indépendament la cale du bateau pour pouvoir inclure d'autres modules comme un module bar ou un bureau d'échange. Tous les prix serons alors ajustée directement depuis le bateau.

Nous avons chois de lister tous les tickets fait pour les véhicules dans le bateau directement mais nous aurions pus les mettres dans la cale, c'est un choix objectif qui a été fait.

Pour enregistrer les tickets nous avons choisis d'utiliser un Set de la bibliothèque java. Tous les tickets sont unique, il n'y a qu'un conducteur pour chaque véhicule qui ont chancun une plaque d'imatriculation unique. On a aussi besoin de les ordonnées selon le nom du conducteur donc on a une seule façon de faire l'implémentation du Set. On implémente le set avec un TreeSet car on ne veux pas d'éléments nulle et il est plus simple de manipuler les élément a l'intérieur.

1.3.3 Wedge

La Cale est la classe qui regroupe toutes les méthodes relatives a la dispositions des véhicules dans le bateau.

Nous avons implémenté une liste statique de Row (rangée) car une collections était inutiles pour les opérations d'ajout et de suppression selon notre implémentation. De plus une fois la créatrion de l'instance Wedge on a aucun ajout ni suppression de rangée. On a selement besoin d'acceder a des rangées spécifiques.

Notre méthode d'ajout ou de suppression ne prend pas en compte la ligne de flotaison, elle permet simplement d'ajouter un véhicule a la rangée sui a le moins de poids et de retirer le vhéhicule a celle qui a le plus de poids.

1.3.4 Row

La classe rangée (ROW) permet l'ajout et la suppression des véhicules dans la rangée. Elle comporte des méthodes pour connaitre son poid, le nombre de véhicules et les méthodes d'ajout et suppressions de véhicule.

Nous avons décidées d'utiliser une Queue pour le stockage des véhicules. Elle nous permet seulement l'ajout de véhicule en fin de que et la suppréssion en fin de queue. On a préféré utiliser la méthode que pour éviter d'enlever en premier le dernier véhicule ajouté ou l'inverse. On a utiliser une LinkedList qui n'est pas forcément la plus éfficace puisque l'operation d'ajout ou de suppression est de O(n). Cependant elle nous évite d'enlever ou ajouter un véhicule qui physiquement ne peux pas l'être.

Cette classe implémente la classe Comparable pour pouvoir ordonner les classes et touver rapidement celle qui a le plus ou le moin de poid selon notre algorithme.

1.3.5 Accounting

Cette classe a pour simple but de gerer les prix de tous les éléments vendables dans le bateau. Elle enregistre les prix de chaques éléments et on peux connaitre le prix de l'élément en appelan la methode gePrice.

1.3.6 Ticket

Cette classe enregistre les informations relative aux tickets. Elle implémente la classe Comparble pour pouvoir fonctionner avec le Set de la classe bateau (Wedge).

Pour éviter la duplications de variables nous avons directement lier le véhicule au ticket On a ajouter une classe Position pour enregistrer la position du véhicule dans la cale duran le voyage.

1.3.7 Position

Cette classe enregistre la position de la voiture liè au ticket. La rangée est enregisté sous forme d'entier pour permettre de mettre un nombre illimité de rangée. Il est demandé d'afficher la rangée droitre par un D et la rangée guauche par un G. Nou traiterons ces cas par l'affichage et seulement pour un bateau a 2 rangées.

1.3.8 Vehicle

La classe véhicule (Vehicle) est hérité par les camions (Truck) et voiture (Car) permettant de les stockées dans une liste unique de véhicules.

Elle possède les fonctions de base qui sont le conducteur, la plaque d'imaticulation, le poid et la longueur qui sont commune aux camions et voiture. Cette classe est abstraite puisqun ne peux charger un véhicule, seulement des camions et voiture.

Pour parler rapidement des classes voiture et camion. La classe voiture perment d'enregistrer un nombre de passager alors que les camions ne transportent pas des passager mais des cargaisons (sauf transport illégal).

1.4 Développement de l'application

1.4.1 Points intéréssants

1.4.2 Partage du travail

1.4.3 Les résultats obtenus

1.4.4 Ce qui a été testé

1.4.5 Ce qui n'as pas été implanté

1.5 Conclusion

1.5.1 Bilan pour le travail en binôme

1.5.2 Bilan du travail pour la formation

1.5.3 Améliorations possibles mais non réalisées