Projet Java IV 2021-2022

HELBPark 2.0

Introduction :

La société HELBPark possède un parking de 20 places. Le parking possède certaines particularités qui le différencient d'un parking conventionnel.

Par exemple, le parking ne propose que des forfaits à la journée. La journée commence à 1h du matin (01:00) et se termine à 23h en soirée (23h:00). Les voitures se situant encore dans le parking après 23h son amenée à la fourrière et le parking est donc vidée chaque jour si des voitures ne sont pas récupérées par leur propriétaire.

Chaque place de parking est numérotée de 0 à 19. Quand un véhicule se présente à l'entrée du parking, une place lui est attribuée. L'attribution des places se fait toujours de la façon suivante, on attribue la place avec le plus petit numéro libre. Par exemple, supposons que toutes les places sont occupées à l'exception de la place 4 et 11, le prochain client se verra attribuer le place numéro 4. Le client ne peut pas choisir sa place.

Le parking n'accepte que trois types de véhicules différents, les motos, les voitures et les camionnettes. Toutes les places peuvent être occupées par tous les types de véhicule. Le prix de base d'une place varie en fonction du type de véhicule : 10 euros pour les motos, 20 euros pour les voitures et 30 euros pour les camionnettes.

Le parking dispose toutefois d'un système de réduction complexe basé sur les jours de la semaine :

Lundi	Prix de base.
Mardi	Moitié prix pour les motos.
Mercredi	25% de réduction pour les véhicules dont la plaque d'immatriculation contient la lettre « P ».
Jeudi	Prix de base.
Vendredi	Moitié prix pour les camionnettes.
Samedi	Si le jour est pair (ex : 14 mars) moitié prix, sinon prix de base.
Dimanche	Prix de base.

Malheureusement, à cause de la pénurie de composants électroniques, le parking ne peut pas être complétement automatisé. Quand un véhicule quitte le parking, cela n'est pas détecté automatiquement, il y a donc un employé chargé de vérifier pendant la journée si une place se libère, et quand c'est le cas, l'employer doit le notifier dans le système informatique de l'application.

L’employé est également mobilisé quand les clients quittent le parking. Ceux-ci doivent payer chez l’employé qui leur remet un ticket de caisse.

Enfin, le parking travaille en collaboration avec le cinéma situé à proximité. Quand un client se présente à l’employer, au moment de quitter le parking, celui-ci se voit attribué un ticket jeu qui permet d’octroyer une réduction sur le prochain ticket de cinéma. Ce ticket jeu est inclus au ticket de caisse du parking. Il existe trois types de tickets jeu, les tickets « standard », les tickets « silver » et les tickets « gold ».

- Les tickets standard contiennent tous un code promo et une valeur qui est soit 5% ou 10%.
- Les tickets silver contiennent tous un code promo et une valeur qui est soit 10% ou 15%, ainsi que 2 symboles, chacun tiré au hasard parmi O,X, et P. Si les deux symboles sont identiques, alors la valeur du ticket de réduction est doublée. Par exemple, le ticket contenant la valeur 10% et les deux symboles « X , P » offre une réduction de 10%, tandis que le ticket contenant la valeur 15% et les deux symboles « X, X » offre une réduction de 30%.
- Les tickets gold contiennent tous un code promo et une valeur qui est soit 20% ou 40%, ainsi qu’une grille 3x3 et dont chaque symbole de la grille est tiré au hasard parmi les lettres de « PARKHELB », si la grille contient une ligne ou une colonne avec deux lettres identiques, alors la valeur du ticket de réduction est doublée.

Chaque code promo est unique. Les tickets sont imprimés par l’employé qui garde chaque ticket du système dans un répertoire dont le nom est la date du jour au format « DDMMYY ».

Chaque ticket est stocké dans un fichier texte dont le nom du fichier est la plaque d’immatriculation du véhicule suivi de _std ou _sil ou _gol en fonction du type de ticket.

