

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		PRÁCTICA DE LABORATORIO	
CARRERA: Computación		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:	02	TÍTULO PRÁCTICA: Examen Final	
OBJETIVO ALCANZADO: <ul style="list-style-type: none"> • Consolidar los conocimientos adquiridos en clase sobre Java. • Identificar las sentencias SQL. • Diseñar e Implementar códigos DDL, DML. 			
ACTIVIDADES DESARROLLADAS			
1. Enunciado para el planteamiento del problema. <p>Se desea simular los posibles beneficios de diversas estrategias de juego en un casino. La ruleta francesa es un juego en el que hay una ruleta con 37 números (del 0 al 36). Cada 2000 (tiempo parametrizable) milisegundos el croupier saca un número al azar y los diversos hilos clientes apuestan para ver si ganan. Todos los hilos empiezan con 1.000 euros y la banca (que controla la ruleta) con 50.000. Cuando los jugadores pierden dinero, la banca incrementa su saldo.</p> <ul style="list-style-type: none"> • Se puede jugar a un número concreto. Habrá 4 hilos clientes que eligen números al azar del 1 al 36 (no el 0) y restarán 10 euros de su saldo para apostar a ese ese número. Si sale su número su saldo se incrementa en 360 euros (36 veces lo apostado). • Se puede jugar a par/impar. Habrá 4 hilos clientes que eligen al azar si apuestan a que saldrá un número par o un número impar. Siempre restan 10 euros para apostar y si ganan incrementan su saldo en 20 euros. • Se puede jugar a la «martingala». Habrá 4 hilos que eligen números al azar. Elegirán un número y empezarán restando 10 euros de su saldo para apostar a ese número. Si ganan incrementan su saldo en 360 euros. Si pierden jugarán el doble de su apuesta anterior (es decir, 20, luego 40, luego 80, y así sucesivamente) • La banca acepta todas las apuestas, pero nunca paga más dinero del que tiene. • Si sale el 0, todo el mundo pierde y la banca se queda con todo el dinero. <p>Adicionalmente, se deberá generar un sistema de base de datos con JPA en donde puede gestionar a los clientes o hilos jugadores, con cada una de las apuestas realizadas, los valores que se están manejando tanto de la banca como de cada cliente y gestionar la simulación es decir se puede iniciar y parar en cualquier intervalo de tiempo en la simulación, además de poder cambiar a cualquier cliente con un nuevo o un anterior y en que modalidad va a jugar. Por otro lado, es parametrizable el tiempo que se demora dar la vuelta a la ruleta con el proceso de apuesta.</p> <p>Es importante destacar que debe existir un sistema de simulación visual y un sistema de gestión de jugadores, transacciones y apuestas en donde se evidencia la apuesta, el jugador, la ruleta el numero generado y como varían los saldos de los que intervienen dentro del juego.</p> <p>Por último, se debe presentar dos reportes o tablas de los datos:</p> <ol style="list-style-type: none"> 1. Clientes y la banca con el número de transacciones o apuestas realizadas, el valor de total y cuantas a perdido y cuantas veces ha ganado además de la modalidad de juego. 2. Dentro de cada cliente se puede acceder al historial de apuestas y transacciones realizadas. 			

