Tommy HoLam

qholam@wpi.edu

Overview

Design and implementation of Alhambra for CS3733 Software Engineering

Alhambra

Solitaire Variant

Contents

[User Cases 2](#_Toc447135133)

[Entities and Boundaries Model 5](#_Toc447135134)

[View 6](#_Toc447135135)

[Entity Classes 7](#_Toc447135136)

[Move Classes 8](#_Toc447135137)

[Move Class Descriptions: 8](#_Toc447135138)

[User Interactions: 9](#_Toc447135139)

[Move Class Pseudo Code: 9](#_Toc447135140)

[Controller Classes 12](#_Toc447135141)

[Controller Mapping: 12](#_Toc447135142)

[Controller Pseudo-code 13](#_Toc447135143)

# User Cases

|  |  |
| --- | --- |
| Use Case Name: | Initialize Game |
| Flow of Events: | 1. Player request to initialize game 2. System loads fresh state of variation |
| Entry Condition: | Game has not yet started |
| Exit Condition: | Deck is shuffled and one card is dealt face up into the waste pile. 8 foundation piles are placed onto the board, leftmost four (one of each suite) are built from up starting with the Ace and the rightmost four(one of each suite) are built down starting with the King. 8 reserve piles each with 4 cards are placed onto the board. |

|  |  |
| --- | --- |
| Use Case Name: | Deal one card |
| Flow of Events: | 1. Player requests to deal top card from deck face up onto to waste pile 2. Board updates to reflect move |
| Entry Condition: | Deck is not empty |
| Exit Condition: | Deck is reduced by one card. Counter for number of cards left in deck is updated accordingly. |

|  |  |
| --- | --- |
| Use Case Name: | Move card from reserve pile to waste pile |
| Flow of Events: | 1. Player requests to move a card from the reserve pile to the waste pile whose top card is the same rank and is one point apart 2. Board updates to reflect move |
| Entry Condition: | There exists at least one card from the reserve pile that has the same rank and is one point higher or lower than the top card of the waste pile. |
| Exit Condition: | The size of the reserve pile is reduced by one and the size of the waste pile is increased by one |

|  |  |
| --- | --- |
| Use Case Name: | Reassemble deck |
| Flow of Events: | 1. Player requests to reassemble the deck using all the cards in the waste pile 2. Board updates to reflect move |
| Entry Condition: | Deck is empty and the waste pile is not empty. The counter for the number of redeals is less than 2. |
| Exit Condition: | The pile of cards in the waste pile is flipped to be face down and placed onto the deck. The waste pile is empty. The counter for the number of redeals is incremented by 1. Counter for number of cards left in deck is updated accordingly. |

|  |  |
| --- | --- |
| Use Case Name: | Complete game |
| Flow of Events: | 1. Player requests to remove card from the deck 2. Game completes |
| Entry Condition: | Deck is empty and all foundation piles are complete |
| Exit Condition: | Game is complete |

|  |  |
| --- | --- |
| Use Case Name: | Move card from Reserve pile to King foundation pile |
| Flow of Events: | 1. Played requests to move card from Reserve pile to the target King  foundation pile whose top card has the same rank and is one point higher 2. Game updates to reflect move |
| Entry Condition: | There exists a card from Reserve pile that is the same rank and one point lower than the top card of a King foundation pile. |
| Exit Condition: | Size of Reserve pile is reduced by one and size of the target King foundation pile is increased by one. Score is updated. |

|  |  |
| --- | --- |
| Use Case Name: | Move card from Waste pile to King foundation pile |
| Flow of Events: | 1. Played requests to move card from a Waste pile to the target King  foundation pile whose top card has the same rank and is one point higher 2. Game updates to reflect move |
| Entry Condition: | There exists a card from a Waste pile that is the same rank and one point lower than the top card of a King foundation pile. |
| Exit Condition: | Size of Waste pile is reduced by one and size of the target King foundation pile is increased by one. Score is updated. |

|  |  |
| --- | --- |
| Use Case Name: | Move card from Reserve pile to Ace foundation pile |
| Flow of Events: | 1. Played requests to move card from a Reserve pile to the target Ace foundation pile whose top card has the same rank and is one point lower 2. Game updates to reflect move |
| Entry Condition: | There exists a card from a Reserve pile that is the same rank and one point higher than the top card of an Ace foundation pile. |
| Exit Condition: | Size of the Reserve pile is reduced by one and size of the target Ace foundation pile is increased by one. Score is updated. |

