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

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Universidad Politécnica Salesiana

Vicerrectorado Docente

Código del Formato:	GUIA-PRL-001
Versión:	VF1.0
Elaborado por:	Directores de Área del Conocimiento Integrantes Consejo Académico
Fecha de elaboración:	2016/04/01
Revisado por:	Consejo Académico
Fecha de revisión:	2016/04/06
Aprobado por:	Lauro Fernando Pesántez Avilés Vicerrector Docente
Fecha de aprobación:	2016/14/06
Nivel de confidencialidad:	Interno

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

Descripción General

Propósito


El propósito del presente documento es definir un estándar para elaborar documentación de guías de práctica de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana, con la finalidad de lograr una homogenización en la presentación de la información por parte del personal académico y técnico docente.

Alcance

El presente estándar será aplicado a toda la documentación referente a informes de prácticas de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana.

Formatos

- Formato de Guía de Práctica de Laboratorio / Talleres / Centros de Simulación – para Docentes
- Formato de Informe de Práctica de Laboratorio / Talleres / Centros de Simulación – para Estudiantes

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



FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES

CARRERA: Computación

ASIGNATURA: Programación Aplicada

NRO. PRÁCTICA:

1

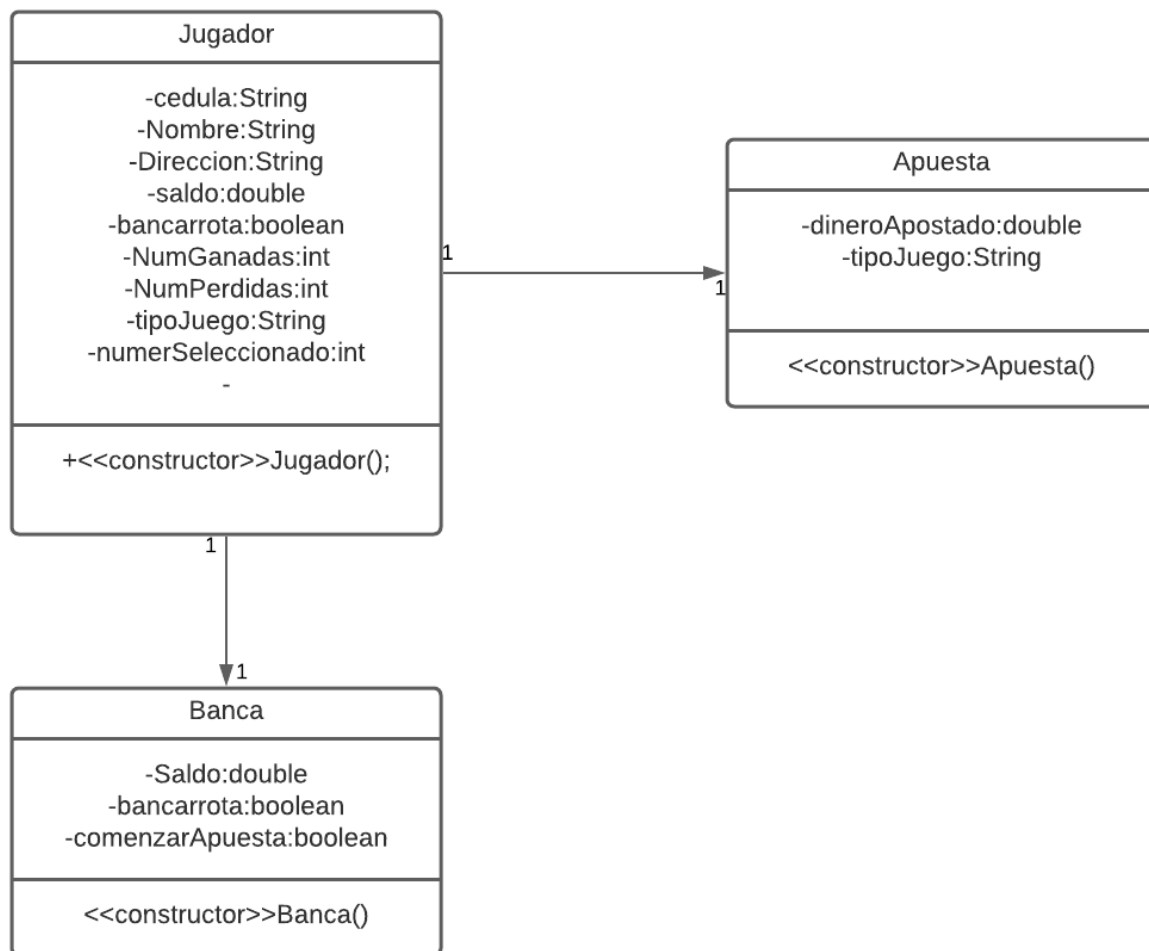
TÍTULO PRÁCTICA: Examen Interciclo


OBJETIVO ALCANZADO:

- Aplicar conocimientos adquiridos durante todo el segundo Interciclo
- Comprender el funcionamiento de hilos (Threads)

ACTIVIDADES DESARROLLADAS

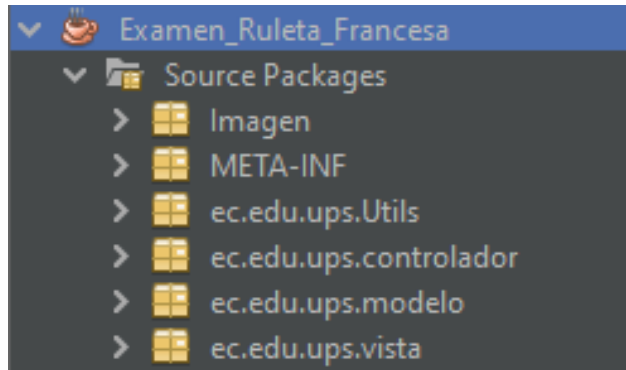
1. Diseñamos el diagrama UML



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

2. Creamos el proyecto con sus respectivos paquetes en los que se crearan las Entity Class, Clases controladores y las ventanas para la interfaz.

a.



3. En el paquete modelo creamos las entity class que serán las tablas de la base de datos, cada una con sus respectivas relaciones

a. Jugador


@Entity

```
public class Jugador implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @Column(name = "Cedula")
    private String cedula;
    @Column(name = "Nombre")
    private String nombre;
    @Column(name = "Direccion")
    private String Direccion;
    @Column(name = "Tipo_Juego")
    private String tipoJuego;
    @Column(name = "Numero_Seleccionado")
    private int numApost;
    @Column(name = "Saldo")
    private double saldo;
    @Column(name = "Bancarrota")
    private boolean bancarrota;
    @Column(name = "NumeroGanadas")
    private int nGanadas;
    @Column(name = "NumeroPerdidas")
    private int nPerdidas;
    @OneToOne
    @JoinColumn(name = "fk_Apuesta")
    private Apuestas apuesta ;
    @OneToOne
    @JoinColumn(name = "fk_Banca")
    private Banca banca ;

    public Jugador() {
    }

    public Jugador(String cedula, String nombre, String Direccion, double saldo,
```

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

```

boolean bancarrota) {
    this.cedula = cedula;
    this.nombre = nombre;
    this.Direccion = Direccion;
    this.saldo = saldo;
    this.bancarrota = bancarrota;
}

public Jugador(String cedula, String nombre, String Direccion, String tipoJuego,
int numSelec, double saldo, boolean bancarrota, int nGanadas, int nPerdidas, Apuestas
apuesta, Banca banca) {
    this.cedula = cedula;
    this.nombre = nombre;
    this.Direccion = Direccion;
    this.tipoJuego = tipoJuego;
    this.numApost = numSelec;
    this.saldo = saldo;
    this.bancarrota = bancarrota;
    this.nGanadas = nGanadas;
    this.nPerdidas = nPerdidas;
    this.apuesta = apuesta;
    this.banca = banca;
}

public Long getId() {
    return id;
}

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

public String getCedula() {
    return cedula;
}

public void setCedula(String cedula) {
    this.cedula = cedula;
}


public String getNombre() {
    return nombre;
}

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

public String getDireccion() {
    return Direccion;
}

public void setDireccion(String Direccion) {
    this.Direccion = Direccion;
}

