**Development Team:**

Paul Nguyen

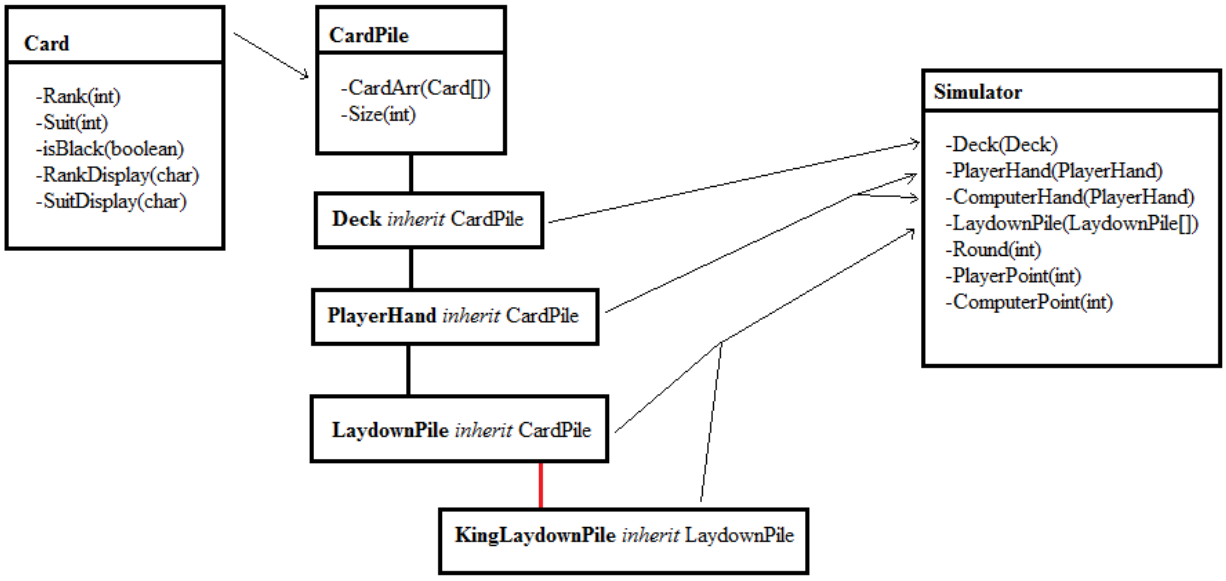
**Project:**

King's Corner version 1.0

**Purpose:**

King's Corner is a card-base game. Each player starts with 7 cards and there are 8 lay down piles. The goal of the game is to lay all the cards down to win. The loser will receive penalty points, base on how many cards they have in their hand. First one to receive a total of 25 penalty points loses the game. Objective is to implement this game in Java, which will allow us to implement this game using object-oriented programming and take advantage of inheritance.