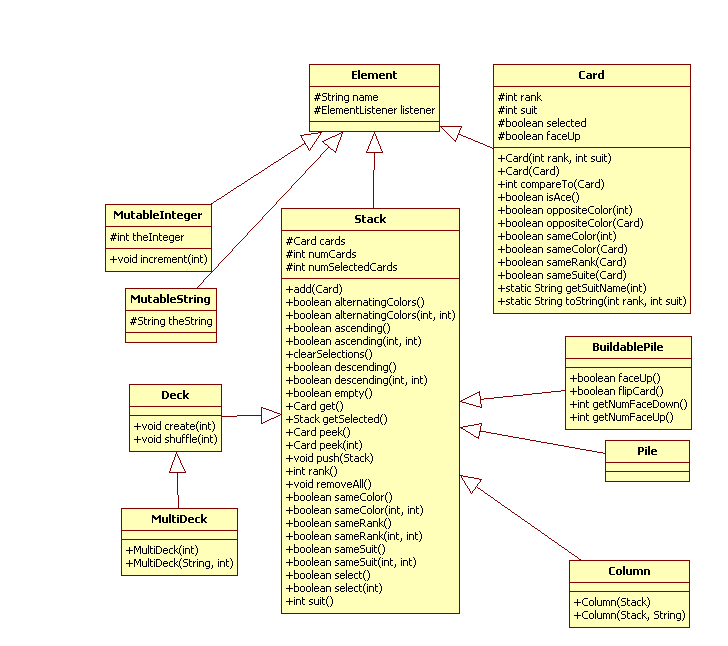
|  |  |
| --- | --- |
| Use Case Name: | Move card from Waste pile to Ace foundation pile |
| Flow of Events: | 1. Played requests to move card from a Waste pile to the target Ace foundation pile whose top card has the same rank and is one point lower 2. Game updates to reflect move |
| Entry Condition: | There exists a card from a Waste pile that is the same rank and one point higher than the top card of an Ace foundation pile. |
| Exit Condition: | Size of Waste pile is reduced by one and size of the target Ace foundation pile is increased by one. Score is updated. |

# Entities and Boundaries Model



# View

# Entity Classes



*NOTE: The above classes were written and provided by Professor George T. Heineman*

## Move Classes



### Move Class Descriptions:

* **DealOneMove:** Remove top card of deck and place it face up onto the waste pile. Number of cards left in deck is reduced by one and the score is left untouched.
* **ReserveToWastePileMove:** Move the single card that is being dragged from a Reserve pile to the Waste pile. The number of cards left in the deck and the score are left untouched.
* **ReserveToUpFoundationMove:** Move the single card that is being dragged from a Reserve pile to a foundation pile that is building up. The number of cards left in the deck is left untouched and the score is updated accordingly.
* **ReserveToDownFoundationMove:** Move the single card that is being dragged from a Reserve pile to a foundation pile that is building down. The number of cards left in the deck is left untouched and the score is updated accordingly.
* **WasteToUpFoundationMove:** Move the single card that is being dragged from the Waste pile to a foundation pile that is building up. The number of cards left in the deck is left untouched and the score is updated accordingly.
* **WasteToDownFoundationMove:** Move the single card that is being dragged from the Waste pile to a foundation pile that is building down. The number of cards left in the deck is left untouched and the score is updated accordingly.
* **ReassembleDeckMove:** The deck is rebuilt from the Waste pile. The number of cards left in the deck is set equal to the total number of cads in the Waste pile and the score is left untouched. There is no undo option for this move.
* **NOTE:** undo is initiated by the right click of the mouse

