Ian Benson, Jonathan Carter, Joey Palchak, Rachel Temple, Cj Zhang

SPECIFICATIONS

The following program specifications are for an implementation of the card game *Moose in the House*. The application will allow a user to play the game through an intuitive, user-friendly graphical interface, against a modifiable number of computer opponents, starting with a randomly shuffled deck of 58 cards and proceeding through the gameplay until either the user-player or a computer opponent has won.

MitH_deck(): class object representing the deck of 58 cards used to play the game Moose in the House. The deck will be represented as an array of the MitH_card class objects.

Data:

- 58 Cards (an array of the Card class)
  - 20 empty room cards (5 each living room, kitchen, bedroom, bathroom)
  - 20 full room cards (5 each as above)
  - 10 There's a Moose in the House cards
  - 5 Door cards
  - 3 Moose trap cards

Methods:

- draw() - draws a new card from the deck and returns it.
- shuffle() - randomizes the ordering of cards in the deck.
- numCards() - returns the current number of cards in the deck.
- reset() - resets the deck back to a full, randomized deck.
- removeMitH() - removes all Moose in the House cards.
  - This is a departure from how the actual game is played, but it makes sense in a computer game - perhaps a "fidelity option" for the user.

MitH_card(): class object representing an individual card in the deck of 58 cards. There are 5 different types of cards: rooms, occupied rooms, moose in the house, door, and trap.

Data:

- Card enum type
  - one of {ROOMEMPTY, ROOMMOOSE, DOOR, MitH, TRAP}
- Card string/enum value
  - If type == ROOMEMPTY or ROOMMOOSE.
  - One of {Bedroom, LivingRoom, Kitchen, Bathroom}.
- Graphics/Images of the card faces.

Methods:

- All values for the cards are set in constructor.
- Getters for type, value and graphic.

MitH_hand(): class object which will keep track of the cards in each individual player's hand, represented as an array of class MitH_card objects. This class takes advantage of class object MitH_card and MitH_deck.

Data:

- Array list, acting as a container for up to five cards.

Methods:

- checkForType(type) - checks the players hand to see what type of cards they hold.
- checkForValue(value) - checks the value on the room cards.
- numCards() - returns the total number of cards in the user/cpu's hand.
- getCard(type, value) - returns the specified type and value of a card and removes the card from the hand.

- addCard() - A card is removed from the deck and added to the hand. The class MitH_deck is called here.
- removeCard(value,type) - a card of a certain value/type is removed from the hand
-

MitH_player(): class object containing the movesets (along with their validation methods) for the user to play the game Moose in the House. This class also creates a computer representation of a player, with basic AI, allowing a user to play against multiple computer players. This class takes advantage of class object MitH_hand (and MitH_card).

Data:

- The player's hand of cards.
- The player's "house" cards.
  - rooms (empty, door, moose, trap), moose in the house
- The number of moose cards in hand.
- Audio clips.

Methods:

- These methods take advantage of MitH_hand & MitH_deck methods.
- numCards() - returns the number of cards in the player's current hand.
- getCard(type, value) - returns the card with type and value, and removes it from the player's hand.
- playCard(card, player) - plays card on chosen player, adding to their "house".
  - Move validation method(s): valid moves depend on the card type & value, and the respective compatible cards in opponent player's house.
- checkHouseType(type) - checks if the player's house contains the card of specified type.
- checkHouseValue(value) - checks the player's house for the specified value of room cards.
- playAudio(sound) - plays specified audio clips.
- Move validation method(s).

MitH_game(): main game file class which manages the game play for Moose in the House. Runs all class files, and the GUI for user input. The class closes when the user quits the game.

MitH_GUI(): class will represent the game board and give a visual representation of the cards, discard pile and deck. As the game progresses the GUI will adjust its representation of the game. The game board consists of each players "house", the deck and the discard pile. The users hand will also be represented. Users will have the option to choose how many CPU players there are per game. The GUI will incorporate all class objects and their methods according to the user input.

Visual representations:

- The card game's deck.
  - The image of the deck diminishes as new cards are pulled; a visual representation of the game's progress.
- The discard pile.
- Representation of each player and their hands / house.
- User's hand of cards / possible moves.

Data:

- A representation of the discard pile.

  - The ability to customize the number of players per game which modifies the game board.
  - Each players house is visible as well as the number of cards they hold.
  - A “Help” button that will open a new window with the instructions for the game.
  - A “Quit” button that ends and closes the game.

  User input:
- Drag and drop abilities on cards
- Registered button clicking

TASK LIST

CLASSES

MitH_deck: (Cj, Rachel) (medium-easy / necessary)

- data
  - 58 Cards (array of card class) (medium / necessary)
    - 20 empty room cards (5 each living room, kitchen, bedroom, bathroom)
    - 20 full room cards (5 each as above)
    - 10 There's a Moose in the House cards
    - 5 Door cards
    - 3 Moose trap cards
- methods
  - draw() (easy / necessary)
  - shuffle() (easy / necessary)
  - numCards() (easy / necessary)
  - reset() (easy / necessary)
  - removeMitH() (easy / desired)

MitH_card: (Joey, Rachel) (medium-easy / necessary)

- data
  - enum type (easy / necessary)
  - string/enum value (easy / necessary)
  - graphic - image of the card (medium-easy / desired)
- methods
  - all values set in constructor
  - getters for type, value and graphic (easy / necessary)

