**Idea1: Online self-discipline system**

Database: stores calendars and diet

Network: user accesses this data by connecting to a remote server.

Muti-thread: multiple users could access to their own diet calendars

UI: upload data and reveal data

Function:

1. User could set goals of inputting calories in one day and upload their every meals’ inputting calories, consuming calories of extra exercises.
2. User could upload their goal of weight and current height, as well as their current weight.
3. System will record their goal, calculate their current BMI and three meals calories input and then sum up to get the total calories input and sub the consuming calories from extra exercises and conclude that whether one day’s diet goal is achieved.
4. Calories input and output in everyday will be calculated and revealed in calendars. Also the total goal achievement of a whole week and whole month will be calculated so that the system could give a big picture to user to know more specific about their daily calories input and weight change.

Main problem:

How to save calendars in database?

How do user access to their own calendars through server?

**Idea2: CardsGame: Blackjack (Point 21)**

UI: game page

Network: multiple player could join at the same time

Muti-thread: each thread attach to one player

Role: dealer(computer) and player\*3

**Rules:**

each player competes against the dealer

blackjack= A +J/Q/K

At a blackjack table, the dealer faces three playing positions from behind a semicircular table. Between one and eight [standard 52-card decks](https://en.wikipedia.org/wiki/Standard_52-card_deck) are [shuffled](https://en.wikipedia.org/wiki/Shuffling) together

1. Initial & check

The dealer deals from their left ("first base") to their far right ("third base").

Each player gets an initial hand of two cards visible to the people playing on it. The dealer's hand gets its first card face up, and, immediately gets a second card face down (the hole card), which the dealer peeks at but only reveals when it makes the dealer's hand a blackjack.

2. Deal with each player: hit or stand

Single cards are dealt to each wagered-on position clockwise from the dealer's left, followed by a single card to the dealer, followed by an additional card to each of the positions in play.

The object of the game is to win money by creating card totals higher than those of the dealer's hand but not exceeding 21, or by stopping at a total in the hope that dealer will bust. On their turn, players choose to "hit" (take a card), "stand" (end their turn and stop without taking a card).

 Number cards count as their number, the jack, queen, and king ("face cards" or "pictures") count as 10, and aces count as 1 or 11 according to the player's choice. If the total exceeds 21 points, it busts.

1. Dealer hit or stand

After the players have finished playing, the dealer's hand is resolved by drawing cards until the hand achieves a total of 17 or higher. The dealer could hit or stand.

Judge:

Blackjack > natural 21 points > other value

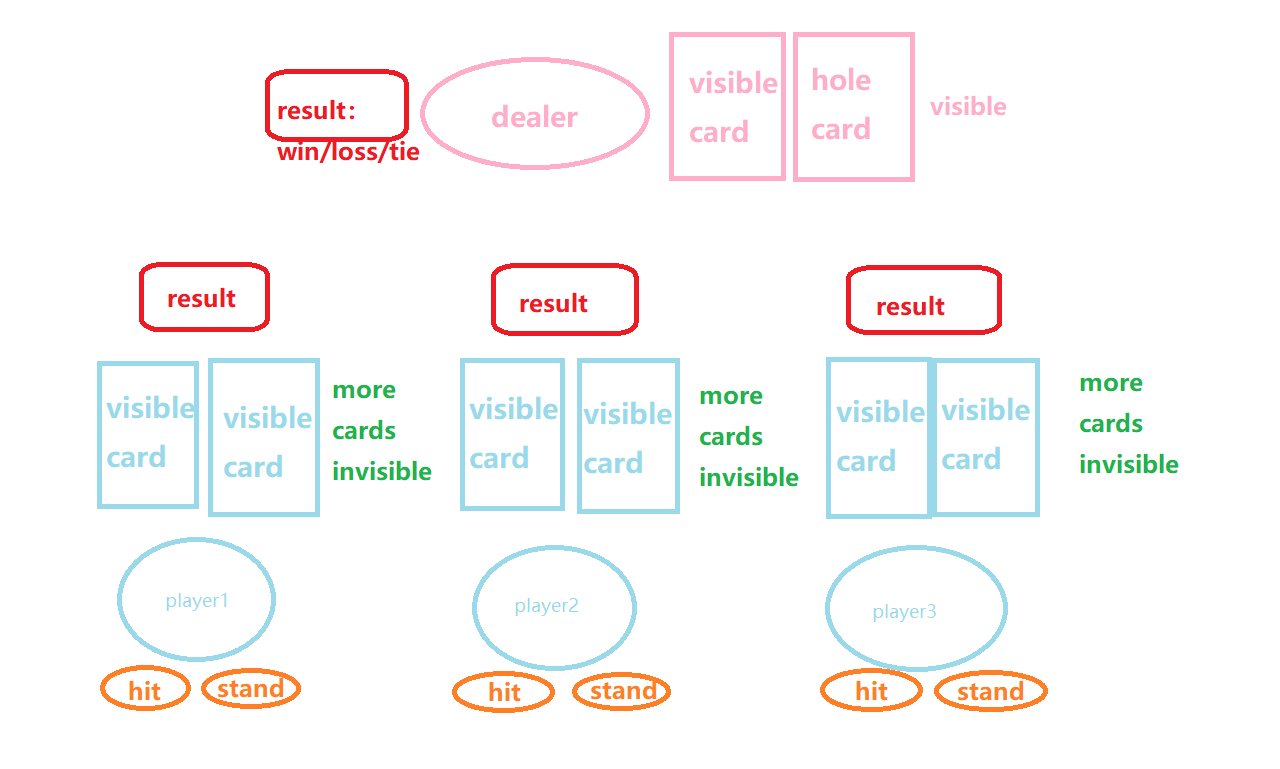
Anyone bust, loss the game.