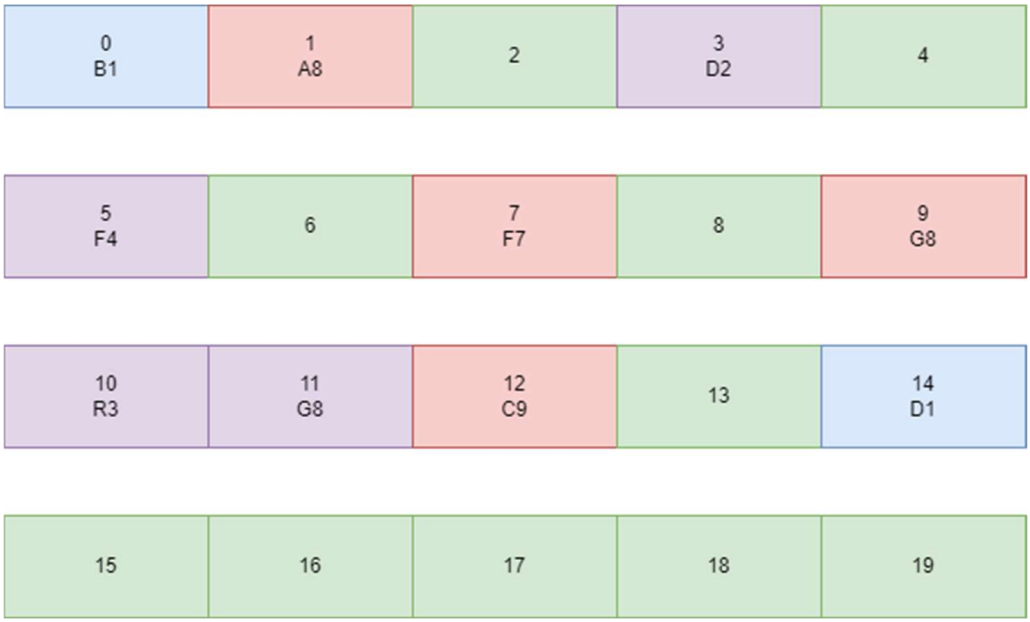
Les tickets ne sont pas tout à fait attribués au hasard. Quand un client vient au parking, il a la possibilité de recevoir n’importe quel type de ticket mais, un client avec une moto aura + de chances de recevoir un ticket standard, un client avec une voiture aura + de chances de recevoir un ticket silver, et un client avec une camionnette aura + de chances de recevoir un ticket gold.

Le concept du parking est très jeune, donc il est fort possible que des choses changent à l’avenir. Par exemple il pourrait y avoir de nouveaux types de véhicules. Certaines réductions pourraient être ajoutées, modifiées ou supprimées. Les tickets de promotion cinéma pourraient aussi évoluer. La technologie de l’interface graphique pourrait changer aussi.

Système visible par l'employé :

Afin de pouvoir travailler correctement, l'employé doit être capable sur un écran de monitoring de visualiser globalement les places de parking, si elles sont occupées ou non, et pour chaque place le type de véhicule présent ainsi que la plaque d'immatriculation du véhicule. Cet écran est dynamique et évolue en fonction des voitures qui entrent dans le parking.

Afin d'illustrer, voici un schéma de cette partie de l'application avec un formatage et une mise en forme minimale :



Le code couleur est le suivant :

- Vert : la place est libre
- Bleu : la place est occupée par une moto
- Rouge : la place est occupée par une voiture
- Mauve : la place est occupée par une camionnette

On voit dans cet exemple que la place numéro 12 est occupée par une voiture dont la plaque d'immatriculation est C9. Dans l'entièreté du projet, on fera une simplification pour le numéro d'immatriculation. On considère que ceux-ci ne sont composés que d'une lettre suivit d'un chiffre.

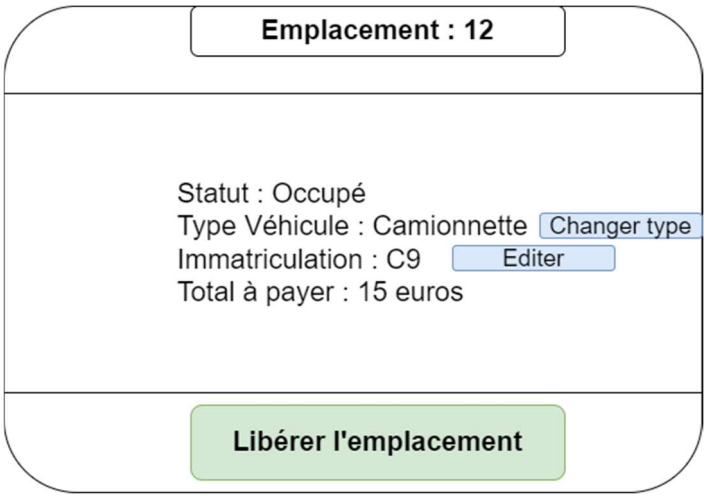
A partir de cet écran, il est possible d'éditer une place de parking en particulier, par exemple, pour libérer la place quand un véhicule s'en va ou encore pour corriger une donnée au cas où celle-ci serait erronée.

