La version d'un seul joueur

La réalisation:

Dans cette partie, nous avons créé
- deux classes *Cards* et *total_Cards* pour représenter respectivement une carte et une liste chaînée qui contient toutes les cartes;
- trois classes *CardDrawable*, *Score* et *LimpidLabel* pour représenter et dessiner les sous-éléments dans la fenêtre principale;
  - *CardDrawable (extends JButton)*: représente une carte qui peut être dessinée sur un bouton. Donc on peut sélectionner et désélectionner une carte. Et si le nombre d'une carte est 0, on ne dessine rien. Donc ça peut présenter une carte vide.
  - *Score (extends JComponent)*: représente la partie "Score" qui peut être dessinée. Il y a le temps et la note dans cette partie, et peut-être les trois petites cartes. Donc il y a un thread *Horloge* qui représente le temps dans cette classe.
  - *LimpidLabel (extends JLabel)*: représente les couvres vertes ou rouges.
- une class *mainFrame* pour structurer la fenêtre principale et réagir de tout ce que l'utilisateur fait.

Dans la class *mainFrame*, il contient 15 objets de *CardDrawable*, 1 objet de *Score* et 3 objets de *LimpidLabel*. Initialement, on tire aléatoirement 12 cartes de la liste. Dès que les 12 cartes sont bien tirées, on vérifie bien que les 12 cartes contiennent un set. Si c'est bien le cas, les 3 cartes restes sont vides, sinon on retire 3 cartes supplémentaires et garantit qu'il y a au moins un set dans ces 15 cartes. Et on ajoute un *ActionListener* pour chaque carte pour qu'elle puisse réagir quand on clique une carte.

Il y a deux threads dans la classe *mainFrame*: *SetDicider* et *AfterDicider*.
- *SetDicider* vérifie si les trois cartes que l'on a choisi sont un set ou pas et on sauvegarde les places de ces trois cartes qui servent à mettre les trois couvres à leur place.
- *AfterDicider* fait ce qu'il faut selon le résultat de *SetDicider*.
  - Si on trouve un set dans *SetDicider*, on met les couvres à couleur verte et on les affiche à leur place pendant 2 secondes. Après, on augmente le "Score" d'un point et on renouvelle les trois petites cartes dans la partie Score. Ensuite, on vérifie si on a choisi des cartes supplémentaires. Si c'est le cas, on dois placer les cartes supplémentaires restes dans les trous des 12 premières cartes et encore une fois, on vérifie bien si maintenant les 12 cartes contiennent un set.
  - Si on n'a pas trouvé de set dans *SetDicider*, on mets les couvres à couleur rouge et on les affiche pendant 2 secondes. Après on diminue le "Score" d'un point. Mais les cartes dans la fenêtre ne change pas.

Pour lancer ce jeu, il suffit d'executer la class *Jeu* et opérer dans la fenêtre qui apparaît.

La version Multi-joueurs

Nous avons défini le protocole suivant:

-Chaque joueur sélectionne des cartes localement.
-Lorsqu'un joueur a sélectionné trois cartes, la machine locale évalue le résultat.
-Un requête est envoyé au serveur pour demander la mis en jour des carte.
-Le résultat d'un joueur n'influence pas d'autres joueurs.
-Comme deux joueurs partagent ensemble les 82 cartes et une fois quand les 82 cartes sont tous distribuées, le jeu termine, donc le but pour le jeu est de trouver les set le plus vite possible.

La réalisation:
Nous avons implémenté le réseau entre un client et un serveur par socket. Le client utilise ObjectInputStream et ObjectOutputStream pour communiquer avec le serveur. Un fois quand le serveur reçoit un InputStream qui est sous la forme d'un String, il chercher l'opération correspondante, et puis il renvoie par un String comme une ordre.

Le côté client est presque pareil que celui de mono-joueur. Le seul changement est que toutes les méthodes "un_Set.tire_une_carte" est changé par plusieurs lignes de codes qui servent à demander le serveur de lui donner les cartes.

Le côté serveur, le serveur attend toujours les inputStreams, qui est dans une boucle infinie. La méthode ObjectInputStream.read est une méthode bloquant. Donc quand le serveur ne reçoit rien, il reste toujours bloqué. Un fois quand le serveur reçoit une requête, il renvoie certain ordre.