CS 340 - Programming Project 1

Card Game: Kings in the Corner

Due: 11:59 pm on Thursday 1/31/2013

For this assignment, you are to implement the card game of Kings Corner. We will use the website as http://www.pagat.com/domino/kingscorners.html as the official rules for the game. No variations will be used for the program. The game will be played by the user and one computer opponent. This game is to be designed using classes and inheritance where-ever possible. The basic classes should include a card class, a card pile class, a game class and a player class.

The game is played using a standard deck of 52 cards. Each card has a Rank and Suit value. There are 13 Ranks and 4 Suits making the 52 cards in the deck. The Suits are colored either Black or Red. The cards are to be listed using a two character sequence for each card. The first character is to specify the rank of the card and the second character is to specify the suit. The ranks and suits with the associated characters are given in the table below. Use upper case letters when specifying cards.

Rank Characters	Suit Characters
• A - Ace • 2 - Two • 3 - Three • 4 - Four • 5 - Five • 6 - Six • 7 - Seven • 8 - Eight • 9 - Nine • T - Ten • J - Jack • Q - Queen • K - King	• C - Clubs (Black) • D - Diamonds (Red) • H - Hearts (Red) • S - Spades (Black)

Note that the ranks and suits are listed in order with Ace being the low rank, King being the high rank, Clubs being the low suit and Spades being the high suit. In the game of Kings Corner, the color of the suit is the important aspect of the game. The order of the Suit has no impact to the game.

The Game's Idea

The general purpose of the game is to lay down all cards in the players hand on one of eight lay-down piles. The lay-down piles must be ordered from high rank to low rank (without skipping ranks) and alternating suit colors. Thus if the bottom card on the pile is a Red Eight (this could be either the Eight of Diamonds "8D" or the Eight of Hearts "8H"), the next card in the pile would need to be a Black Seven (7C or 7S) and then a Red Six (6D or 6H), etc. Four of the lay-down piles can have any card at the bottom. Four of the lay-down piles must have Kings as the bottom card. The eight lay-down piles will be numbered from 1 to 8. Lay-down piles 1 through 4 can allow any card at the bottom of the pile. Lay-down piles 5 through 8 must have a king at the bottom of the pile.

How to Deal

When the program is started, the computer player will "deal" the first round. The deal is made up of the following steps:

1. Shuffle the 52 cards in the deck.
2. Deal out seven cards to each player. This is done by giving the top card in the deck to the player that is not dealing. Then give the new top card to the dealer. Cards are continued to be given out in this alternating fashion until both players have seven cards in their hands.
3. Put one card on each of the lay-down piles 1 through 4.
4. The remaining 34 cards are put in the draw pile.

I would make dealing a method (or methods) of the draw pile that way step 4 it automatically taken care of.

The play now alternates between the user player and the computer player. The player that did not deal goes first.

The Prompt for the User Player

The user player is given a prompt and enters a command to be read in by the program. The prompt should show all cards in all eight lay-down piles, all cards in the user player's hand and the number of cards in the computer player's hand. This prompt **might** appear as follows:

```
Pile 1: 8S 7H 6C 5D
Pile 2: AD
Pile 3: JD TC
Pile 4: 6H 5S 4D 3S
Pile 5: KH
Pile 6: KS QH JS TD 9S
Pile 7:
Pile 8:
Computer Player has 6 cards
Your hand: KC QS TH 9D 6D 2S
Move>
```

When displaying the user player's hand, show the cards in sorted order. The user player does not have to make any moves during his/her turn; however, since the goal of the game is to get rid of all the cards in his/her hand, most players attempt to make as many moves as possible during a turn. A player's turn is over when the player draws a card from the draw pile or has laid down all cards in his/her hand. When a player has laid down all of the cards, the current round is over and the score for that round is determined.

The User Player's Commands

The user player has the following commands to play the game. The commands can be given in either upper or lower case. Only one command per line is expected and each command is expected to be given on a single line of input (note: this information is to make it easier for you to deal with the user interface and not to cause additional error checking on your part).

- **Q** - Quit the program.

  The program is to stop executing after some appropriate message has been displayed. You may prompt the user with an "Are you sure you want to quit?" message, but this is not required.

- **H** - Help

  This command causes some help message to be displayed about the user commands in the game.

- **A** - About

  This command should display some about message to give information about the programmer and the program.

- **D** - Draw a Card from the Draw Pile

  This command will take the top card from the Draw Pile and add it to the user player's hand and end the user player's turn. You may display a message about what card was drawn (but this is not required).

- **L <Card> <Pile>** - Lay a Card on a Pile

  This command will have the user specify a card from his/her hand and a pile. The card is to be specified as a two character string with the rank as the first character and the suit as the second character (these can be in either upper or lower case). The pile is to be specified by the number 1 through 8. If the card does not exist in the player's hand or the pile number is out of range, give an appropriate error message. If the card specified can not be laid down on the specified pile, give an appropriate error message. If the specified card can be laid down on the specified pile, move the card from the player's hand to the pile. If the specified pile is pile 1 through 4 and the pile is empty, any card can be laid down on the pile. If the specified pile is pile 5 through 8 and the pile is empty, only a king can be laid down on the pile.

