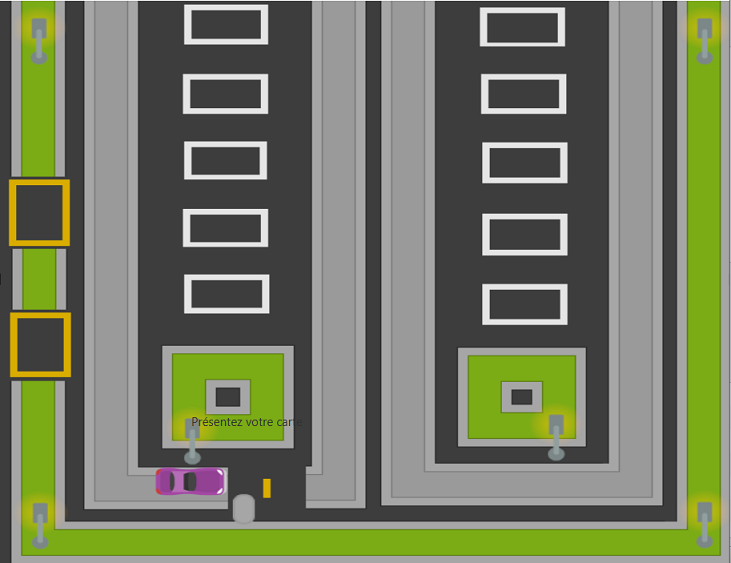
Uml - Java

Projet Parking



Formation TP CDA

Greta des Yvelines : Pôle Guyancourt

Robelot Geoffrey

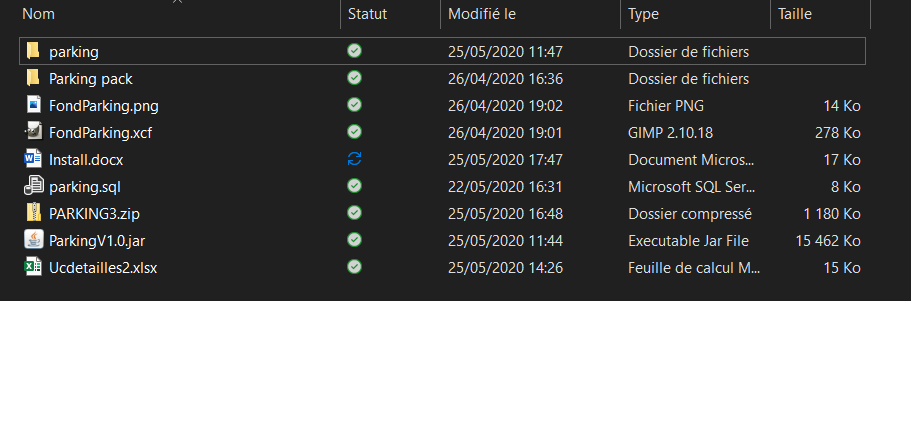
Année 2019 - 2020

Sommaire :