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

2. JPA en el proyecto.

Clase JPAUtils

```
package ec.edu.ups.utils;

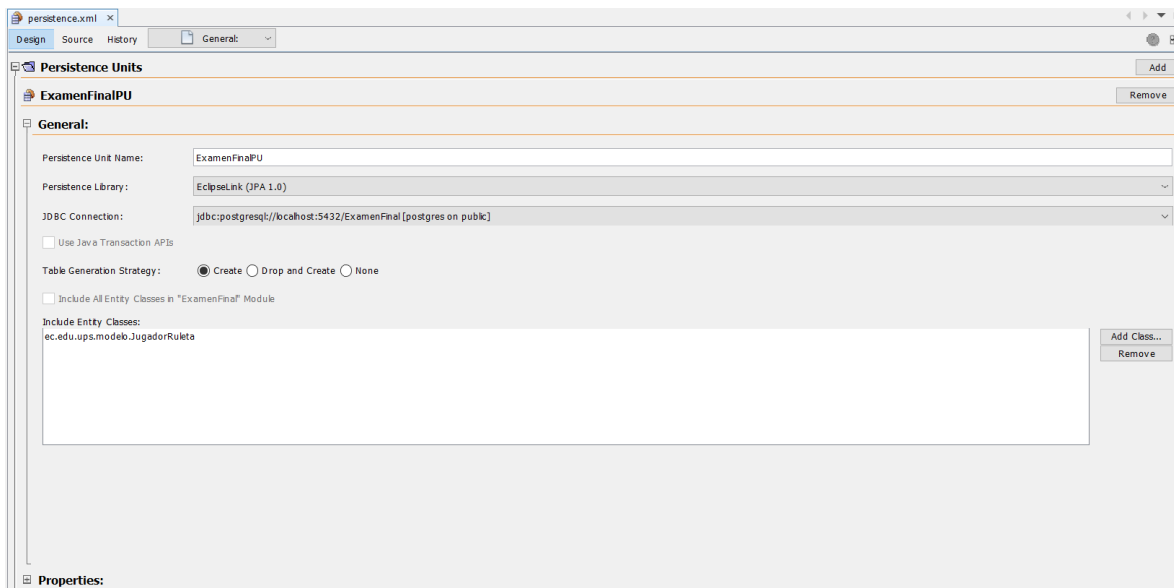
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author user
 */
public class JPAUtils {

    private static final EntityManagerFactory emf =
Persistence.createEntityManagerFactory("ExamenFinalPU");

    public static EntityManager getEntityManager() {
        return emf.createEntityManager();
    }

}
```




3. Hilos en el proyecto.

Ruleta

```
package ec.edu.ups.Hilos;

import ec.edu.ups.controlador.controladorJugador;
import ec.edu.ups.modelo.JugadorRuleta;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JTextArea;
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

import java.util.concurrent.ThreadLocalRandom;
import javax.swing.JLabel;

/**
 *
 * @author user
 */
public class Ruleta implements Runnable {

    private ArrayList<JugadorRuleta> jugadores;
    private String juego;
    private JTextArea descripcion;
    private int segundos;
    private controladorJugador control;
    private boolean iterar = true;
    private JugadorRuleta j;

    private JLabel saldoBanca;

    private JLabel numeroB;

    public Ruleta(List<JugadorRuleta> jugadores, int segundos, JTextArea
descripcion, JLabel saldoBanca, JLabel numeroB, controladorJugador control, String
juego) {
        this.jugadores = (ArrayList<JugadorRuleta>) jugadores;
        this.segundos = segundos;
        this.descripcion = descripcion;
        this.control = control;
        this.juego = juego;
        this.saldoBanca = saldoBanca;
        this.numeroB = numeroB;
    }

    int numeroBanca = 0;

    @Override
    public void run() {

        while (iterar) {

            numeroBanca = RandomNumber();
            restarSaldoJugador();
            tiempoEsperar(segundos);
            if (numeroBanca == 0) {

            }
            switch (juego) {
                case "concreto":
                    jugadores.stream().map(jugador -> {
                        if (numeroBanca == jugador.getNumero()) {
                            j = jugador;
                            j.setSaldo(jugador.getSaldo() + (360));
                            j.setnWins(jugador.getnWins() + 1);
                        }
                    });
            }
        }
    }
}

