

**Testeo y Prueba de Software**

**Trabajo práctico: UNO Game**

**Carrera**: Técnico en Programación de Computadoras

**Docentes**: Didier Renard

**Turno**: Noche

**Materia**: Testing

**Fecha entrega**: 14/06/19

Integrantes:

|  |  |
| --- | --- |
| **Matrícula** | **APELLIDO, Nombres** |
| 56003193 | Franco, Federico |
| 56003194 | De la cruz, Andres |
| 56003163 | Lehmacher, Alex |

Grilla de calificación

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Indicador** | 1 | 2 | 3 | 4 | 5 |
| **Muy Bien** |  |  |  |  |  |
| **Bien** |  |  |  |  |  |
| **A corregir** |  |  |  |  |  |
| **NOTA** |  | | | | |

package GUI;

import Juego.NuevaPartida;

import Juego.Partida;

public class MenuPrincipal extends javax.swing.JFrame {

public MenuPrincipal() {

initComponents();

setLocationRelativeTo(null);

}

public MenuPrincipal(int i) {

initComponents();

bloquearReanudar();

setLocationRelativeTo(null);

}

public void bloquearReanudar(){

btnReanudar.setEnabled(false);

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

btnNueva = new javax.swing.JButton();

btnReanudar = new javax.swing.JButton();

btnSalir = new javax.swing.JButton();

jLabel2 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setTitle("UNO");

setMaximumSize(new java.awt.Dimension(828, 390));

setMinimumSize(new java.awt.Dimension(828, 390));

setPreferredSize(new java.awt.Dimension(828, 390));

getContentPane().setLayout(null);

jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/unoLogo.png"))); // NOI18N

getContentPane().add(jLabel1);

jLabel1.setBounds(0, 0, 415, 378);

btnNueva.setBackground(new java.awt.Color(0, 0, 0));

btnNueva.setFont(new java.awt.Font("Dialog", 1, 18)); // NOI18N

btnNueva.setForeground(new java.awt.Color(255, 255, 255));

btnNueva.setText("Nueva Partida");

btnNueva.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnNuevaActionPerformed(evt);

}

});

getContentPane().add(btnNueva);

btnNueva.setBounds(521, 61, 234, 64);

btnReanudar.setBackground(new java.awt.Color(0, 0, 0));

btnReanudar.setFont(new java.awt.Font("Dialog", 1, 18)); // NOI18N

btnReanudar.setForeground(new java.awt.Color(255, 255, 255));

btnReanudar.setText("Reanudar Partida");

btnReanudar.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnReanudarActionPerformed(evt);

}

});

getContentPane().add(btnReanudar);

btnReanudar.setBounds(521, 143, 234, 64);

btnSalir.setBackground(new java.awt.Color(0, 0, 0));

btnSalir.setFont(new java.awt.Font("Dialog", 1, 18)); // NOI18N

btnSalir.setForeground(new java.awt.Color(255, 255, 255));

btnSalir.setText("Salir");

btnSalir.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnSalirActionPerformed(evt);

}

});

getContentPane().add(btnSalir);

btnSalir.setBounds(521, 225, 234, 64);

jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/Webp.net-resizeimage.jpg"))); // NOI18N

getContentPane().add(jLabel2);

jLabel2.setBounds(0, -5, 830, 400);

pack();

}// </editor-fold>

private void btnNuevaActionPerformed(java.awt.event.ActionEvent evt) {

this.setVisible(false);

NuevaPartida partida2 = new NuevaPartida();

}

private void btnReanudarActionPerformed(java.awt.event.ActionEvent evt) {

this.setVisible(false);

Partida partida = new Partida();

}

private void btnSalirActionPerformed(java.awt.event.ActionEvent evt) {

System.exit(0);

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MenuPrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MenuPrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MenuPrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MenuPrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new MenuPrincipal().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton btnNueva;

private javax.swing.JButton btnReanudar;

private javax.swing.JButton btnSalir;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

// End of variables declaration

}

package GUI;

import Juego.JuegoNormal;

import Models.Carta;

import Models.Jugador;

import java.util.ArrayList;

import java.util.Arrays;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPasswordField;

public class VistaJugador extends javax.swing.JFrame {

/\*\*

\* Creates new form VistaJugador

\*/

public VistaJugador() {

initComponents();

juntarCartas();

cambiarImagen();

setLocationRelativeTo(null);

}

public VistaJugador(Carta c,int NumeroJugador) {

initComponents();

juntarCartas();

cambiarImagen();

asignarPozo(c);

asignarNombre();

checkearEntradaMenu();

vecesEnMenu=0;

setLocationRelativeTo(null);

}

public ArrayList<JLabel>listaCartas=new ArrayList<>();

public ArrayList<JLabel>cartasLibres=new ArrayList<>();

public static int vecesEnMenu=0;

public void incrementFocus() {

int lastPlayer = JuegoNormal.jugadorFocus;

if (JuegoNormal.rondaHoraria) {

if (lastPlayer + 1 == JuegoNormal.listaJugadores.size()) {

JuegoNormal.jugadorFocus = 0;

} else {

JuegoNormal.jugadorFocus = lastPlayer + 1;

}

} else {

if (lastPlayer - 1 < 0) {

JuegoNormal.jugadorFocus = JuegoNormal.listaJugadores.size() - 1;

} else {

JuegoNormal.jugadorFocus = lastPlayer - 1;

}

}

}

public void siguienteTurno(){

incrementFocus();

cambiarImagen();

asignarNombre();

}

public void siguienteTurno(String s){

cambiarImagen();

asignarNombre();

}

public void asignarNombre(){

lblNombre.setText(JuegoNormal.listaJugadores.get(JuegoNormal.jugadorFocus).getNombre());

}

public void asignarPozo(Carta c){

String carta=c.getValor()+c.getColor();

JuegoNormal.pozo.setColor(c.getColor());

JuegoNormal.pozo.setTipo(c.getTipo());

JuegoNormal.pozo.setValor(c.getValor());

cartaPozo.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/"+carta+".png")));

cartaPozo.setName(JuegoNormal.pozo.getValor()+" "+JuegoNormal.pozo.getColor()+" "+JuegoNormal.pozo.getTipo());

}

public void cambiarImagen2(){

String carta="+2azul";

for (JLabel label : listaCartas) {

label.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/"+carta+".png")));

}

}

public void verificarEspecial(Carta c) {

if (c.getTipo().equals("especial")) {

aplicarCartaEspecial(c);

}

}

public void aplicarCartaEspecial(Carta c) {

checkMazoVacio();

checkCantidadCartas(c);

String tipoCarta = c.getTipo();

String valorCarta = c.getValor();

if (tipoCarta.equals("numero")) return;

switch (valorCarta) {

case ("+2"):

int numeroRandom = (int) (Math.random() \* JuegoNormal.listaMazos.get(0).getMazoPrincipal().size()-1);

Carta ca1 = JuegoNormal.listaMazos.get(0).getMazoPrincipal().get(numeroRandom);

int numeroRandom2 = (int) (Math.random() \* JuegoNormal.listaMazos.get(0).getMazoPrincipal().size() - 2);

Carta ca2 = JuegoNormal.listaMazos.get(0).getMazoPrincipal().get(numeroRandom2);

JuegoNormal.listaJugadores.get(numeroSiguiente()).agregarCartas(Arrays.asList(ca1, ca2));

JuegoNormal.listaMazos.get(0).removerCartas(Arrays.asList(ca1, ca2));

incrementFocus();

break;

case ("+4"):

int n1 = (int) (Math.random() \* JuegoNormal.listaMazos.get(0).getMazoPrincipal().size()-1);

Carta c1 = JuegoNormal.listaMazos.get(0).getMazoPrincipal().get(n1);

int n2 = (int) (Math.random() \* JuegoNormal.listaMazos.get(0).getMazoPrincipal().size()-1);

Carta c2 = JuegoNormal.listaMazos.get(0).getMazoPrincipal().get(n2);

int n3 = (int) (Math.random() \* JuegoNormal.listaMazos.get(0).getMazoPrincipal().size()-1);

Carta c3 = JuegoNormal.listaMazos.get(0).getMazoPrincipal().get(n3);

int n4 = (int) (Math.random() \* JuegoNormal.listaMazos.get(0).getMazoPrincipal().size()-1);

Carta c4 = JuegoNormal.listaMazos.get(0).getMazoPrincipal().get(n4);

JuegoNormal.listaJugadores.get(numeroSiguiente()).agregarCartas(Arrays.asList(c1, c2, c3, c4));

JuegoNormal.listaMazos.get(0).removerCartas(Arrays.asList(c1, c2, c3, c4));

cambioColor();

break;

case ("skip"):

incrementFocus();

incrementFocus();

break;

case ("spin"):

JuegoNormal.rondaHoraria = !JuegoNormal.rondaHoraria;

incrementFocus();

break;

case ("color"):

cambioColor();

break;

}

}

public void preguntarMov(){

this.setVisible(false);

VistaJugador vistaJugador = new VistaJugador(JuegoNormal.pozo,JuegoNormal.jugadorFocus);

vistaJugador.setVisible(true);

}

public int verValor(Carta c){

int valor=0;

if(c.getTipo().equals("especial")){

switch(c.getValor()){

case("spin"):valor=7;break;

case("skip"):valor=4;break;

case("color"):valor=10;break;

case("+2"):valor=8;break;

case("+4"):valor=6;break;

}

}else{

valor=Integer.parseInt(c.getValor());

}

return valor;

}

public int verTipo(Carta c){

if(c.getTipo().equals("especial")){

return 10;

}else{

return 5;

}

}

public int verColor(Carta c){

int valor;

String colorCarta=c.getColor();

switch(colorCarta){

case("rojo"):valor=15;break;

case("amarillo"):valor=25;break;

case("azul"):valor=20;break;

case("verde"):valor=10;break;

case("joker"):valor=5;break;

default:valor=0;break;

}

return valor;

}

public void checkCantidadCartas(Carta c){

int tamañoActual=JuegoNormal.listaMazos.get(0).getMazoPrincipal().size();

if((c.getTipo().equals("especial") && (c.getValor().equals("+4") || c.getValor().equals("+2"))) && tamañoActual<4 ){

rellenarMazo();

}

}

public String mString(){

String data="MAZO\n-";

for (Carta carta : JuegoNormal.listaMazos.get(0).getMazoPrincipal()) {

data+=carta.getValor()+" "+carta.getTipo()+" "+carta.getColor();

if (JuegoNormal.listaMazos.get(0).getMazoPrincipal().indexOf(carta)!=JuegoNormal.listaMazos.get(0).getMazoPrincipal().size()-1){

data+=",";