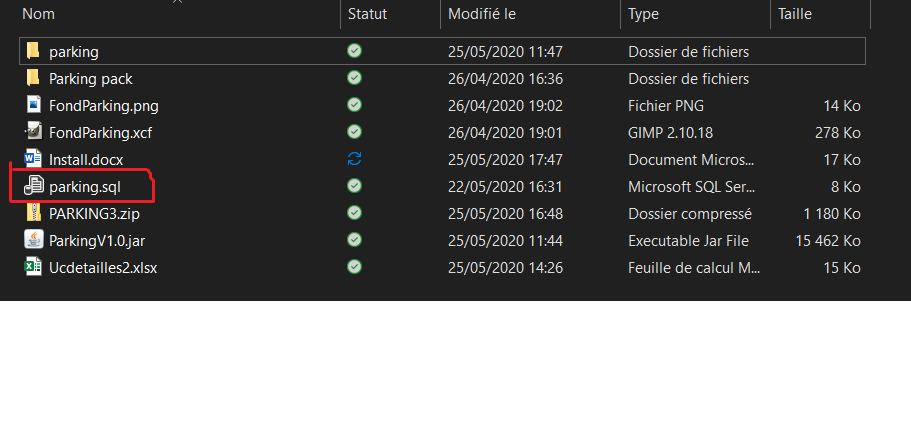
1. Configuration requise ………………………………………………………………………………………………………. p.3
   1. Import de la base de données ……………………………………………………………………………………..…. p.3
   2. Vérification de la version de Java sur la machine …………………………………………………………….. p.4
   3. Gestionnaire de base de données (Xampp ou Wamp) ……………………………………………………… p.5
   4. Exécution du fichier .jar ……………………………………................................................................. p.5
2. Exécution de l’application ……………………………………………………………………………………………….. p.7
3. Conduite de projet ………………………………………………………………………………………………………….. p.8
   1. Participants, planning, livrables ……………………………………………………………………………………… p.8
   2. Démarche utilisée ………………………………………………………………………………………………………….. p.8
   3. Environnement de développement ……………………………………………………………………………….. p.9
4. Analyse …………………………………………………………………………………………………………………………... p.9
5. Conception ……………………………………………………………………………………………………………………… p.9
6. Les tests …………………………………………………………………………………………………………………………. p.11
7. **Configuration requise :**

* Base de données « parking »
* Java avec la version 1.8 ou ultérieure.
  1. Import de la base de données « parking »
* Dézipper le package *Parking* dans le répertoire de votre choix.

*La base de données est fournie dans le package Parking.*



* Ouvrir PhpMyAdmin.
* Créer la base de données « parking ».
* Cliquer sur « Importer » dans le menu du haut.
* Sélectionner la base de données « parking.sql » dans le package *Parking.*



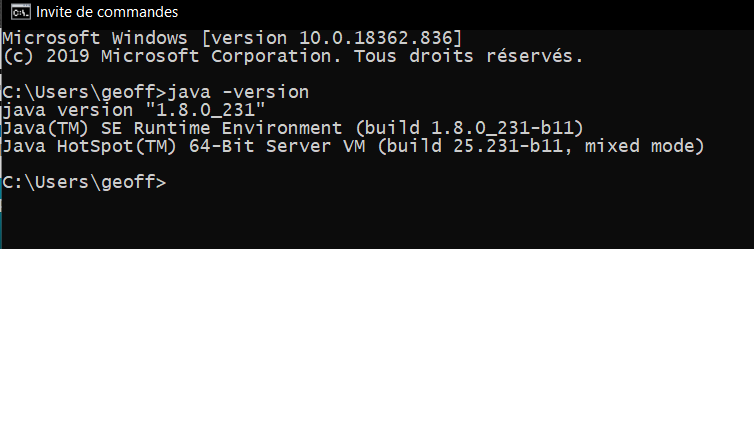
* Cliquer sur le bouton « Exécuter ».

Vous êtes maintenant en possession de la base de données « parking » dans votre gestionnaire de base de données PhpMyAdmin.

* 1. Vérification de la version de Java installée sur la machine

Vérifier la version de l’application Java sur le poste de travail. Pour cela :

* Ouvrir l’invite de commande
* Taper la commande suivante « java -version »



Si aucune application java n’est détectée, il faut la télécharger.

Si la version de java n’est pas assez récente, il faut la mettre à jour.

* 1. Gestionnaire de base de données

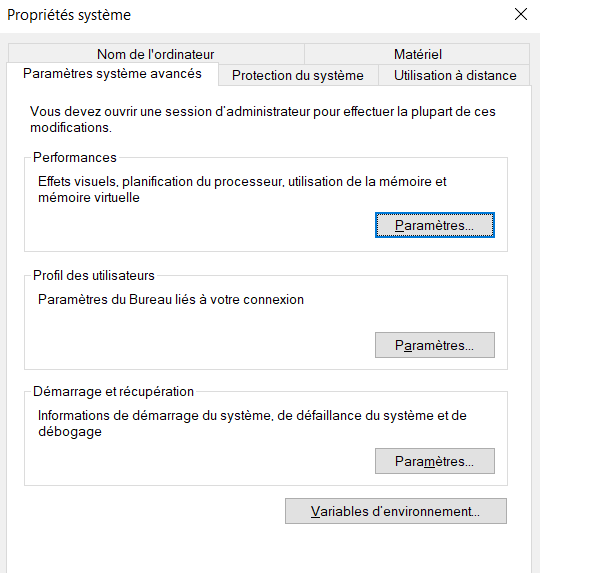
Ouvrir votre gestionnaire de base de données (Xampp ou Wamp) puis démarrer les services Apache et MySQL.

