ONE CARD GAME

Go to Github

자바스크립트로 구현한 원카드 게임(싱글 플레이)입니다.

Go To Play!!

주의

원할한 게임은 직접 다운받아서 하시길 바랍니다...

게임 규칙

게임 인원은 4명(플레이어 포함) 입니다.

게임 진행은 시계방향입니다.

시작 기준은 랜덤하게 정해집니다.

시작 카드는 5장입니다.

보이게 뒤집어 놓은 카드 뭉치의 맨 위에 있는 카드와 동일한 숫자 또는 무늬의 카드를 뭉치 위에 계속해서 한 장씩 올려놓으면서, 자신의 모든 카드를 소모하면 승리합니다.

자신을 제외한 모두가 파산(소유 카드가 12장을 초과하는 사람)이 되면 승리합니다.

예를 들어, 정통 룰 기준 ♥5가 제시되어 있다면 같은 하트 무늬의 카드 중 하나(♥2, ♥3, ♥4, ♥6, ♥7, ♥8, ♥9, ♥10, ♥J, ♥Q, ♥K, ♥A)나 ♠5, ♣5, ♦5 중 한 장만 낼 수 있다.

만약 자기가 가진 카드 중에서 올려놓을 수 있는 카드가 없다면 대신 카드 한 장을 가져가고 다음 사람에게 순서를 넘겨야 한다.

카드 뭉치의 카드가 전부 사라지면, 이미 쌓인 카드를 맨 위의 카드만 남겨놓고 다시 섞어서 엎어놓아 카드 뭉치를 만든다. (딜러)

그리고 자기가 가진 마지막 카드를 낼 때, '원 카드'를 선언해야 승리할 수 있다. 물론 선언 자체를 생략하기도 한다. (미구현)

능력을 갖춘 카드

공격계열 카드

'2'카드

능력 : 자신의 다음 차례 사람에게 2장을 먹인다.

상대가 다른 무늬 2를 냈을 때 지원공격이 가능하다.

'A'카드

능력 : 자신의 다음 차례 사람에게 3장[12]을 먹인다.

상대가 다른 무늬 A를 냈을 때 지원공격이 가능하다.

'조커'카드

능력 : 자신의 다음 차례 사람에게 주로 일정 수의카드를 먹인다.(흑백:5 컬러:7)

흑백 조커는 흑백계열의 카드 다음에만 카드를 낼수있다.(ex:클로버,스페이드)

컬러 조커는 컬러계열의 카드 다음에만 카드를 낼수있다.(ex:하트,다이아)

조커카드는 다른 무엇도 지원공격 할 수 없다.

단 컬러조커는 흑백조커에 지원공격 할 수 있다.

조커를 두 장 가지고 있을 경우, 동시에 내는 게 불가능.

기능 계열 카드

'3'카드

능력 : 같은 모양의 2를 방어할 수 있다.(속칭 방어.)

방어에 성공한 경우 먹여야할 장수를 0장으로 바꾼다.

'7'카드

능력 : 카드를 낸 후에 자신이 원하는 무늬로 바꿀 수 있다.(속칭 체인지.)

예를들어 ♣7을 내고 '스페이드'라고 입력하면 그건 '스페이드'로 취급한다.

'J'카드

능력 : 현재 진행 방향으로 한 사람을 건너 뛰어 그 다음 사람에게 차례를 넘긴다.(속칭 점프.)

'Q'카드

능력 : 현재 진행자의 이전 차례의 플레이어로 차례가 넘어간다.

'K'카드

능력 : 한번 더 자신의 차례가 된다. 즉, 카드를 한 장 더 낸다.

파산

소지하고 있는 카드 수가 12장 초과시

코드

javascript

DoubleLinkedList.js

node는 prev,next를 가지고 있으며 양방향 더블링크드리스트입니다.

Loger.js

Loger 는 각 객체들의 행동에 대하여 플레이어가 알 수 있도록 로그를 남겨줍니다.

로그 출력 외에도 현재 누구의 차례인지와 우승자가 판별 될 시에 관련 문구를 보여줍니다.

Dealer.js

Dealer 는 게임시작과 끝의 권한을 가지고 있습니다.

플레이어들의 각각의 행동들은 Dealer 객체에게 관련 데이터를 넘겨주어 그에 따른 행동을 취합니다.

Dealer 의 다음차례를 선별하는 알고리즘은 recursive function 을 이용했습니다.

user(사람)플레이어 일때까지 반복하여 nextTurn() 호출합니다.

다음 차례가 AI 플레이어 일시에는 setTimeout 메서드를 이용하여 일정 시간후에 nextTurn() 를 호출합니다.

```
nextTurn = function(times) {
  if (!this.gameOver) {
    if (this.playerList._ptr.data.playerCardList.length > 13) {
      this.diePlayer(this.playerList._ptr.data);
    }
    if (this.playerList._ptr.data.playerCardList.length == 0 ||
this.playerList._length == 1) {
      this.GAMEOVER();
      loger.displayWinner(this.playerList._ptr.data.id);
      return;
    }
    this.playerList._ptr.data.setTurn(false);
    for (var i = 0; i < times; i++) {
      this.playerList._ptr = this.playerList._ptr.next;
    }
    if (times == -1) {
      this.playerList._ptr = this.playerList._ptr.prev;
    }
    this.playerList._ptr.data.setTurn(true);
    this.loger.displayCrruntTurn(this.playerList._ptr.data.id);
    if (this.playerList._ptr.data instanceof User) {
      return;
    }
    this.timer(this);
  }
};
```

Deck.js

TrumpDeck 과 DummyDeck 은 Deck 에게서 상속받았습니다.

TrumpDeck 은 플레이어들이 Draw 시 필요한 카드 뭉치입니다.

DummyDeck 은 플레이어들이 카드를 내고 남은 카드 뭉치입니다.

Player.js

User 와 AI 는 Player 에게서 상속 받았습니다.

```
Player.prototype.playCard = function(cardNumber) {};
Player.prototype.sortCardId = function(cardId){~};
```

카드를 내는 행위와 카드를 내려고 할때 알맞은 카드인지 선별해주는 메서드입니다.

User의 playCard()는 소지한 카드 이미지에 따른 클릭이벤트를 발생시켜 sortCardId() 로 적절한지를 판단하여 진행합니다.

```
AI.prototype.playCard = function() {
  var cardId = null;
  for (var i = 0; i < this.playerCardList.length; i++) {
    var resultTimes = this.sortCardId(this.playerCardList[i]);
    if (resultTimes != null) {
      this.loger.appendPlayCardLog(this.id, this.playerCardList[i]);

this.playerCardList.splice(this.playerCardList.indexOf(this.playerCardList[i]),
1);
      this.repaintCardField();
      return resultTimes;
    }
  }
  this.draw();
  return 1;
};
```

AI는 현재 소지카드 중에 dummyDeck의 마지막 카드와 비교하여 적절한 카드가 있을시 카드를 내고 없으면 Draw() 합니다.

javascript.js

게임시작전에 필요한 객체초기화 및 각종 셋팅을 담당합니다.