Supposons que, ce vendredi 1er avril 2022, le véhicule C9 situé à la place numéro 12 décide de quitter le parking. Ce client va alors chez l'employé pour payer, il reçoit ensuite un ticket de caisse. Il s'agissait d'un ticket gold. L'employé va alors trouver le fichier C9_gol.txt dans le répertoire du jour (010422) pour l'imprimer. Voici à quoi ressemble le ticket de caisse.

Date : 01/04/2022
Place occupée : 12
Type véhicule : Camionnette
Immatriculation : C9
Prix de base : 30 euros
Réduction : Vendredi – Moitié Prix Pour les Camionnettes.
Total à payer : 15 euros.
Code Promo Ciné : BTF1989
Ticket Gold : Valeur : 20%
Jeu :

AKE
RHL
ELB

L'employé doit ensuite libérer la place de parking. Il clique alors sur l'emplacement 12 et se voit proposer une interface d'édition. Encore une fois, afin d'illustrer, voici un schéma de cette partie de l'application avec un formatage et une mise en forme minimale :



Il s'agit d'une fenêtre contenant des informations de base sur l'emplacement numéro 12. Le bouton, « libérer l'emplacement » permet comme son nom l'indique de changer son statut. Il est également possible pour l'employé de changer le type du véhicule ou le numéro d'immatriculation de celui-ci s'il s'avère incorrecte.

Travail demandé :

Pour ce projet, vous allez devoir implémenter **une application JavaFX** en appliquant les concepts vus en cours tel que **la conception pilotée par les responsabilités** et **l'application adéquate de design patterns**.

Il vous est demandé de concevoir et développer **le programme utilisé par l'employé du parking**.

Le projet devra être **développé individuellement**.

Contrainte de développement :

- Votre programme devra être compilable et exécutable dans un environnement Linux Ubuntu tel qu'une des machines virtuelles utilisées en cours et disponible sur le SharePoint d'eCampus. Un script bash nommé « run.sh » devra permettre la compilation et l'exécution de l'application en utilisant seulement la commande suivante «bash run.sh » en terminale. Si le fichier n'est pas fourni, ou si la compilation et l'exécution ne fonctionnent pas dans l'environnement spécifié, le projet ne sera pas corrigé et sanctionné d'un **zéro**. **Testez donc avant que tout fonctionne !**

- Votre programme devra prendre en compte le fichier de simulation.

- A l'exception des commentaires, l'entièreté de votre programme devra être codé en anglais (nom de variables/fonction/classes/etc…).

- Votre code devra respecter les principes de designs orientés objet comme vu au cours. Pensez donc à faire des choix logiques de design de classes afin de produire un code propre et maintenable.

- Votre code devra présenter une structure correcte et maintenable. Notamment :
    - Evitez la duplication de code.
    - Evitez les constantes magiques.
    - Evitez le code mort.
    - Commentez intelligemment et suffisamment votre code.
    - Nommez correctement vos variables (ex : pas de majuscule comme première lettre).

  Un code dont la qualité sera jugée insuffisante sera sanctionné d'un zéro.

- Votre programme devra être développé en utilisant **exclusivement la librairie graphique JavaFX**.

De manière générale, l'application devra faire un usage suffisant et correcte des concepts vus au cours.

Les contraintes et fonctionnalités du projet sont susceptibles d'évoluer au cours du temps. Pensez donc à adapter une stratégie de développement agile.

Note : certains points de la description ne sont pas précisés ou sont laissés volontairement vagues. Il revient à vous de faire certains choix d'interprétations. Veillez toutefois à ce que votre approche soit logique et justifiée.

Simulation :

Afin de tester et simuler votre code en situation, un fichier avec des arrivées de clients est disponible sur eCampus. Le fichier est composé de plusieurs lignes et chaque ligne décrit l'information suivante :

[Nombre de secondes, type véhicule, immatriculation]

Pour la première ligne, « Nombre de secondes » est le nombre de secondes depuis le début du lancement du programme. Pour toutes les lignes suivantes, c'est le nombre de secondes depuis l'arrivée dans le parking du véhicule précédent. Ceci permet donc de simuler le flux de clients avec une moyenne de 5 à 10 secondes entre chaque client.