Si ce distributeur de logiciels n’est pas installé sur votre machine, suivre les étapes suivantes :

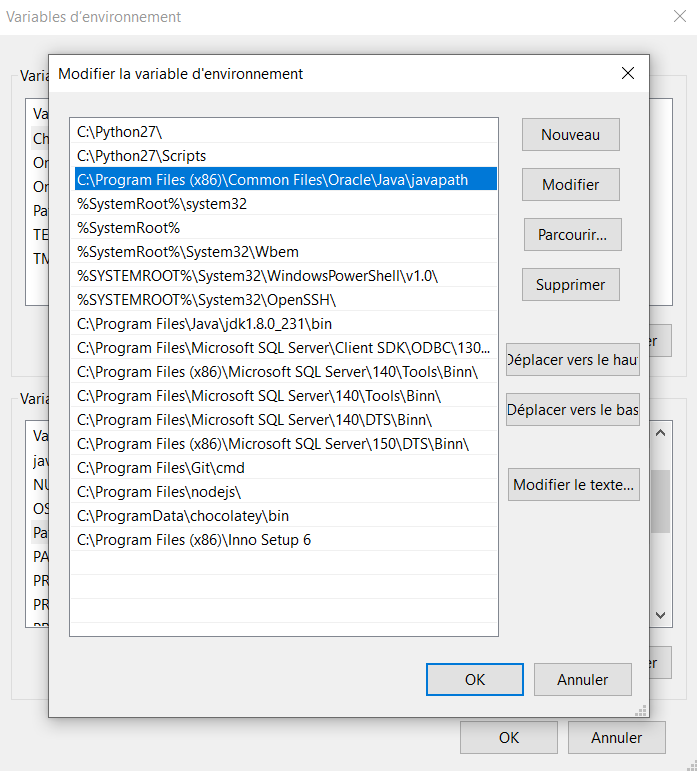
* Choisir l’un des deux distributeurs et le télécharger
* Par défaut, l’installation se fait à la racine de votre disque dur
* Pour Xampp, ouvrir le « control panel »
* Lancer les services Apache et MySQL
  1. Exécution du fichier .jar

Deux solutions sont possibles :

* Placer le fichier exécutable .jar dans le répertoire d’exécution par défaut de java : « c:\Program Files (x86)\JAVA\Jrx\Bin ».
* Ajouter l’application java dans les variables d’environnement : Ouvrir les propriétés système, puis « variables d’environnement ».



Il faut modifier la variable PATH en indiquant le chemin d’accès à l’application java.