### User Interactions:

* **DealOneMove**: mouseClick on the DeckView when it is not empty.
* **ReserveToWastePileMove:** mousePress on a PileView to initiate the move by extracting the top card of the pile. mouseDragged to have the card being moved cover the PileView. mouseReleased on the target PileView to complete the move.
* **ReserveToUpFoundationMove:** mousePress on a PileView to initiate the move by extracting the top card of the pile. mouseDragged to have the card being moved cover the PileView. mouseReleased on the target PileView to complete the move.
* **ReserveToDownFoundationMove:** mousePress on a PileView to initiate the move by extracting the top card of the pile. mouseDragged to have the card being moved cover the PileView. mouseReleased on the target PileView to complete the move.
* **WasteToUpFoundationMove:** mousePress on a PileView to initiate the move by extracting the top card of the pile. mouseDragged to have the card being moved cover the PileView. mouseReleased on the target PileView to complete the move.
* **WasteToDownFoundationMove:** mousePress on a wsatePileView to initiate the move by extracting the top card of the pile. mouseDragged to have the card being moved cover the PileView. mouseReleased on the target PileView to complete the move.
* **ReassembleDeckMove:** mouseClick on deckView when it is empty.

### Move Class Pseudo Code:

|  |  |
| --- | --- |
| **DealOneMove** | **Logic** |
| valid | 1. Deck is not empty |
| doMove | 1. Move top card of deck to be face up on top of the waste pile 2. Reduce the counter for the number of card in the deck by one |

|  |  |
| --- | --- |
| **ReserveToWastePileMove** | **Logic** |
| valid | 1. Waste pile is not empty 2. Top card of the waste pile and the card being dragged are both of the same suite and one point apart(King wraps to Ace) |
| doMove | 1. Move top card of deck to be face up on top of the waste pile 2. Reduce the counter for the number of card in the deck by one |

|  |  |
| --- | --- |
| **ReserveToUpFoundationMove** | **Logic** |
| valid | 1. Foundation pile is not empty 2. Foundation pile has less than 13 cards 3. Top card of the foundation pile is of the same suite and one point lower than the card being dragged |
| doMove | 1. Move card being dragged to the top of the foundation pile. Update score. |

|  |  |
| --- | --- |
| **ReserveToDownFoundationMove** | **Logic** |
| valid | 1. Foundation pile is not empty 2. Foundation pile has less than 13 cards 3. Top card of the foundation pile is of the same suite and one point higher than the card being dragged |
| doMove | 1. Move card being dragged to the top of the foundation pile. Update score. |

|  |  |
| --- | --- |
| **WasteToUpFoundationMove** | **Logic** |
| valid | 1. Foundation pile is not empty 2. Foundation pile has less than 13 cards 3. Top card of the foundation pile is of the same suite and one point lower than the card being dragged |
| doMove | 1. Move card being dragged to the top of the foundation pile. Update score. |

|  |  |
| --- | --- |
| **WasteToDownFoundationMove** | **Logic** |
| valid | 1. Foundation pile is not empty 2. Foundation pile has less than 13 cards 3. Top card of the foundation pile is of the same suite and one point higher than the card being dragged |
| doMove | 1. Move card being dragged to the top of the foundation pile. Update score. |

|  |  |
| --- | --- |
| **ReassembleDeckMove** | **Logic** |
| valid | 1. Deck is empty |
| doMove | 1. Remove the card on top of waste pile one by one and place it on top of the deck |
| undo | 1. Return false |

# Controller Classes

## Controller Mapping:

|  |  |
| --- | --- |
| **View Widget** | **Controller Type** |
| DeckView | DeckController |
| PileView(waste pile) | WastePileController |
| PileView(ace up foundation) | AceUpFoundationController |
| PileView(king down foundation) | KingDownFoundationController |
| ColumnView(reserve column) | ReserveColumnController |

