[1]

TP2 – Monopoly

Objectifs du travail

- Utiliser les tableaux
- Utiliser les traitements d'exceptions
- Programmation orientée-objet :
    - Encapsulation, Héritage, Polymorphisme et classes abstraites
    - Classes, objets, constructeurs, getter, setter etc.
- Utiliser la sérialisation pour enregistrer et lire les fichiers binaires
- Savoir bien organiser son code en méthodes et sous-méthodes

Modalités de remise

- Vous devez remettre, **sur Léa**, un fichier zip comprenant :
    - Le projet complet;
    - Une auto-évaluation de votre programme (ce qui fonctionne très bien, moins bien, pas du tout et pourquoi).
- Le travail est individuel.

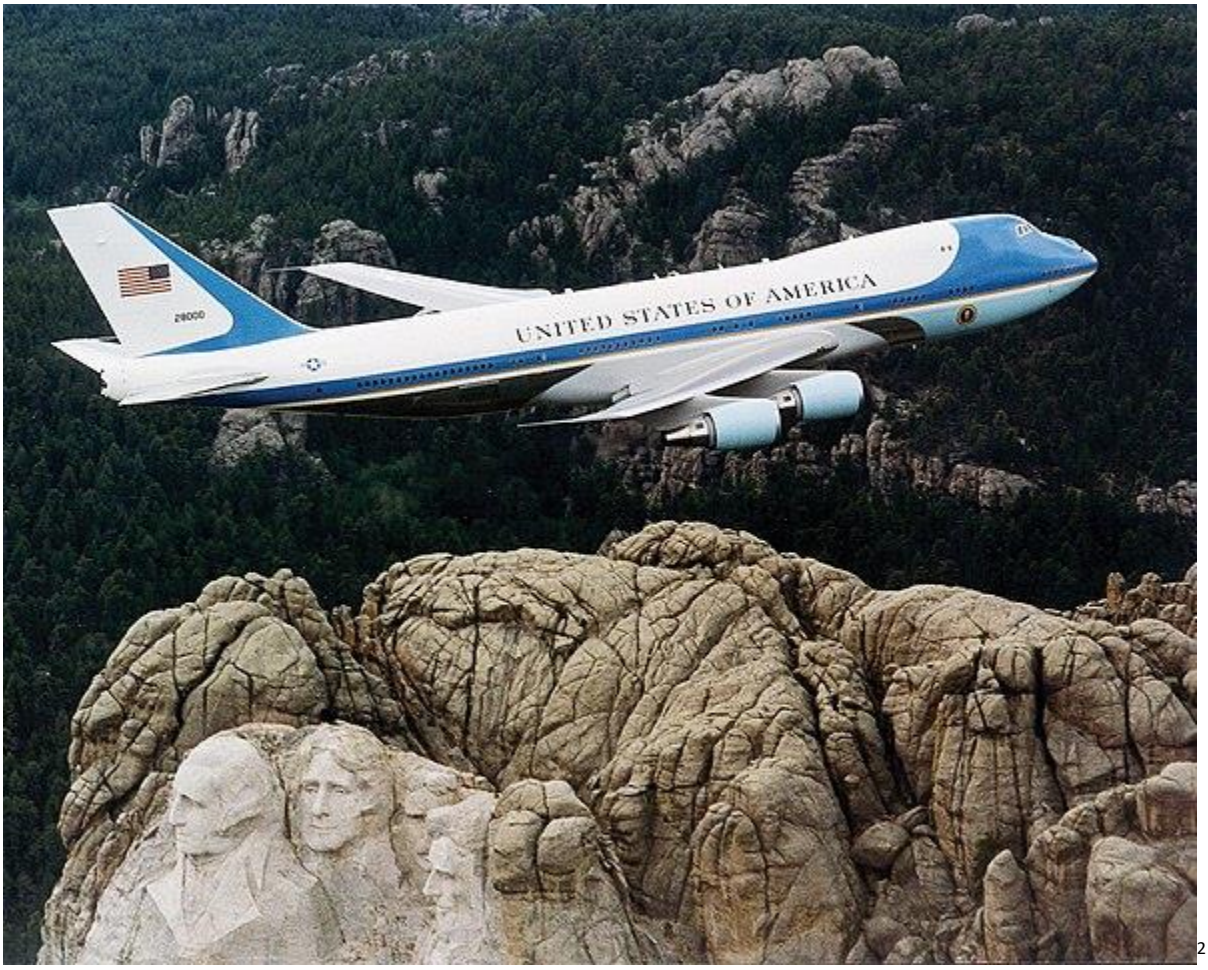
Mise en situation

Pour fuir l'hystérie collective entourant la pandémie de coronavirus (COVID-19). Notre cher voisin Donald décide de s'enfuir à bord de son jet privé (rempli de rouleaux de papier hygiénique et de masques N95) sur une île privée perdue dans l'océan Pacifique.

