

M1 MIAGE / MAPC / TD3

Analyse de code

1. Analyser [Version \(JetUML\)](#) et l'aspect de comparaison.
2. Analyser [CardStack \(Solitaire\)](#) et l'aspect d'itération. Représenter cette application de patron avec un diagramme de classe.
3. Analyser [PlayingStrategy \(Solitaire\)](#) et ses sous-classes, [NullPlayingStrategy](#) et [GreedyPlayingStrategy](#). Représenter cette application de patron avec un diagramme de classe.

Conception

1. concevez et implémentez une abstraction bien encapsulée représentant la 'main' d'un joueur de cartes avec une classe `Hand`. Une main doit pouvoir contenir de $0 \leq N$ cartes, avec N un paramètre qui dépend du jeu (5 au Poker, 13 au Bridge, etc). Les services fournis par cette classe sont : `add(Card)`, `remove(Card)`, `contains(Card)`, `isEmpty()`, `size()` et `isFull()`, ainsi qu'un moyen (correct) d'accès aux cartes.
2. permettez la comparaison de deux 'mains' et le(s) tri(s) associé(s). Vous ferez une version en utilisant `Comparable` (utilisez le nombre de cartes en main) et deux en utilisant `Comparator` (vous utiliserez des factory methods dans `Hand` pour récupérer les comparateurs). Pensez à tester votre code.