# **Program: WarGame**

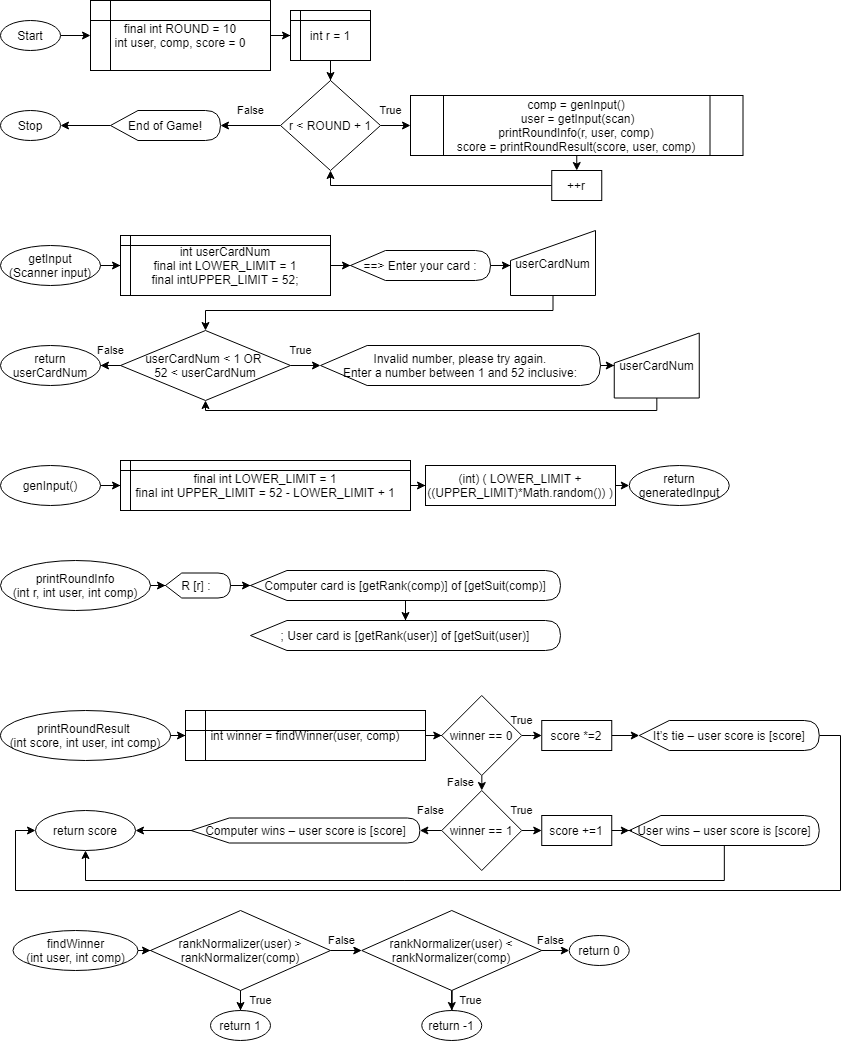
**Purpose:** Plays a card game called WarGame with the computer in these trying times.

