| **Group** | **Module** | **Task** | **Description** | **Uses methods of** | **Used by** | **Depends on** |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | Controller | general | Contains the main (the entry point).  If a command line arg for loading a saved game is present, read the file name. |  |  |  |
| Dealing new hand or loading saved game | Deletes old game and gameView, if necessary.  Creates new game (uses factory method and IGame).  Creates new gameView (uses factory method and IGameView). Passes the game to the gameView.  For each cardAreaView in the gameView, create a subwindow and pass the cardAreaView to the subwindow. Get the position and extent for the subwindow from gameView.  For each buttonLabel in the gameView, get its position and create a button. | IGame  newGame()  IGameView  newGameView() |  | IGame  IGameView  CardAreaView |
| Painting the screen | Tell each subwindow to repaint itself.  Each subwindow tells its cardAreaView to display itself. | CardAreaView |  | CardAreaView |
| When a mouse click occurs | Each subwindow receives its own mouse clicks, but they all do the same thing.  Use cardAreaView to find out which card was selected. An ICardCollection, ICardList, and card index describe the selected card.  Pass these objects to the game to tell it which card was selected.  Then make sure all subwindows are repainted. | CardAreaView  IGame |  | CardAreaView  ICardCollection  ICardList  IGame |
| When a button click occurs | Pass the command (e.g. the button label) to the game. If the command was to save the game state, then first generate a file name and pass that also.  Then make sure all subwindows are repainted. | IGame |  | IGame |
| 3 | IGameView |  | An interface and factory method used to keep the controller from knowing about the different GameViews. |  |  |  |
| 3 | Solitaire  GameView | general | Implements IGameView, and supports the factory method for creating itself.  Created and deleted by Controller.  Holds 4 cardAreaViews (5 for RummyGameView).  Knows the position and extent for each cardAreaView.  Knows what buttons should be created and the position of each. |  | Controller |  |
| Initialization | Uses the game to get the appropriate cardCollection for each cardAreaView, then creates each cardAreaView and tells each its position and extent.  Passes the game’s stock, discard, foundation, tableau, hands (as appropriate for the game) to the cardAreaView. | ISolitaireGame  CardAreaView | Controller | ISolitaireGame  ICardCollection  CardAreaView |
| 3 | Rummy  GameView | Like Solitaire  GameView. |  | IRummyGame  CardAreaView | Controller | IRummyGame  ICardCollection  CardAreaView |
| 6 | CardArea  View | general | Created and deleted by a gameView.  Holds an ICardCollection reference (which is really a Deck, Stack, …). Knows/computes the position and extent of each card in the cardCollection.  You may want to use different subclasses in order to handle different kinds of ICardCollections (esp. RowOfStacks vs. the others). In this case the gameView creates the subclass rather than CardAreaView. But the Controller always thinks it’s dealing with a CardAreaView. |  |  |  |
| Display | Iterates through the cards in cardCollection. The first card goes to a fixed location, and each subsequent card goes to the previous location plus an increment.  Tells each card to display itself at the appropriate location.  Iterate from first to last in order to get overlaps showing correctly. | ICardCollection  IListOfLists  ICardList  ICard | Controller | ICardCollection  IListOfLists  ICardList  ICard |
| Determine which card is under the cursor (for a mouse click). | Iterates through the cards in cardCollection. If the cursor lies within that location and extent, return the appropriate ICardCollection, ICardList (could be same as ICardCollection for simple lists), and cardIndex.  Iterate from last to first in order to handle overlaps correctly. | ICardCollection  IListOfLists  ICardList | Controller | ICardCollection  IListOfLists  ICardList |
| 6 | ICardView |  | An interface and factory method used to keep Card from knowing which class implements ICardView. |  | Card |  |
| 6 | CardView | general | Implements ICardView and supports the factory method for creating itself.  Created by CardAreaView but owned and deleted by Card.  Holds references to the images for the front and back of the card. |  |  |  |
| Initialization | Receives the suit and rank, then uses CardImage to get references to the bitmaps for the front and back of the card. | CardImage | CardAreaView | CardImage |
| Display | Given (x, y) position and booleans for isFaceUp and isSelected, it calls the Win32 draw method to display the appropriate bitmap. A selected card can be drawn by applying XOR to the usual bitmap. |  | CardAreaView |  |
| 6 | CardImage | general | This is never instantiated; it contains only class method and class attributes. |  |  |  |
| Display | Given a suit and rank, returns the card front and back (references to bitmaps). Load the bitmap if it hasn’t already been loaded. | - | CardView | - |
| 4 | IGame |  | An interface and factory method that provides a generic interface for Controller, with methods for the Controller to call when a card is selected or a button is clicked. |  | Controller |  |
| 2 | ISolitaireGame |  | Inherits from IGame. Provides getters for the stock, discard, foundation, and tableau. |  | Solitaire GameView | IGame |
| 5 | IRummyGame |  | Inherits from IGame. Provides getters for the stock, discard, foundations, and both hands. |  | Rummy GameView | IGame |
| 4 | Klondike SolitaireGame | general | Implements ISolitaireGame and supports the factory method for creating itself.  Holds the stock, discard, foundation, etc. It creates and destroys these when it is created and destroyed (by the Controller). |  |  | ISolitaireGame  Deck  Stack  RowOfStacks  Card  Matchmaker |
| Initialization | Allows controller to either initialize a new hand or load a saved game, immediately after the controller creates it.  For a new hand, it creates the appropriate cards for the stock, then deals the first hand.  For loading a saved game, it creates a card for each one in the file, and places it in the correct place according to the file (e.g. in stock, in discard, etc.). | Deck  Stack  RowOfStacks  Card | Controller |  |
| Initialization | Provides getters for stock, discard, foundation, etc. |  | Solitaire GameView |  |
| Card selected | Determine what move to make; e.g. set one card as selected, set a group as selected, unselect a selected card, move a set of cards from one stack to another, deal a set of cards to the stacks, etc.  The game must be in the appropriate state and the command must be legal before executing it. This object keeps track of whether it’s in the appropriate state, and the Matchmaker helps check for a legal move. | Matchmaker  Deck  Stack  RowOfStacks  Card |  |  |
| Button command | Perform the requested command (e.g. save to file). |  |  |  |
| 2 | Spider SolitaireGame |  | Similar to KlondikeSolitaireGame. |  |  |  |
| 1 | RummyGame |  | Implements IRummyGame.  Similar to KlondikeSolitaireGame, but when the player’s turn is finished, it uses the AIOpponent to update the model. (Beware: this will be a recursive call.) | AIOpponent |  | IRummyGame  Deck  Hand  Stack  RowOfStacks  Card  Matchmaker AIOpponent |
| 5 | GinRummy Game |  | Similar to RummyGame. |  |  |  |
| 2 | Matchmaker | general | Either created and deleted by one of the Games, or it’s never instantiated (class methods and attributes only). |  |  |  |
| Checking a move | Computes things like whether the cards are in an ascending run, a descending run, all the same suit, alternating colors, all the same rank, etc. | ICardList  ICard | any Game | ICardList  ICard |
| 5 | AIOpponent | general | Created and deleted by one of the RummyGames. Gets a reference to its hand and the foundation. |  | a Rummy Game |  |
| Player’s turn is over | Given a ref to the top card in the discard pile, it checks its hand and the foundation stacks to figure out whether to draw from the stock or the discard, what cards to meld/lay off, and what card to discard.  It uses the same methods as the Controller to make a move. | IListOfLists  ICardList  ICard  IRummyGame | a Rummy Game | IListOfLists  ICardList  ICard  IRummyGame |
| 6 | ICardCollection |  | Generic interface used by CardAreaView. |  | Controller  CardAreaView |  |
| 5 | IListOfLists |  | Inherits from ICardCollection.  Contains methods to iterate over its list, and returns its elements as ICardList. |  | CardAreaView  AIOpponent | ICardCollection  ICardList |
| 1 | RowOfStacks |  | Implements IListOfLists.  Creates and stores the elements of the list as Stack.  Provides additional methods for adding and remove elements, and reading the list from a file, writing the list to a file.  Handles destruction of its Stacks. |  |  | IListOfLists  Stack |
| 4 | ICardList |  | Inherits from ICardCollection.  Contains methods to iterate over its list, and returns its elements as ICard. |  | Controller  CardAreaView  Matchmaker  AIOpponent | ICardCollection  ICard |
| 4 | CardList |  | Implements ICardList.  Creates and stores the elements of the list as Card.  Provides additional methods for reading the list from a file and writing the list to a file.  Handles destruction of its cards. |  |  | ICardList  Card |
| 4 | Deck |  | Is a subclass of CardList.  Provides methods such as for shuffling, removing the top card, etc. |  | all Games | CardList  ICard |
| 1 | Stack |  | Is a subclass of CardList.  Provides methods such as push, pop, pushGroup, etc. |  | all Games | CardList  ICard |
| 4 | Hand |  | Is a subclass of CardList.  Provides methods such as sortBySuit, sortByRank, add, remove, etc. |  | a Rummy Game | CardList  ICard |
| 1 | ICard |  | Provides methods such as getSuit, getRank, isFaceUp, isSelected, getCardView. |  | CardAreaView  ICardList  Deck  Stack  Hand |  |
| 1 | Card |  | Implements ICard.  Holds a cardView (as an ICardView).  Handles destruction of its cardView.  Uses ICardView and a factory method to create the cardView. |  | CardList | ICard  ICardView |