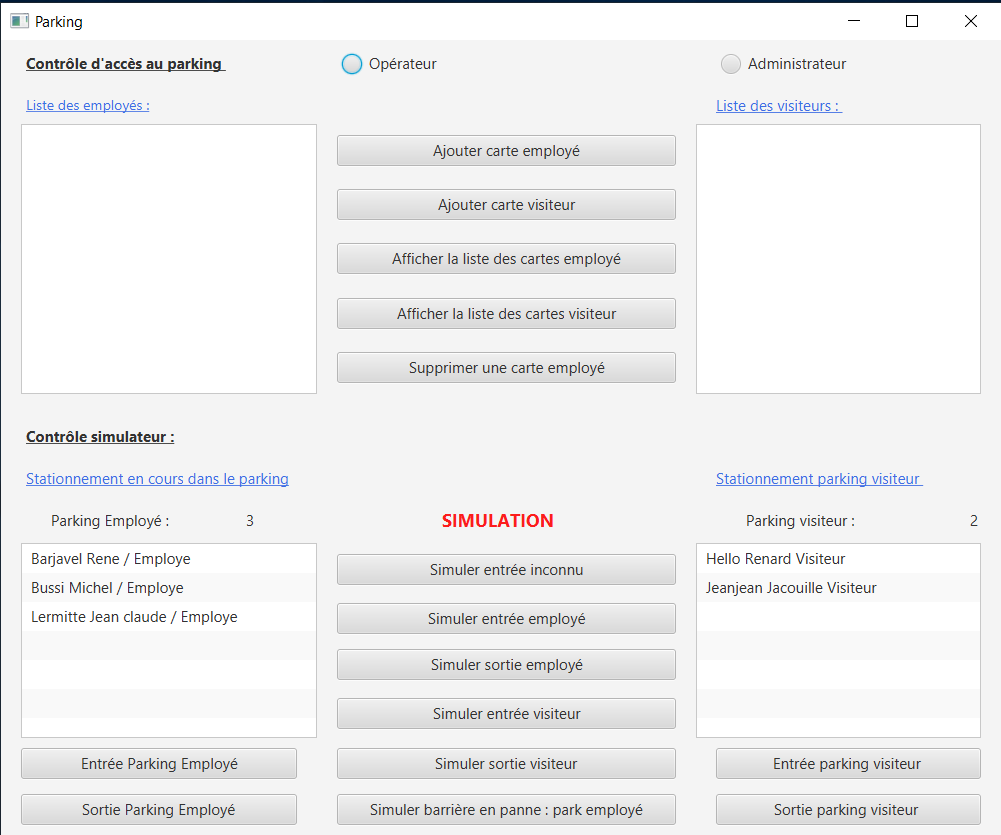
1. **Exécution de l’application :**

Lancer l’exécution de l’application Parking depuis l’invite de commande :

Il est nécessaire de se positionner au niveau de l’emplacement du fichier jar puis de taper la commande suivante : « JAVA ParkingV1.0.jar ».

Si vous avez indiqué le chemin d’accès de l’application Java dans la variable d’environnement Path, vous pouvez simplement double cliquer sur le fichier « ParkingV1.0.jar ».

L’application Parking doit se lancer. Vous obtenez l’interface suivante :



1. **Conduite de projet** 
   1. Participants, planning, livrables

*Participants*

Le projet devait être réalisé par trois participants : Luc Roux, Agnieszka Rogoza et moi. Nous avons tous les trois pris part à la conception du projet : Use Case, Diagramme de classe.   
La crise sanitaire (Covid-19) de ce début d’année 2020 a cependant perturbé la réalisation en groupe de notre projet.   
Il en découle que pour la partie « développement », j’ai réalisé seul le codage de l’application, le maquettage et les différentes interfaces graphiques de cette application et du simulateur.

*Planning*

Nous avions décidé de nous accorder une séance de deux heures par semaine pour mettre en commun la partie UML du projet, travaillée chacun de notre côté en amont.   
  
Côté codage, je me suis fixé comme « contraintes » :

* Deux jours pour la réalisation de l’image de fond du simulateur et de la maquette de l’application.
* Une semaine pour le développement des différentes vues de l’application.
* Une semaine pour la conception et le développement de la base de données.
* Deux semaines pour le codage des différentes classes.

*Livrables*

Le projet doit contenir les livrables suivants :

* Une archive zip des sources
* Une version exécutable, Jar ou Exe
* Une documentation du projet
  1. Démarche utilisée

Après la conception du projet vient la partie codage. Celle-ci traduit les fonctionnalités et les exigences techniques définies lors de la conception.   
Dans ce projet, la partie programmation se fait en Java. Pour entamer le codage de l’application, je me sers du diagramme de classe. S’agissant de l’architecture du projet, j’utilise le pattern MVC (Model, View, Controller).  
Le modèle permet de stocker les données.   
Le contrôleur fait le lien entre la vue et le modèle.   
La vue est ce que l’utilisateur voit de la programmation de l’application.