**Packages needed:**

* java.util.Scanner

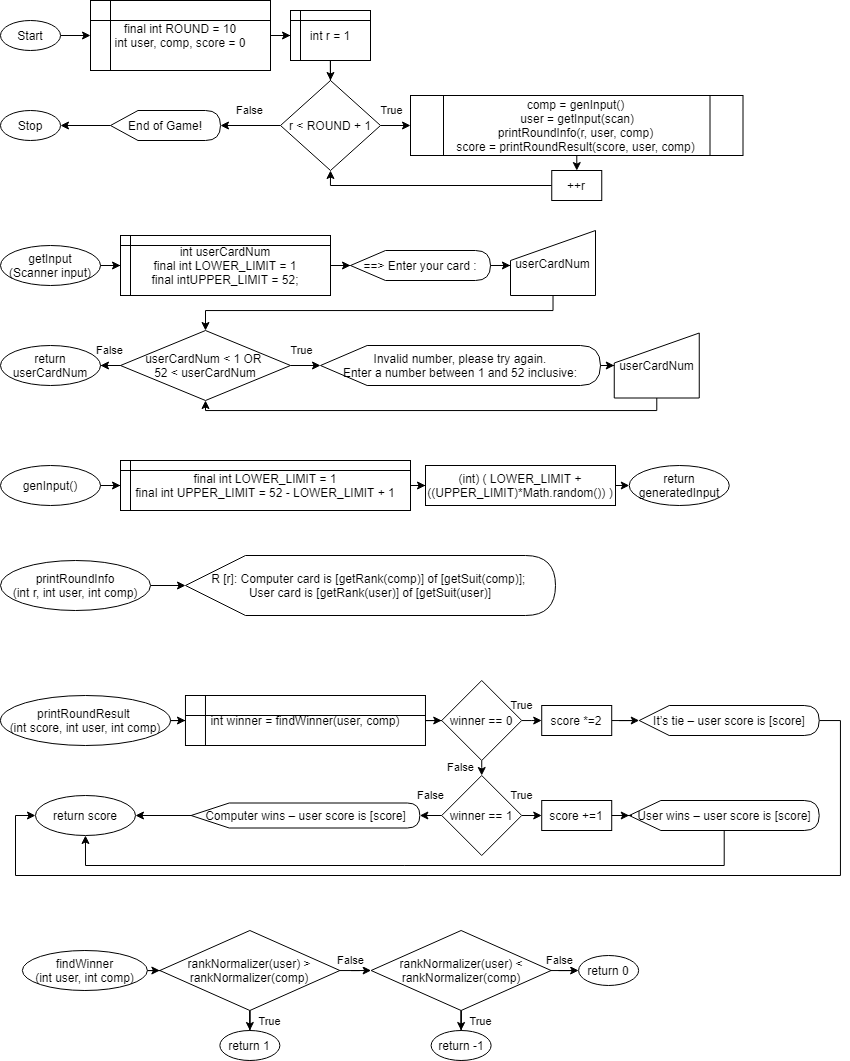
**Input:** Number of   
**Output:** Tells you if you win, lose or draw. Also outputs the current score and round number. Also shows the type of card you and the computer had.  
**Limitations:** Can only accepts integers, will give an error/exception otherwise

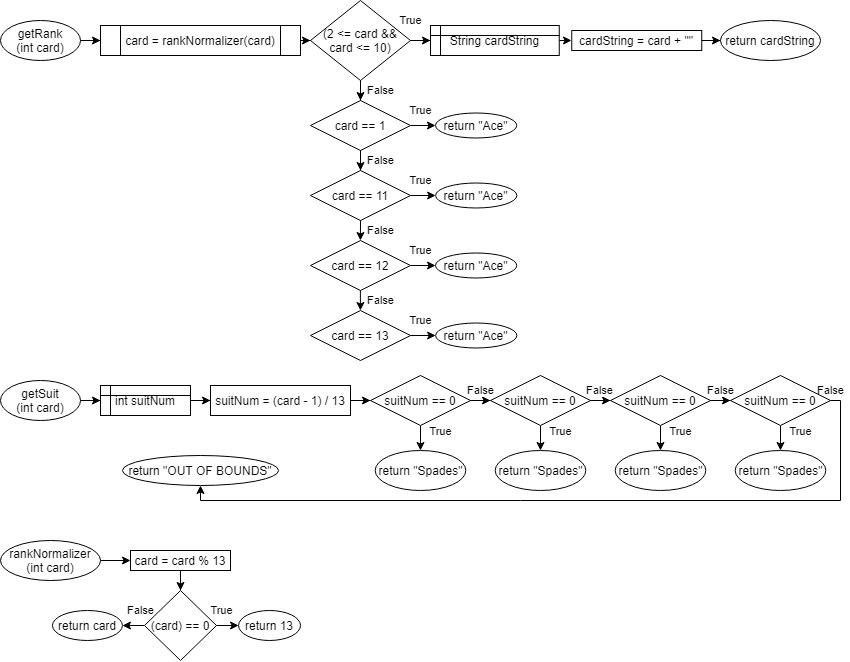
**Sample output :**

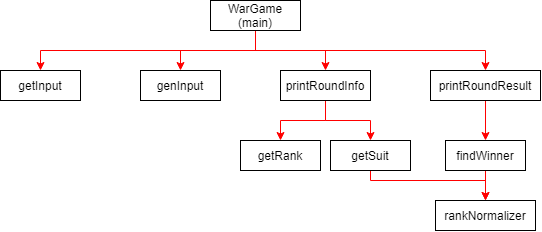
**Flowchart:**

[***continued***]

