Biplab Thapa Magar

OPL – Fall 2020

Manual for Pinochle for C++

**Bug Report**

No known bugs in program

**Feature Report:**

Missing features:

All features have been implemented

Extra features:

No extra features were implemented

**Description of Data Structures/Classes:**

**Classes:**

1. **Card**: Represents a card in a game of Pinochle. Each card is comprised of a rank (of enum type Rank), suit (of enum type suit), and id.
2. **Deck**: Represents the stack of 48 cards to be used for the game of Pinochle. It also acts as the stock pile once cards have been taken out of the deck and distributed. Consists of a vector of 48 card objects
3. **Game:** The Game class represents the game of Pinochle. It manages all the rounds in a game and keeps track of the overall game score
4. **Round:** The Round class represents a round in a game of Pinochle. It manages everything about a round, including saving and loading data from files to populate a round in a game. It makes use of Player objects, Card objects, and Deck objects. It handles the playing out of a round of Pinochle and mediates the game between the players.
5. **Serialization**: This class handles the conversion of string data from file into object data concerning the cards that each player possesses in their various card piles. It also handles converting the same object data into serialized data to store in a file. In doing this, it interprets and translates how game state is stored in the save file and in the program during gameplay. It makes use of GroupOfCards and MeldsStorage objects.
6. **Player:** Represents a player in a game of Pinochle. It also generates the logic for what card to play and what melds to play. This logic is used differently by its child classes. It has two child class: Human and Computer. Its composition includes MeldServices object and GroupOfCards object.
7. **Computer:** Represents the computer player in a game of Pinochle. It inherits from the Player class. It uses logic from Player class to play cards and melds during a game of Pinochle.
8. **Human:** Represents a human player (user) in a game of Pinochle. It handles prompting user for input on what cards to throw and what cards to play for meld. It inherits from the Player class. It also provided suggestions, using logic from the player class, to the user on what card/melds to play
9. **GroupOfCards:** Represents a group of cards, in any order, in a game of Pinochle. It handles retrieval and searching of cards in terms of various parameters.It has one child class: MeldInstance.
10. **MeldInstance:** Represents a group of cards that combine to create a meld. It is a child class of GroupOfCards. A meld instance can contain cards that do no combine to create a meld as well. However, the MeldInstance object marks itself as an invalid meld if that happens. This class self-contains the logic to validate whether a group of cards combine to create a meld or not. MeldInstance makes use of Card class objects.
11. **MeldsStorage:** Represents a group of MeldInstance objects. MeldsStorage stores MeldInstance objects by separating them based on meld types. A MeldsStorage object can be used to store the melds a player has played during a round.
12. **MeldServices:** This class encapsulates various services pertaining to melds. It is used especially by the Player class and its subclasses to help figure out what melds can be played and should be played. It also keeps track of the melds a player has played during a round. It uses a MeldsStorage object to store players’ melds.
13. **StringUtilities:** This class encapsulates the various functionalities for cleaning, interpreting, parsing, and converting strings in the Pinochle program. It is a utility class whose functions are all static functions.
14. **PinochleException:** A class that inherits from the C++ exception class. This class represents an exception in the Pinochle program and has to do with exceptions pertaining especially to the game’s logic.

**Enums:**

**Rank** : Ace, Ten, King, Queen, Jack, Nine

**Suit:** Clubs, Hearts, Diamonds, Spades

**Meld:** Flush, RoyalMarriage, Marriage, Dix, FourAces, FourKings, FourQueens, FourJacks, and Pinochle

**Log**

**-**Sept 8, 2020

-UML designed classes for Pinochle

Total: 2.5hours

-Sept 9, 2020

-Scaffolded the Card class

-Scaffolded the Deck class

-Implemented getters and setters functions in the Card Class

-Implemented constructor and shuffle function in Deck Class

Total: 2 hours

-Sept 10, 2020

-implemented functions for Card class for getting string representation of the class

-Created the Pinochle exception class and added exception handling in Card and Deck classes

-Scaffolded the Player class, Human class, and Computer class

Total: 3 hours

-Sept 13, 2020

-implemented playLeadCard, playChaseCard, promptUser and promptCardToThrow functions in Human class

Total: 2 hours

-Sept 14, 2020

-Redesigned classes to separate out logic concerning melds into a new class: MeldsServices.

