# Travail pratique (20%)

## Objectifs

* Vous utiliserez plusieurs notions vues pendant le cours
  + Respect de l’architecture MVC
  + Héritage
  + Polymorphisme
  + Agrégation/Composition
  + Exception
  + UserControl
  + Surcharge d’opérateur
  + Statique
  + Abstract/Virtual

## Modalité :

* Équipe de 2-3. Les équipes doivent être formées d’ici le **16 novembre**.
* La remise se fait par Github le **2 décembre avant minuit**. Vous devez inviter **elacassecegepba** comme collaborateur à votre répertoire. Je vais prendre le code se trouvant dans la branche master/main.
* Les modalités d’intégrité intellectuelle et plagiat s’appliquent.
* Votre programme doit compiler (40% de pénalité)

# Description

## Model (47 pts)

### Card (15 pts)

* Il existe plusieurs types de carte qui interagissent de façon différente
  + Sorcery (5 pts)
    - Faite une factory méthode qui crée correctement 3 Sorcery selon leur nom. Aidez-vous des images pour cela.
  + Permanent (5 pts)
    - Ajouter une propriété virtuelle pour savoir sur une carte est de type Permanent.
    - Creature
      * Faite une factory méthode qui crée correctement 3 Creature selon leur nom. Aidez-vous des images pour cela.
    - Land
      * Une Land prend en paramètre un objet de type Mana qui représente le type de mana qui sera généré par la Land lors de son activation.
      * Faite une Factory dans Land qui crée soit une Forest, Island, Mountain, Plains ou Swamp. Assurez-vous qu’elles génèrent la bonne couleur de mana. Aidez-vous des images pour cela.
  + Il y a aussi 2 types spéciaux de carte : CardBack et DarkCardBack qui font office de placeholder.
* Ajoutez une liste de 10 Card au dictionnaire de LibraryManager en utilisant vos factory. (5 pts)
  + Vous devez avoir au moins une Card de chaque type.

### Zone (27 pts)

* Il existe plusieurs types de Zone
  + Lorsque l’utilisateur clique sur une carte, la Zone qui contient la carte doit gérer l’événement à sa façon.
  + Chaque Zone override la propriété Name qui est ensuite utilisé dans ToString de Zone.
  + Library (10 pts)
    - La TopCard de la Library est une DarkCardBack lorsqu’elle est vide, sinon c’est une CardBack.
    - La Library doit permettre de mélanger ses Card.
    - Lorsqu’on clique sur une Card, la Card du dessus doit être retirée de la Library et le Player la place dans sa Hand (pensez à comment le Player peut être informé qu’une carte a été retirée de sa Library). S’il n’y a plus de Card dans la Library, lancez une exception avec un message significatif et affichez-le.
  + Hand (10 pts)
    - Lorsqu’on clique sur une Card, le Player tente de la jouer.
      * Pour ce fait, il tente payer le coût en Mana de la carte. Par exemple, si la carte demande deux Mana verte, le Player doit pouvoir retirer deux Mana verte de son ManaPool. La gestion des Mana Colorless se fait en bonus.
      * Utilisez la surcharge des opérateurs de comparaison dans ManaColor pour savoir si le Player a suffisamment d’une couleur.
      * Si le Player n’a pas assez de Mana, lancez une exception avec un message significatif et affichez-le.
      * S’il a pu payer le coût en Mana, la Card est jouée. Si elle est un Permanent, le Player la déplace sur le Battlefield, sinon elle est envoyée au Graveyard.
  + Battlefield (5 pts)
    - Lorsqu’on clique sur une Card, on inverse son statut Tapped. Si la Card est une Land, on ajoute au ManaPool du Player la Mana qui est générée par cette Land.
  + Graveyard (2 pts)
    - Lorsqu’on clique sur une Card, rien ne se passe.

### Player (5 pt)

* Le Player est composé d’un Battlefield, Graveyard, Hand et Library.
  + Le Player mélange la Library à sa création.

## View (5 PT)

* Lorsqu’une Card est tapped l’affichage de l’image doit être affiché à l’envers (RotateFlipType).
* À chaque fois que l’utilisateur fait une action qui influence ce qui est affiché à l’écran, l’affichage doit être mis à jour.
* L’affichage du Battlefield est divisé en deux.
  + En haut, affichez les Creature.
  + En bas, affichez les Land.

## Git (5 pts)

* De bonnes pratiques de développement doivent être mises en place.
* Le nombre de « commits » doit être supérieur à 10.
* Chaque membre de l’équipe doit avoir des « commits ».
  + Dans un cas extrême où un membre n’aurait aucun « commits » ou que ses « commits » ne fassent rien, la note 0 pourrait être obtenue pour cette personne.

## Bonus – Gestion des colorless (5 pts)

* Après avoir vérifié toutes les Mana de couleur, vérifier qu’il resterait assez de Mana pour payer le coût en Mana Colorless
* Il est possible d’utiliser de la Mana de n’importe quelle couleur pour payer du Colorless.