Classe: Carta

Carta (int valor, tipuscoll coll) Carta
Retorna una nova carta de valor i coll.

getValor () int valor
Retorna el valor d'una carta (de 1 a 13 no?)

getColl () tipuscoll coll
Retorna el coll d'una carta

getColor () tipuscolor color
Retorna el color d'una carta

Notes: En principi és trivial i tots l'hauríem de fer servir.

Classe: MàDeCartes

MàDeCartes (bool baralla) MàDeCartes
Retorna un nou objecte MàDeCartes sense cap carta a excepció que baralla sigui cert, que crea una MàDeCartes amb totes les cartes de la baralla.

getCartes () Carta []
Retorna el vector de cartes que conté

afegirCarta(Carta c) MàDeCartes
Retorna una mà de cartes resultat d'afegir la carta c a la mà actual de cartes.

treureCarta(Carta c) MàDeCartes
Retorna una mà de cartes resultat de treure la carta c a la mà actual de cartes.

operator + (MàDeCartes a, MàDeCartes b) MàDeCartes
Retorna una mà de cartes que conté la unió de les mans *a* i *b*. No té sentit parlar de repeticions de cartes.

Específic pòquer:
getMillorJugada () MàDeCartes
Retorna la millor mà de 5 cartes. Només aplicable a mans de 7 cartes en principi

avaluarJugada () int
Retorna el valor numèric associat a la jugada per apoder

comparar amb altres jugades.

Notes: Classe a compartir. Aventuro que cada grup l'estendrà mitjançant un controlador per afegir funcions que necessitin (calcular valor els del BlackJack o trobar cartes "sueltes" (sense combinar) els del Rummy).

Classe: Partida

Partida (Jugador [] llista_jugadors_ordenats, int diners, int apostaMàxima, int MaxMans) Partida
Crea una nova partida amb la seva baralla completa i crea també les instàncies de participants associats a la partida inicialitzant:
    torn: 1, ronda: 1, apostaMàxima: apostaMàxima, mansJugades: 0, dinersInicials: diners, maxMans: maxMans
    Cada participant
        ordre: situació en la llista
        diners: diners
        va: true
        preparat: false
        apostaAct: 0

obtenirParticipantActual() Participant
Retorna el participant actual (el que té el torn).

apostarParticipant(int diners, Participant p)
Aposta la quantitat al participant. Avança de jugador també.

retiraParticipant(Participant p)
Retira el participant de la ronda. Avança de jugador també.

obtenirMínimaApostaRonda() int
Retorna l'aposta que s'ha d'igualar per a continuar a la partida.

getApostaMàxima() int
Retorna l'aposta màxima de la partida (si no hi ha límit és infinit).

getMaxMans() int
Retorna el nombre màxim de mans.

getParticipants() Participant []

Retorna la llista d'objectes participant

getCartesTaula() MàDeCartes
Retorna les cartes vistes

getCartesBaralla() MàDeCartes
Retorna les cartes de la baralla

Classe: Participant

Participant (Jugador j, Partida p, int ordre) Participant
Crea un nou participant d'un jugador i d'una partida.
Ordre indica l'ordre en la partida actual.

MínimaAposta () int
Retorna la mínima aposta que ha de fer aquest participant per a seguir en la ronda (pot ser 0 per exemple en el cas de un all-in anterior).

MàximaAposta () int
Retorna la màxima aposta que pot fer aquest participant. (Pot ser 0)

getJugador() Jugador
Retorna el jugador associat al participant.

set/get ready
set/get va
set/get apostaAct
set/get diners
set/get cartes
No crec útil fer milers de funcions aquí. Millor un accés directe als membres de la classe.

Classe: Jugador

Jugador (string nom) Jugador
Crea un nou jugador amb nom "nom".

GetTipusJugador () TipusJugador
Retorna el tipus de jugador (huma o machine)

getNom() string
Retorna el nom del jugador

set/getAvatar (string)
Set o get el avatar del jugador.

Classe: JugadorHumà

JugadorHumà (string nom, string password) JugadorHumà
Crea un nou jugador humà amb nom i password.

Classe: JugadorCPU

JugadorCPU (string nom, ...) JugadorCPU
A vere qui té ous de definir el sistema de regles.

(...)

Implementació de Partida (parcial i guia pel futur)

Nota: ens vam deixar l'atribut de mínima aposta de la ronda oi? En el seu lloc hi ha un apostaMàxima (mira UML)! Assumeixo que també existeix apostaMínimaRonda

```
obtenirParticipantActual() Participant {
 for each Participant pa in this.participants do
  if (pa.ordre == this.torn)
   retorna pa
  fisi
 loop
}

apostarParticipant(int diners, Participant p) {
 p.setDiners(p.getDiners() - diners);
 p.setapostaAct (p.getapostaAct () + diners);
 p.setReady(true);

 if (this.apostaMínimaRonda < p.apostaAct) {
   // El jugador ha pujat, posem a no ready els que van
   for each Participant pa in this.participants do
    if (pa.getVa()) pa.setReady(false);
   loop
 }

 // Avancem de jugador (saltant els que no van clar)
 do
  this.torn = (this.torn + 1) % this.participants.size()
 loop while (not this.obtenirParticipantActual().getVa)

 // Comprovem si ha finalitzat la ronda/partida
 // cridant a una funció privada. Repartirà les cartes si s'escau
 bool ready = true;
 for each Participant pa in this.participants do
  ready = ready AND pa.getReady();
 loop

 if (ready) this.CanviDeRonda();
}

retiraParticipant(Participant p) {
 p.setReady(true);
 p.setVa(false);

 // Comprovem si ha finalitzat la ronda/partida
 // cridant a una funció privada. Repartirà les cartes si s'escau
 bool ready = true;
 for each Participant pa in this.participants do
  ready = ready AND pa.getReady();
 loop

 if (ready) this.CanviDeRonda();
```

```
}
```