Malheureusement (ou heureusement pour nous) il a oublié d'emporter avec lui son jeu favori : Le Monopoly.

Il organise donc un concours pour lui fournir le meilleur jeu de Monopoly possible qu'il pourra télécharger sur Léa (car un de ses amis russe a réussi à trouver une faille de sécurité dans le système). Le gagnant pourra le rejoindre sur son île.

[1] https://en.wikipedia.org/wiki/Monopoly_(game)#/media/File:Monopoly_pack_logo.png



[2]

Consignes

Pour ce TP, vous devrez coder un jeu fonctionnel et simplifié de Monopoly :

Le Plateau de Jeu peut accueillir de 2 à 3 joueurs et il est composé de 10 cases, il y a 4 types de cases :

- Cases propriété
- Cases taxes
- Case départ
- Case stationnement gratuit

Toutes les cases ont ceci en commun, elles ont un nom, une description et on peut effectuer une action lorsqu'on atterrit sur l'une d'elle, l'action est déterminée par le type de case et s'applique au joueur courant.

[2] https://upload.wikimedia.org/wikipedia/commons/thumb/7/7d/Air_Force_One_over_Mt._Rushmore.jpg/597px-Air_Force_One_over_Mt._Rushmore.jpg

Les cases propriété :

Elles sont de 2 types :

- Les Terrains : Le prix du loyer est fixe, il est doublé si le joueur possède plus d'une propriété.
- Les services publics : le loyer est 10x la valeur du dé qui a servi à atterrir sur cette case.

Toutes ont en commun : un propriétaire (qui peut être « `null` »), un prix d'achat, un loyer.

Le plateau de jeu

Le plateau de jeu :

- Sera chargé à partir d'un fichier texte nommé `plateau.csv` qui contient les cases avec leur type et le prix des propriétés. Ce fichier sera à la racine du projet;
- A une taille fixe (10 cases);
- Est considéré comme valide s'il a au moins une case de chaque type, commence par la case départ et ne contient qu'une seule case départ.

Déroulement du jeu

Un menu présent au début du jeu doit permettre à l'utilisateur de charger la partie précédente, de créer une nouvelle partie ou de quitter le jeu.

Lorsque l'utilisateur crée une nouvelle partie, vous devrez commencer par charger le fichier plateau.csv et en valider le contenu. Ensuite, vous devrez saisir les noms des différents joueurs. La partie ne peut pas démarrer s'il y a moins de 2 joueurs et elle démarre automatiquement après la saisie du 3ème joueur. Le tableau n'a pas besoin d'être chargé si vous charger la partie précédente puisque les informations du tableau seront contenues dans la sauvegarde.

Chaque joueur commence la partie avec 400$ en poche.

Les joueurs jouent à tour de rôle:

1. Au début de chaque tour un menu propose :
    a. De lancer le dé.
    b. De quitter en sauvegardant, (un fichier binaire sera utilisé pour sauvegarder l'état de la partie)
    c. De mettre fin à la partie, le joueur le plus riche (argent en sa possession et valeur de ses propriétés) est alors désigné comme gagnant.
2. Le joueur lance 1 dé (génération de nombres aléatoires de 1 à 6) et se déplace du nombre de cases correspondant au résultat.
3. En arrivant sur la case, on affiche le nom de la Case, le nom du joueur et le montant d'argent qu'il possède puis l'action correspondant au type de case est exécutée.
4. Le tour est fini, on passe au joueur suivant.

Les actions

Les actions seront gérées par la méthode abstraite `public void effectuerAction(Joueur j, int valeurDe)` qui sera héritée de la classe abstraite `Case`.

À chaque case visitée (le joueur passe par cette case sans s'y arrêter) on appellera la méthode abstraite `public void survolerCase(Joueur j)` pour gérer les cases de type Départ et Taxe. Cette méthode sera aussi héritée de la classe abstraite `Case`.