**Flowcharts:**

*[continued]*******

****

**Design Diagram:**

**Question 2 :**

public static int partition(int[] list){

    int pivot = list[0];

    int countGreater = 0;

    int lastIndex = list.length - 1;

    int temp;

*//Counts numbers greater than pivot*

*for* (int i = 1; i < list.length; ++i){

*if* (pivot < list[i]){

            countGreater++;

        }

    }

*//Place pivotIndex in place that is less than future integers greater than pivot*

    int pivotIndex = lastIndex - countGreater;

*//Swaps first element with pivotIndex*

    temp = list[pivotIndex];

    list[pivotIndex] = pivot;

    list[0] = temp;

*//Checks if left or right side gives the lowest iteration for the worst case*

    int countLesser = list.length - countGreater - 1;

    int lowestIterationSide;

*if*(countLesser < countGreater)

        lowestIterationSide = countLesser;

*else*

        lowestIterationSide = countGreater;

*//Initializes booleans if left, right and both are the last elements*

    boolean leftRightLastElement = false;

    boolean leftLastElement = false;

    boolean rightLastElement = false;

*//Swaps numbers greater than pivot on the left with numbers less than pivot on the right*

*for*(int i = 0, left = 0, right = pivotIndex + 1; i < lowestIterationSide; ++i){

*//Checks if it is last element on both sides, and if so everything is checked*

*while* (!leftRightLastElement){

*//Goes through each left side to find first left number that is greater than pivot*

*if* (left < pivotIndex){

*while*(list[left] <= list[pivotIndex]){

                    leftLastElement = (left >= (pivotIndex - 1));

*if*(!leftLastElement)

                        left++;

*else*

*break*;

                }

            }

*//Goes through each right side to find first right number that is less than pivot*

*if* (right < list.length){

*while* (list[right] > list[pivotIndex]){

                    rightLastElement = (right >= list.length - 1);

*if* (!rightLastElement)

                        right++;

*else*

*break*;

                }

            }

*//Swaps left and right numbers if left is bigger than right*

*if*((left < pivotIndex) && (right < list.length)){

*if* (list[left] > list[right]){

                    temp = list[right];

                    list[right] = list[left];

                    list[left] = temp;

                }

            }

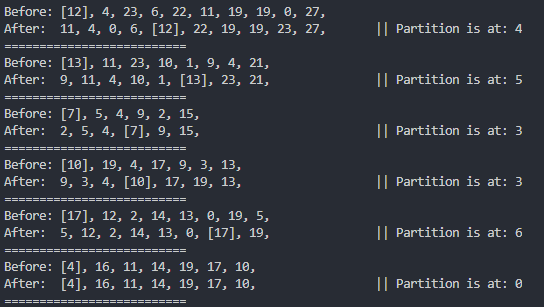
            leftRightLastElement = (leftLastElement && rightLastElement);

        }

    }

*return* pivotIndex;

}



**Question 3 :**

public static String reverseDislay(String value){

    int lastIndex = value.length() - 1;

    String newValue = "";

    String reverse = "";

*//Returns value if it's the only character*

*if* (value.length() == 1){

*return* value;

    }

*else*{

*//Creates new string without the last character*

*for*(int i = 0; i < lastIndex; ++i){

            newValue += value.charAt(i);

        }

*//Adds the last character and calls method for string without last character*

        reverse += (value.charAt(lastIndex) + reverseDislay(newValue));

*return* reverse;

    }

}

**Sample:**

