

Nombre: Andrés Alejandro Cárdenas de la Cruz		Matricula: 2896424
Nombre del curso: Computación En Java	Nombre del profesor: Manuel Cruz Serrano	
Tema 9. Entrada y Salida de datos 1	Actividad 9. Entrada estándar	
Fecha: 19/10/2020		

Solución Implementada

```
package poker;

import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Gui extends JFrame implements ActionListener {
    //Elementos graficos de la Aplicacion
    private JButton shuffle, head, hand, pick, reset;
    private TextArea result;
    //instancia a la clase Deck
    Deck deck = new Deck();
    public static void main(String[] args) {
        //Se establece los valores predeterminados de la ventana
        Gui Frame = new Gui();
        Frame.setSize(500, 350);
        Frame.setTitle("Actividad 8 Baraja");
        Frame.interfaz();
        Frame.setVisible(true);
    }
    public void interfaz() {
        //Se indica cuando se cierra la aplicacion
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        //establece un panel a la ventana
        Container window = getContentPane();
        //establece el acomodo o layout de la ventana
        window.setLayout(new FlowLayout());
        //Establece los valores para los elementos graficos
        JLabel titulo = new JLabel("Que Accion Desea Realizar ?");
        JLabel formato = new JLabel("");
        result = new TextArea();
        shuffle = new JButton("Shuffle");
        head = new JButton("Head");
        pick = new JButton("Pick");
        hand = new JButton("Hand");
        reset = new JButton("Reset");
        //Establece la accion al presionar los botones
        shuffle.addActionListener(this);
        head.addActionListener(this);
        pick.addActionListener(this);
        hand.addActionListener(this);
        reset.addActionListener(this);
        //agrega los elementos graficos al panel
        window.add(titulo);
        window.add(formato);
        window.add(shuffle);
        window.add(head);
        window.add(pick);
        window.add(hand);
        window.add(reset);
        window.add(result);
        //Establece un tamaño a los elementos graficos indicados
        formato.setPreferredSize(new Dimension(400, 0));
        shuffle.setPreferredSize(new Dimension(200, 25));
    }
}
```

```

        head.setPreferredSize(new
Dimension(200,25));
        pick.setPreferredSize(new Dimension(200,25));
        hand.setPreferredSize(new Dimension(200,25));
        reset.setPreferredSize(new Dimension(400,25));
    }
    //Metodo que establece la accion de los botones
    @Override
    public void actionPerformed(ActionEvent event) {
        //accion al presionar el boton shuffle
        if (event.getSource().equals(shuffle))
            result.setText(deck.shuffle());
        //accion al presionar el boton head
        else if (event.getSource().equals(head))
            result.setText(deck.head());
        //accion al presionar el boton pick
        else if (event.getSource().equals(pick))
            result.setText(deck.pick());
        //accion al presionar el boton hand
        else if (event.getSource().equals(hand))
            result.setText(deck.hand());
        //accion al presionar el boton reset
        else {
            //restablece la baraja a su acomodo inicial
            Deck restablecer = new Deck();
            deck.baraja= restablecer.aux;
            result.setText("Quedan "+(deck.baraja.size())+" Cartas En Deck");
        }
    }
}

```

```

package poker;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Random;

public class Deck {
    //atributos de tipo lista para agregar las cartas
    ArrayList<Object> baraja = new ArrayList<>();
    ArrayList<Object> aux = new ArrayList<>();

    public Deck () {
        //declaracion de variables
        String p,c,v;
        //establece aleatoriamente el color de la baraja negro o rojo
        int d =(new Random().nextInt(2));
        if (d==1) c="Rojo";
        else c = "Negro";
        //ciclo for para agregar las 52 cartas
        for (int i=1;i <= 52;i++){
            //condicionales para cambiar los valores determinados por sus
respectivas letras en la baraja
            if (i==1 || i==14||i==27||i==40)
                v="A";
            else if (i==11 || i==24||i==37||i==50)
                v="J";
            else if (i==12 || i==25||i==38||i==51)
                v="Q";
            else if (i==13 || i==26||i==39||i==52)
                v="K";
        }
    }
}