}

}

data+="-";

data+="\nPOZO\n"+JuegoNormal.pozo.getValor()+" "+JuegoNormal.pozo.getTipo()+" "+JuegoNormal.pozo.getColor();

return data;

}

public void asignarValores(ArrayList<Carta> listaCartas){

for (Carta carta : listaCartas) {

int aux=0;

aux+= verTipo(carta)+verColor(carta)+verValor(carta);

carta.setCodigo(aux\*31);

}

}

public String jString(){

String data="";

for (Jugador jug : JuegoNormal.listaJugadores) {

int codigoJug=0;

data+="JUGADOR\n";

data+=jug.getNombre()+"\n"+jug.getClave()+"\n-";

asignarValores(jug.getManoCartas());

for (Carta carta : jug.getManoCartas()) {

codigoJug+=carta.getCodigo();

data+=carta.getValor()+" "+carta.getTipo()+" "+carta.getColor();

if (jug.getManoCartas().indexOf(carta)!=jug.getManoCartas().size()-1){

data+=",";

}

}

data+="-\n";

data+=codigoJug+"\n";

}

data+="FOCUS\n"+JuegoNormal.jugadorFocus;

return data;

}

public void guardarData(){

JuegoNormal.persistencia.escribirArchivo(jString(), mString());

}

private int numeroSiguiente() {

incrementFocus();

return JuegoNormal.jugadorFocus;

}

public void cambioColor() {

String msj = "Elija a que color quiere cambiar:\n"

+ "1-rojo\n2-azul\n3-amarillo\n4-verde";

String choice = JOptionPane.showInputDialog(msj);

switch (choice) {

case ("1"):

JuegoNormal.pozo.setColor("rojo");

break;

case ("2"):

JuegoNormal.pozo.setColor("azul");

break;

case ("3"):

JuegoNormal.pozo.setColor("amarillo");

break;

case ("4"):

JuegoNormal.pozo.setColor("verde");

break;

default:

JOptionPane.showMessageDialog(null, "La opcion ingresada no es valida!\nIntenta de nuevo.");

cambioColor();

break;

}

incrementFocus();

}

public Jugador nextPlayer() {

for (Jugador jug : JuegoNormal.listaJugadores) {

jug.setVecesEnMenu(0);

}

incrementFocus();

int nextPlayer = JuegoNormal.jugadorFocus;

return JuegoNormal.listaJugadores.get(nextPlayer);

}

public void checkManoJug(Jugador j){

int tamañoMano= j.getManoCartas().size();

if(tamañoMano == 1){

JOptionPane.showMessageDialog(null, "UNO! "+j.getNombre());

}

if(tamañoMano==0){

JOptionPane.showMessageDialog(null, "El jugador "+j.getNombre()+" se ha quedado sin cartas!");

JuegoNormal.listaGanadores.add(j);

JuegoNormal.listaJugadores.remove(j);

}

}

public boolean checkPerder(){

if(JuegoNormal.listaJugadores.size()==1){

return true;

}

return false;

}

public void checkearEntradaMenu(){

vecesEnMenu++;

if(vecesEnMenu==1){

if(otraValidacion()){

JOptionPane.showMessageDialog(null, "Bienvenido!");

}

}

}

public boolean otraValidacion(){

Jugador j = JuegoNormal.listaJugadores.get(JuegoNormal.jugadorFocus);

JPasswordField pwd = new JPasswordField();

JOptionPane.showConfirmDialog(null, pwd, "Ingrese su clave "+j.getNombre(),JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

if(pwd.getText().equals(j.getClave())){

return true;

}else{

JOptionPane.showMessageDialog(null, "Contraseña incorrecta");

otraValidacion();

}

return false;

}

public void finPartida(){

if(checkPerder()){

String ganadores="";

for (Jugador jug : JuegoNormal.listaGanadores) {

ganadores+=JuegoNormal.listaGanadores.indexOf(jug)+1+" Puesto-";

ganadores+=jug.getNombre()+"\n";

}

JOptionPane.showMessageDialog(null, "Queda solo un jugador, la partida termino!"

+ "\nLos ganadores fueron:\n"+ganadores);

System.exit(0);

}

}

public boolean validarTiro(String carta){

String [] cartaPartida=carta.split(" ");

Carta cartaTirada = new Carta(cartaPartida[1],cartaPartida[2],cartaPartida[0]);

if(JuegoNormal.pozo.validarCarta(cartaTirada)){

JOptionPane.showMessageDialog(null, "Carta Tirada!");

JuegoNormal.pozo.setColor(cartaTirada.getColor());

JuegoNormal.pozo.setValor(cartaTirada.getValor());

JuegoNormal.pozo.setTipo(cartaTirada.getTipo());

String carta2=JuegoNormal.pozo.getValor()+JuegoNormal.pozo.getColor();

cartaPozo.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/"+carta2+".png")));

int lugarJugador=JuegoNormal.jugadorFocus;

JuegoNormal.listaJugadores.get(lugarJugador).removerCarta(cartaTirada);

verificarEspecial(JuegoNormal.pozo);

checkManoJug(JuegoNormal.listaJugadores.get(JuegoNormal.jugadorFocus));

finPartida();

if(JuegoNormal.pozo.getTipo().equals("especial")){

preguntarMov();

}else{

incrementFocus();

preguntarMov();

}

return true;

}else{

JOptionPane.showMessageDialog(null, "Jugada No Valida!");

return false;

}

}

public void cambiarImagen(){

int tamañoMano2=JuegoNormal.listaJugadores.get(JuegoNormal.jugadorFocus).getManoCartas().size();

for (int i = 0; i < tamañoMano2; i++) {

String card=JuegoNormal.listaJugadores.get(JuegoNormal.jugadorFocus).getManoCartas().get(i).getValor()+JuegoNormal.listaJugadores.get(JuegoNormal.jugadorFocus).getManoCartas().get(i).getColor();

listaCartas.get(i).setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/"+card+".png")));

listaCartas.get(i).setName(JuegoNormal.listaJugadores.get(JuegoNormal.jugadorFocus).getManoCartas().get(i).getValor()+" "+JuegoNormal.listaJugadores.get(JuegoNormal.jugadorFocus).getManoCartas().get(i).getColor()+" "+JuegoNormal.listaJugadores.get(JuegoNormal.jugadorFocus).getManoCartas().get(i).getTipo());

}

for (int i = tamañoMano2; i < listaCartas.size(); i++) {

cartasLibres.add(listaCartas.get(i));

}

for (int i = tamañoMano2; i < listaCartas.size(); i++) {

listaCartas.remove(i);

}

for (JLabel car : cartasLibres) {

car.setIcon(null);

car.setName("");

}

}

public ArrayList<JLabel> getListaCartas() {

return listaCartas;

}

public void setListaCartas(ArrayList<JLabel> listaCartas) {

this.listaCartas = listaCartas;

}

public void checkMazoVacio() {

if (JuegoNormal.listaMazos.get(0).getMazoPrincipal().isEmpty()) {

rellenarMazo();

}

}

public void rellenarMazo() {

JuegoNormal.listaMazos.get(1).mezclarMazo();

JuegoNormal.listaMazos.get(0).getMazoPrincipal().addAll(JuegoNormal.listaMazos.get(1).getMazoPrincipal());

}

public void levantarCartaMazo(Jugador j) {

checkMazoVacio();

if (!tieneCartaParaJugar(j)){

int numeroRandom = (int) (Math.random() \* JuegoNormal.listaMazos.get(0).getMazoPrincipal().size());

j.getManoCartas().add(JuegoNormal.listaMazos.get(0).getMazoPrincipal().get(numeroRandom));

JuegoNormal.listaMazos.get(0).removerCarta(JuegoNormal.listaMazos.get(0).getMazoPrincipal().indexOf(numeroRandom-1));

} else {

JOptionPane.showMessageDialog(null, "Tienes al menos una carta válidad para jugar");

}

}

/\*

Método para verificar si el jugador tiene alguna carta válida antes de levantar una carta del mazo.

\*/

public boolean tieneCartaParaJugar (Jugador j){

checkMazoVacio();

for (int i = 0; i <j.getManoCartas().size() ; i++) {

if (JuegoNormal.pozo.validarCarta(j.getManoCartas().get(i)))

return true;

}

return false;

}

public void juntarCartas(){

listaCartas.add(carta1);

listaCartas.add(carta2);

listaCartas.add(carta3);

listaCartas.add(carta4);

listaCartas.add(carta5);

listaCartas.add(carta6);

listaCartas.add(carta7);

listaCartas.add(carta8);

listaCartas.add(carta9);

listaCartas.add(carta10);

listaCartas.add(carta11);

listaCartas.add(carta12);

listaCartas.add(carta13);

listaCartas.add(carta14);

listaCartas.add(carta15);

listaCartas.add(carta16);

listaCartas.add(carta17);

listaCartas.add(carta18);

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

carta2 = new javax.swing.JLabel();

carta1 = new javax.swing.JLabel();

carta3 = new javax.swing.JLabel();

carta4 = new javax.swing.JLabel();

carta5 = new javax.swing.JLabel();

carta6 = new javax.swing.JLabel();

carta7 = new javax.swing.JLabel();

carta8 = new javax.swing.JLabel();

carta9 = new javax.swing.JLabel();

carta10 = new javax.swing.JLabel();

carta11 = new javax.swing.JLabel();

carta12 = new javax.swing.JLabel();

carta13 = new javax.swing.JLabel();

carta14 = new javax.swing.JLabel();

carta15 = new javax.swing.JLabel();

carta16 = new javax.swing.JLabel();

carta17 = new javax.swing.JLabel();

carta18 = new javax.swing.JLabel();

cartaPozo = new javax.swing.JLabel();

btnPasar = new javax.swing.JButton();

btnLevantar = new javax.swing.JButton();

jButton1 = new javax.swing.JButton();

lblNombre = new javax.swing.JButton();

lblNombre1 = new javax.swing.JButton();

jLabel3 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setTitle("UNO");

setMaximumSize(new java.awt.Dimension(1280, 1024));

setMinimumSize(new java.awt.Dimension(1280, 1024));

setPreferredSize(new java.awt.Dimension(1280, 1024));

setResizable(false);

setSize(new java.awt.Dimension(1280, 1024));

getContentPane().setLayout(null);

carta2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta2.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta2MouseClicked(evt);

}

});

getContentPane().add(carta2);

carta2.setBounds(190, 60, 150, 190);

carta1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta1.setName("Pepe");

carta1.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta1MouseClicked(evt);

}

});

getContentPane().add(carta1);

carta1.setBounds(20, 60, 150, 190);

carta3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta3.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta3MouseClicked(evt);