```

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

```

}

public String getTipoJuego() {
    return tipoJuego;
}

public void setTipoJuego(String tipoJuego) {
    this.tipoJuego = tipoJuego;
}

public int getNumApost() {
    return numApost;
}

public void setNumApost(int numSelec) {
    this.numApost = numSelec;
}

public double getSaldo() {
    return saldo;
}

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

public boolean isBancarrota() {
    return bancarrota;
}

public void setBancarrota(boolean bancarrota) {
    this.bancarrota = bancarrota;
}

public int getnGanadas() {
    return nGanadas;
}


public void setnGanadas(int nGanadas) {
    this.nGanadas = nGanadas;
}

public int getnPerdidas() {
    return nPerdidas;
}

public void setnPerdidas(int nPerdidas) {
    this.nPerdidas = nPerdidas;
}

public Apuestas getApuesta() {
    return apuesta;
}

```

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

```

public void setApuesta(Apuestas apuesta) {
    this.apuesta = apuesta;
}

public Banca getBanca() {
    return banca;
}

public void setBanca(Banca banca) {
    this.banca = banca;
}

@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 Jugador)) {
        return false;
    }
    Jugador other = (Jugador) 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.Jugador[ id=" + id + " ]";
}

```


b. Banca

```

@Entity
public class Banca implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @Column(name = "Saldo")
    private double saldo;
    @Column(name = "Bancarrota")
    private boolean bancarrota;
    @Column(name = "Comenzar_Apuesta")
    private boolean comenzar_Apuesta;
}

```

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

```

public Banca() {
}

public Banca(double saldo, boolean bancarrota, boolean comenzar_Apuesta) {
    this.saldo = saldo;
    this.bancarrota = bancarrota;
    this.comenzar_Apuesta = comenzar_Apuesta;
}

public Banca(Long id, double saldo, boolean bancarrota, boolean comenzar_Apuesta)
{
    this.id = id;
    this.saldo = saldo;
    this.bancarrota = bancarrota;
    this.comenzar_Apuesta = comenzar_Apuesta;
}

public Long getId() {
    return id;
}

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

public double getSaldo() {
    return saldo;
}

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

public boolean isBancarrota() {
    return bancarrota;
}


public void setBancarrota(boolean bancarrota) {
    this.bancarrota = bancarrota;
}

public boolean isComenzar_Apuesta() {
    return comenzar_Apuesta;
}

public void setComenzar_Apuesta(boolean comenzar_Apuesta) {
    this.comenzar_Apuesta = comenzar_Apuesta;
}

// public Jugador getJugador() {
//     return jugador;
// }
//

```


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

```
// public void setJugador(Jugador jugador) {
//     this.jugador = jugador;
// }

@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 Banca)) {
        return false;
    }
    Banca other = (Banca) 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.Banca[ id=" + id + " ]";
}

}
```

c. Apuestas

```
@Entity
public class Apuestas implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @Column(name = "Dinero_Apostado")
    private double dinero_Apostado;
    @Column(name = "Tipo_Juego")
    private String tipo_Juego;

    public Apuestas() {
    }

    public Apuestas(double dinero_Apostado, String tipo_Juego) {
        this.dinero_Apostado = dinero_Apostado;
        this.tipo_Juego = tipo_Juego;
    }

    public Long getId() {
```

```
        return id;
    }

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

    public double getDinero_Apostado() {
        return dinero_Apostado;
    }

    public void setDinero_Apostado(double dinero_Apostado) {
        this.dinero_Apostado = dinero_Apostado;
    }

    public String getTipo_Juego() {
        return tipo_Juego;
    }


    public void setTipo_Juego(String tipo_Juego) {
        this.tipo_Juego = tipo_Juego;
    }

    // public Jugador getJugador() {
    //     return jugador;
    // }
    // public void setJugador(Jugador jugador) {
    //     this.jugador = jugador;
    // }

    @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 Apuestas)) {
            return false;
        }
        Apuestas other = (Apuestas) object;
        if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
```

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

```

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

}

```

4. Las clases controladores nos ayudara para crear al usuario, la apuesta que realiza y que se registre en la basa de datos y la banca en la que se encuentra jugando.

a. ControladorAbstract: Sera la clase padre, tendrá los métodos para crear al jugador, registrar la apuesta realizada, etc.