- **M <Pile1> <Pile2>** - Move One Pile on top of Another Pile.

  This command will have the user specify two piles. The first pile is to be moved on top of the second pile. Both piles are to be specified by a number 1 through 8. If either number is out of range give an appropriate error message. If the first pile cannot be laid on top of the second pile, give an appropriate error message. Otherwise move the file pile on top of the second pile. Note that only piles 1 through 4 can be moved on top of another pile. This move would empty the first pile specified.

The Computer Player's AI Algorithm

For the computer player's turn, use the following AI algorithm.

1. Lay down all kings in the computer player's hand onto an empty pile 5 through 8. If a king exists in a player's hand, at least one of the piles 5 through 8 will be empty. **Also check if there is a King at the bottom of piles 1 through 4. If so, move this pile to an empty pile 5 though 8.**
2. If a pile can be moved onto of another pile, do so. If successful, repeat.
3. Lay down a card on top of any non-empty pile. If successful, goto step 2.
4. Lay down a card on top of any empty pile 1 through 4. If successful, goto step 2.
5. If the computer player has not laid down all cards in its hand, draw a card from the draw pile and end the computer player's turn.

Remember that when moving piles and laying down cards in steps 2 and 3 that cards in the pile must be in decreasing order and of alternating color. As each move is made by the computer player, an appropriate message should be displayed. You may use any method you wish when picking a card to lay down or a pile to move if multiple cards could be laid down or multiple piles could be moved. The simplest method is to randomly choose between the possible choices. This assignment does not require an intelligent choice to be made; however, you may add this if you wish as long as the above sequence is followed. If you note in the example above, the user could lay down all of his/her cards if they are played correctly. If the cards are not played correctly the user will not lay down all of his/her cards. If such a situation was presented to the computer player in your program, the AI in your program is not required to figure out proper sequence of moves to lay down all cards.

When the computer player makes a move, it would be nice to have a message displayed that explains the moves. This would inform the user player of everything the computer did thus creating a better user interface in your program (and make it easier for the TA to verify your program is running correctly).

Completing a Round

Once one player has laid down all cards in the player's hand, the current round is over immediately. The other player receives penalty points for the cards left in that player's hand. A king costs 10 points and the other cards cost one point. A game is played until one player gets 25 penalty points. This means that multiple rounds may need to be played to finish the game. The dealer for the next round is the one who did not deal in the last round and the player than moves first is the one who dealt in the last round.

Completing a Game

When a player has laid down all cards and the round is over, a message is displayed stating the number of penalty points earned in that round and the total number of penalty points for each player in the game. If the game is not over (neither player has 25 penalty points), the deal for the next round is then immediately started and then the turns for that round commence. When a game is over (once one player has earned 25 penalty points), a message is displayed stating the winner of the game and the user player is asked if he/she would like to play another game. If the player answers "no", end the program with an appropriate message. If the player answers "yes", reset the penalty points for both player's to zero and begin a new game. The dealer for the first round in the new game should be the player that did not deal in the last round of the last game.

C++ Class Requirement

Your program must have at least two classes. One class will be a card class. The other class will be a "card pile" class.

The class of card pile is to hold any number of cards. The laydown piles are card piles in which the cards must be added in the proper order (as specified above). You will also need to have a collection of cards for each player's hand (the user player and the computer player). The cards that have not been played must be held in a draw pile. Your program must name the card pile class called "CardPile".

Ideally, the CardPile class could to be the base class for the classes of LaydownPile, PlayerHand and DrawPile. This means the classes for the laydown piles, player's hands and draw pile should all inherit information from the card pile class. **However, the use of inheritance is NOT required in this program.**

Another place for inheritance in your program is the player class. The base player class will allow the interaction of the player's hand with the draw pile and the laydown piles. The player class will need to be expanded for the user player and the computer player. This expansion will determine how the interaction is to occur. In the case of the user player, the class will need to prompt the user for which move to make. In the case of the computer player, the class will need to use the AI feature of the game to determine which move to make.

Another suggested class would be class to keep track of all the game information such as:

- verifying whether two cards are in valid sequence (in decreasing order and of alternate color),
- the number of cards to deal to each player,
- keeping track of who deals next,
- keeping track of whose turn it is,
- the number of points to play to,
- etc.

General Comments

Your program must be written in good programming style. This includes (but is not limited to) meaningful identifier names, a file header at the beginning of each source code file, a function header at the beginning of the function, proper use of blank lines and indentation to aide in the reading of your code, explanatory "value-added" in-line comments, etc.

The work you turn in must be 100% done by yourself and written by yourself. You are not allowed to used any pre-defined classes in your code. You are not allowed to share code with any other students (inside this class or not) You may discuss the project with other students; however, you may not show any code you write to another student nor may you look at any other student's written code. Also, you are not to find code on the internet and turn it in as your own.

You are to submit your project using the Assignment link on the Assignments page in Blackboard.