}

});

getContentPane().add(carta3);

carta3.setBounds(360, 60, 150, 190);

carta4.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta4.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta4MouseClicked(evt);

}

});

getContentPane().add(carta4);

carta4.setBounds(520, 60, 150, 190);

carta5.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta5.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta5MouseClicked(evt);

}

});

getContentPane().add(carta5);

carta5.setBounds(690, 60, 150, 190);

carta6.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta6.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta6MouseClicked(evt);

}

});

getContentPane().add(carta6);

carta6.setBounds(850, 70, 150, 190);

carta7.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta7.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta7MouseClicked(evt);

}

});

getContentPane().add(carta7);

carta7.setBounds(20, 260, 150, 190);

carta8.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta8.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta8MouseClicked(evt);

}

});

getContentPane().add(carta8);

carta8.setBounds(190, 260, 150, 190);

carta9.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta9.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta9MouseClicked(evt);

}

});

getContentPane().add(carta9);

carta9.setBounds(360, 260, 150, 190);

carta10.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta10.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta10MouseClicked(evt);

}

});

getContentPane().add(carta10);

carta10.setBounds(520, 260, 150, 190);

carta11.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta11.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta11MouseClicked(evt);

}

});

getContentPane().add(carta11);

carta11.setBounds(690, 260, 150, 190);

carta12.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta12.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta12MouseClicked(evt);

}

});

getContentPane().add(carta12);

carta12.setBounds(850, 260, 150, 190);

carta13.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta13.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta13MouseClicked(evt);

}

});

getContentPane().add(carta13);

carta13.setBounds(20, 470, 150, 190);

carta14.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta14.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta14MouseClicked(evt);

}

});

getContentPane().add(carta14);

carta14.setBounds(190, 470, 150, 190);

carta15.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta15.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta15MouseClicked(evt);

}

});

getContentPane().add(carta15);

carta15.setBounds(360, 470, 150, 190);

carta16.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta16.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta16MouseClicked(evt);

}

});

getContentPane().add(carta16);

carta16.setBounds(520, 470, 150, 190);

carta17.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta17.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta17MouseClicked(evt);

}

});

getContentPane().add(carta17);

carta17.setBounds(690, 470, 150, 190);

carta18.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/+4joker.png"))); // NOI18N

carta18.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

carta18MouseClicked(evt);

}

});

getContentPane().add(carta18);

carta18.setBounds(850, 470, 150, 190);

cartaPozo.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/1verde.png"))); // NOI18N

cartaPozo.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

cartaPozoMouseClicked(evt);

}

});

getContentPane().add(cartaPozo);

cartaPozo.setBounds(1040, 270, 150, 190);

btnPasar.setBackground(new java.awt.Color(0, 0, 0));

btnPasar.setFont(new java.awt.Font("Dialog", 1, 18)); // NOI18N

btnPasar.setForeground(new java.awt.Color(255, 255, 255));

btnPasar.setText("Pasar Turno");

btnPasar.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnPasarActionPerformed(evt);

}

});

getContentPane().add(btnPasar);

btnPasar.setBounds(1030, 480, 190, 42);

btnLevantar.setBackground(new java.awt.Color(0, 0, 0));

btnLevantar.setFont(new java.awt.Font("Dialog", 1, 18)); // NOI18N

btnLevantar.setForeground(new java.awt.Color(255, 255, 255));

btnLevantar.setText("Levantar Carta");

btnLevantar.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnLevantarActionPerformed(evt);

}

});

getContentPane().add(btnLevantar);

btnLevantar.setBounds(1030, 530, 190, 42);

jButton1.setBackground(new java.awt.Color(0, 0, 0));

jButton1.setFont(new java.awt.Font("Dialog", 1, 18)); // NOI18N

jButton1.setForeground(new java.awt.Color(255, 255, 255));

jButton1.setText("Guardar y Salir");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

getContentPane().add(jButton1);

jButton1.setBounds(1030, 580, 190, 40);

lblNombre.setBackground(new java.awt.Color(0, 0, 0));

lblNombre.setFont(new java.awt.Font("Dialog", 1, 24)); // NOI18N

lblNombre.setForeground(new java.awt.Color(255, 255, 255));

lblNombre.setText("asd");

lblNombre.setFocusPainted(false);

lblNombre.setFocusable(false);

getContentPane().add(lblNombre);

lblNombre.setBounds(1000, 100, 240, 100);

lblNombre1.setBackground(new java.awt.Color(0, 0, 0));

lblNombre1.setFont(new java.awt.Font("Dialog", 1, 24)); // NOI18N

lblNombre1.setForeground(new java.awt.Color(255, 255, 255));

lblNombre1.setText("Carta del Pozo");

lblNombre1.setFocusPainted(false);

lblNombre1.setFocusable(false);

getContentPane().add(lblNombre1);

lblNombre1.setBounds(1000, 220, 240, 40);

jLabel3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/maxresdefault.jpg"))); // NOI18N

getContentPane().add(jLabel3);

jLabel3.setBounds(0, 0, 1280, 720);

pack();

}// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

guardarData();

JOptionPane.showMessageDialog(null, "La partida fue guardada!");

System.exit(0);

}

private void btnLevantarActionPerformed(java.awt.event.ActionEvent evt) {

if(cartasLibres.isEmpty()){

JOptionPane.showMessageDialog(null, "Cupo Maximo de cartas en mano!");

return;

}

checkMazoVacio();

Carta cartaLevantada = JuegoNormal.listaMazos.get(0).getMazoPrincipal().get(0);

//Asigna una carta al, primer slot libre que haya

String carta=cartaLevantada.getValor()+cartaLevantada.getColor();

JuegoNormal.listaJugadores.get(JuegoNormal.jugadorFocus).agregarCartas(Arrays.asList(cartaLevantada));

cartasLibres.get(0).setIcon(new javax.swing.ImageIcon(getClass().getResource("/GUI/Resources/recartas/"+carta+".png")));

cartasLibres.get(0).setName(cartaLevantada.getValor()+" "+cartaLevantada.getColor()+" "+cartaLevantada.getTipo());

cartasLibres.remove(0);

JuegoNormal.listaMazos.get(0).getMazoPrincipal().remove(0);

}

private void btnPasarActionPerformed(java.awt.event.ActionEvent evt) {

JOptionPane.showMessageDialog(null, "Turno Cedido!");

incrementFocus();

preguntarMov();

}

private void carta1MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta1.getName())){

carta1.setName("");

carta1.setIcon(null);

cartasLibres.add(carta1);

}

}

private void carta2MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta2.getName())){

carta2.setName("");

carta2.setIcon(null);

cartasLibres.add(carta2);

}

}

private void carta3MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta3.getName())){

carta3.setName("");

carta3.setIcon(null);

cartasLibres.add(carta3);

}

}

private void carta4MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta4.getName())){

carta4.setName("");

carta4.setIcon(null);

cartasLibres.add(carta4);

}

}

private void carta5MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta5.getName())){

carta5.setName("");

carta5.setIcon(null);

cartasLibres.add(carta5);

}

}

private void carta6MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta6.getName())){

carta6.setName("");

carta6.setIcon(null);

cartasLibres.add(carta6);

}

}

private void carta7MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta7.getName())){

carta7.setName("");

carta7.setIcon(null);

cartasLibres.add(carta7);

}

}

private void carta8MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta8.getName())){

carta8.setName("");

carta8.setIcon(null);

cartasLibres.add(carta8);

}

}

private void carta9MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta9.getName())){

carta9.setName("");

carta9.setIcon(null);

cartasLibres.add(carta9);

}

}

private void carta10MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta10.getName())){

carta10.setName("");

carta10.setIcon(null);

cartasLibres.add(carta10);

}

}

private void carta11MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta11.getName())){

carta11.setName("");

carta11.setIcon(null);

cartasLibres.add(carta11);

}

}

private void carta12MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta12.getName())){

carta12.setName("");

carta12.setIcon(null);

cartasLibres.add(carta12);

}

}

private void carta13MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta13.getName())){

carta13.setName("");

carta13.setIcon(null);

cartasLibres.add(carta13);

}

}

private void carta14MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta14.getName())){

carta14.setName("");

carta14.setIcon(null);

cartasLibres.add(carta14);

}

}

private void carta15MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta15.getName())){

carta15.setName("");

carta15.setIcon(null);

cartasLibres.add(carta15);

}

}

private void carta16MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta16.getName())){

carta16.setName("");

carta16.setIcon(null);

cartasLibres.add(carta16);

}

}

private void carta17MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta17.getName())){

carta17.setName("");

carta17.setIcon(null);

cartasLibres.add(carta17);

}

}

private void carta18MouseClicked(java.awt.event.MouseEvent evt) {

if(validarTiro(carta18.getName())){

carta18.setName("");

carta18.setIcon(null);

cartasLibres.add(carta18);

}

}

private void cartaPozoMouseClicked(java.awt.event.MouseEvent evt) {

String msj="El color del pozo es: "+JuegoNormal.pozo.getColor();

JOptionPane.showMessageDialog(null, msj);

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(VistaJugador.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(VistaJugador.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(VistaJugador.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(VistaJugador.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

Jugador jugador = new Jugador("Player 1");

Carta carta = new Carta("rojo", "numero", "3");

Carta carta2 = new Carta("azul", "numero", "9");

Carta carta3 = new Carta("verde", "numero", "2");

Carta carta4 = new Carta("verde", "especial", "+2");

Carta carta5 = new Carta("joker", "especial", "+4");

Carta carta6 = new Carta("rojo", "numero", "9");

jugador.agregarCartas(Arrays.asList(carta,carta2,carta3,carta4,carta5));

new VistaJugador(carta6,0).setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton btnLevantar;

private javax.swing.JButton btnPasar;

private javax.swing.JLabel carta1;

private javax.swing.JLabel carta10;

private javax.swing.JLabel carta11;

private javax.swing.JLabel carta12;

private javax.swing.JLabel carta13;

private javax.swing.JLabel carta14;

private javax.swing.JLabel carta15;

private javax.swing.JLabel carta16;

private javax.swing.JLabel carta17;

private javax.swing.JLabel carta18;

private javax.swing.JLabel carta2;

private javax.swing.JLabel carta3;

private javax.swing.JLabel carta4;

private javax.swing.JLabel carta5;

private javax.swing.JLabel carta6;

private javax.swing.JLabel carta7;

private javax.swing.JLabel carta8;

private javax.swing.JLabel carta9;

private javax.swing.JLabel cartaPozo;

private javax.swing.JButton jButton1;

private javax.swing.JLabel jLabel3;

private javax.swing.JButton lblNombre;

private javax.swing.JButton lblNombre1;

public JButton getLblNombre() {

return lblNombre;