**High-Level Design:**

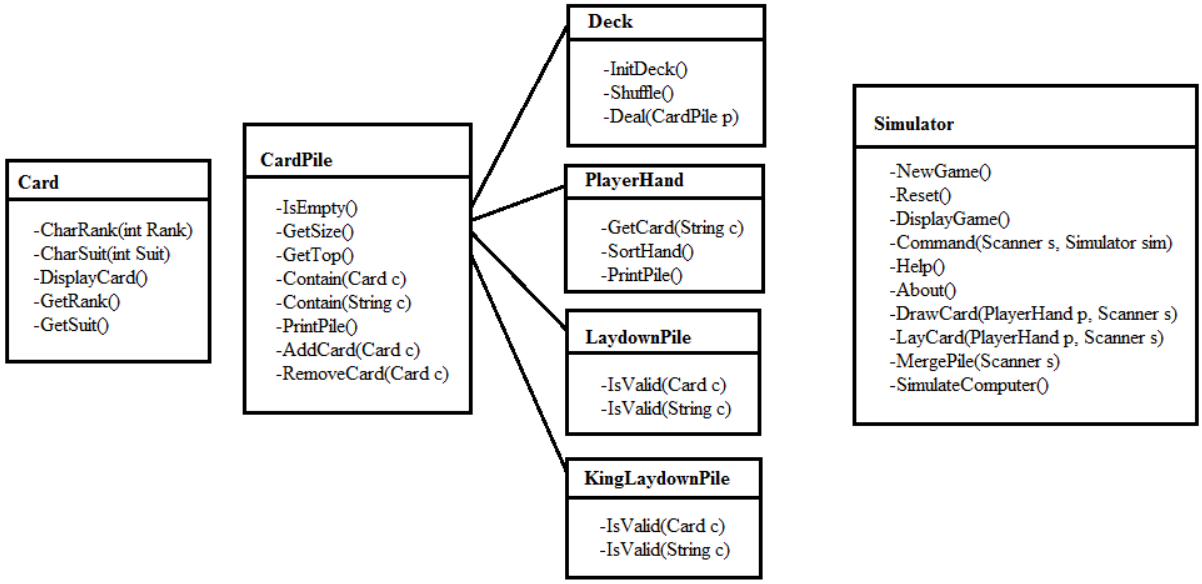



*Card:* This object will hold the content of a basic card. The reason why I set the rank and suit as integer is to determine which card has a higher priority. Also, we will store the character so we can easily display the card to the user. isBlack will determine if the suit is either black or red.

*CardPile:* This object holds an array of the object, Card. I will be implementing this as if it was a stack. The instance variable, size, will determine the size of the deck, and determine the top of the CardPile. This will act as a parent object for all the piles we need in this game, such as Deck, PlayerHand, LaydownPile, and KingLaydownPile.

*Simulator:* This object will simulate the game, by constructing the board and simulating the computer's move. The concept of the game requires a basic deck and a total of 8 laydown pile. We have a total of 2 players in this game, the computer and the user. Some bookkeeping is also required, such as penalty points for each player and the round of games.

**Low Level Design:**




*Set-Up:* Initialize the simulator will set up all the objects we need for this game. First we initialize the deck and shuffle it by calling InitDeck() and Shuffle(). Follow by calling NewGame(), which will use the Deal() function from the Deck object. Deal() also uses the function IsEmpty(), AddCard(), RemoveCard(), and GetTop(). Once NewGame() is called, each player should have 7 cards and pile 1-4 will have 1 card.

*GUI:* To display the game, we call DisplayGame(), which will take advantage of using PrintPile() and DisplayCard(). The cards are to display in descending order, although we have it set up to where the piles are already displayed in order. The only thing that needs to be modified is the player's hand. So we override PrintPile() for the object PlayerHand, and the only difference is before we display the pile we call SortHand().

*User-Interface:* The program will initialize a scanner that will take the user input. The scanner will get pass through Command(Scanner s, Simulator sim). Base on the user's input, it will perform one of the following command; Help(), About(), DrawCard(PlayerHand p, Scanner s), LayCard(PlayerHand p, Scanner s), or MergePile(Scanner s).

*Draw:* The draw command uses the DrawCard(PlayerHand p, Scanner s), which uses the Deal(CardPile p). Using the IsEmpty() to see of the deck is empty, will prompt a question to see if the user wants to quit the game. Otherwise, the game continues on. After the user activates draw, the computer will simulated their turn by calling SimulateComputer().

*Lay:* The lay command uses LayCard(PlayerHand p, Scanner s), which will use Contain(String c) to see if the player has the card. Then we want to check if the card IsValid(String c) to lay down in the selected pile. The computer uses Contain(Card c) and IsValid(Card c) since we are using GetCard(String c). If all test has pass, we RemoveCard(Card c) and AddCard(Card c) to transfer the card over.

*Merge:* Similar to Lay, we created a function called MergePile(Scanner s), which takes the user input and see if we are able to merge the two piles.

**Benefits/Assumptions/Risk & Issue:**

The reason why I choice to use array instead of linked list is because we know each pile has a capacity of 52 cards. So there is no need to make anything dynamic when we can statically set the capacity of each pile. The downfall of this choice would probably be memory allocation, but this method makes it easier for the programmer to code. Since we are programming in Java we do not have to worry about garbage collection since this is already implemented in Java.