# Game of Cards Design Document

## *Mairead Manifold, G00294426, Summer 2015*

The following are the classes used in the project:

HumanPlayer

ComputerPlayer

Card

Deck

Hands

GameOfHearts

The GameOfHearts is the runner class for the project. It is where main resides.

The class GameOfHearts' main dictates the order of activity in the game:

dealCards

pass3Cards

playCards

checkRoundResults

checkGameResults

Following is an explanation of each of these activities.

## dealCards()

This method creates an instance of the deck class and uses a shuffle method to shuffle the cards in the new deck and then the program deals 13 cards to each player; using the following names for the computer players: cp1, cp2, cp3 and and using the name 'me' for the human player.

## pass3cards

This method gets all players to pass their 3 worst cards to the player on their left. The computer player cards are auto chosen for these players and it auto choses cards based on the following order preferenes:

- Queen of spades
- any heart; highest first
- any high value card of other suits

The human player is asked to choose a card. If they choose a card that they don't have, they are prompted to resubmit a card. To help the human player choose a correct card, they are presented with a list of cards in the chosen suit of cards.

## playCards()

This method runs until all players have no cards left or one player exceeds 100 penalty points.

The playCards method keeps track on whether a first round is being played or not. It uses the '**firstRoundFlag**' for this. It helps decide who plays first as the player with the two of clubs starts the new game. There is a method called hasTwoOfClubs in the hand class that checks if a player has the two of clubs, so this is used to work out who goes first in a new game.

Depending on which player has the two of clubs, they are assigned to the **firstplayer** variable.

When a player plays a card, it is stored in a card array called **table[ ]**.

The program makes use of a Scanner variable called **pause** to halt the program as each player plays a card. This gives the player a chance to review cards played by the computer.

The game is broken up into rounds of cards (aka a trick). This is when all players have played a card. As played cards are stored in the table[] array, so there is a test to ensure that a card is taken from each player and the code for this is:

```
while (table[0]==null || table[1]==null || table[2]==null || table[3]==null)
```

On the second and subsequent rounds (prints out 'Starting new round'), the system gets player to play card based on setting for **firstPlayer**.  At the same time, the system specifies the next player using the variable **nextPlayer**. This ensure there is an appropriate sequence of play from player to player.

When a round has finished, there is a method checkRoundResults to see who has won the round and to assign firstPlayer to the winner of the round as the round winner always starts the next round. This method also calculates penalties for holding a heart or the queen of spade if you have won the rounds. It is a bit of an anomaly calling a player a winner as winning means you are awarded penalty points. But such is the way the game is played. The eventual winner of the game is the person with the least number of penalty points. So the aim of the game is to win as few games as possible.

The playCards method continues until players run out of cards or a player exceeds 100 points and then the **CheckGameResults()** method is run.

This method finds out who has the least number of penalty points and awards them the winner.

## A few game features that have yet to be implemented:

- Hearts cannot be played unless a player has no trump card, so boolean **heartsPlayed** is set to false until a player plays a heart when they have no trump card. The heartsPlayed variable defaults to false. Players should not be allowed play a heart until the heartsPlayed is set to true.
- At end of each round, it might be useful to provide players with a roundup of scores: showing how much each player had accumulated in penalty points.
- Keep a tally of games played and how many games each player has won.
- Change the text display of cards to graphical images.