}

}

package GUI;

public class ingresoClave extends javax.swing.JFrame {

public ingresoClave() {

initComponents();

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jLabel1.setText("HOLA");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(158, 158, 158)

.addComponent(jLabel1)

.addContainerGap(204, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(125, 125, 125)

.addComponent(jLabel1)

.addContainerGap(160, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(ingresoClave.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(ingresoClave.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(ingresoClave.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(ingresoClave.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new ingresoClave().setVisible(true);

}

});

}

private javax.swing.JLabel jLabel1;

}

package Juego;

import GUI.MenuPrincipal;

import GUI.VistaJugador;

import Models.Carta;

import Models.Jugador;

import Models.Mazo;

import Persistencia.Persistencia;

import javax.swing.\*;

import java.util.ArrayList;

import java.util.Arrays;

public class JuegoNormal {

public static ArrayList<Jugador> listaJugadores = new ArrayList<>();

public static ArrayList<Jugador> listaGanadores = new ArrayList<>();

public static ArrayList<Mazo> listaMazos = new ArrayList<>();

public static int jugadorFocus = 0;

public static boolean rondaHoraria = true;

public static Carta pozo = new Carta();

static int acumulador = 0;

public static Persistencia persistencia = new Persistencia();

private ArrayList<String> kdena= persistencia.getKdena();

private final int tamañoMano=7;

public boolean isValid=true;

public void setJugadorFocus (int num){

jugadorFocus = num;

}

public boolean isIsValid() {

return isValid;

}

public void setIsValid(boolean isValid) {

this.isValid = isValid;

}

public JuegoNormal() {}

/\*

Crea el mazo principal del juego y uno de respaldo. Los dos se añaden a la lista

de mazos

\*/

public void llenarMazos() {

Mazo mazoPrincipal = new Mazo();

mazoPrincipal.llenarMazo();

Mazo mazoSecundario = new Mazo();

mazoSecundario.llenarMazo();

listaMazos.addAll(Arrays.asList(mazoPrincipal, mazoSecundario));

}

/\*

Crea algunos jugadores y les asigna una mano de cartas.

\*/

public void llenarJugadores() {

Jugador jugador = new Jugador("Jugador1");

jugador.setManoCartas(generarMano());

Jugador jugador2 = new Jugador("Jugador2");

jugador2.setManoCartas(generarMano());

Jugador jugador3 = new Jugador("Jugador3");

jugador3.setManoCartas(generarMano());

listaJugadores.addAll(Arrays.asList(jugador, jugador2, jugador3));

for (Jugador jug : listaJugadores) {

jug.setClave("test");

}

}

/\*

Crea un jugador con todas cartas +4 para probar la partida

\*/

public void jugTrucado(){

Jugador jugadorr = new Jugador("Cheater");

for (int i = 0; i < tamañoMano; i++) {

Carta cartaa = new Carta("joker", "especial", "+4");

jugadorr.agregarCartas(Arrays.asList(cartaa));

}

// Carta cartaa2 = new Carta("verde", "especial", "spin");

// Carta cartaa3 = new Carta("rojo", "especial", "spin");

// Carta cartaa4 = new Carta("azul", "especial", "spin");

// Carta cartaa5 = new Carta("amarillo", "especial", "spin");

// jugadorr.addCartas(Arrays.asList(cartaa2,cartaa3,cartaa4,cartaa5));

jugadorr.setClave("q");

listaJugadores.add(jugadorr);

}

/\*

Devuelve la lista con los jugadores

\*/

public ArrayList<Jugador> getListaJugadores() {

return listaJugadores;

}

/\*

Devuelve la lista con los 2 mazos

\*/

public ArrayList<Mazo> getListaMazos() {

return listaMazos;

}

/\*

Devuelve el numero de jugador que esta en juego

\*/

public int getJugadorFocus() {

return jugadorFocus;

}

/\*

Devuelve la mano de un jugador (7cartas) aleatorias

\*/

public ArrayList<Carta> generarMano() {

ArrayList<Carta> mano = new ArrayList<>();

Mazo mazo = listaMazos.get(0);

for (int i = 0; i < tamañoMano; i++) {

int random = (int) (Math.random() \* mazo.getMazoPrincipal().size());

mano.add(mazo.getMazoPrincipal().get(random));

}

return mano;

}

public boolean validarClave(String claveInput,Jugador j){

return j.getClave().equals(claveInput);

}

/\*

Elige una carta aleatoria del mazo, que no sea una carta especial, y la saca del

mazo una vez que se tira.

\*/

public void primerCartaPozo() {

listaMazos.get(0).mezclarMazo();

for (int i = 0; i < listaMazos.get(0).getMazoPrincipal().size(); i++) {

Carta c = listaMazos.get(0).getMazoPrincipal().get(i);

if (!"especial".equals(c.getTipo())) {

pozo.setColor(c.getColor());

pozo.setTipo(c.getTipo());

pozo.setValor(c.getValor());

listaMazos.get(0).removerCarta(i);

}

}

}

public void asignarValores(ArrayList<Carta> listaCartas){

for (Carta carta : listaCartas) {

int aux=0;

aux+= verTipo(carta)+verColor(carta)+verValor(carta);

carta.setCodigo(aux\*31);

}

}

public int verValor(Carta c){

int valor=0;

if(c.getTipo().equals("especial")){

switch(c.getValor()){

case("spin"):valor=7;break;

case("skip"):valor=4;break;

case("color"):valor=10;break;

case("+2"):valor=8;break;

case("+4"):valor=6;break;

}

}else{

valor=Integer.parseInt(c.getValor());

}

return valor;

}

public int verTipo(Carta c){

if(c.getTipo().equals("especial")){

return 10;

}else{

return 5;

}

}

public int verColor(Carta c){

int valor;

String colorCarta=c.getColor();

switch(colorCarta){

case("rojo"):valor=15;break;

case("amarillo"):valor=25;break;

case("azul"):valor=20;break;

case("verde"):valor=10;break;

case("joker"):valor=5;break;

default:valor=0;break;

}

return valor;

}

/\*

Trae una carta aleatoria de la mano del jugador.

\*/

public Carta generarCarta(Jugador j) {

int random = (int) (Math.random() \* j.getManoCartas().size());

Carta cartaJugada = j.getManoCartas().get(random);

return cartaJugada;

}

/\*

Tira una carta aleatoria valida

\*/

public Carta generarCartaValida(Jugador j) {

if (j.validarMano(pozo)) {

return j.traerCartaValida(pozo);

}

return null;

}

/\*

Simular tiro carta valida de bot

\*/

public void tirarValida(Jugador j) {

if (j.validarMano(pozo)) {

Carta cartaJugada = generarCartaValida(j);

String msj = "";

msj += ("Jugada exitosa!\n");

msj += ("Pozo:" + pozo.getValor() + " " + pozo.getTipo() + " " + pozo.getColor() + "\n");

msj += ("Carta tirada: " + cartaJugada.getValor() + " " + cartaJugada.getTipo() + " " + cartaJugada.getColor() + "\n");

JOptionPane.showMessageDialog(null, msj);

pozo.setValor(cartaJugada.getValor());

pozo.setColor(cartaJugada.getColor());

if (cartaJugada.getTipo().equals("especial")) {

pozo.setTipo("especial");

} else {

pozo.setTipo("numero");

}

j.removerCarta(j.getManoCartas().indexOf(cartaJugada));

if (pozo.getTipo().equals("especial")) aplicarCartaEspecial(pozo);

} else {

JOptionPane.showMessageDialog(null, "El jugador no posee cartas validas!");

}

}

public void cambioColor() {

String msj = "Elija a que color quiere cambiar:\n"

+ "1-rojo\n2-azul\n3-amarillo\n4-verde";

String choice = JOptionPane.showInputDialog(msj);

switch (choice) {

case ("1"):

pozo.setColor("rojo");

preguntarMovida(nextPlayer());

break;

case ("2"):

pozo.setColor("azul");

preguntarMovida(nextPlayer());

break;

case ("3"):

pozo.setColor("amarillo");

preguntarMovida(nextPlayer());

break;

case ("4"):

pozo.setColor("verde");

preguntarMovida(nextPlayer());

break;

default:

JOptionPane.showMessageDialog(null, "La opcion ingresada no es valida!\nIntenta de nuevo.");

cambioColor();

break;

}

}

public void verificarEspecial(Carta c) {

if (c.getTipo().equals("especial")) {

aplicarCartaEspecial(c);

}

}

/\*

Asigna como jugador que esta jugando a la posicion de j en el array. Valida la carta que tira el jugador

contra la carta que esta en el pozo. Y aplica las consecuencias si la carta es especial.

Y saca la carta de la mano del jugador, siempre que la jugada haya sido valida.

\*/

public void turnoJugador(Jugador j) {

Carta cartaJugada = generarCarta(j);

String msj = "";

boolean valid = true;

if (pozo.validarCarta(cartaJugada)) {

msj += ("Jugada exitosa!\n");

msj += ("Pozo:" + pozo.getValor() + " " + pozo.getTipo() + " " + pozo.getColor() + "\n");

msj += ("Carta tirada: " + cartaJugada.getValor() + " " + cartaJugada.getTipo() + " " + cartaJugada.getColor() + "\n");

pozo.setValor(cartaJugada.getValor());

if (cartaJugada.getTipo().equals("especial")) {

pozo.setTipo("especial");

} else {

pozo.setTipo("numero");

pozo.setColor(cartaJugada.getColor());

}

j.removerCarta(j.getManoCartas().indexOf(cartaJugada));

checkManoJug(j);

if (pozo.getTipo().equals("especial")) aplicarCartaEspecial(pozo);

} else {

msj += ("Jugada no valida!\n");

msj += ("Pozo:" + pozo.getValor() + " " + pozo.getTipo() + " " + pozo.getColor() + "\n");

msj += ("Carta tirada: " + cartaJugada.getValor() + " " + cartaJugada

.getTipo() + " " + cartaJugada.getColor() + "\n");

valid = false;

}

JOptionPane.showMessageDialog(null, msj);

if (!valid) preguntarMovida(j);

}

public boolean validarChoice(String choice,int tamañoMano){

try {

Integer.parseInt(choice);

} catch (Exception e) {

return false;

}

if(Integer.parseInt(choice) < -1 || Integer.parseInt(choice) >= tamañoMano){

return false;

}

return true;

}

public Carta elegirCarta(Jugador j){

int tamañoMano=j.getManoCartas().size();

String choice="";