b. ControladorJugador

```

public abstract class ControladorAbstract<T> {
    private List<T> listaObjetos;
    private Class<T> clase;
    private EntityManager em;

    ///
    public ControladorAbstract() {
        listaObjetos = new ArrayList<>();
        Type t = getClass().getGenericSuperclass();
        ParameterizedType pt = (ParameterizedType) t;
        clase = (Class) pt.getActualTypeArguments()[0];
        em = JAPUtils.getEntityManager();
    }

    public ControladorAbstract(EntityManager em) {
        listaObjetos = new ArrayList<>();
        Type t = getClass().getGenericSuperclass();
        ParameterizedType pt = (ParameterizedType) t;
        clase = (Class) pt.getActualTypeArguments()[0];
        this.em = em;
    }

    public T create(T objeto){
        em.getTransaction().begin();
        em.persist(objeto);
        em.getTransaction().commit();
        listaObjetos.add(objeto);
        return objeto;
    }

    // public boolean delete(T objeto){
    //     em.getTransaction().begin();
    //     em.remove(em.merge(objeto));
    //     em.getTransaction().commit();
    //     listaObjetos.remove(objeto);
    //     return true;
    // }

    public T update(T objeto){
        em.getTransaction().begin();
        objeto = em.merge(objeto);
        em.getTransaction().commit();
        this.findAll();
    }
}

