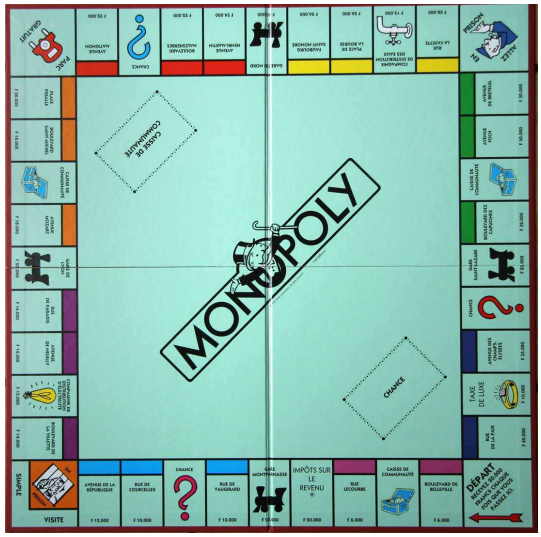
1 Descriptif du projet

On souhaite réaliser une version informatique du célèbre jeu de plateau « Monopoly », dont l'objectif est d'éliminer les autres joueurs en les ruinant à travers une concurrence foncière débridée. Ce projet sera réalisé en Java avec interface graphique en Swing. Les caractéristiques du jeu (lieux représentés sur le plateau, nature et prix des propriétés, événements, etc.) doivent être personnalisables et permettre des extensions.

Règles du jeu

Chaque joueur, doté initialement d'une certaine somme en espèces (150.000F), est représenté par un pion dont les déplacements sont principalement régis par le tirage de deux dés. Lorsqu'un pion arrive sur une case, plusieurs événements sont susceptibles de se produire :

- Si la case contient une propriété : le joueur doit acquitter un certain **loyer** au propriétaire, dont le montant dépend de la nature de la case et de la présence éventuelle de constructions (maisons ou hôtel). Si la propriété n'a pas de propriétaire, le joueur peut décider (ou non) de l'acheter.
- Si la case contient un « événement », le joueur doit effectuer certaines actions : par exemple, payer un impôt, aller en prison, tirer une carte qui elle-même provoque l'apparition d'un nouvel événement, etc.
- Certaines cases n'ont pas d'effet particulier (visite de la prison, parc gratuit). D'autres comme la case Départ n'ont d'effet que lors du passage du joueur même s'il ne s'y arrête pas (il reçoit son salaire).



Le plateau de jeu traditionnel.

Bien évidemment il arrive qu'un joueur ne soit plus en mesure de payer ce qu'il doit faute d'espèces : il doit alors se procurer de l'argent soit en **hypothéquant** des propriétés (dont le loyer ne peut alors plus être perçu), soit en **revendant** des constructions ou des titres (à la banque, ou à d'autres joueurs). Si cela ne suffit pas le joueur est éliminé.

Pour pouvoir jouer de façon intéressante il faut au minimum implémenter les règles suivantes :

- Le tour de jeu : pour chaque joueur, lancer les dés, avancer sur la case correspondante, effectuer les actions associées, rejouer si nécessaire.
- Représenter les différentes sortes de propriétés :
  - les **terrains** appartiennent à un groupe de couleur ; lorsqu'un joueur possède tous les terrains d'une couleur, le loyer est doublé ;
  - les **monopoles** entraînent un loyer variable : pour les gares il est de 2500, multiplié par 2 pour chaque autre gare possédée par le propriétaire, pour les compagnies, c'est 400 ou 1000 fois la valeur des dés selon qu'on possède une compagnie ou les deux.
- Gérer correctement les événements suivants :
  - proposer à la vente une propriété sans propriétaire ;
  - gérer automatiquement les dépenses et les recettes obligatoires (taxes, Départ...) ;
  - gérer les déplacements y compris le fait de rejouer quand on tire un double (deux dés identiques) ;
  - gérer l'emprisonnement du joueur : un joueur reste en prison trois tours.
- Pouvoir charger les données de configuration du jeu en format texte (CSV).
- Gérer les cartes Chance et Caisse de Communauté les plus simples (recettes ou dépenses immédiates, déplacements).
- Gérer le manque d'espèces d'un joueur au moins avec un algorithme « glouton » (par exemple vendre automatiquement les constructions d'abord, puis hypothèquer, puis vendre les titres qui couvrent le mieux la dette).

2 Mise en œuvre

Analyse

L'application sera découpée en 4 packages au minimum, assurant les fonctionnalités suivantes :

- `monopoly.gui` sera en charge de toute la partie graphique (interface utilisateur, dessin du plateau de jeu, affichage des pions, etc.) ;
- `monopoly.jeu` contiendra tout ce qui concerne la gestion du cycle de jeu proprement dit, à savoir le traitement des joueurs, le positionnement des pions sur les cases du plateau, la gestion effective des transactions bancaires et la gestion de la fin de partie ;
- `monopoly.proprietes` contiendra tout ce qui concerne les titres pouvant être acquis par les joueurs : terrains et monopoles (comme les compagnies et les gares) qui sont caractérisés par des loyers et la possibilité ou non de construire des maisons ou hôtels ;
- `monopoly.evenements` sera en charge de la définition des classes destinées à gérer les événements du jeu : déplacer un joueur (à une position absolue ou relative), tirer les dés, régler les dépenses ou recettes obligatoires, proposer d'acheter ou de vendre, tirer une carte, aller en prison ou en sortir, rejouer, etc.

Documents fournis

Vous disposez sur Moodle d'une archive `projet-long.tgz` contenant :

- un répertoire `sources` contenant les interfaces, classes abstraites et concrètes permettant de spécifier un certain nombre de fonctionnalités attendues ;
- un répertoire `config` contenant les fichiers CSV permettant la configuration du jeu ainsi qu'un exemple de partie sauvegardée ;
- des répertoires pour le bytecode, la javadoc et le manuel utilisateur.