String cartaPozo = "La carta del pozo es: \n";

cartaPozo += pozo.getTipo() + " " + pozo.getValor() + " " + pozo.getColor()+"\n";

while (!validarChoice(choice=JOptionPane.showInputDialog(j.devolverStringMano()+

"\n"+cartaPozo+"\nIngrese el numero de carta que quiere mostrar\nIngrese -1 para salir"),tamañoMano)) {

JOptionPane.showMessageDialog(null, "Numero ingresado no es valido");

}

if(Integer.parseInt(choice) == -1){

return null;

}

return j.getManoCartas().get(Integer.parseInt(choice));

}

public void tirarEleccion(Jugador j){

Carta cartaJugada = elegirCarta(j);

if(cartaJugada==null){

preguntarMovida(j);

}

String msj = "";

boolean valid = true;

if (pozo.validarCarta(cartaJugada)) {

msj += ("Jugada exitosa!\n");

msj += ("Pozo:" + pozo.getValor() + " " + pozo.getTipo() + " " + pozo.getColor() + "\n");

msj += ("Carta tirada: " + cartaJugada.getValor() + " " + cartaJugada.getTipo() + " " + cartaJugada.getColor() + "\n");

pozo.setValor(cartaJugada.getValor());

if (cartaJugada.getTipo().equals("especial")) {

pozo.setTipo("especial");

} else {

pozo.setTipo("numero");

pozo.setColor(cartaJugada.getColor());

}

j.removerCarta(j.getManoCartas().indexOf(cartaJugada));

checkManoJug(j);

if (pozo.getTipo().equals("especial")) aplicarCartaEspecial(pozo);

} else {

msj += ("Jugada no valida!\n");

msj += ("Pozo:" + pozo.getValor() + " " + pozo.getTipo() + " " + pozo.getColor() + "\n");

msj += ("Carta tirada: " + cartaJugada.getValor() + " " + cartaJugada.getTipo() + " " + cartaJugada.getColor() + "\n");

valid = false;

}

JOptionPane.showMessageDialog(null, msj);

if (!valid) preguntarMovida(j);

}

public void mensajeEntrada(){

JOptionPane.showMessageDialog(null, "¡Bienvenido al UNO!");

}

public void checkManoJug(Jugador j){

int tamañoMano= j.getManoCartas().size();

if(tamañoMano == 1){

JOptionPane.showMessageDialog(null, "UNO! "+j.getNombre());

}

if(tamañoMano==0){

JOptionPane.showMessageDialog(null, "El jugador "+j.getNombre()+" se ha quedado sin cartas!");

listaGanadores.add(j);

listaJugadores.remove(j);

}

}

public void checkCantidadCartas(Carta c){

int tamañoActual=listaMazos.get(0).getMazoPrincipal().size();

if(c.getTipo().equals("especial") && c.getValor().equals("+4") || c.getValor().equals("+2") && tamañoActual<4 ){

rellenarMazo();

}

}

/\*

Checkea si el mazo principal esta vacio, si lo esta lo rellena usando el refil

\*/

public void checkMazoVacio() {

if (listaMazos.get(0).getMazoPrincipal().isEmpty()) {

rellenarMazo();

}

}

/\*l

Toma el mazo secundario, lo mezcla y agrega al mazo principal todas las cartas ya mezcladas

\*/

public void rellenarMazo() {

listaMazos.get(1).mezclarMazo();

listaMazos.get(0).getMazoPrincipal().addAll(listaMazos.get(1).getMazoPrincipal());

}

public boolean otraValidacion(Jugador j){

JPasswordField pwd = new JPasswordField();

JOptionPane.showConfirmDialog(null, pwd, "Ingrese su clave "+j.getNombre(),JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

if(pwd.getText().equals(j.getClave())){

return true;

}else{

JOptionPane.showMessageDialog(null, "Contraseña incorrecta");

otraValidacion(j);

}

return false;

}

public void preguntarMovida(){

VistaJugador vista=new VistaJugador(pozo, jugadorFocus);

vista.setVisible(true);

}

/\*

Accion a tomar en el turno

(Faltan implementar)

\*/

public void preguntarMovida(Jugador j) {

j.setVecesEnMenu(j.getVecesEnMenu()+1);

if(checkPerder()){

String ganadores="";

for (Jugador jug : listaGanadores) {

ganadores+=jug.getNombre()+"\n";

}

JOptionPane.showMessageDialog(null, "Queda solo un jugador, la partida termino!"

+ "\nLos ganadores fueron:\n"+ganadores);

System.exit(0);

}

if(j.getVecesEnMenu()==1){

otraValidacion(j);

}

String msj = "1-Ver mano" +

"\n2-Ver Pozo" +

"\n3-Tirar carta" +

"\n4-Levantar carta del mazo" +

"\n5-Validar mano" +

"\n6-Pasar turno" +

"\n7-Tirar Valida"+

"\n8-Elegir Carta para Tirar"

+ "\n10-Guardar y Salir" +

"\n -------------------------------"+

"\n POZO "+

"\n " + getPozo().toString() +

"\n -------------------------------";

int opcion;

try {

opcion = Integer.parseInt(JOptionPane.showInputDialog(msj));

} catch (NumberFormatException e) {

opcion = -1;

}

//agregar constantes

final int verMano=1;

final int verCartaPozo=2;

final int tirarCartaRandom=3;

final int levantarCartaMazo=4;

final int validarMano=5;

final int pasarTurno=6;

final int tirarValida=7;

final int elegirCartaJugador=8;

final int salir=10;

final int casodefault=-1;

switch (opcion) {

case (verMano):

j.imprimirMano();

preguntarMovida(j);

break;

case (verCartaPozo):

String cartaPozo = "La carta del pozo es: \n\n";

cartaPozo += pozo.getTipo() + "\n" + pozo.getValor() + "\n" + pozo.getColor();

JOptionPane.showMessageDialog(null, cartaPozo);

preguntarMovida(j);

break;

case (tirarCartaRandom):

turnoJugador(j);

verificarEspecial(pozo);

JOptionPane.showMessageDialog(null, "Tu turno: " + listaJugadores.get(jugadorFocus+1).getNombre());

preguntarMovida(nextPlayer());

break;

case (levantarCartaMazo):

levantarCartaMazo(j);

JOptionPane.showMessageDialog(null, "Carta Levantada!");

preguntarMovida(j);

break;

case (validarMano):

String alert;

if (j.validarMano(pozo)) {

alert = "Tu mano cuenta con una carta valida!";

} else {

alert = "No posees ninguna carta valida! Toma del pozo";

}

JOptionPane.showMessageDialog(null, alert);

preguntarMovida(j);

break;

case (pasarTurno):

JOptionPane.showMessageDialog(null, "Turno cedido!");

preguntarMovida(nextPlayer());

break;

case (tirarValida):

tirarValida(j);

checkManoJug(j);

checkPerder();

preguntarMovida(nextPlayer());

break;

case (elegirCartaJugador):

tirarEleccion(j);

checkPerder();

preguntarMovida(nextPlayer());

break;

case (salir):

guardarData();

System.exit(0);

case(150):

VistaJugador vista=new VistaJugador(pozo, jugadorFocus);

vista.setVisible(true);

break;

case (casodefault):

JOptionPane.showMessageDialog(null, "¡El valor ingresado no es valido!");

preguntarMovida(j);

break;

default:

JOptionPane.showMessageDialog(null, "¡El numero ingresado no es valido!");

//Ignorar Warning.

preguntarMovida(j);

break;

}

}

/\*

Toma una carta aleatoria del mazo y la agrega a la mano del jugador

\*/

/\*

Se modificó el método para que no deje levantar cartas si el jugador tiene alguna carta válida.

\*/

public void levantarCartaMazo(Jugador j) {

checkMazoVacio();

if (!tieneCartaParaJugar(j)){

int numeroRandom = (int) (Math.random() \* listaMazos.get(0).getMazoPrincipal().size());

j.getManoCartas().add(listaMazos.get(0).getMazoPrincipal().get(numeroRandom));

listaMazos.get(0).removerCarta(listaMazos.get(0).getMazoPrincipal().indexOf(numeroRandom-1));

} else {

JOptionPane.showMessageDialog(null, "Tienes al menos una carta válidad para jugar");

preguntarMovida(j);

}

}

/\*

Método para verificar si el jugador tiene alguna carta válida antes de levantar una carta del mazo.

\*/

public boolean tieneCartaParaJugar (Jugador j){

checkMazoVacio();

for (int i = 0; i <j.getManoCartas().size() ; i++) {

if (getPozo().validarCarta(j.getManoCartas().get(i)))

return true;

}

return false;

}

/\*

Metodo que va corriendo el focus de los jugadores

\*/

public void incrementFocus() {

int lastPlayer = jugadorFocus;

if (rondaHoraria) {

if (lastPlayer + 1 == listaJugadores.size()) {

jugadorFocus = 0;

} else {

jugadorFocus = lastPlayer + 1;

}

} else {

if (lastPlayer - 1 < 0) {

jugadorFocus = listaJugadores.size() - 1;

} else {

jugadorFocus = lastPlayer - 1;

}

}

}

/\*

Metodo que calcula quien es el siguiente jugador

(A mejorar el codigo, pero funciona)

\*/

public Jugador nextPlayer() {

for (Jugador jug : listaJugadores) {

jug.setVecesEnMenu(0);

}

incrementFocus();

int nextPlayer = jugadorFocus;

return listaJugadores.get(nextPlayer);

}

public int numeroSiguiente(){

int lastPlayer = jugadorFocus;

int siguiente;

if (rondaHoraria) {

if (lastPlayer + 1 == listaJugadores.size()) {

siguiente = 0;

} else {

siguiente = lastPlayer + 1;

}

} else {

if (lastPlayer - 1 < 0) {

siguiente = listaJugadores.size() - 1;

} else {

siguiente = lastPlayer - 1;

}

}

return siguiente;

}

/\*

Falta implementar

Metodo que toma la carta especial, y lleva a cabo la accion que trae la carta.