```

```
        return objeto;
    }

    public T read(Object id){
        return (T) em.find(clase, id);
    }

    public List<T> findAll(){
        return em.createQuery("Select t from " + clase.getSimpleName() + "
t").getResultList();
    }

    public List<T> getListaObjetos() {
        return listaObjetos;
    }

    public void setListaObjetos(List<T> listaObjetos) {
        this.listaObjetos = listaObjetos;
    }

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

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

    public EntityManager getEm() {
        return em;
    }

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

    }

    c. ControladorApuesta
    public class ControladorJugador extends ControladorAbstract<Jugador>{
        private DefaultTableModel modelo2;
        private JTable tablaJugadores;
        private int nS;
        private List<Jugador> listaJ;
        private JTextField NumS;
        private JTextArea NumA;
        private VntRuleta vntR;

        public ControladorJugador() {
            super();
            modelo2 = new DefaultTableModel();
        }

        public ControladorJugador(JTable tablaJugadores, List<Jugador> listaJ, JTextField
```

```

NumS, JTextArea NumA,VntRuleta vntR) {
    super();
    modelo2 = new DefaultTableModel();
    this.tablaJugadores = tablaJugadores;
    this.nS = nS;
    this.listaJ = listaJ;
    this.NumS = NumS;
    this.NumA = NumA;
    this.vntR = vntR;
    this.jugar(listaJ, NumS, NumA);
    this.llenarTbl(listaJ);
    //this.jugar(nS, listaJ, NumS);
}

public ControladorJugador( JTable tablaJugadores) {
    super();
    modelo2 = new DefaultTableModel();
    this.tablaJugadores = tablaJugadores;
}

// @Override
// public void run() {
//     while (true) {
//         llenarTbl(listaJ);
//         //vntRuleta.llenarTabla(findAll());
//     }
// }

public synchronized void jugar(List<Jugador> listaJ, JTextField txtNum, JTextArea
NumA) {
    int numSeleccionadp = (int) (Math.random() * 36);
    System.out.println("numS----- " + numSeleccionadp);
    txtNum.setText(String.valueOf(numSeleccionadp));

    for (int i = 0; i < listaJ.size(); i++) {
        try {
            Jugador jugadorP = listaJ.get(i);
            NumA.append("Numero Ruleta " + numSeleccionadp + " || Numero J " +
jugadorP.getNumApost() + "\n");
            //System.out.println("Numero Ruleta " + numSeleccionadp + " ||
Numero J " + jugadorP.getNumApost() + "\n");
            if (!jugadorP.isBancarrota()) {
                //System.out.println("tlba " +
(tablaJugadores.getSelectedRow()));
                if (jugadorP.getTipoJuego().equalsIgnoreCase("Par")) {
                    if (pOim(numSeleccionadp)) {
                        jugadorP.setSaldo(jugadorP.getSaldo() + 10);
                        System.out.println("= " + jugadorP.getSaldo());
                        // update(jugadorP);
                        // llenarTbl(listaJ);
                    } else {
                        jugadorP.setSaldo(jugadorP.getSaldo() - 10);
                        System.out.println("= " + jugadorP.getSaldo());
                        // update(jugadorP);
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```
//          llenarTbl(listaJ);
        }
        System.out.println("par = " + jugadorP.getNombre());
    } else if (jugadorP.getTipoJuego().equalsIgnoreCase("Impar")) {
        if (!pOim(numSeleccionadp)) {
            jugadorP.setSaldo(jugadorP.getSaldo() + 10);
            System.out.println("= " + jugadorP.getSaldo());
//          update(jugadorP);
//          llenarTbl(listaJ);
        } else {
            jugadorP.setSaldo(jugadorP.getSaldo() - 10);
            System.out.println("= " + jugadorP.getSaldo());
//          update(jugadorP);
//          llenarTbl(listaJ);
        }
        System.out.println("impar= " + jugadorP.getNombre());
    } else if (jugadorP.getTipoJuego().equalsIgnoreCase("Numero")) {
        if (jugadorP.getNumApost() == numSeleccionadp) {
            jugadorP.setSaldo(jugadorP.getSaldo() + 360);
            System.out.println("= " + jugadorP.getSaldo());
//          update(jugadorP);
//          llenarTbl(listaJ);
        } else {
            jugadorP.setSaldo(jugadorP.getSaldo() - 10);
            System.out.println("= " + jugadorP.getSaldo());
//          update(jugadorP);
//          llenarTbl(listaJ);
        }
        System.out.println("Numero= " + jugadorP.getNombre());
    } else if
(jugadorP.getTipoJuego().equalsIgnoreCase("Martingala")) {
        if (jugadorP.getNumApost() == numSeleccionadp) {
            jugadorP.setSaldo(jugadorP.getSaldo() + 10);
            System.out.println("= " + jugadorP.getSaldo());
//          update(jugadorP);
//          llenarTbl(listaJ);
        } else {
            jugadorP.setSaldo(jugadorP.getSaldo() - 10);
            System.out.println("= " + jugadorP.getSaldo());
//          update(jugadorP);
//          llenarTbl(listaJ);
        }
        System.out.println("martingala= " + jugadorP.getNombre());
    }
//          llenarTbl(listaJ);
        }
        //update(jugadorP);
        Thread.sleep(500);
    } catch (InterruptedException ex) {
        Logger.getLogger(ControladorJugador.class.getName()).log(Level.SEVERE, null, ex);
    }
}
//vntR.llenarTabla(listaJ);
```

```
llenarTbl(listaJ);
}

public boolean pOim(int num) {
    int v = num % 2;
    return v == 0;
}

public void llenarTbl(List<Jugador> jugadores) {
    modelo2 = (DefaultTableModel) tablaJugadores.getModel();
    modelo2.setRowCount(0);
    for (int i = 0; i < jugadores.size(); i++) {
        Jugador ju = jugadores.get(i);
        Object[] pl = {ju.getCedula(), ju.getNombre(), ju.getSaldo(),
ju.getNumApost(), ju.isBancarrota()};
        modelo2.addRow(pl);
    }
    tablaJugadores.setModel(modelo2);
}

public DefaultTableModel getModelo2() {
    return modelo2;
}

public void setModelo2(DefaultTableModel modelo2) {
    this.modelo2 = modelo2;
}

public JTable getTablaJugadores() {
    return tablaJugadores;
}

public void setTablaJugadores(JTable tablaJugadores) {
    this.tablaJugadores = tablaJugadores;
}


public int getnS() {
    return nS;
}

public void setnS(int nS) {
    this.nS = nS;
}

public List<Jugador> getListaj() {
    return listaJ;
}

public void setListaj(List<Jugador> listaJ) {
    this.listaJ = listaJ;
}

public JTextField getNumS() {
    return NumS;
}
```