Par exemple :

2,camionnette,C9
6,voiture,B8
…

Ceci signifie qu'après 2 secondes une camionnette, immatriculée C9, entre dans le parking. Ensuite, 6 secondes après, une voiture immatriculée B8 entre à son tour dans le parking.

Votre programme devra **obligatoirement pouvoir être lancé avec le fichier de simulation**.

Rapport :

Il vous est demandé de rédiger un rapport décrivant votre projet. Votre rapport devra contenir au minimum les sections suivantes :

- Introduction
- Présentation de l'interface graphique
- Analyse et Application des Design Patterns
- Limitations
- Conclusion

Introduction : Cette section devra introduire votre projet. Décrire ce qui a été réalisé et présenter brièvement la structure de votre rapport.
Présentation de l'interface graphique : Cette section devra illustrer et expliquer la partie de votre application qui concerne l'interface graphique.
Analyse et Application des Design Patterns : Cette section devra expliquer la structure de votre implémentation en utilisant les outils d'analyses déjà vus durant votre parcours. Attention tous les diagrammes doivent être commentés. Vous devrez également dans cette section décrire les patterns utilisés et expliquer comment et pourquoi vous les avez utilisés. Les diagrammes d'analyse pourront vous aider à illustrer vos propos.
Limitations : Les limites de votre application, par exemple : dans quels cas d'utilisation votre application pourrait ne pas fonctionner comme prévu ? Y a-t-il des aspects techniques qui n'ont pas été traité ? Si vous aviez plus de temps pour le projet, qu'auriez-vous amélioré ? Plusieurs points de vue sont possibles, il revient à l'étudiant de choisir les points qu'il considère les plus pertinents pour réaliser son autocritique.
Conclusion : Votre conclusion sur le projet. Ce que vous avez réussi à faire ou non durant le projet et les apprentissages que vous en tirez.

Le rapport sera notamment évalué sur la qualité écrite et l'effort de présentation ainsi que la pertinence et la complétude des points abordés.

Deadline et remise :

La date limite pour la remise du projet est le **Lundi 30 Mai à 23h59**. Le projet devra être déposé sur eCampus à l'intérieur d'un fichier .zip contenant toutes les sources de votre projet ainsi que le rapport **au format PDF** (et pas word).

Développement et Triche :

- Tout acte de triche sera sanctionné par **un zéro pour l'entièreté du projet**. Des parties de code réutilisés d'un projet existant (d'un autre étudiant ou

disponible sur le net) sans références dans votre rapport et sans mention de l'utilité du code utilisé est considéré comme une fraude.

- Pour ce projet, vous ne pouvez pas reprendre des parties du code d'un autre étudiant (que ce soit de cette année ou d'une des années précédentes).

- Pour ce projet vous pouvez vous inspirer/servir d'un code disponible sur internet. N'oubliez pas toutefois d'en mentionner la source.

- Si vous avez un doute, contactez l'enseignant le plus tôt possible afin d'éviter du refactoring inutile, ou pire, **un zéro**.

Conseil pratique :

Voici quelques conseils qui j'espère pourront vous aider.

- Veillez à ce que votre code ne contienne pas de constantes magique et/ou de duplication qui serait facilement évitable avec l'utilisation de méthodes.
- Veillez à effectivement implémenter les designs patterns quand cela est pertinent.
- Réfléchissez en terme d'« attribution de responsabilités » en accord avec les principes vu au cours (segmentation logique en classes).
- **Ne négligez pas la théorie du cours (vous serez interrogés dessus).**
- Ne négligez pas votre rapport. Tachez d'y expliquer/justifier explicitement vos choix d'implémentation. (Exemple : pourquoi un héritage ici ? pourquoi avoir choisi (ou non) d'implémenter ce design pattern ? quel avantage en termes de structure ? …)
- Prenez le temps de bien comprendre tout l'énoncé avant de vous lancer (et lisez la FAQ).
- **Vérifiez que votre code compile et run effectivement via le script bash prévu sur la machine Ubuntu présente dans le SharePoint du cours.**

FAQ :

- **Puis je ajouter d'autres sections ou sous-sections dans le rapport ?**

Oui. La partie rapport de ce document donne seulement la structure minimum.

- **Puis je coder ou rendre mon rapport en anglais ?**

Oui. Pour ce qui est du code vous devez obligatoirement coder en anglais.