+2---> funciona

+4---> suma las 4 cartas pero falta resolver el tema del cambio de color.

skip--->

spin--->cambia el sentido de la ronda, falta implementar que le pregunte al siguiente jugador que accion quiere tomar

\*/

public void aplicarCartaEspecial(Carta c) {

checkMazoVacio();

checkCantidadCartas(c);

String tipoCarta = c.getTipo();

String valorCarta = c.getValor();

if (tipoCarta.equals("numero")) return;

switch (valorCarta) {

case ("+2"):

nextPlayer();

int numeroRandom = (int) (Math.random() \* listaMazos.get(0).getMazoPrincipal().size()-1);

Carta ca1 = listaMazos.get(0).getMazoPrincipal().get(numeroRandom);

int numeroRandom2 = (int) (Math.random() \* listaMazos.get(0).getMazoPrincipal().size() - 2);

Carta ca2 = listaMazos.get(0).getMazoPrincipal().get(numeroRandom2);

listaJugadores.get(numeroSiguiente()).agregarCartas(Arrays.asList(ca1, ca2));

listaMazos.get(0).removerCartas(Arrays.asList(ca1, ca2));

break;

case ("+4"):

nextPlayer();

int n1 = (int) (Math.random() \* listaMazos.get(0).getMazoPrincipal().size()-1);

Carta c1 = listaMazos.get(0).getMazoPrincipal().get(n1);

int n2 = (int) (Math.random() \* listaMazos.get(0).getMazoPrincipal().size()-1);

Carta c2 = listaMazos.get(0).getMazoPrincipal().get(n2);

int n3 = (int) (Math.random() \* listaMazos.get(0).getMazoPrincipal().size()-1);

Carta c3 = listaMazos.get(0).getMazoPrincipal().get(n3);

int n4 = (int) (Math.random() \* listaMazos.get(0).getMazoPrincipal().size()-1);

Carta c4 = listaMazos.get(0).getMazoPrincipal().get(n4);

listaJugadores.get(numeroSiguiente()).agregarCartas(Arrays.asList(c1, c2, c3, c4));

listaMazos.get(0).removerCartas(Arrays.asList(c1, c2, c3, c4));

cambioColor();

break;

case ("skip"):

nextPlayer();

preguntarMovida(nextPlayer());

break;

case ("spin"):

rondaHoraria = !rondaHoraria;

preguntarMovida(nextPlayer());

break;

case ("color"):

cambioColor();

break;

}

}

public void preguntarJugadores() {

int numero;

final int minJugadores=2;

final int maxJugadores=6;

try {

numero = Integer.parseInt(JOptionPane.showInputDialog("Ingrese la cantidad de jugadores\nEntre 2 y 6 Jugadores"));

if (numero < minJugadores || numero > maxJugadores) {

JOptionPane.showMessageDialog(null, "El numero ingresado no es valido");

preguntarJugadores();

}

} catch (NumberFormatException e) {

numero = -1;

JOptionPane.showMessageDialog(null, "El numero ingresado no es valido");

preguntarJugadores();

}

for (int i = 0; i < numero; i++) {

String name = "";

while (!validarNombre(name = JOptionPane.showInputDialog("Ingrese el nombre del jugador"))) {

JOptionPane.showMessageDialog(null, "Nombre ingresado no es valido");

}

String pass = "";

while (!validarClave(pass = JOptionPane.showInputDialog("Ingrese la clave del jugador"))) {

JOptionPane.showMessageDialog(null, "Clave ingresada no es valido");

}

Jugador jugador = new Jugador(name, pass);

jugador.agregarCartas(generarMano());

listaJugadores.add(jugador);

JOptionPane.showMessageDialog(null, "Jugador " + jugador.getNombre() + " agregado con exito!");

}

//Descomentar jugador trucado para usar cartas especificas (Clave = q)

//jugTrucado();

for (Jugador jugador : listaJugadores) {

asignarValores(jugador.getManoCartas());

}

//preguntarMovida(listaJugadores.get(jugadorFocus));

preguntarMovida();

}

public boolean validarNombre(String nombre) {

for (int i = 0; i < nombre.length(); i++) {

if (nombre.charAt(i) != ' ') return true;

}

return false;

}

public boolean validarClave(String clave) {

for (int i = 0; i < clave.length(); i++) {

if (clave.charAt(i) != ' ') return true;

}

return false;

}

public boolean checkPerder(){

if(listaJugadores.size()==1){

return true;

}

return false;

}

public boolean isRondaHoraria() {

return rondaHoraria;

}

public void setRondaHoraria(boolean rondaHoraria) {

this.rondaHoraria = rondaHoraria;

}

public Carta getPozo() {

return pozo;

}

public void setPozo(Carta pozo) {

this.pozo = pozo;

}

public ArrayList<Jugador> getListaJug(){

return listaJugadores;

}

public ArrayList<Carta> getMazo(){

return listaMazos.get(0).getMazoPrincipal();

}

public String mString(){

String data="MAZO\n-";

for (Carta carta : getMazo()) {

data+=carta.getValor()+" "+carta.getTipo()+" "+carta.getColor();

if (getMazo().indexOf(carta)!=getMazo().size()-1){

data+=",";

}

}

data+="-";

data+="\nPOZO\n"+pozo.getValor()+" "+pozo.getTipo()+" "+pozo.getColor();

return data;

}

public String jString(){

String data="";

for (Jugador jug : listaJugadores) {

int codigoJug=0;

data+="JUGADOR\n";

data+=jug.getNombre()+"\n"+jug.getClave()+"\n-";

asignarValores(jug.getManoCartas());

for (Carta carta : jug.getManoCartas()) {

codigoJug+=carta.getCodigo();

data+=carta.getValor()+" "+carta.getTipo()+" "+carta.getColor();

if (jug.getManoCartas().indexOf(carta)!=jug.getManoCartas().size()-1){

data+=",";

}

}

data+="-\n";

data+=codigoJug+"\n";

}

data+="FOCUS\n"+jugadorFocus;

return data;

}

public void guardarData(){

persistencia.escribirArchivo(jString(), mString());

}

public void volverMenu(){

MenuPrincipal menu = new MenuPrincipal(1);

menu.setVisible(true);

}

public void cargarData(){

persistencia.leerArchivo();

persistencia.agregarData();

if(persistencia.corrupto){

JOptionPane.showMessageDialog(null, "El fichero fue alterado, y fue eliminado!\n"

+ "O el fichero no tiene contenido!\n"

+ "Para Jugar crea una nueva partida.");

isValid=false;

return;

}

setPozo(persistencia.getPozo());

listaJugadores=persistencia.getListaJugadores();

Mazo mazo = new Mazo();

mazo.agregarCartas(persistencia.getMazo());

Mazo aux = new Mazo();

aux.llenarMazo();

listaMazos.add(mazo);

listaMazos.add(aux);

jugadorFocus=persistencia.getJugadorFocus();

}

}

package Juego;

import Models.Jugador;

import javax.swing.JOptionPane;

/\*

pasos:

llenar mazos

llenar jugadores

primer carta pozo

preguntar accion para el primer jugador

\*/

public class NuevaPartida {

public NuevaPartida(){

JuegoNormal juego = new JuegoNormal();

juego.llenarMazos();

juego.asignarValores(juego.getListaMazos().get(0).getMazoPrincipal());

//juego.llenarJugadores();

//juego.cargarData();

juego.primerCartaPozo();

juego.mensajeEntrada();

juego.preguntarJugadores();

}

}

package Juego;

public class Partida {

/\*

pasos:

llenar mazos

llenar jugadores

primer carta pozo

preguntar accion para el primer jugador

\*/

public Partida(){

JuegoNormal juego = new JuegoNormal();

juego.cargarData();

if(juego.isValid){

//juego.primerCartaPozo();

juego.mensajeEntrada();

juego.preguntarMovida();

}else{

juego.volverMenu();

}

}

public static void main(String[] args) {

Partida partida = new Partida();

}

}

package Models;

import java.util.Objects;

public class Carta {

private String color;

private String valor;

private String tipo;

private int codigo;

public Carta() {}

public void setColor(String color) {

this.color = color;

}

public void setTipo(String tipo) {

this.tipo = tipo;

}

public Carta(String color, String tipo,String valor) {

this.color = color;

this.tipo = tipo;

this.valor=valor;

}

public int getCodigo() {

return codigo;

}

public void setCodigo(int codigo) {

this.codigo = codigo;

}

public String getColor() {

return color;

}

public String getValor() {

return valor;

}

public String getTipo() {

return tipo;

}

public void setValor(String valor) {

this.valor = valor;

}

/\*

Agregar validacion de que pasa cuando la carta que se tira es un +4 o un cambio de color.

Ahora si la carta es un +4 o un cambio de color, el color al cual se pasa lo define la

siguiente carta que se tira.

\*/

public boolean validarCarta(Carta c){

if(c.getTipo().equals("especial") && (c.getValor().equals("+4") || c.getValor().equals("color"))){

return true;

}

if(getColor().equals(c.getColor()) || getValor().equals(c.getValor())){

return true;

}

return false;

}

@Override

public String toString() {

return this.tipo + " " + this.valor + " " + this.color;

}

@Override

public int hashCode() {

int hash = 7;

return hash;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

final Carta other = (Carta) obj;

if (!Objects.equals(this.color, other.color)) {

return false;

}

if (!Objects.equals(this.valor, other.valor)) {

return false;

}

if (!Objects.equals(this.tipo, other.tipo)) {

return false;

}

return true;

}

}

package Models;

import java.util.ArrayList;

import java.util.Collection;

import javax.swing.JOptionPane;

public class Jugador {

private String nombre;

private ArrayList<Carta> manoCartas= new ArrayList<>();

private String clave;

private int vecesEnMenu;

public Jugador(String nombre) {

this(nombre, "");

//this.nombre = nombre;

}

public Jugador(String nombre,String pass) {

this.nombre = nombre;

this.clave=pass;

}

public int getVecesEnMenu() {

return vecesEnMenu;

}

public void setVecesEnMenu(int vecesEnMenu) {

this.vecesEnMenu = vecesEnMenu;

}

public void setClave(String clave) {

this.clave = clave;

}

public String getClave() {

return clave;

}

public String getNombre() {

return nombre;

}

public ArrayList<Carta> getManoCartas() {

return manoCartas;

}

public void setManoCartas(ArrayList<Carta> manoCartas) {

this.manoCartas = manoCartas;

}

public void agregarCartas(Collection<Carta> sumadas){

manoCartas.addAll(sumadas);

}

public String devolverStringMano(){

String mano="Tu mano "+getNombre()+" es:\n\n";

for (int i = 0; i < getManoCartas().size(); i++) {

mano+=i+"-";

mano+=(getManoCartas().get(i).getTipo()+" "+getManoCartas().get(i).getValor()+" "+getManoCartas().get(i).getColor())+"\n";

}

return mano;

}

public void imprimirMano(){

String mano="Tu mano "+getNombre()+" es:\n\n";

for (int i = 0; i < getManoCartas().size(); i++) {

mano+=i+"-";

mano+=(getManoCartas().get(i).getTipo()+" "+getManoCartas().get(i).getValor()+" "+getManoCartas().get(i).getColor())+"\n";

}

JOptionPane.showMessageDialog(null, mano);