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

```

}

public void setNumS(JTextField NumS) {
    this.NumS = NumS;
}

public JTextArea getNumA() {
    return NumA;
}

public void setNumA(JTextArea NumA) {
    this.NumA = NumA;
}

public VntRuleta getVntR() {
    return vntR;
}

public void setVntR(VntRuleta vntR) {
    this.vntR = vntR;
}

    }
    d. ControladorBanca
public class ControladorBanca extends ControladorAbstract<Banca>{


    public ControladorBanca() {
        super();
    }

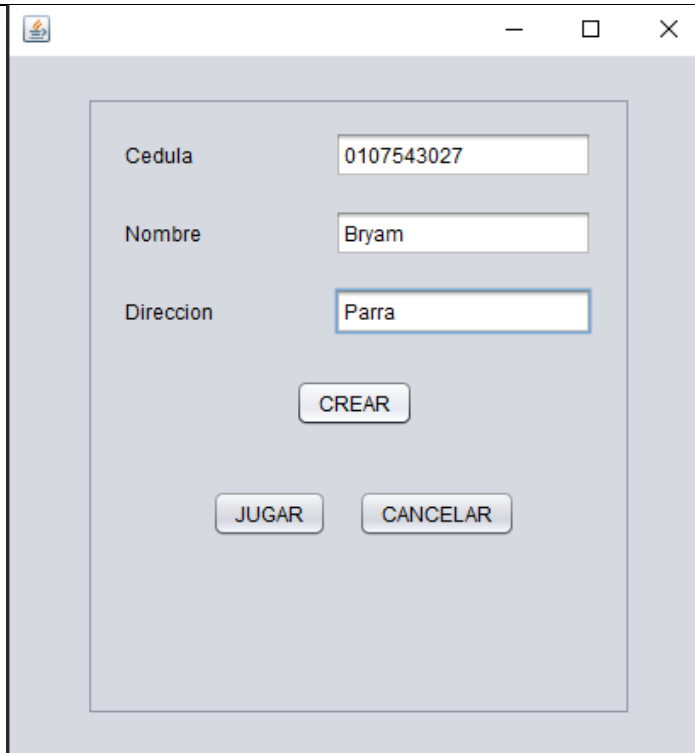
    }

```

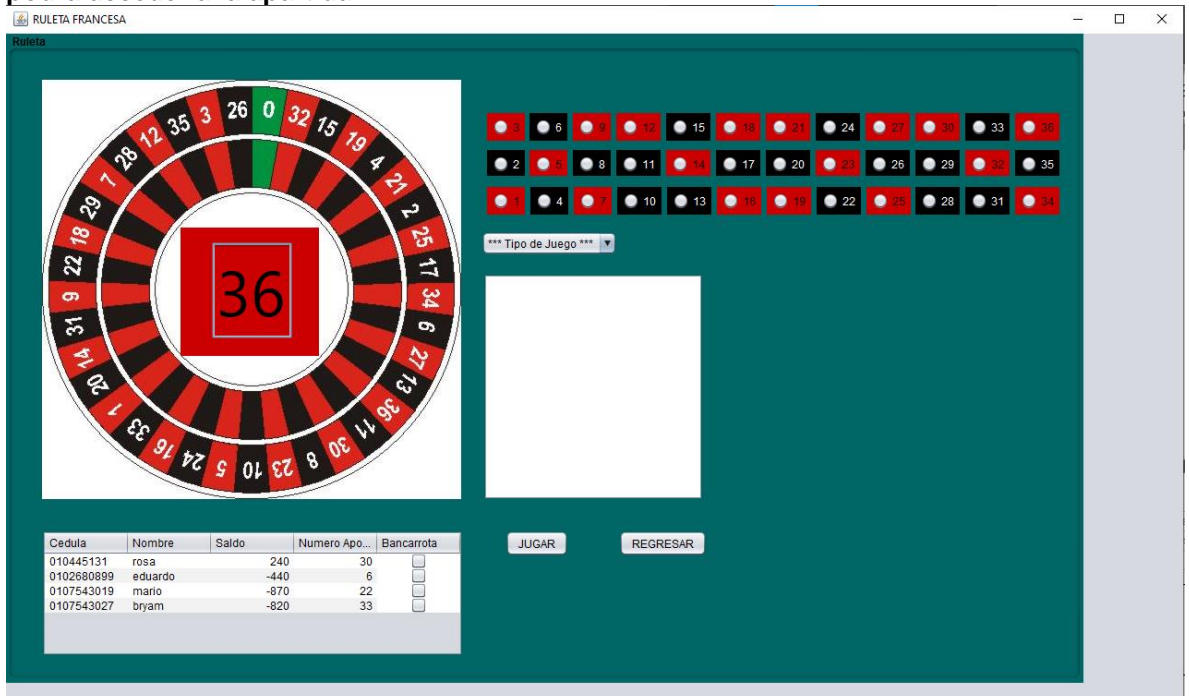
5. Ventanas

- a. VntCrearJugador: Para ingresar al juegos se tiene que crear a los jugadores si hay un mínimo de cuatro jugadores no se podrá acceder a la partida