```

```

        descripcion.setText(jugador.getNombre() + "Gana 360
euros");
        saldoBanca.setText("" +
(Float.parseFloat(saldoBanca.getText() + "") - 360));
        tiempoEsperar(segundos);
    }
    return jugador;
}).filter(jugador -> (numeroBanca !=
jugador.getNumero())).map(jugador -> {
    j = jugador;
    j.setnLost(jugador.getnLost() + 1);
    descripcion.setText(jugador.getNombre() + "Pierde");
    return jugador;
}).forEachOrdered((JugadorRuleta _item) -> {
    tiempoEsperar(segundos);
    iterar = false;
});
control.actualizar(j);
break;

case "parImpar":
    jugadores.forEach(jugador -> {
        if (parImpar(numeroBanca) == jugador.getIsPar()) {
            j = jugador;
            j.setSaldo(jugador.getSaldo() + 20);
            j.setnWins(jugador.getnWins() + 1);
            descripcion.setText(jugador.getNombre() + "Gana 20
euros");
            saldoBanca.setText("" +
(Float.parseFloat(saldoBanca.getText() + "") - 20));
            tiempoEsperar(segundos);
        } else {
            j = jugador;
            j.setnLost(jugador.getnLost() + 1);
            descripcion.setText(jugador.getNombre() + "Pierde");
            tiempoEsperar(segundos);
            iterar = false;
        }
    });
    control.actualizar(j);
    break;

case "martingala":
    jugadores.stream().map(jugador -> {
        if (numeroBanca == jugador.getNumero()) {
            j = jugador;
            j.setSaldo(jugador.getSaldo() + (360));
            j.setnWins(jugador.getnWins() + 1);
            descripcion.setText(jugador.getNombre() + "Gana 360
euros");
            saldoBanca.setText("" +
(Float.parseFloat(saldoBanca.getText() + "") - 360));
        }
    })
    return jugador;

```

```

        })).filter(jugador -> (numeroBanca !=
jugador.getNumero()))).forEachOrdered(jugador -> {
            j = jugador;
            j.setnLost(jugador.getnLost() + 1);
            j.setIsDuplicar(true);
            descripcion.setText(jugador.getNombre() + "Pierde y se
duplica apuesta");

            tiempoEsperar(segundos);
            iterar = false;
        });
        control.actualizar(j);

        break;

    default:
        break;
}

}

}

public void reanudar() {
    iterar = true;
}

public boolean isIterar() {
    return iterar;
}

public void setIterar(boolean iterar) {
    this.iterar = iterar;
}

private void tiempoEsperar(int segundos) {

    segundos = segundos * 1000;
    try {
        descripcion.setText("Esperando..... " + segundos);
        Thread.sleep(segundos);


    } catch (InterruptedException e) {

    }

}

private void restarSaldoJugador() {
    for (JugadorRuleta jugador : jugadores) {
        if (jugador.isIsDuplicar()) {
            JugadorRuleta j = jugador;
            j.setSaldo(jugador.getSaldo() - (jugador.getCantidadApuesta()) * 2);
            j.setCantidadApuesta((jugador.getCantidadApuesta()) * 2);
            j.setnApuestas(jugador.getnApuestas() + 1);
            tiempoEsperar(segundos);
            //control.actualizar(j);

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

    } else {

        descripcion.setText(jugador.getNombre() + " apuesta 10 euros al
numero " + jugador.getNumero() + "\n");
        JugadorRuleta j = jugador;
        j.setSaldo(jugador.getSaldo() - 10);
        j.setCantidadApuesta(jugador.getCantidadApuesta() + 10);
        j.setnApuestas(jugador.getnApuestas() + 1);
        tiempoEsperar(segundos);
        // control.actualizar(j);
    }

}

private int RandomNumber() {
    int numero = ThreadLocalRandom.current().nextInt(0, 36 + 1);
    numeroB.setText("" + numero);
    return numero;
}

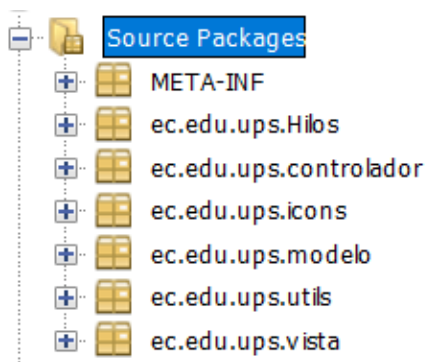
private int parImpar(int numero) {
    if (numero % 2 == 0) {
        return 2;
    } else {
        return 1;
    }
}

public void stop() {
    iterar = false;
}
}

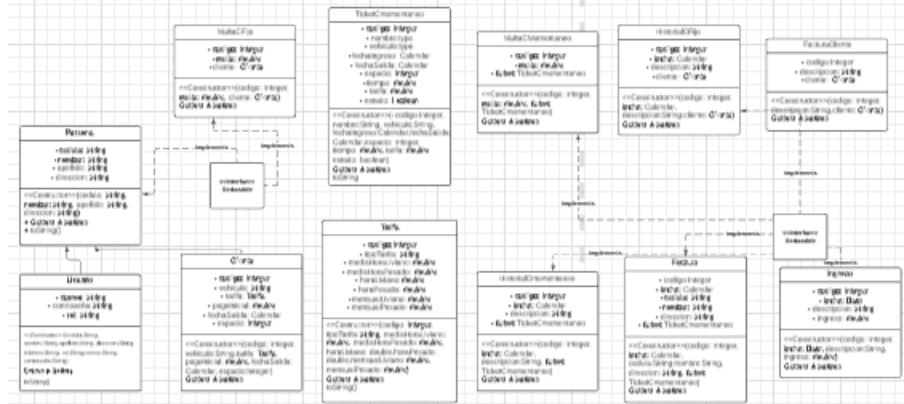
```

