Plan klas i struktur projektu Makao

1. CLI

a. **MainInterface** - klasa odpowiedzialna generalny interfejs z użytkownikiem. Zawiera wypisywanie dostępnych algorytmów, ile iteracji ma zostać wykonanych itp. To ona również odpowiada za uruchamianie gier.

2. Obsługa gry

a. **Game** - klasa zarządzająca przebiegiem gry. Zawiera tablicę algorytmów biorących udział w grze oraz *Deck*. Wywołuje kolejnych graczy do ruchu i jest przerwana w przypadku skończenia się kart w *Deck*-u lub gdy któryś z graczy wygra.

b. **Deck** - klasa odpowiedzialna za posiadanie i rozdawanie kart. Posiada metody takie jak *shuffleIn* pozwalające na wtasowanie oraz *takeOne* - zabranie jednej karty

c. **Player** - klasa bazowa, po której dziedziczą wszystkie klasy algorytmów. Posiada wirtualną metodę *choose* odpowiedzialną za dokonywanie wyboru podczas ruchu gracza - zwracającą listę kart, w którym karta położona jako pierwsza znajduje się w elemencie głowy listy. Zawiera on informacje o swoich karach (*pull* oraz *delay*), listę *Hand* zawierającą wszystkie karty posiadane przez gracza, oraz finalne metody *didWin* - zwracającą true jeśli gracz wygrał, *doesMatch* - pozwalającą na sprawdzenie czy gracz może położyć kartę na drugą oraz *canStack* - sprawdzającą czy karty mogą być położone jedna na drugiej. Posiada również metody takie jak *pullCards*, oraz *addPull* oraz *addDelay* modyfikujące jego pola.

3. Karty

a. **Card** - klasa bazowa dla wszystkich kart biorących udział w grze. Posiada pola number oraz color zawierające podstawowe informacje pozwalające zdefiniować typ, *matchingValues* - listę wartości kart kompatybilnych oraz *matchingColor* - wartość koloru, na którego podstawie można położyć kartę (może być różne od color w przypadku asa). Posiada również metodę *selfRestore* - przywracającą kartę do stanu początkowego

(anulującą żądania) wywoływaną podczas wtasowywania do decku. Dodatkowo zawiera funkcję *execute* wykonywaną podczas dawania kart na stół - różnie przysłoniętą zależnie od typu karty

b. **AgroCard** - klasa kart agresywnych - posiadającą metodę nałożenia kary na gracza następnego lub poprzedniego oraz modyfikację pól gracza wykonującego ruch (np odbicie żądania itp)
c. **Jack** - klasa kart waleta - umożliwiająca żądanie figury (modyfikację *matchingColor* oraz *matchingValues*)
d. **Ace** - klasa kart as - umożliwiająca zmianę koloru (modyfikację *matchingValues*)
e. **QueenOfSpades** - klasa karty umożliwiającej zwolnienie z kary (*delay* lub *pull*) bez konieczności pobierania karty.

4. Inne

Karty oraz żądania przesyłane są między metodami klas w postaci list. Do tego służy szablonowa klasa **LinkedList** - zawierająca rozbudowaną listę jednokierunkową z iteratorem. Na etapie implementacji mogą pojawić się dodatkowe klasy pomocnicze - w celu zoptymalizowania pewnych procesów lub obsłużenia nieprzewidzianych przypadków granicznych. Dodatkowo pojawi się **placeholder dla algorytmu** - klasa dziedzicząca po **Player** posiadająca najprostszą strategię gry pozwalająca na testy symulatora.

Bartłomiej Pogodziński, gr. dziek. 2, sem. 3 INF AEiI