ECM 2414: Software Development

Coursework

Student Numbers: 690009073 and xxxxxxxxx

(weighting 50:50)

Class Diagrams:

|  |
| --- |
| Card |
| * cardValue -> int |
| + readValue() -> int |

|  |
| --- |
| Player |
| * playerNumber -> int * hand -> LinkedList<Card> * pullDeck -> Deck * putDeck -> Deck |
| + drawCard() -> Card  + checkWin() -> Void  + setPullDeck(Deck) -> void  + setPutDeck(Deck) -> void  + findDiscardLocation() -> int  + addCardToHand(Card) -> void  + discardCard(Card) -> void |

|  |
| --- |
| Deck |
| * deckNumber -> int * deck -> LinkedList<Card> |
| + addCard(Card) -> void  + getCard() -> Card |

|  |
| --- |
| Pack |
| * cards -> LinkedList<Card> |
| + loadPack() -> void throws IOException  + readFile(String) -> void  + getCard() -> Card  + addCard(Card) -> void |

Design Choices:

|  |  |
| --- | --- |
| Design Choice | Justification |
| Implementing the deck and pack as a linked list. | Java has a very easy to use library for linked lists. The deck and pack are ordered structures or a fixed length, so a linked list is a good choice. It is also easy to reach the top and bottom of the structure, which are the positions that need to be accessed most often in the program. |
| Choosing 4 classes | We felt that adding a Pack and Deck class would aid the development of the program, because it allows each deck to be an object in its own right. This way, we can create methods around the deck, rather than storing it as a data structure in the CardGame file. This in turn makes the code more readable, and easier to develop and maintain. Adding a Pack class allows us to keep all of the code relating to the handling of the initial text file in one place, once again making the code more organised. |
| Private attributes in the classes | This allows for encapsulation, which means that during the development, there will be fewer errors, and a smoother development process. |
| Card Class | The card class is kept very simple, as it is an object that needs to be instantiated lots of times. It only has one method, which is to read the value of the card. |
| Player Class | The player class has a ‘hand’ attribute, which holds the cards that the player has in a linked list. This allows us to separate each player’s hands to make threading easier, and we can set the size of the list to be 4, so that each player can only hold 4 cards as specified. We also defined the deck that the player will draw from and discard to as separate attributes to improve readability, and also separate the decks for easier threading. We kept the checkWin() method inside the player class so that each player can check through their own hand simultaneously. |
| Deck Class | In the deck class, we also use a linked list to store the cards. The linked list does not have to have a fixed length, so it is easy for the code to be flexible for the number of cards in the pack. The addCard() method is in the deck class so that the deck can be kept private to enable encapsulation. |
| Pack Class | Once again, a linked list is used for the pack because it allows for a variable pack size. |