MainController {

read command line arg to find out whether to load a saved game.

create main window with buttons.

set model, view, and controller to null.

call initializeGame().

go into an infinite loop.

}

method initializeGame() {

if model, view, or controller is not null, call the destructor(s).

create model, telling it whether to load a saved game or deal a new game.

create view (passing model to view)

create controller (passing model and view to controller)

}

main window {

knows its sub windows.

creates its sub windows.

main window method repaintAll():

invalidate each subwindow (which will trigger the paint event).

main window callback:

case button click:

if it was “save game state”, invoke controller’s saveGameState().

}

sub window {

knows its view object.

sub window callback:

case click on pane:

pass cursor position and my view object to controller (invoke handleCardSelect()).

invoke main window’s repaintAll() method

case paint:

tell my view object to show itself

}

controller {

method handleCardSelect(IList viewObj, cursorPosition)

cardIndex = viewObj->getCardIndex(cursorPosition);

game->selectedCard(viewObj, cardIndex);

method handleCardSelect(IListOfLists viewObj, cursorPosition)

listIndex = viewObj->getList(cursorPosition);

cardIndex = viewObj->getCardIndex(listIndex, cursorPosition);

game->selectedCard(viewObj, listIndex, cardIndex);

method saveGameState()

game->selectedSaveGame();

}