- **Puis je programmer sur Windows avec Eclipse ?**

Oui vous pouvez programmer comme vous le désirez mais vous devez respecter les contraintes de ce document, notamment : votre code doit être exécutable sur un environnement Linux Ubuntu via un script run.sh que vous devez fournir en

même temps que vos sources (voir les contraintes de développement). Une machine préconfigurée sera disponible sur le SharePoint, celle-ci pourrait être utilisée pour la démo de votre application lors de la défense du projet.

- **Le rapport est-il important ?**

Oui. Le rapport est une **pièce centrale de votre projet** et c'est le premier outil de communication qui me servira à juger de la bonne réalisation du projet, pas seulement du point de vue du code mais également de la méthodologie utilisée. Il convient pour le rapport d'être pertinent et complet quant aux informations contenues.

- **Que voulez-vous dire par « tous les diagrammes doivent être commentés ».**

Les diagrammes doivent servir à illustrer et appuyer vos explications sur la structure de votre implémentation. Ils ne remplacent aucunement un texte explicatif revenant sur les points d'attention.

- **Je n'ai pas réussi à tout réaliser. Est-ce que ça vaut la peine de vous rendre le projet ?**

Oui, veillez toutefois à être claire sur les parties non implémentées. Il est très déconseiller de dissimuler ou d'« oublier » de mentionner qu'une partie n'a pas été réalisée. Veillez toutefois à bien respecter les consignes. Par exemple, votre code doit pouvoir compiler avec le script bash demandé, la qualité du code doit être suffisante, etc…

- **Puis je réaliser le projet en groupe ?**

Non, le projet doit être réalisé individuellement.

- **Que voulez-vous dire par « Votre code devra respecter les principes de design orienté objet comme vu au cours. »**

L'orienté objet fait intervenir certains principes comme l'héritage ou les méthodes statiques. Il revient à vous de décider quand les mettre en œuvre ou non. Votre approche devra toutefois être logique et justifiée. Cela implique notamment, de faire apparaitre de l'héritage quand cela a du sens, de rendre une classe abstraite quand cela a du sens, d'utiliser intelligemment l'encapsulation etc…

- **Dois-je vraiment faire de l'orienté objet ? Mon programme peut fonctionner sans.**

Vous devez absolument mettre en œuvre l'orienté objet pour ce projet. Cela fait partie des contraintes du projet. Un non-respect de ces contraintes sera pénalisé. L'utilisation de l'orienté objet n'est pas une contrainte faible. Veillez donc à la respecter.

- **Doit-on faire de l'encapsulation ou de l'héritage et du polymorphisme ?**

Vous devez toujours en faire usage quand cela a du sens et ne pas le faire est généralement un signe de mauvaise conception.

- **Dois-je vraiment utiliser des design patterns ? Mon programme peut fonctionner sans.**

Vous devez absolument mettre en œuvre les design patterns pour ce projet. Cela fait partie des contraintes du projet. Un non-respect de ces contraintes sera pénalisé. L'utilisation des design patterns n'est pas une contrainte faible. Veillez donc à la respecter. Il ne s'agit toutefois pas d'utiliser les patterns sans réflexion quant à leur utilité dans le projet. Leur utilisation et choix ou non choix doit être pertinente et justifiée. Une partie non négligeable des points sera donnée pour l'implémentation adéquate des designs patterns comme vu au cours

- **L'aspect graphique et l'ergonomie sont-ils des critères importants ?**

Ces critères seront susceptibles d'être pris en compte dans l'évaluation mais sont nettement moins importants que l'implémentation des fonctionnalités et le respect des contraintes.
Dis grossièrement : Mieux vaut un programme moche mais avec toutes les fonctionnalités implémentées qu'un programme magnifique mais avec des fonctionnalités manquantes et un design logique (architecture du code) désastreuse.

- **Puis je utiliser un outil comme Scene Builder pour la création d'interfaces graphique ?**

Oui tant que le code généré fasse partie de la bibliothèque d'interfaces utilisateur JavaFX.

- **Puis je utiliser des designs patterns qui n'ont pas été vu au cours ?**

Oui mais vous devez les définir, justifier leur utilisation et fournir des sources. Vous devez dans tous les cas justifier leur utilisation.

- **L'interface graphique doit-elle être exactement celle présentée dans l'énoncé ?**

Les informations et fonctionnalités présentes dans les maquettes doivent apparaitre dans l'application mais vous êtes libres d'ajouter des informations ou fonctionnalités ainsi que de parfaire l'esthétique et l'ergonomie.