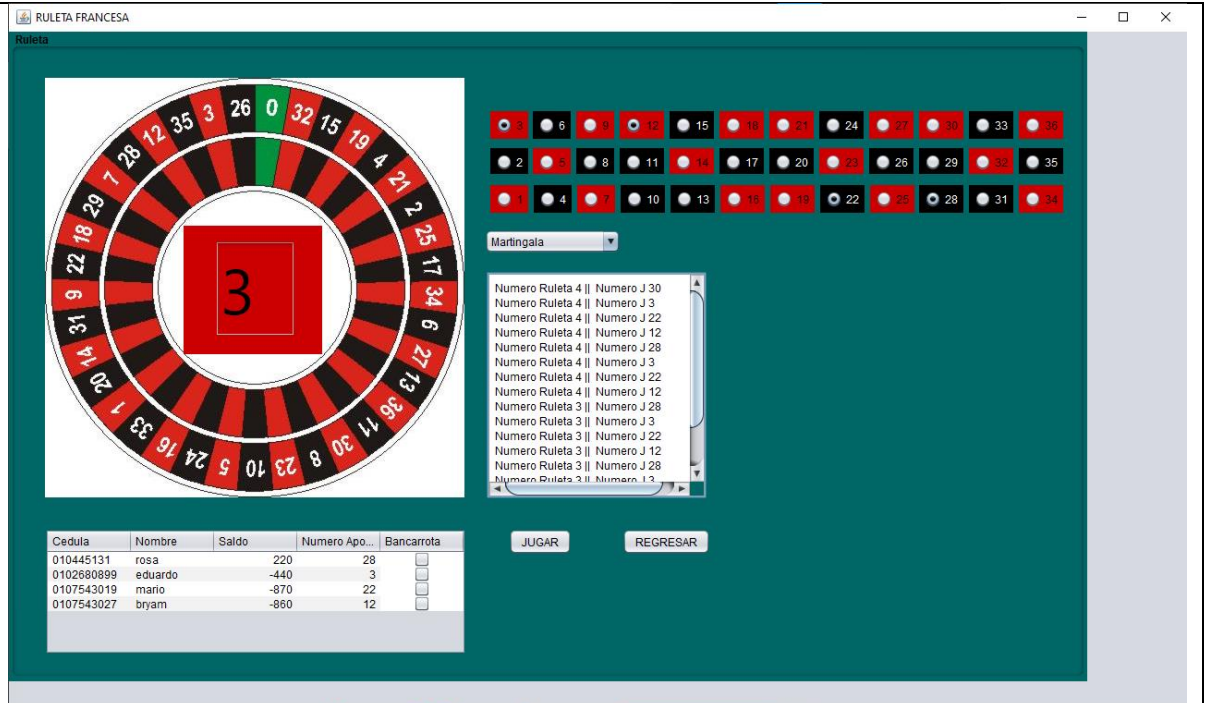
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		



- b. VntRuleta: Cuando el sistema detecte que ya hay cuatro o mas jugadores registrados se podrá acceder a la partida



- c. Se mostrar una imagen con la información de todos los valores que contiene la ruleta francesa. Al presionar en jugar inmediatamente comenzara la partida



- d. Depende de la auesta ganara el jugador o perderá, si aposto a un numero la cantidad de apuesta será de 10\$, si gana recibirá 350 más sus 10\$ que aposto, si fue para o impar, si gana su total aumentara 10\$ y si pierde restara 10\$, en el caso de que sea la martingala si gana recibirá 10\$ pero cada vez que pierda el valor de su apuesta de duplicara.

5.

6.


N.

RESULTADO(S) OBTENIDO(S):

- Entender el funcionamiento de hilos
- Analizar e implementar la lógica par los hilos con JPA en base de datos

CONCLUSIONES:

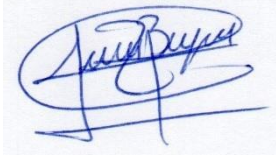
- Se aprendió como es la lógica y el funcionamiento al momento de jugar a la ruleta francesa
- Se tuvo mas conocimiento sobre las relaciones entre las tablas y el funcionamiento de los hilos

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

RECOMENDACIONES:

- Consultar al profesor si se tiene alguna duda respecto al examen practico
- Acudir a la web si no se tiene el debido conocimiento sobre algún tema que sea necesario para realizar el examen

Nombre de estudiante: Bryam Parra



Firma de estudiante: _____