```

```
//condicional que reasigna el valor
para tener los valores en un rango del 2 al 10
    else {
        if (i<15)
            v = Integer.toString(i);
        else if (i<28)
            v = Integer.toString((i-13));
        else if (i<41)
            v = Integer.toString((i-26));
        else
            v = Integer.toString((i-39));
    }
    //condicionales para establecer el palo de la carta
    if (i<=13)
        p="Tréboles";
    else if (i <= 26)
        p="Corazones";
    else if (i <= 39)
        p = "Picas";
    else
        p = "Diamantes";
    //instancia a la clase Cards con los valores del constructor
    Cards carta = new Cards(p,c,v);
    //agrega la carta a los atributos de tipo ArrayList
    baraja.add("Palo: "+carta.Palo+", Color: "+carta.Color+", Valor:
"+carta.Valor);
    aux.add("Palo: "+carta.Palo+", Color: "+carta.Color+", Valor:
"+carta.Valor);
}

//metodo para barajar el deck con las cartas actuales
public String shuffle(){
    if (baraja.size()==0)
        return "Deck Vacio";
    else {
        Collections.shuffle(baraja);
        return "Se mezclo el deck";
    }
}

//Metodo para mostrar la carta del final y quitarla de la lista
public String head(){
    if (baraja.size()==0)
        return "Quedan "+ 0 +" Cartas En Deck";
    else {

        String cadena = (String) baraja.get(baraja.size() - 1);
        baraja.remove(baraja.get(baraja.size() - 1));
        return cadena + "\nQuedan " + baraja.size() + " Cartas En Deck";
    }
}

//Metodo para mostrar una carta aleatoria y quitarla de la lista
public String pick(){
    if (baraja.size()==0)
        return "Quedan "+ 0 +" Cartas En Deck";
    else {
        int random = (new Random().nextInt(baraja.size()));
        String cadena = (String) baraja.get(random);
        baraja.remove(random);
        return cadena + "\nQuedan " + baraja.size() + " Cartas En Deck";
    }
}
```

```

    }
    //Metodo para mostrar las 5 cartas del final y quitarlas de la lista
    public String hand() {
        StringBuilder cadena = new StringBuilder();
        int ciclo = Math.min(baraja.size(), 5);
        for (int cont = 0; cont < ciclo; cont++) {
            cadena.append(baraja.get(baraja.size() - 1)).append("\n");
            baraja.remove(baraja.size() - 1);
        }
        return cadena + "Quedan " + baraja.size() + " Cartas En Deck";
    }
}

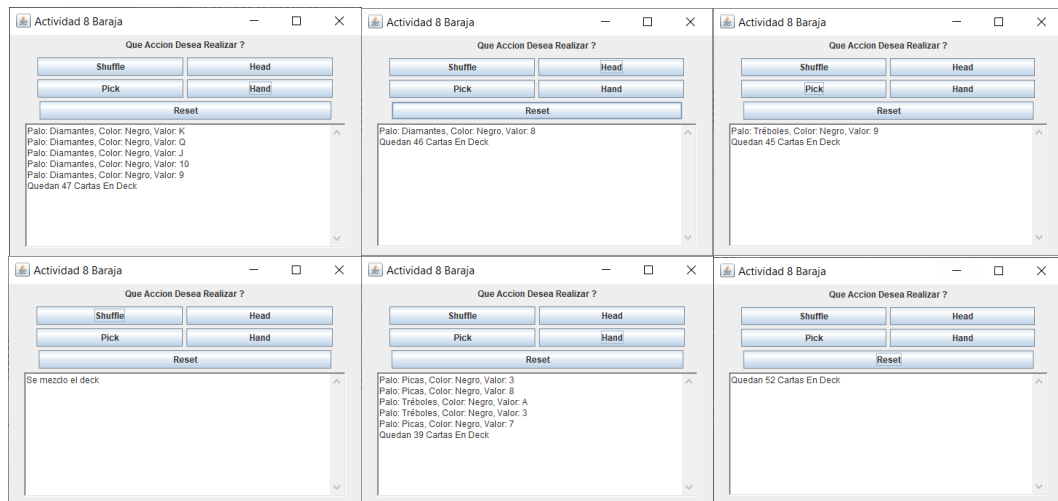
```

```

package poker;

public class Cards {
    //atributos de las cartas
    String Palo, Color, Valor;
    //constructor para establecer valor por determinado
    public Cards(String Palo, String Color, String Valor) {
        this.Color = Color;
        this.Valor = Valor;
        this.Palo = Palo;
    }
}

```



La baraja se maneja principalmente de una lista con las librería `import java.util.ArrayList;`
`import java.util.Collections;` lo que permite un mejor manejo en cuestión de los métodos principales ya que permite sacar los elementos que vayas mostrando y barajarlos a través de sus métodos

<https://github.com/alex-crdnz/actividades.git>