4. Excepciones en el proyecto.

5. MVC en el proyecto.



6. Diagrama de clases en el proyecto.



7. Usabilidad – Vista – Simulación:

Abstract Controlador

```
package ec.edu.ups.controlador;


import ec.edu.ups.utils.JPAUtils;
import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.persistence.EntityManager;

/**
 *
 * @author user
 * @param <E>
 */
public abstract class AbstractControlador<E> {

    private List<E> lista;
    private Class<E> clase;
    public EntityManager em;

    /**
     *
     */
    public AbstractControlador() {
        lista = new ArrayList<>();
        Type t = getClass().getGenericSuperclass();
        ParameterizedType pt = (ParameterizedType) t;
        clase = (Class) pt.getActualTypeArguments()[0];
        em = JPAUtils.getEntityManager();
    }

    public AbstractControlador(EntityManager em) {
        lista = new ArrayList<>();
        Type t = getClass().getGenericSuperclass();
    }
}
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

ParameterizedType pt = (ParameterizedType) t;
this.clase = (Class) pt.getActualTypeArguments()[0];
this.em = em;
}

public boolean crear(E objeto) {
    try {
        if (this.validar(objeto)) {
            em.getTransaction().begin();
            em.persist(objeto);
            em.getTransaction().commit();
            lista.add(objeto);
            return true;
        }
    } catch (Exception ex) {
        Logger.getLogger(AbstractControlador.class.getName()).log(Level.SEVERE,
null, ex);
    }

    return false;
}

public boolean eliminar(E objeto) {
    em.getTransaction().begin();
    em.remove(em.merge(objeto));
    em.getTransaction().commit();
    lista.remove(objeto);
    return true;
}

public boolean actualizar(E objeto) {
    try {
        if (this.validar(objeto)) {
            em.getTransaction().begin();
            objeto = em.merge(objeto);
            em.getTransaction().commit();
            this.lista = buscarTodo();
            return true;
        }
    } catch (Exception ex) {
        Logger.getLogger(AbstractControlador.class.getName()).log(Level.SEVERE,
null, ex);
    }

    return false;
}

public E buscar(Object id) {
    return (E) em.find(clase, id);
}

public List<E> buscarTodo() {

```



```
        return em.createQuery("Select t from " + clase.getSimpleName() + "
t").getResultList();
    }

    public abstract boolean validar(E objeto) throws Exception;

    public List<E> getLista() {
        return lista;
    }

    public void setLista(List<E> lista) {
        this.lista = lista;
    }

    public Class<E> getClase() {
        return clase;
    }

    public void setClase(Class<E> clase) {
        this.clase = clase;
    }

    public EntityManager getEm() {
        return em;
    }

    public void setEm(EntityManager em) {
        this.em = em;
    }
}
```


