CSE 331: Deep Gold API
Fall 2004
Dr. John Stewman

Group Members:
Garrett Britten
Daniel Mack
Jared Sylvester
Brian Bien

GAME CLASS:

Constructors:

game(struct GameFlow*);

- Default constructor takes a pointer to the GameFlow struct and initializes the data members to default values while also generating the table of weights for hole card possibilities

Functions:

void init(int num, const vector<int>& stacks, int dealerNumber)

- A quick initializer for a game class instance that takes arguments num to set the number of players in the game, stacks to set the chip count for each player, and dealerNumber to set the initial dealer

double stackSize(int n)

- Function to ask for the active player n and return that player's chip count as a double

void betRaise(double amnt)

- Function to bet or raise the amount indicative of both the limit and the round of betting and then moves the active player to the next eligible player

void callCheck(double amnt)

- Function that take an amount amnt and call checks that amount for the active player and then moves the active player to the next eligible player

void fold()

- Function to fold the active player and increment the active player to the next eligible player

void newHand()

- newHand() completes all necessary logic to initialize the game for a new hand. The dealer is incremented to the next player the blind positions are set, and the ante is forced. The potSize is updated as well as the new active player.

Void dealCard(string val)

- Function to take a string val in and deal a card to the computer player beginning with the first flop card. The val is parsed to construct a card and that card is added to the computer’s knowledge

Void pickWinner(int n)

- Function to declare winner based on passed in player number n

Void addHole(string c)

- Function that takes string c, parses it to create a card, and pushes it onto the hole vector. If hole is size 2, then it sets the hole cards for the computer

Enum actionNames think()

- Function the processes all knowledge of the current game state and proceeds to simulate and return a recommendation

Void genTable()

- Read from text file the weights of all hole card combinations and place information into a map

HAND CLASS:

Constructors:

Hand()

- Basic contructor that initializes one data member

Functions:

Void init(const card& 0, const card& t, const card& th, const card& fr, const card& fv)

- Initializer that takes in five cards into a cards vector and sorts them by rank

Void addCard(const card& next)

- Function to add card next to the hand in order

handType getType()

- Function to return a type of hand in the form of an object type defined as handType

handType typeOf(const vector<card>& h)

- Pass a vector of cards to determine and return a handType

void hand::addcombo(int x1, int x2, int x3, int x4, int x5)

- Given five ints as index into a card vector, it creates a possible hand

Void clear()

- Clear out vectors

Bool isStraight(const vector<card>& h, handType& type)

- Determines whether or not the cards are a straight and returns true or false

Bool isFlush(const vector<card>& h, handType& type)

- Determines whether or not the cards are a flush and returns true or false

bool is4kind(const vector<card>& h, handType& type)

- Determines whether or not the cards are a 4 of a kind and returns true or false

bool is3kind(const vector<card>& h, handType& type)

- Determines whether or not the cards are a 3 of a kind and returns true or false

bool isFull(const vector<card>& h, handType& type)

- Determines whether or not the cards are a full house and returns true or false

bool is2pair(const vector<card>& h, handType& type)

- Determines whether or not the cards are a set of pairs and returns true or false

bool isPair(const vector<card>& h, handType& type)

- Determines whether or not the cards are a pair and returns true or false

HOLECARDS CLASS:

Constructors:

holeCards()

Default Constructor, nothing is initialized or set

Functions:

Void setCards(const card& c1, const card& c2)

- Takes two card objects and orders them and sets them in the object’s data member

String getCards()

- Returns holeCards as a string

Card firstCard()

- Returns a card object that hold the first card in the hole card object

Card secondCard()

- Returns a card object that hold the second card in the hole card object

STATE CLASS

Constructor:

State(list<int> ops, double pt, double ms):pot(pt), my_share(ms), gameOver(false), compout(false)

- Basic constructor to setup default values for data members

State(State& cpy)

- Copy constructor

Functions:

Void Fold()

- Lets state know that fold action has occurred

Void Bet()

- Lets state know that bet action has occurred

Void Call()

- Lets state know that call action has occurred

double gainloss()

- Returns winnings

Int getIndex(int n)

- Function that takes int n as player number and returns index into player vector for that player

void setHole(int n, holeCards hc)

- Set hole cards for a player n

Void dealCard(string s)

- Deals a card matching string

Void updateRound(bool called)

- Taked called and round to move to next round or end hand

Void pickWinner()

- Uses hand evaluation to determine the winner of a hand among remain players in state

HUMAN CLASS

Constructors:

humanPlayer()

- Default constructor

humanPlayer(int cc = 0)

- Constructor to set default values and chip count set to int cc

Functions:

Void addChips(int amnt)

- Take amnt and adds to the chip count and resets bust

Double aggressiveness()

- Returns aggressiveness metric

Double tightness()

- Returns tightness metric

Void fold()

- Set player flag to out

Void betRaise(double amt)

- Function that takes amt and decrements from player chip count

Void checkCall(double amt)

- Function that takes amt and decrements from player chip count

Void busted()

- Sets bust to true

Map<string, double>* getWeights()

- Links to the weight map

Double stackSize()

- Returns player chip count

Void unfold()

- Changes out to false

Bool checkBust()

- Returns state of bust

Bool checkFold()

- Returns state of fold

Double getWgt(holdCards hc)

- Takes holdCards hc, determines strength of the hole cards and returns that weight as double

Void pointToTable(map<string,couble>* mptr)

- Set weightTable to point to mptr

Void wonHand(double n)

- Increments chip cound for player n

Void addFlopSeen()

- Increments flop cards seen

Void foldBeforeFlop()

- Decrements flop card seen

COMP CLASS

Constructor:

compPlayer()

- Constructor that initializes the compPlayer

Functions:

Void addChips(int amnt)

- Take amnt and adds to the chip count and resets bust

Void setHoleCards(card a, card b)

- Set the hole cards for the computer player

Void fold()

- Set player flag to out

Void betRaise(double amt)

- Function that takes amt and decrements from player chip count

Void checkCall(double amt)

- Function that takes amt and decrements from player chip count

Void busted()

- Sets bust to true

Void newHand()

- Creates a new hand that resets some of the class data members

Void addCard(card next)

- Adds a card to the hand of the computer player

Void clearCards()

- Clears the cards variables used by the computer

Void unfold()

- Changes out to false

bool checkBust()

- Returns state of bust

bool checkFold()

- state of fold

enum actionNames makeDec()

- Calls the computer decision making function and returns the appropriate action

Void pointToOppontents(vector<humanPlayer*> x)

- Points opposition member to x pointer

Void opponentBet()

- Sets betPlaced flag to true

Void opponentRaised()

- Set raiseMade flag to true

Void setSeatNumber(int n)

- Sets posAtTable to position n

Void setPotSize(double x)

- Set potSize to value x

Void wonHand(double n)

- Increase chip count by n

Void setDealer(int n)

- Set dealer to passed int n