Format des données

1. Le fichier `monopoly.csv` contient la description du plateau de jeu avec une ligne pour chaque case, donnant :
   - son numéro (identifiant unique)
   - son nom (celui qui doit être affiché)
   - le type d'événement susceptible de se produire sur cette case (suivi d'éventuels paramètres séparés par des virgules) : par exemple `carte,CC` pour tirer une carte Caisse de Communauté, ou `dépense,20000` pour l'impôt sur le revenu, ou encore `prison` quand il faut aller en prison
   - le type de propriété éventuel : `terrain` ou `monopole`
   - le groupe des propriétés : couleur pour le terrain, `gares` ou `compagnies` pour les monopoles
   - le prix d'achat (quand le titre peut être acheté)
   - le coût des constructions (maisons et hôtel)
   - les valeurs des loyers pour le terrain nu et les niveaux de constructions suivants (de 1 maison à 4 maisons puis 1 hôtel), séparés par des virgules.
2. Le fichier `cartes.csv` contient la description de chaque carte Chance et Caisse de Communauté, donnant :
   - son numéro (identifiant unique)
   - son groupe (chance ou CC)
   - son intitulé (à afficher)
   - le type d'événement : `dépense`, `recette`, `frais immo`, `aller`, etc.
   - les paramètres éventuels de l'événement : par exemple `recette;5000` donne 5000F au joueur, `aller;1` le place sur la case Départ, `déplacement relatif;-3` le fait reculer de trois cases, `frais immo;4000,11500` impose des réparations à raison de 4000F par maison et 11500F par hôtel. Certains événements sont plus complexes, comme par exemple `choix;dépense:1000,carte:chance` dans lequel on demande au joueur de choisir entre deux autres événements : payer 1000 ou tirer une carte chance.
3. Pour sauvegarder une partie deux fichiers sont nécessaires :
   - la description des joueurs : numéro, nom, somme disponible en espèces, position (numéro de case), emprisonné ou pas, titres possédés et cartes possédées (par exemple « Sortie de prison »)
   - la description des modifications apportées aux propriétés : numéro de case, hypothèque ou pas, niveau de construction (de 0 : rien à 5 : hôtel)

3 Travail demandé

Vous ne serez pas jugé exclusivement sur le rendu graphique du jeu. L'accent doit porter avant tout sur une implémentation efficace des classes et de leurs relations, ainsi que sur la documentation.

Objectifs minimaux à atteindre

Le point auquel vous devrez arriver au minimum est le suivant :

- un programme portable (au moins Windows/Linux) ;
- une interface graphique en Swing, parfaitement fonctionnelle, avec affichage d'un plateau de jeu et du pion de chaque joueur ;
- la possibilité de jouer à N joueurs humains selon les règles obligatoires décrites page 1 ;
- une gestion propre des fins de partie lorsqu'un joueur a gagné.

Améliorations souhaitées

Vous devez également choisir quelques améliorations à votre gré, parmi les idées suivantes :

- Gestion des hypothèques : un titre peut être hypothéqué pour la moitié de sa valeur d'achat (pendant ce temps le propriétaire ne perçoit plus de loyer pour ce titre) ; l'hypothèque peut être levée en remboursant le montant + 10 %.
- Gestion des constructions immobilières : seuls les terrains sont constructibles, et seulement si le joueur possède tout le groupe de couleur. On peut construire jusqu'à 4 maisons par terrain, après quoi les maisons sont remplacées par un hôtel. On doit construire uniformément, c'est-à-dire qu'il ne doit pas y avoir plus d'un niveau de différence entre les propriétés d'un même groupe.
- Gérer tous les événements des cartes Chance et Caisse de Communauté (y compris « Payez une amende de 1000F ou tirez une carte Chance »)
- En cas de manque de liquidités, permettre au joueur de gérer lui-même ses hypothèques et ventes.
- Possibilité de jouer contre l'ordinateur dans un mode simple (par exemple l'ordinateur achète systématiquement tout ce qu'il peut) ;
- Sauvegarde et chargement de parties au format CSV (voir l'exemple) ;
- Obliger la vente d'une propriété qui n'appartient à personne lorsque le joueur qui est arrivé dessus ne souhaite pas l'acheter, en la mettant aux enchères.
- Gérer les règles détaillées d'emprisonnement : le joueur qui fait un double trois fois de suite va en prison. Mais un joueur emprisonné continue de lancer les dés et doit sortir de prison s'il fait un double ; il peut sortir en payant un amende ou en utilisant une carte « Sortie de prison ».
- Proposer une variante « Cité Scientifique »...

Documents à rendre

L'ensemble du projet doit être rendu sur la plateforme Moodle dans une archive au format `tar/gz` contenant :

- tout le répertoire de travail (sources, doc, classes et config) ;
- un exécutable sous forme d'archive JAR (voir documentation de Java) ;
- un fichier texte `README` décrivant la procédure à suivre pour compiler les sources et démarrer le jeu ;
- un court manuel utilisateur sous la forme d'une page HTML, indiquant comment jouer (buts du jeu, règles ayant été implémentées, ajouts introduits, description de l'interface graphique, etc.).

Quel que soit le degré d'avancement de votre projet, tout doit compiler et se conformer aux interfaces et fichiers de configuration fournis.

Cette archive (code documenté !) sera rendue au plus tard le **mardi 17 décembre 2011**, avant la démonstration du programme. Cette démonstration (de 10 à 15 minutes pendant le créneau 16h–18h) doit permettre d'illustrer les points réalisés.