Jugador Ruleta

```
package ec.edu.ups.modelo;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

/**
 *
 * @author paul_
 */
@Entity
public class JugadorRuleta implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

private Integer id;

@Column(name = "nombre")
private String nombre;

@Column(name = "numero")
private int numero; //el número de la ruleta o 0/1

@Column(name = "saldo")
private float saldo;

@Column(name = "cantidadApuesta")
private int cantidadApuesta;

@Column(name = "isPar")
private int isPar; //Se verifica si se escogió par caso contrario, impar

@Column(name = "nApuestas")
private int nApuestas; //numero de apuestas

@Column(name = "nGanadas")
private int nWins; //numero de apuestas ganadas

@Column(name = "nPerdidas")
private int nLost; //numero de apuestas perdidas

@Column(name = "dApuesta")
private boolean isDuplicar; //Se verifica si se escogió 'martingala' y se
duplican sus proximas apuestas

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}


public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public int getNumero() {
    return numero;
}

public void setNumero(int numero) {
    this.numero = numero;
}

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

public float getSaldo() {
    return saldo;
}

public void setSaldo(float saldo) {
    this.saldo = saldo;
}

public int getCantidadApuesta() {
    return cantidadApuesta;
}

public void setCantidadApuesta(int cantidadApuesta) {
    this.cantidadApuesta = cantidadApuesta;
}

public int getIsPar() {
    return isPar;
}

public void setIsPar(int isPar) {
    this.isPar = isPar;
}

public int getnApuestas() {
    return nApuestas;
}

public void setnApuestas(int nApuestas) {
    this.nApuestas = nApuestas;
}

public int getnWins() {
    return nWins;
}

public void setnWins(int nWins) {
    this.nWins = nWins;
}

public int getnLost() {
    return nLost;
}

public void setnLost(int nLost) {
    this.nLost = nLost;
}

public boolean isIsDuplicar() {
    return isDuplicar;
}

public void setIsDuplicar(boolean isDuplicar) {
    this.isDuplicar = isDuplicar;
}

```

```
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not
set
    if (!(object instanceof JugadorRuleta)) {
        return false;
    }
    JugadorRuleta other = (JugadorRuleta) object;
    if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "ec.edu.ups.modelo.JugadorRuleta[ id=" + id + " ]";
}

}

Vista Principal

package ec.edu.ups.vista;


import ec.edu.ups.controlador.ControladorJugador;

/**
 *
 * @author paul_
 */
public class VentanaPrincipal extends javax.swing.JFrame {

    RegistroJugador ventanaJugador;
    InformeJugador ventanaHistorial;
    JuegoRuleta juegoRuleta;

    ControladorJugador controladorJugador;

    public VentanaPrincipal() {
        initComponents();
        controladorJugador = new ControladorJugador();
        ventanaJugador = new RegistroJugador(controladorJugador);
        ventanaHistorial = new InformeJugador(controladorJugador);
        juegoRuleta = new JuegoRuleta(controladorJugador);
    }
}
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

    }
    private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void openMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        desktopPane.add(ventanaJugador);
        ventanaJugador.setVisible(true);
    }

    private void saveAsMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        desktopPane.add(juegoRuleta);
        juegoRuleta.setVisible(true);
    }

    private void saveMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        desktopPane.add(ventanaHistorial);
        ventanaHistorial.setVisible(true);
    }


    // Variables declaration - do not modify
    private javax.swing.JDesktopPane desktopPane;
    private javax.swing.JMenuItem exitMenuItem;
    private javax.swing.JMenu fileMenu;
    private javax.swing.JMenuBar menuBar;
    private javax.swing.JMenuItem openMenuItem;
    private javax.swing.JMenuItem saveAsMenuItem;
    private javax.swing.JMenuItem saveMenuItem;
    // End of variables declaration
}

