Sivananda Rajananda
204322840
PIC 20A
Prof. Haberland

Homework 3

WarCard:

Instance Variables:
Declare an int to hold the value the card
Declare a string to hold the suit
Declare another string to hold the rank

Constructor:
Declare a constructor that takes in an int as the value (from 0 to 51)
-Check to make sure that the value is between 0 to 51
    -Set the value to the value from the parameter
    -set the rank by using integer division and adding 2
    -set the suit by %4
-If not in the range, then print out an error statement

Methods
Declare a toString() method
-Make a string that has the output that we want to read the card it (i.e. rank + “of” + suit)
-Return the string

Declare a compareTo() method to compare cards
-Takes a card as a parameter
-Takes both cards and does integer division by 4
-Compare them to see which is larger
-If the same, return 0
-If this is larger, return 1
-If this is smaller, return -1

Declare a setRank method()
-Private
-Takes an int as a parameter
-Checks to see if the number is between 2 and 14 inclusive
    -Sets the rank based on the number (e.g. parameter 2 gets “Two” set as rank)
-If not inside range, then print error message

Declare a setSuit() method
-Private
-Takes an int as a parameter
-Check to see that parameter is between 0 and 3 inclusive

-Set the suit based on the number (e.g. 0 gets “Diamond” set as suit)
-If outside range, print out error message

WarDeck

Import ArrayList
Import Collections

Instance Variables:
Declare ArrayList<WarCard> cardStack to hold the cards

Constructor:
Declare a WarDeck() constructor
-Declare a new ArrayList as the cardStack
-For loop from 0 to 51{
-Create a card
-Add the card to the cardStack
}

Methods
Declare Void print() method {
-Enhanced for loop{
-Print WarCard using print function from WarCard
}
}

Declare Void shuffle() method{
-calls the shuffle for ArrayLists on the cardStack
}

Void deal(p1, p2){
-Takes 2 players as the arguments
-While cardStack is not empty{
-Player1 accept top card;
-Remove top card from deck;
-Player 2 accept top card;
-Remove top card from deck
}
}

War Player

Instance Variables:
Declare String name to hold the name
Declare ArrayList<WarCard> hand to hold cards in the hand
Declare ArrayList<WarCard> pile to hold cards in the pile

Constructor
WarPlayer(String name){
-Sets the name to the name from the parameter
-Creates new ArrayList for hand
-Creates new ArrayList for pile
}

Methods
Declare Void acceptCards(WarCard incomingCard) to accept the card from deck{
- add incomingCard to the hand
}

Declare Void acceptCards(ArrayList<WarCard> incomingArrayList) to accept card from pile{
-enhanced for loop goes through the ArrayList
    -adds the card to the hand
}

Declare Void playCard () to move the card from the hand to the pile{
-Copy card top from hand to pile
-Remove top card from hand
}

Declare Void printCards() to print the cards in both hand and pile{
-If there are cards in the pile
    -Print out the name + "'s Pile"
    -If the number of cards in the pile is less than 4 inclusive
        -Print all cards separated by comma
    -Else if the number of cards in the pile is more than 4
        -print out the first 2 cards
        -print out the number of cards -4 + "more"
        -print out the last 2 cards
-If there are cards in both the pile and the deck
    -Print out the "|" separator
-If there are cards in the hand
    -Print out the name + "'s Hand"
    -If the number of cards in the hand is less than 4 inclusive
        -Print all cards separated by comma
    -Else if the number of cards in the hand is more than 4
        -print out the first 2 cards
        -print out the number of cards -4 + "more"
        -print out the last 2 cards

Declare getTopPileCard()
-Returns the card at the top of the pile (the lastest card)

Declare getPile()
-Returns the pile

Declare clearPile()
-clears the pile so that it is empty

Declare gethand()
-Returns the hand

Declare getName()
-Returns the name

War Game

```
main{
        Create the players
        Create the deck and shuffle
        Deal the deck to the players

        Print out the initial hand for both players

        Set the isWar flag to false

        Declare and initialize the round counter to 1

        Declare a player called “winner” to assign the winner of each round to

        While (both have cards still){
                Print out the round number
                Both play card

                If isWar, then play another card

                Compare the latest cards
                If (war){
                        Print “War!”
                        Set isWar to true
                        continue;
                }
                Else if (p1 is better){
                        Winner = p1;
                }
```

```
            Else if (p2 is better){
                Winner = p2;
            }

            Increment the counter

            Set a random value to determine the order of adding the pile to the hand
        randomly

            If even{
                Winner takes p1's pile first, then p2's pile
            }
            If odd{
                Winner takes p2's pile first, then p1's pile
            }

            Remove cards from the pile
            Print Win statement
            Print took pile statement
        }//end of while loop

        Print "Game Over!"
        Print name of winner + "wins!"

}//end of main
```

Finally, consider what you would need to do to write superclasses of WarCard, WarDeck, and WarPlayer that could be used in a more general card game.
What might the superclasses be called?
What fields, methods, and constructors would be appropriate to move from the subclasses you wrote into the superclasses (where they could be used in more general card games)?

Since not all card sets and games are the same, the only characteristic they have in common is to have a value for identification.
Superclass: Card
Constructors: Card(int value)
Fields: int value
Methods: print()

Decks need to hold a dynamic array of cards, and be able to be shuffled
Superclass: Deck
Constructors: Deck()
Fields: ArrayList<Card> deck
Methods: print(), shuffle()

Players need a name for identification, and to print out their cards
Superclass: Player
Constructors: Player(String name)
Fields: String name, ArrayList<Card> hand
Methods: print()