MitH_hand: (Jon, Joey) (medium-easy / necessary)

- data
  - container for up to five cards (easy / necessary)
- methods
  - checkForType(type) (easy / necessary)
  - checkForValue(value) (easy / necessary)
  - numCards() (easy / necessary)
  - getCard(type, value) (easy / necessary)
  - addCard() (easy / necessary)
  - removeCard(value, type) (easy / necessary)

MitH_player: (Jon, Ian) (medium / necessary)

- player / computer subclasses (medium / necessary)
  - computer AI (medium-hard / desired)
- Data:
  - player's hand of cards (easy / necessary)
  - player's "house" cards (medium-easy / necessary)
  - number of moose cards in hand (easy / necessary)
  - audio clips (medium-hard / wish list)
- Methods:
  - makes use of MitH_hand & MitH_deck methods

- numCards() (easy / necessary)
- getCard(type, value) (easy / necessary)
- playCard(card, player) (medium / necessary)
- move validation method(s) (medium / necessary)
- checkHouseType(type) (easy / necessary)
- checkHouseValue(value) (easy / necessary)
- playAudio(sound) (medium-hard / wish list)

MitH_game (CJ, Rachel) (easy / necessary)

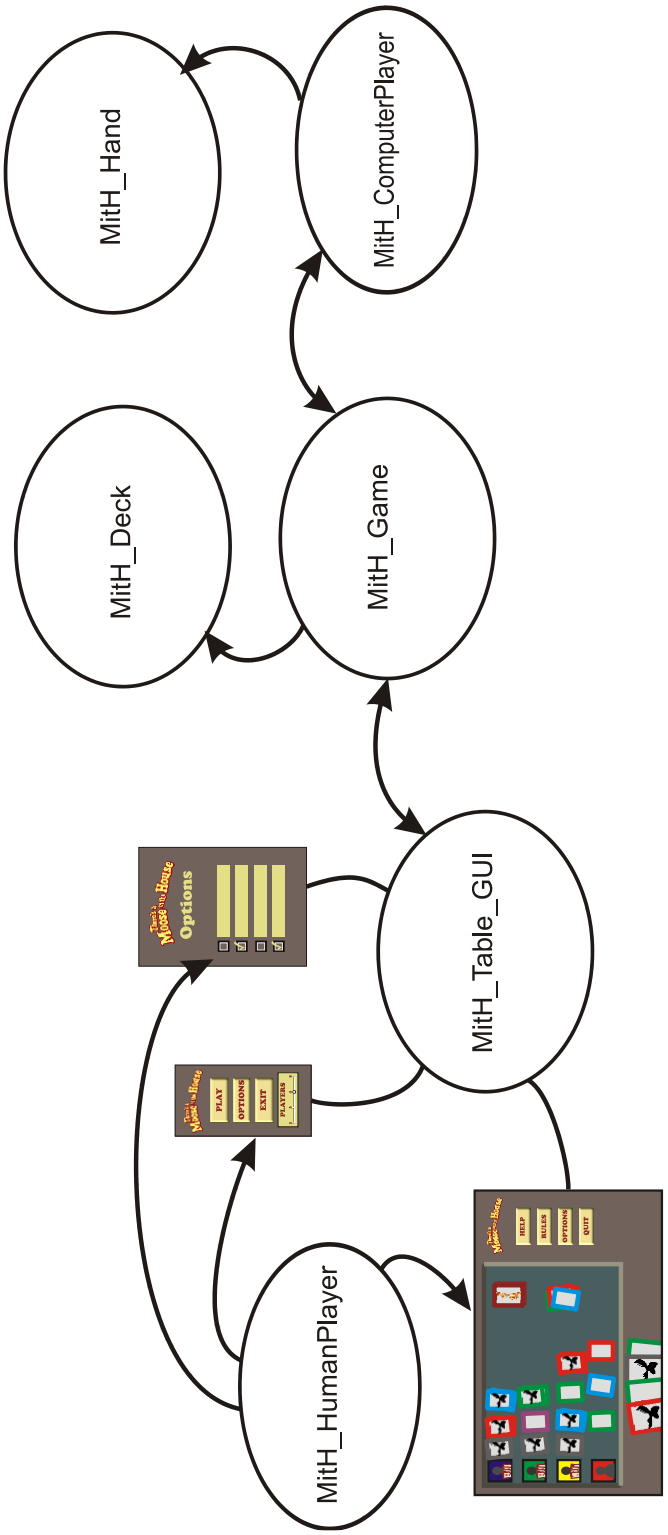
- Manages the gameplay of There's a Moose in the House
  - run GUI class for user input, closes when user quits

MitH_table_GUI (Ian, Joey) (medium / necessary)

- The graphical representation of the game table
  - scan cards (easy / necessary)
  - animations (hard / desired)
  - "pretty" GUI (medium / desired)
- Incorporate class objects & methods according to user input (medium / necessary)
- Data:
  - representation of the discard area (medium / necessary)
  - representation of player hands and houses (medium / necessary)
    - number of players customizable, modifies the game table
  - representation of card deck (medium / necessary)
    - have deck diminish as cards are removed; shows game progress
  - buttons for Help/Quit/… (medium / necessary)
    - help and menus should either open in overlays or separate windows
  - user input: drag & drop, clicking/buttons (medium / necessary)
  - settings (hard / wish list)

GUI Elements

Control Flow



Class Hierarchy