}

public void removerCarta(int posicion){

manoCartas.remove(posicion);

}

public void removerCarta(Carta c){

int posicion=-1;

for (int i = 0; i < manoCartas.size(); i++) {

if(manoCartas.get(i).equals(c) ){

posicion=i;

break;

}

}

manoCartas.remove(posicion);

}

public boolean validarMano(Carta c){

for (Carta carta : manoCartas) {

if(carta.validarCarta(c))return true;

}

return false;

}

public Carta traerCartaValida(Carta c){

for (Carta carta : manoCartas) {

if(carta.validarCarta(c))return carta;

}

return null;

}

}

package Models;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collection;

import java.util.Collections;

public class Mazo {

private ArrayList<Carta> mazoPrincipal= new ArrayList<>();

public Mazo() {

}

/\*

Llena el mazo con cartas numericas de los 4 colores mas las cartas especiales

\*/

public void llenarMazo(){

ArrayList<Carta>numeros=new ArrayList<>();

ArrayList<Carta>especiales=new ArrayList<>();

/\*

Add cartas numericas

\*/

for(int i=0;i<4;i++){

String temp="";

if(i==0)temp=("rojo");

if(i==1)temp=("azul");

if(i==2)temp=("verde");

if(i==3)temp=("amarillo");

for (int j = 1; j < 10; j++) {

Carta c = new Carta(temp, "numero", String.valueOf(j));

numeros.add(c);

}

}

/\*

Add cartas +2, spin, skip

\*/

for(int i=0;i<4;i++){

String temp="";

if(i==0)temp=("rojo");

if(i==1)temp=("azul");

if(i==2)temp=("verde");

if(i==3)temp=("amarillo");

Carta s= new Carta(temp, "especial", "skip");

Carta s2= new Carta(temp, "especial", "skip");

Carta t= new Carta(temp, "especial", "spin");

Carta t2= new Carta(temp, "especial", "spin");

Carta p= new Carta(temp, "especial", "+2");

Carta p2= new Carta(temp, "especial", "+2");

especiales.addAll(Arrays.asList(s,s2,t,t2,p,p2));

}

/\*

Add cartas especiales +4 / cambio color

\*/

Carta c1= new Carta("joker","especial", "color");

Carta c2= new Carta("joker","especial", "color");

Carta c3= new Carta("joker", "especial", "color");

Carta c4= new Carta("joker", "especial", "color");

Carta m1= new Carta("joker","especial","+4");

Carta m2= new Carta("joker","especial","+4");

Carta m3= new Carta("joker","especial","+4");

Carta m4= new Carta("joker","especial","+4");

especiales.addAll(Arrays.asList(m1,m2,m3,m4,c1,c2,c3,c4));

mazoPrincipal.addAll(numeros);

mazoPrincipal.addAll(numeros);

mazoPrincipal.addAll(especiales);

//Modificación para que el mazo principal contengo solo 108 cartas

Carta carta = new Carta("rojo", "numero", "0");

Carta carta2 = new Carta("azul", "numero", "0");

Carta carta3 = new Carta("verde", "numero", "0");

Carta carta4 = new Carta("amarillo", "numero", "0");

mazoPrincipal.addAll(Arrays.asList(carta, carta2, carta3, carta4));

//cartas totales 108

}

public void imprimirMazo(){

for (Carta carta : mazoPrincipal) {

System.out.println(carta.getTipo()+" "+carta.getValor());

}

public void mezclarMazo(){

Collections.shuffle(mazoPrincipal);

}

public ArrayList<Carta> getMazoPrincipal() {

return mazoPrincipal;

}

public void removerCarta(int posicion){

mazoPrincipal.remove(posicion);

}

public void removerCartas(Collection<Carta> lista){

mazoPrincipal.removeAll(lista);

}

public void agregarCartas(Collection<Carta> lista){

mazoPrincipal.addAll(lista);

}

//Se agrega este método para verificar que el mazo se haya llenado.

public int tamañoMazo (){

return mazoPrincipal.size();

}

public static void main(String[] args) {

Mazo mazo = new Mazo();

mazo.llenarMazo();

}

}

package Persistencia;

import Models.Carta;

import Models.Jugador;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.Arrays;

import javax.swing.JOptionPane;

public class Persistencia {

private File archivo=new File("Partida.txt");

private ArrayList<String> kdena = new ArrayList<>();

private ArrayList<Jugador> listaJugadores = new ArrayList<>();

private Carta pozo = new Carta();

private ArrayList<Carta> mazo = new ArrayList<>();

private int jugadorFocus=-1;

private int valid=0;

public boolean corrupto=false;

public Persistencia() {

}

public int getJugadorFocus() {

return jugadorFocus;

}

public ArrayList<Jugador> getListaJugadores() {

return listaJugadores;

}

public void setListaJugadores(ArrayList<Jugador> listaJugadores) {

this.listaJugadores = listaJugadores;

}

public Carta getPozo() {

return pozo;

}

public void setPozo(Carta pozo) {

this.pozo = pozo;

}

public ArrayList<Carta> getMazo() {

return mazo;

}

public void setMazo(ArrayList<Carta> mazo) {

this.mazo = mazo;

}

public void leerArchivo(){

try{

String cadena;

FileReader f = new FileReader(archivo);

BufferedReader b = new BufferedReader(f);

while((cadena = b.readLine())!=null) {

kdena.add(cadena);

}

b.close();

}catch(Exception e){

System.out.println(e);

}

}

public ArrayList<String> getKdena() {

return kdena;

}

public String generarValidKdena(String algo){

int tamaño=algo.length();

return "V\n"+String.valueOf(tamaño\*31);

}

public void limpiarFichero(){

try {

PrintWriter writer = new PrintWriter(archivo);

writer.print("");

writer.close();

} catch (IOException ex) {}

}

public void escribirArchivo(String j,String m){

//limpiar el fichero

try {

PrintWriter writer = new PrintWriter(archivo);

writer.print("");

writer.close();

} catch (IOException ex) {}

//reescribir fichero

try{

FileWriter escribir = new FileWriter(archivo,true);

BufferedWriter f = new BufferedWriter(escribir);

//agrega los jugadores y sus cartas como string

String conjunto=j+"\n"+m;

f.write(conjunto);

f.newLine();

f.write(generarValidKdena(conjunto));

f.newLine();

f.close();

}catch(IOException e){}

}

public void agregarData(){

boolean archivoValido=true;

if(!kdena.isEmpty()){

for (int i = 0; i < kdena.size(); i++) {

if (kdena.get(i).equals("JUGADOR")) {

Jugador jugador = new Jugador(kdena.get(i+1), kdena.get(i+2));

String aux1= kdena.get(i+3).replaceAll("-", "");

String [] cartaLista = aux1.split(",");

int codigoAuxiliar=Integer.parseInt(kdena.get(i+4));

for (String carta : cartaLista) {

String [] aux2=carta.split(" ");

jugador.agregarCartas(Arrays.asList(new Carta(aux2[2], aux2[1], aux2[0])));

}

if(calcularValores(jugador.getManoCartas())==codigoAuxiliar){

listaJugadores.add(jugador);

}else{

archivoValido=false;

corrupto=true;

return;

}

}

if (kdena.get(i).equals("POZO")) {

String [] aux=kdena.get(i+1).split(" ");

pozo.setValor(aux[0]);

pozo.setTipo(aux[1]);

pozo.setColor(aux[2]);

}

if(kdena.get(i).equals("MAZO")){

String [] aux = (kdena.get(i+1).replaceAll("-", "")).split(",");

for (String string : aux) {

String [] aux2=string.split(" ");

Carta carta = new Carta(aux2[2], aux2[1], aux2[0]);

mazo.add(carta);

}

}

if(kdena.get(i).equals("FOCUS")){

jugadorFocus=Integer.parseInt(kdena.get(i+1));

}

if(kdena.get(i).equals("V")) valid=Integer.parseInt(kdena.get(i+1));

}

}else{

corrupto=true;

}

if (validarData() || archivoValido || !corrupto) {

System.out.println("Es valido el fichero");

}else{

limpiarFichero();

corrupto=true;

//falta implementar que haga algo como no cargar la partida

}

}

public int calcularValores(ArrayList<Carta> listaCartas){

int codigoAux=0;

for (Carta carta : listaCartas) {

int aux=0;

aux+= verTipo(carta)+verColor(carta)+verValor(carta);

codigoAux+=aux;

}

return codigoAux\*31;

}

public int verValor(Carta c){

int valor=0;

if(c.getTipo().equals("especial")){

switch(c.getValor()){

case("spin"):valor=7;break;

case("skip"):valor=4;break;

case("color"):valor=10;break;

case("+2"):valor=8;break;

case("+4"):valor=6;break;

}

}else{

valor=Integer.parseInt(c.getValor());

}

return valor;

}

public int verTipo(Carta c){

if(c.getTipo().equals("especial")){

return 10;

}else{

return 5;

}

}

public int verColor(Carta c){

int valor;

String colorCarta=c.getColor();

switch(colorCarta){

case("rojo"):valor=15;break;

case("amarillo"):valor=25;break;

case("azul"):valor=20;break;

case("verde"):valor=10;break;

case("joker"):valor=5;break;

default:valor=0;break;

}

return valor;

}

public boolean validarData(){

String conjunto=jString()+"\n"+mString();

int validNumber=conjunto.length()\*31;

return validNumber==valid;

}

public String mString(){

String data="MAZO\n-";

for (Carta carta : getMazo()) {

data+=carta.getValor()+" "+carta.getTipo()+" "+carta.getColor();

if (getMazo().indexOf(carta)!=getMazo().size()-1){

data+=",";

}

}

data+="-";

data+="\nPOZO\n"+pozo.getValor()+" "+pozo.getTipo()+" "+pozo.getColor();

return data;

}

public String jString(){

String data="";

for (Jugador jug : listaJugadores) {

data+="JUGADOR\n";

data+=jug.getNombre()+"\n"+jug.getClave()+"\n-";

for (Carta carta : jug.getManoCartas()) {

data+=carta.getValor()+" "+carta.getTipo()+" "+carta.getColor();

if (jug.getManoCartas().indexOf(carta)!=jug.getManoCartas().size()-1){

data+=",";

}

}

data+="-\n";

}

data+="FOCUS\n"+jugadorFocus;

return data;

}

public void imprimirMazo(){

for (Carta carta : mazo) {

System.out.println(carta.getValor()+" "+carta.getTipo()+" "+carta.getColor());

}

}

public void imprimirPozo(){

System.out.println(pozo.getValor()+" "+pozo.getTipo()+" "+pozo.getColor());