-Added checkMeldValidity class to validate if a group of cards is a meld.

Total: 4.5 hours

-Sept 15, 2020

-Refactored code to reflect new logic. Each Card class now also has an id property that uniquely identifies each card. Refactored the populate function in Deck class to generate 48 unique cards

Total: 1 hour

-Sept 16, 2020

-Created two new classes: Group Of Cards and MeldInstance in order to store a collection of cards.

-Moved meld validation logic to Group of Cards

-used GroupOfCards to store the players’ hands and capture pile instead of a vector of cards

Total: 3 hours

-Sept 17, 2020

-Created MeldsStorage class to store MeldInstance objects. MeldsStorage is now used by MeldServices to keep track of the melds a player has played

-Redesigned getMeldsFromHand, getDixes, getPinochles, getMarriages, and all the other functions that calculate the possible melds from hand so that now these functions return the actual meld instances instead of just the number of melds of a particular type that a player has created in the round

Total: 3 hours

Sept 18, 2020

-Implemented potentialPointsFromHand, compareHandsForMelds, meldIsNotARepeat, and allCardsPresentInHand functions in MeldServices class

-Added isCardUsedByMeld, isCardUsedByAnyMeld, and cardsUsedForSameMeld function in MeldsStorage class

-fixed issues with getMarriage function

Total: 3.5 hours

Sept 19, 2020

-Refactored the function getSameSuitMelds in MeldServices class so that it makes sure a group of cards that have together already made a meld before cannot be used to create a meld again.

-Implemented Game class

-Scaffolded Round class, and implemented prompting user for action

Total: 2 hours

Sept 20, 2020

-Moved logic for string declaration into a separate StringUtilities file, which contains a list of functions for string usage. Did this because lots of classes are using the same code frequently

-Added several utility functions to MeldsStorage, MeldServices and MeldInstance, including addCard, addCards, getAllMeldsUsingCard

Total: 1 hours

Sept 23, 2020

-Implemented computer logic for best lead card and best chase card to play

-Implemented computer logic for best meld to play

Total: 4 hours

Sept 24, 2020

-Debugged suggestMeld card and extended it to include logic for what to do when two melds produce the same points.

-Added public functions to Player class to interface it with the Round class, which can now ask player to play a lead card, chase card, and meld

-Added to Human and Computer class so that they now respond to the client asking them to make a move

-Implemented Round class to get the game loop going

Total: 5 hours

Sept 25, 2020

-Scaffolded Serialization class and implemented functions to convert object representations of cards into string representations

-Implemented the Save File feature in Round class

Total: 2 hours

Sept 26, 2020

-Added the load file feature in the Round class and function to read and interpret the file content

-Added more StringUtility functions. Changed StringUtility into an actual class consisting solely static functions

-Implemented functions to convert hand, capture pile, and melds string into their object representations

Total: 5 hours

Sept 27, 2020

-Completed the logic in Round class for loading from a save file

-Completed the setting up of the game class

Total: 1.5 hours

Sept 28, 2020

-Tested the entire program.

-Found and solved problem in Serialization class, wherein I incorrectly accounted for Melds within the serialization file that have asterisks

-Further debugged minor errors

-Tweaked console user interface to make it easier for user

Total: 3 hours

Sept 29, 2020

-Finished documenting the program

Total: 2 hours

**How to Run**

1. In terminal, navigate into the src directory
2. Enter “make” into the terminal and press enter
3. Enter “./pinochle” into the terminal to run the game of pinochle
4. Enter “make clean” into the terminal once you are done with the game

**Screenshots**

1. Starting a new game

**Text

Description automatically generated**

1. Coin tossText

   Description automatically generated
2. Asking computer to make a move

Text

Description automatically generated

1. Asking for help throwing a cardText

   Description automatically generated
2. Computer Winning a turn

Text

Description automatically generated

1. Human making a move

Text

Description automatically generated

1. Asking for help for a meld

Text

Description automatically generated

1. Human creating a meldText

   Description automatically generated
2. Human winning a turn

Text

Description automatically generated

1. Asking if user wants to play another round

Text

Description automatically generated

1. Player refusing to play another round

Text

Description automatically generated

1. Loading a previous game from a save fileText

   Description automatically generated
2. Player quitting in the middle of a gameText

   Description automatically generated
3. Saving a game

Text

Description automatically generated