## Controller Pseudo-code

|  |  |
| --- | --- |
| **DeckController** |  |
| constructor | 1. Store the Alhambra game |
| mousePressed | 1. Find the deck from model and all waste pile 2. If deck is not empty, make a **DealOneMove** object using the deck and the waste pile. Attempt to make the move. If the move is valid, store it in the game 3. If the deck is empty, make a **ReassembleDeckMove** object using the deck and the waste pile. Attempt to make the move. If the move is valid, store it in the game. 4. Refresh all widgets that have been affected |

|  |  |
| --- | --- |
| **WastePileController** |  |
| constructor | 1. Store the Alhambra game and PileView for the waste pile |
| mousePressed | 1. Get a CardView for the top card in the PileView for the waste pile. If no such CardView exists, then ignore the action and return 2. Tell the Container that this CardView is being dragged and that the PileView for the waste pile is the source 3. Redraw the PileView for the waste pile |
| mouseReleased | 1. Get CardView for card being dragged from the Container which will be called *draggedObject* 2. Get card model element for that CardView which will be called *draggedCard* 3. Get the Widget of the source from the Container that initiated the drag; From this determine the model element which will be called *source* 4. If *source* is not a Column, then return dragged card to the source 5. Get PileView from this controller’s constructor; From that determine the pile model element which will be called *target* 6. Make a **ReserveToWastePileMove** using *source, target,* and *draggedCard*. Attempt to make the move. If the move is valid, then store it in the game; otherwise return *draggedObject* to the PileView it came from 7. Redraw all widgets affected 8. Have the Container release the dragger object and repaint |

|  |  |
| --- | --- |
| **AceUpFoundationController** |  |
| constructor | 1. Store the Alhambra game and PileView for the ace up foundation |
| mouseReleased | 1. Get CardView for card being dragged from the Container which will be called *draggedObject* 2. Get card model element for that CardView which will be called *draggedCard* 3. Get the Widget of the source from the Container that initiated the drag; From this determine the model element which will be called *source* 4. Get PileView from this controller’s constructor; From that determine the pile model element which will be called *target* 5. If *source* is a Column then make a **ReserveToUpFoundationMove** using *source, target,* and *draggedCard*. Attempt to make the move. If the move is valid, then store it in the game; otherwise return *draggedObject* to the PileView it came from 6. If *source* is a Pile then make a **WasteToUpFoundationMove** using *source, target,* and *draggedCard*. Attempt to make the move. If the move is valid, then store it in the game; otherwise return *draggedObject* to the PileView it came from 7. Redraw all widgets affected 8. Have the Container release the dragger object and repaint |

|  |  |
| --- | --- |
| **ReserveColumnController** |  |
| constructor | 1. Store the Alhambra game and ColumnView for the reserve column |
| mousePressed | 1. Get a CardView for the top card in the ColumnView for the reserve column. If no such CardView exists, then ignore the action and return 2. Tell the Container that this CardView is being dragged and that the ColumnView for the reserve column is the source 3. Redraw the ColumnView for the reserve column |

|  |  |
| --- | --- |
| **KingDownFoundationController** |  |
| constructor | 1. Store the Alhambra game and PileView for the king down foundation |
| mouseReleased | 1. Get CardView for card being dragged from the Container which will be called *draggedObject* 2. Get card model element for that CardView which will be called *draggedCard* 3. Get the Widget of the source from the Container that initiated the drag; From this determine the model element which will be called *source* 4. Get PileView from this controller’s constructor; From that determine the pile model element which will be called *target* 5. If *source* is a Column then make a **ReserveToDownFoundationMove** using *source, target,* and *draggedCard*. Attempt to make the move. If the move is valid, then store it in the game; otherwise return *draggedObject* to the PileView it came from 6. If *source* is a Pile then make a **WasteToDownFoundationMove** using *source, target,* and *draggedCard*. Attempt to make the move. If the move is valid, then store it in the game; otherwise return *draggedObject* to the PileView it came from 7. Redraw all widgets affected 8. Have the Container release the dragger object and repaint |