* 1. Environnement de développement

La partie programmation du projet se fait grâce à l’IDE (Environnement de développement intégré) Eclipse. Cet IDE simplifie le développement d’application Java mais il est aussi considéré comme multi langages.

1. **Analyse**

*L’analyse fonctionnelle d’un projet informatique permet de concevoir un système qui correspond aux besoins des utilisateurs en tenant compte des contraintes.*

Le système Parking doit permettre de gérer les cartes des employés d’une société ainsi que les cartes de ses visiteurs.   
Il permettra aussi de contrôler l’accès des employés et des visiteurs au parking en simulant leurs entrées et sorties.  
Le simulateur du parking doit intégrer une barrière qui s’ouvre et se ferme en fonction de la validité d’une carte.   
Une borne à l’entrée du parking vérifie si la carte est valide ou non.  
Une barrière s’ouvre si la carte est valide. Sinon elle reste en position fermée.   
Un feu bicolore passe au vert lorsque la barrière est ouverte et passe au rouge lorsque la barrière est fermée.

1. **Conception**

Une carte est délivrée à chaque employé de la société ainsi qu’à chaque visiteur. La carte d’un utilisateur indique son numéro de carte, son nom, son prénom, son type (employé ou visiteur) et la date de création de la carte. Celle-ci est stockée dans la table « carte » de la base de données « parking ».

Un stationnement « employé » doit contenir les informations suivantes : le numéro du stationnement, la date d’entrée, la date de sortie (initialisée à nulle tant que l’utilisateur n’est pas sortie du parking) et le numéro de la carte de l’employé. Les stationnements « employé » sont stockés dans la table « stationnement » de la base de données parking.   
  
Un stationnement « visiteur » doit contenir les informations suivantes : le numéro du stationnement, la date d’entrée, la date de sortie (initialisée à nulle tant que l’utilisateur n’est pas sortie du parking) et le numéro de la carte du visiteur. Les stationnements « visiteur » sont stockés dans la table « stationnement visiteur » de la base de données parking.  
Un visiteur qui a une carte peut revenir dans le parking si le nombre de places le permet.

Dans le cadre de ce projet, il est décidé que le nombre de places « visiteur » dans le parking est de 2.   
Le nombre de places « employé » dans le parking est de 10.

L’application devra être composée de deux parties distinctes :

* Une partie contrôle d’accès au parking
* Une partie contrôle simulateur

*Partie contrôle d’accès au parking*

Un opérateur / administrateur pourra :

* Ajouter une carte « employé » ou une carte « visiteur » dans la base de données.
* Supprimer une carte « employé ».

Il n’est pas possible de supprimer une carte « visiteur » afin de conserver une trace des cartes « visiteur ».

Les listes « cartes employé » et « cartes visiteur » seront indépendantes.

*Partie contrôle simulateur*

L’application permettra de :

* Simuler un inconnu qui s’approche du parking
* Simuler l’entrée d’un employé dans le parking
* Simuler la sortie d’un employé
* Simuler l’entrée d’un visiteur dans le parking
* Simuler la sortie d’un visiteur
* Simuler une défaillance de la barrière côté parking salarié
* Faire entrer un employé dans le parking
* Faire sortir un employé du parking
* Faire entrer un visiteur dans le parking
* Faire sortir un visiteur du parking

Un compteur « parking employé » actualisera en temps réel le nombre d’employés dans le parking.   
Un compteur « parking visiteur » actualisera en temps réel le nombre de visiteurs dans le parking.   
Si un employé se présente alors que le parking est plein, un indicateur affiche « Complet ».   
Si un visiteur se présente alors que le parking est plein, un indicateur affiche « Complet ».

Un utilisateur présent dans le parking ne peut pas se représenter à l’entrée du parking. Une alerte affichera un message d’erreur « utilisateur déjà présent dans le parking ».

1. **Tests**

Des tests sont réalisés lors du développement du projet grâce à des classes spécifiques de test.  
Une classe TestDao permet de tester les méthodes la Data Access Object class.   
Une classe MainTestParking permet de tester le fonctionnement du système parking.