Les actions possibles sont :

- Cases propriété :
  - Acheter si elle n'a pas de propriétaire.
  - Payer le loyer si la propriété appartient à un autre joueur.
    - Si le joueur n'a pas assez d'argent pour payer le loyer, il fait alors faillite.
  - Si c'est un service public, le joueur doit payer 10$ x le montant la valeur du dé qui l'a fait atterrir sur cette case
  - Si la propriété appartient au joueur, il n'a rien à payer.
- Case départ :
  - Si lors d'un lancer de dés le joueur s'arrête sur cette case ou la dépasse il touchera 50$
- Case taxe :
  - Le joueur qui s'arrête sur cette case doit payer le montant de la taxe, s'il n'a pas assez d'argent il fait faillite.
  - Le joueur qui dépasse cette case doit payer 10% de la taxe.
- Case stationnement gratuit :
  - Aucune action particulière.

Fin de partie

La partie termine lorsqu'un joueur fait faillite, le gagnant est alors le joueur le plus riche (argent en sa possession et valeur de ses propriétés).

Directives de programmation

Découpage du programme

Votre classe principale sera nommée `Tp2`. C'est dans cette classe que sera le `main`.

Vous devrez avoir une classe Partie qui devra conserver l'état d'une partie. Cet état devra contenir :

- Le tableau de jeu
- Les joueurs et les informations sur les joueurs
- Un indicateur pour savoir qui est le prochain joueur à jouer.

C'est cette classe qui sera sérialisée afin de créer votre fichier de sauvegarde.

Votre classe Tp2 sera responsable de :

- La lecture du fichier `plateauDeJeu.csv`;
- La lecture et l'écriture du fichier de sauvegarde `sauvegarde.bin`.
- La communication avec la classe Partie afin de s'assurer du bon déroulement de la partie.

Vous devrez ensuite définir au minimum les classes suivantes :

- `Joueur`
- `Case (abstraite)`
  - `Propriete (abstraite) :`
    - `Terrain`
    - `ServicePublic`
  - `Taxe`
  - `Depart`
  - `StationnementGratuit`

Vous devrez utiliser la classe De fournie dans le fichier « `De.java` »

À chaque tour, vous devrez afficher l'état complet de la partie. Ce qui inclus :

- Les informations sur le tableau
- Les informations sur les joueurs
- Qui est le prochain joueur

Pour afficher les informations sur une case, vous devrez utiliser la méthode `toString()`.

Programmation

Aucune méthode n'aura de clause `throws` dans sa déclaration.

Aucune exception de type `NullPointerException`, `ArrayIndexOutOfBoundsException`, `StringIndexOutOfBoundsException`, `NegativeArraySizeException` ou `Exception` ne doit être traitée par un `try..catch`. Le programme doit faire les vérifications nécessaires pour éviter qu'elles soient lancées.

Aucun attribut de type `public` ou sans modificateur, tous les attributs devront être `private`.

Votre code devra être dans un package nommé `ca.qc.bdeb.info202.tp2`.

Votre code ne devra pas planter sur des entrées invalides ou des fichiers manquants. Vous devrez dans ces cas, afficher des messages d'erreur pertinents.

Votre code sera découpé en méthodes paramétrées. L'utilisation de la clause `static` est permise seulement pour les déclarations des méthodes de la classe `Tp2` et les constantes.

Votre code sera conforme aux directives du guide de style Java disponible sur Léa.

Exemple d'exécution

Voir la trace fournie en annexe.

Critères d'évaluation

Critère	Nombre de points accordés
Qualité du code • Traitement adéquat des exceptions • Utilisation des classes spécifiées par l'énoncé • Découpage des méthodes • Pertinence des choix algorithmiques • Utilisation correcte des éléments du langage	**/30**
Traitement du fichier csv et du fichier de sauvegarde • Lecture correcte du fichier csv • Sauvegarde et Chargement de partie	**/30**
Résultats • Déroulement du jeu • Gestion des actions • Respect des consignes • Affichage du gagnant	**/40**
Total	**/100**

NOTES : La qualité de la programmation et le fonctionnement sont deux critères différents. Toutefois, pour courir la chance d'avoir tous vos points pour la qualité de la programmation, vous devez tenter d'avoir fait l'ensemble du travail. Si des parties du travail sont absentes, vous serez pénalisé dans la qualité de la programmation en proportion de la partie du travail qui n'a pas été faite.

Le non-respect des directives de style Java entraîne une perte de points de présentation pouvant aller jusqu'à 10% de la note.