WarPrep

WarCard

Declare a private string variable to store the suit. Call it *suit*
Declare a private string variable to store the rank. Call it *rank*
Declare a private int variable to store the value. Call it *value*

Declare a WarCard() constructor that accepts an integer and sets the rank suit and value accordingly.

Declare a toString() method that returns a string with the card suit and rank

Declare a compareTo() that takes another WarCard as an argument and return -1 if its this.value is smaller, 0 if equal and 1 if larger.

Constructor WarCard(int val)

value is equal to val%13

IF val is between 0-12

suit is Hearts

End IF

ELSEIF val is between 13-25

suit is Clubs

End ELSE IF

ELSEIF val is between 26–38

suit is Spades

End ELSE IF

ELSE

suit is Diamonds

End ELSE

IF value is 0

rank s Ace

End IF

```
        ELSE IF value is 1

                rank is One
        End ELSE IF
…..



        ELSE IF value is 12

                rank is King

        End ELSE IF


End Method



Method toString() is void

        Output the rank and value

End Method


Method compareTo(WarCard othercard) return int

        IF this.value smaller than othercard.value

                return -1
        End IF

        ELSEIF this.value equal to othercard.value

                return 0

        End ELSEIF

        ELSE

                return 1

        End ELSE

End Method
```

WarDeck

import java.util.ArrayList
import java.util.Collections

Declare an ArrayList of type WarCard named cardStack

Constructor WarDeck()

    For loop with i from 0 to 51

        create a card with each i

        insert it to the end of the array list

    End For loop

End constructor

Method print() returns void

    declare a stack_size variable with the size of the stack

    for Loop through array list size

        print each card with toString from WarCard

    End for Loop

End Method

Method shuffle() returns void

    shuffle the card stack with Collections.shuffle()

End Method

Method deal(WarPlayer[] players, int numberOfCards) returns void

    Declare an int counter set it to 0

    declare an int size set it to the size of the array players

    While number of card is larger than 0

give the index zero of deck card of the deck to player with the index counter modulo size

remove index 0 card from deck

counter plus one
numberOfCards minus one

End While loop

End Method

WarPlayer

Declare a String called name
Declare a ArrayList called hand
Declare a ArrayList for their pile called pile

Constructor WarPlayer (string n)

set the name to be n

End constructor

Method add(WarCard card) returns void

add card to the end of the hand

end method

Method play returns void

add card from index 0 of hand to end of pile

end method

Method take(WarPlayer rival) returns void

For loop through rivals pile

add card from rivals pile to end of this.hand

end For loop

For loop through this.pile

add card from this.pile to end of this.hand

end For loop

empty rivals pile

empty pile

end method

Method print(WarPlayer p) return void

If pile size is less than or equal to four

print pile in the correct format with out the parentheses int he middle

else print the pile with the parenthesis

If hand is less than or equal to four

print hand in the correct format with out the parentheses in the middle

else print the hand with the parenthesis

end method

WarGame

Declare a WarDeck name it deck
shuffle the deck
Declare two players p_1 and p_2

Declare an array of players

put p_1 and p_2 in the array

Declare and int to serve as a counter of round initialized to 0. Name it counter

While both players still have cards in their hand or one player has cards in his pile

if counter is zero

print Initial Hands

```
end if statement

else if one player has an empty hand and the piles are not empty

        print Round + counter

        if p_1 has empty hand

                p_2 plays
                print p_1 and p_2

                compare both last card of the pile

                if p_1 wins

                        p_1 takes the piles

                end if

                else if p_2 wins

                        p_2 takes the piles

                end else if

                else //if its a tie

                        write war
                        p_2 plays
                        counter minus one

                end else
        end if

        else

                p_1 plays
                print p_1 and p_2

                compare both last card of the pile

                if p_1 wins

                        p_1 takes the piles

                end if

                else if p_2 wins
```

```
                    p_2 takes the piles

                end else if

                else //if its a tie

                    write war
                    p_1 plays
                    counter minus one

                end else

        end else

else

        p_1 and p_2 play
        print p_1 and p_2

        compare both last card of the pile

        if p_1 wins

                p_1 takes the piles

        end if

        else if p_2 wins

        p_2 takes the piles

        end else if

        else //if its a tie

                write war
                if p_1 hand is not empty
                        play p_1
                end if

                if p_2 hand is not empty
                        play p_2
                end if

                counter minus one

        end else
end else
```

counter plus one

End While

print Game Over!

Questions:

We could call them

Card

Constructor:
- Accept one arguments int for the value

Variables:
- private int value;
//no suit has we might want to play a game like uno

Methods:
- compare // compares value

Deck

Constructor:
- accepts as an argument a int saying how many cards we want in the deck

Variables:
- ArrayList of cards

Methods:
- print
- deal //accepts a array of players and an int to say how many cards each player
- shuffle

Player

Constructor:

- accepts argument: one string for name

variables:

- name
- points //many games count the points of each player
- hand

method

- card_on_table // puts a card down
- is_hand_empty
- print_hand
- take card from deck // many games have a option of taking a card from the deck if you can't play.