```

Play al juego



```
package ec.edu.ups.vista;
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

import ec.edu.ups.controlador.ControladorJugador;
import ec.edu.ups.modelo.JugadorRuleta;
import javax.swing.JOptionPane;

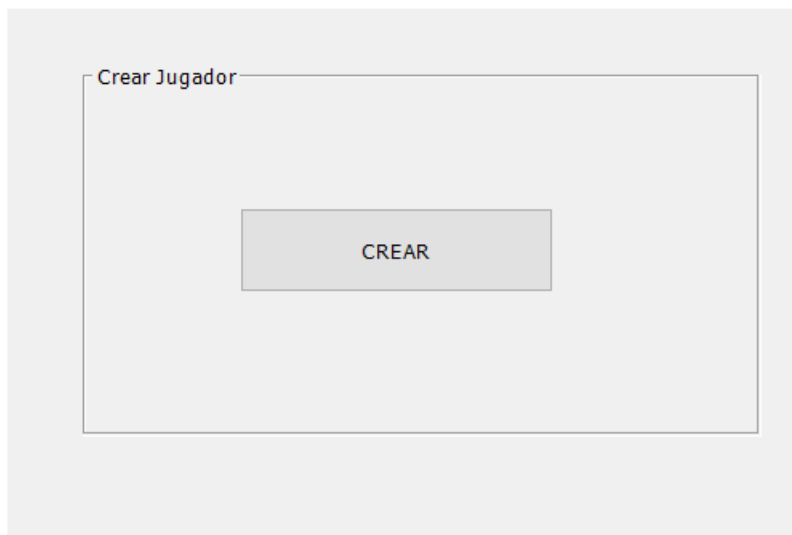
/**
 *
 * @author paul_
 */
public class RegistroJugador extends javax.swing.JInternalFrame {

    ControladorJugador controladorJugador;

    public RegistroJugador(ControladorJugador controladorJugador) {
        initComponents();
        this.controladorJugador = controladorJugador;
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        JugadorRuleta j = new JugadorRuleta();
        String nombre = JOptionPane.showInputDialog(this, "Ingrese un nombre para el
jugador");
        j.setNombre(nombre);
        j.setSaldo(1000);
        controladorJugador.crear(j);
        JOptionPane.showMessageDialog(this, "Cliente creado exitosamente");
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration
}

```



Juego Ruleta

```
package ec.edu.ups.vista;

import ec.edu.ups.Hilos.Ruleta;
import ec.edu.ups.controlador.ControladorJugador;
import ec.edu.ups.modelo.JugadorRuleta;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;

/**
 *
 * @author paul_
 */
public class JuegoRuleta extends javax.swing.JInternalFrame {

    private ControladorJugador controladorJugador;
    private List<JugadorRuleta> jugadores;//Lista temporal
    private int segundos;
    private boolean isStop;
    private String isPause;

    public JuegoRuleta(ControladorJugador controladorJugador) {
        initComponents();
        this.controladorJugador = controladorJugador;
        jugadores = new ArrayList<>();
        segundos = 0;
        isStop = false;
        isPause = "verdad";
    }

    public int numList() {
        int cont = 0;
        for (JugadorRuleta r : jugadores) {
            cont++;
        }
        return cont;
    }


    private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {
        if (jugadores != null) {
            String id = JOptionPane.showInputDialog(this, "Ingrese id de jugador");

            String num = JOptionPane.showInputDialog(this, "Ingrese numero a apostar
1-36");

            JugadorRuleta j = new JugadorRuleta();

            if (Integer.parseInt(num) >= 1 && Integer.parseInt(num) <= 36) {
                j.setNumero(Integer.parseInt(num));
            }

            j = controladorJugador.buscar(Integer.parseInt(id.trim()));
            jugadores.add(j);
            //controladorJugador.actualizar(j);
        }
    }
}
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

JOptionPane.showMessageDialog(this, "Jugador: " + j.getNombre() + "
asignado exitosamente");
    }
}

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

    if (jugadores != null) {
        Ruleta[] vHilos = new Ruleta[numList()];
        for (int i = 0; i < numList(); i++) {

            vHilos[i] = new Ruleta(jugadores, segundos, txtArea, lblValorS,
lblValorBanca, controladorJugador,
cbxItemJuego.getSelectedItem().toString().trim());
            Thread hilo = new Thread(vHilos[i]);
            hilo.start();
            /* if(isStop){
hilo.stop();
}
if(isPause.equals("verdad")){
vHilos[i].stop();
}else if(isPause.equals("resume")){
vHilos[i].reanudar();
}*/

        }
    }
}


private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    String ms = JOptionPane.showInputDialog(this, "Ingrese tiempo de intervalo
entre procesos");
    segundos = Integer.parseInt(ms);
}

private void btnPauseActionPerformed(java.awt.event.ActionEvent evt) {
    isPause = "resume";
}

private void btnStopActionPerformed(java.awt.event.ActionEvent evt) {
    isStop = true;
}

// Variables declaration - do not modify
private javax.swing.JButton btnBuscar;
private javax.swing.JButton btnPause;
private javax.swing.JButton btnStart;
private javax.swing.JButton btnStop;
private javax.swing.JComboBox<String> cbxItemJuego;
private javax.swing.JButton jButton4;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JLabel lblSaldo;
private javax.swing.JLabel lblSaldo1;