}

public static void main(String[] args) {

Persistencia persistencia = new Persistencia();

persistencia.leerArchivo();

persistencia.agregarData();

}//fin main

}

Test Cases

package GUI;

import Juego.JuegoNormal;

import Models.Carta;

import Models.Jugador;

import Models.Mazo;

import org.junit.jupiter.api.AfterEach;

import org.junit.jupiter.api.Assertions;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import javax.swing.\*;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

import Juego.JuegoNormal;

import Models.Carta;

import Models.Jugador;

import Models.Mazo;

import java.security.InvalidParameterException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import static org.junit.Assert.assertEquals;

import static org.junit.Assert.assertFalse;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

class AlexsTest {

Jugador jugador;

Jugador j1;

Mazo mazo;

JuegoNormal juego;

ArrayList<Mazo> listaMazos;

Carta carta;

Carta c;

List<Jugador>listaJugadores;

@BeforeEach

void setUp1() {

j1 = new Jugador ("Alex", "al");

listaJugadores = new ArrayList<>();

listaJugadores.addAll(Arrays.asList(j1));

}

/\*Se verifica que una vez que la mano sea una mano vacia, el jugador sea borrado de la lista de jugadores\*/

@Test

void testCheckManoJug() {

ArrayList<Carta> manoCartas = new ArrayList<>();

manoCartas.clear();

assertFalse(JuegoNormal.listaJugadores.isEmpty());

}

@BeforeEach

void setUp2() {

jugador = new Jugador("Alex","alex");

}

@Test

void crearJugador() throws Exception {

assertEquals("Alex", jugador.getNombre());

assertNotEquals("alex", jugador.getNombre());

assertEquals("alex", jugador.getClave());

assertNotEquals("Alex", jugador.getClave());

}

@BeforeEach

void setUp3(){

c = new Carta ("verde", "número", "3");

}

@Test

void testVerTipo() {

assertEquals(5, juego.verValor(c));

}

@BeforeEach

void setUp4() {

carta = new Carta ("verde", "número", "3");

}

@Test

void testVerValor() {

assertEquals(3,juego.verValor(carta));

}

/\*@BeforeEach

void setUp5() throws Exception {

c = new Carta ("verde", "especial", "skip");

}

@Test

void testVerificarEspecial() {

assertFalse(juego.aplicarCartaEspecial(c),juego.verificarEspecial(c));

}\*/

}

package GUI;

import Juego.JuegoNormal;

import Models.Carta;

import Models.Jugador;

import Models.Mazo;

import org.junit.jupiter.api.AfterEach;

import org.junit.jupiter.api.Assertions;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import javax.swing.\*;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import static org.junit.jupiter.api.Assertions.\*;

class VistaJugadorTest {

Mazo mazo;

JuegoNormal juego;

ArrayList<Mazo> listaMazos;

Carta carta;

Carta cartaEspecial;

Carta skip;

Carta mas4;

Carta spin;

Carta cambioColor;

List<Jugador> listaJugadores;

Jugador jugador1;

Jugador jugador2;

Jugador jugador3;

VistaJugador vistaJugador;

@BeforeEach

void setUp() {

mazo = new Mazo();

mazo.llenarMazo();

juego = new JuegoNormal();

listaMazos = new ArrayList<>();

juego.setRondaHoraria(true);

carta = new Carta("rojo", "numero", "9");

cartaEspecial = new Carta("verde", "especial", "+2");

skip = new Carta("azul", "especial", "skip");

mas4 = new Carta("joker", "especial", "+4");

spin = new Carta("amarillo", "especial", "spin");

cambioColor = new Carta("joker", "especial", "color");

juego.setIsValid(true);

listaJugadores = new ArrayList<>();

jugador1 = new Jugador("jugador 1", "123");

jugador2 = new Jugador("jugador 2", "456");

jugador3 = new Jugador("jugador 3", "789");

listaJugadores.addAll(Arrays.asList(jugador1, jugador2, jugador3));

juego.llenarMazos();

jugador1.setManoCartas(juego.generarMano());

jugador2.setManoCartas(juego.generarMano());

jugador3.setManoCartas(juego.generarMano());

juego.getListaJugadores().addAll(listaJugadores);

juego.setJugadorFocus(0);

juego.setPozo(carta);

vistaJugador = new VistaJugador();

}

@AfterEach

void tearDown() {

}

@Test

void incrementarFocusTest(){

Assertions.assertEquals(0, juego.getJugadorFocus());

vistaJugador.incrementFocus();

Assertions.assertEquals(1, juego.getJugadorFocus());

}

@Test

void asignarNombreTest(){

vistaJugador.asignarNombre();

Assertions.assertEquals("jugador 1", vistaJugador.getLblNombre().getText());

vistaJugador.incrementFocus();

vistaJugador.asignarNombre();

Assertions.assertEquals("jugador 2", vistaJugador.getLblNombre().getText());

}

@Test

void asignarPozoTest (){

vistaJugador.asignarPozo(carta);

Assertions.assertEquals(carta, juego.getPozo());

Assertions.assertEquals(carta.getColor(), juego.getPozo().getColor());

Assertions.assertEquals(carta.getValor(), juego.getPozo().getValor());

Assertions.assertEquals(carta.getTipo(), juego.getPozo().getTipo());

}

@Test

void verificarEspecialTest(){

Assertions.assertEquals(0, juego.getJugadorFocus());

vistaJugador.verificarEspecial(cartaEspecial);

Assertions.assertEquals(2, juego.getJugadorFocus());

Assertions.assertEquals(9, jugador2.getManoCartas().size());

vistaJugador.verificarEspecial(skip);

Assertions.assertEquals(1, juego.getJugadorFocus());

vistaJugador.verificarEspecial(spin);

Assertions.assertEquals(false, juego.isRondaHoraria());

Assertions.assertEquals(0, juego.getJugadorFocus());

vistaJugador.verificarEspecial(cambioColor);

Assertions.assertEquals(2, juego.getJugadorFocus());

Assertions.assertEquals("verde", juego.getPozo().getColor());

vistaJugador.verificarEspecial(mas4);

Assertions.assertEquals(0, juego.getJugadorFocus());

Assertions.assertEquals(13, jugador2.getManoCartas().size());

Assertions.assertEquals("azul", juego.getPozo().getColor());

}

@Test

void verValorTest (){

Assertions.assertEquals(4, vistaJugador.verValor(skip));

Assertions.assertEquals(7, vistaJugador.verValor(spin));

Assertions.assertEquals(10, vistaJugador.verValor(cambioColor));

Assertions.assertEquals(8, vistaJugador.verValor(cartaEspecial));

Assertions.assertEquals(6, vistaJugador.verValor(mas4));

Assertions.assertEquals(9, vistaJugador.verValor(carta));

}

@Test

void verTipoTest(){

Assertions.assertEquals(10, vistaJugador.verTipo(mas4));

Assertions.assertEquals(10, vistaJugador.verTipo(cartaEspecial));

Assertions.assertEquals(10, vistaJugador.verTipo(skip));

Assertions.assertEquals(10, vistaJugador.verTipo(spin));

Assertions.assertEquals(10, vistaJugador.verTipo(cambioColor));

Assertions.assertEquals(5, vistaJugador.verTipo(carta));

}

@Test

void verColorTest(){

Assertions.assertEquals(15, vistaJugador.verColor(carta));

Assertions.assertEquals(20, vistaJugador.verColor(skip));

Assertions.assertEquals(10, vistaJugador.verColor(cartaEspecial));

Assertions.assertEquals(25, vistaJugador.verColor(spin));

Assertions.assertEquals(5, vistaJugador.verColor(mas4));

Assertions.assertEquals(5, vistaJugador.verColor(cambioColor));

}

@Test

void checkCantidadCartasTest(){

juego.getListaMazos().get(0).removerCartas(mazo.getMazoPrincipal());

Assertions.assertEquals(0, juego.getListaMazos().get(0).getMazoPrincipal().size());

vistaJugador.checkCantidadCartas(mas4);

Assertions.assertEquals(108, juego.getListaMazos().get(0).getMazoPrincipal().size());

}

@Test

void asignarValoresTest(){

ArrayList<Carta> lista = new ArrayList<>(Arrays.asList(carta, cartaEspecial));

vistaJugador.asignarValores(lista);

Assertions.assertEquals(899, carta.getCodigo());

Assertions.assertEquals(868, cartaEspecial.getCodigo());

}

@Test

void cambioColorTest(){

vistaJugador.cambioColor();

Assertions.assertEquals("amarillo", juego.getPozo().getColor());

Assertions.assertEquals(1, juego.getJugadorFocus());

}

@Test

void checkManoJugTest(){

Jugador jugador4 = new Jugador("jugador 4", "123");

juego.getListaJugadores().add(jugador4);

vistaJugador.checkManoJug(jugador4);

Assertions.assertEquals(3, juego.getListaJugadores().size());

Assertions.assertEquals(1, juego.getListaGanadores().size());

}

@Test

void checkPerderTest(){

juego.getListaJugadores().remove(jugador1);

juego.getListaJugadores().remove(jugador2);

Assertions.assertTrue(vistaJugador.checkPerder());

juego.getListaJugadores().add(jugador2);

Assertions.assertFalse(vistaJugador.checkPerder());

}

@Test

void validarTiroTest(){

Carta carta2 = new Carta("rojo", "numero", "7" );

jugador1.getManoCartas().add(carta2);

Assertions.assertTrue(vistaJugador.validarTiro("7 rojo numero"));

Assertions.assertEquals(carta2, juego.getPozo());

Assertions.assertFalse(vistaJugador.validarTiro("9 amarillo numero"));

}

@Test

void checkMazoVacioTest(){

juego.getListaMazos().get(0).removerCartas(mazo.getMazoPrincipal());

Assertions.assertEquals(0, juego.getListaMazos().get(0).getMazoPrincipal().size());

juego.checkMazoVacio();

Assertions.assertEquals(108, juego.getListaMazos().get(0).getMazoPrincipal().size());

}

@Test

void rellenarMazoTest(){

juego.getListaMazos().get(0).removerCartas(mazo.getMazoPrincipal());

Assertions.assertEquals(0, juego.getListaMazos().get(0).getMazoPrincipal().size());

juego.rellenarMazo();

Assertions.assertEquals(108, juego.getListaMazos().get(0).getMazoPrincipal().size());

}

@Test

void levantarCartaMazoTest(){

Jugador jugador4 = new Jugador("jugador 4", "123");

Carta carta2 = new Carta("amarillo", "numero", "3" );

jugador4.getManoCartas().add(carta2);

vistaJugador.levantarCartaMazo(jugador4);

assertEquals(2, jugador4.getManoCartas().size());

}

}