If the dealer busts, all remaining player hands win. If the dealer does not bust, each remaining player wins if its hand is higher than the dealer's and loses if it is lower.

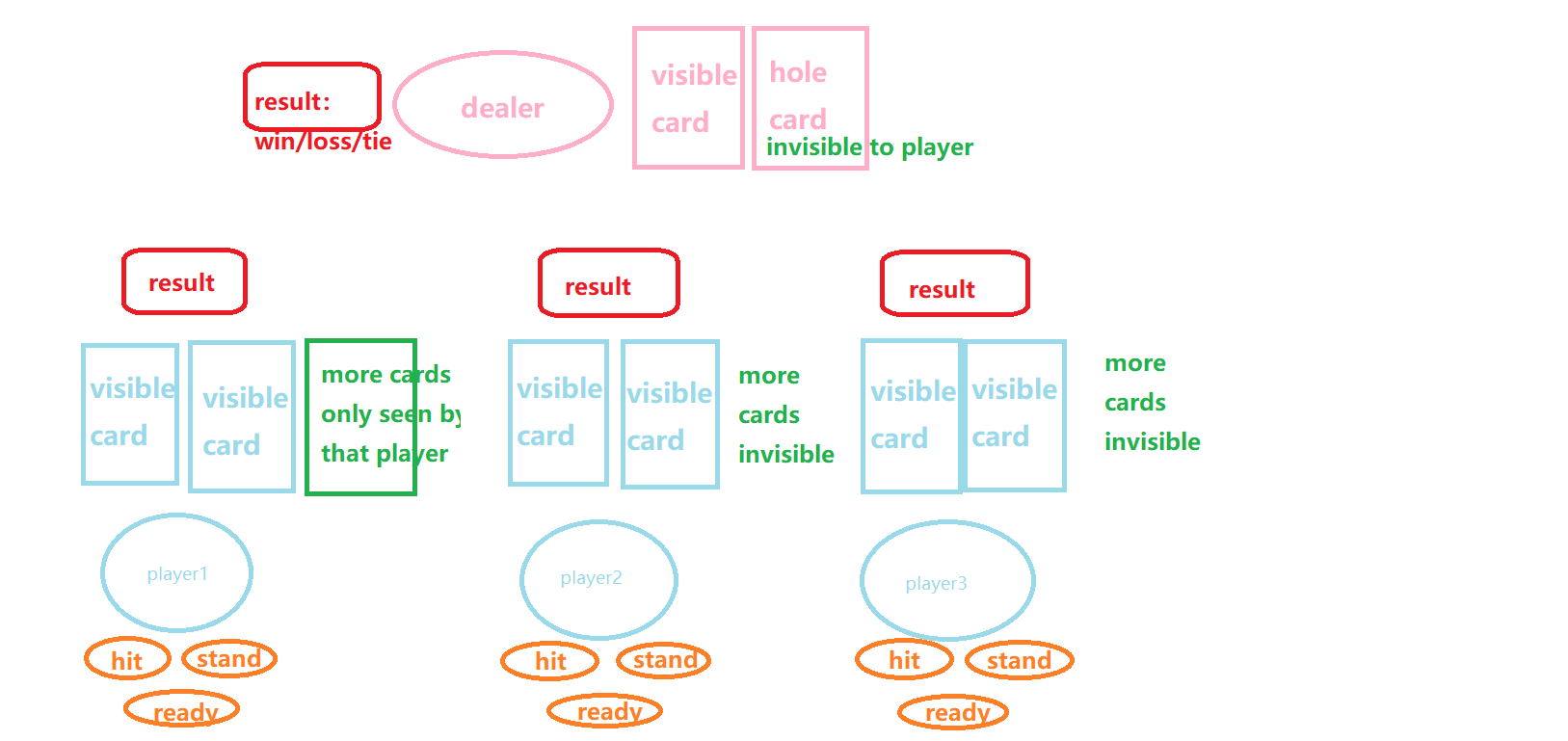
A player total of 21 on the first two cards is a "natural" or "blackjack," and the player wins immediately unless dealer also has one, in which case the hand ties. In the case of a tie ("push" or "standoff") . A blackjack beats any hand that is not a blackjack, even one with a value of 21.

UI design:

From dealer aspect:



From player1 aspect:



Class design

Dealer:

fields:

int status; // -1 lose; 0 playing; 1 win

ArrayList [] int HandCards; // cards in dealer’s hand

ArrayList [] int CardsBox; // draw cards from shuffled cards box

Static Map<String,int> card2val; // map cards to its value

Int CurValue; // current value of cards in hand

ArrayList [] int players; // record the players who are playing in the game;

methods:

public int checkStatus(); // check whether players are ready to play

public void Draw(); // randomly draw one card from CardsBox

public void Deal(int playerNum); // deal with player using playerNum

public void hit(); // take that card

public void stand(); // stop taking

public void judge(); // reveal the result of the game win/lose/tie

Player:

Fields:

private int playerNum; // player’s number to identify;

int status;

ArrayList [] HandCards;

private Int CurValue;

Methods:

public void getReady(); // ready to play

public void hit();

public void stand();

public int upDateValue(); // everytime hit a cards update curValue

public int getCurValue();