```


	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```
private javax.swing.JLabel lblValorBanca;
private javax.swing.JLabel lblValorS;
private javax.swing.JTextArea txtArea;
// End of variables declaration
}
```



Informe de Jugador

```
package ec.edu.ups.vista;

import java.util.Iterator;
import javax.swing.table.DefaultTableModel;
import ec.edu.ups.controlador.ControladorJugador;
import ec.edu.ups.modelo.JugadorRuleta;

/**
 *
 * @author paul_
 */
public class InformeJugador extends javax.swing.JInternalFrame {

    ControladorJugador controladorJugador;

    public InformeJugador(ControladorJugador controladorJugador) {
        initComponents();
        this.controladorJugador = controladorJugador;
    }

    public void limpiarTabla() {
        DefaultTableModel modelo = (DefaultTableModel) tblJugadores.getModel();
        modelo.setRowCount(0);
    }

    public void cargarJugadoresTbl() {
        DefaultTableModel modelo = (DefaultTableModel) tblJugadores.getModel();
        modelo.setRowCount(0);
        if (controladorJugador.buscarTodo() != null) {
            for (Iterator it = controladorJugador.buscarTodo().iterator();
            it.hasNext();) {
                JugadorRuleta jugador = (JugadorRuleta) it.next();
            }
        }
    }
}
```

```

        Object[] rowData = {jugador.getId(), jugador.getNombre(),
jugador.getnApuestas(), jugador.getSaldo(), jugador.getnWins(), jugador.getnLost()};
        modelo.addRow(rowData);
        tblJugadores.setModel(modelo);
    }
    } else {
        System.out.println("LISTA VACIA");
    }
}
private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    limpiarTabla();
}

private void btnMostrarTodosActionPerformed(java.awt.event.ActionEvent evt) {
    cargarJugadoresTbl();
}

// Variables declaration - do not modify
private javax.swing.JButton btnBuscar;
private javax.swing.JButton btnLimpiar;
private javax.swing.JButton btnMostrarTodos;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JLabel lblId;
private javax.swing.JTable tblJugadores;
private javax.swing.JTextField txtId;
// End of variables declaration
}

```


INFORME JUGADORES

Ingrese id de jugador:

ID	Participante	Numero apuestas	Saldo del participante	Veces ganadas	Veces perdidas

RESULTADO(S) OBTENIDO(S):

- Realizar procesos de Hilos en Java.
- Entender las aplicaciones de codificación de las nuevas características de concurrencia.
- Entender las funcionalidades de sincronización y manejo de grupo de Thread dentro de Java.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

CONCLUSIONES:

- Aprenden a trabajar en grupo dentro de plazos de tiempo establecidos, manejando el lenguaje de programación de Java y SQL.
- La aplicación de patrones de diseño y arquitecturas dentro de la aplicación creada son fundamentales y facilitan el trabajo a la vez que simplifican el código para que no venga a ser muy pesado y tenga una mayor aceptabilidad dentro de los ámbitos en los que se deseen proyectar su respectivo uso.

RECOMENDACIONES:

- Realizar el trabajo dentro del tiempo establecido.
- Para que una vista de interfaz sea agradable a un lector se debe realizar un estudio de los colores que agraden a los usuarios para que de esta manera se pueda tener una respuesta factible cuando el o los usuarios utilicen la aplicación y den su visto bueno.

Nombre de estudiante:

PAUL ALEXANDER GUAPUCAL CARDENAS

Firma de estudiante:

