

	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:		TÍTULO PRÁCTICA: Examen Final	
OBJETIVO ALCANZADO: Aplicar los conocimientos adquiridos a lo largo de la asignatura en la creación de un juego.			
ACTIVIDADES DESARROLLADAS			
<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 ultimo se debe presentar dos reportes o tablas de los datos:</p> <ol style="list-style-type: none"> 1. Clientes y la banca con el numero de transacciones o apuestas realizadas, el valor de total y 			

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

cuantas a perdido y cuantas veces a ganado ademas de la modalidad de juego.

2. Dentro de cada cliente se puede acceder al historial de apuestas y transacciones realizadas.

ec.edu.ups.modelo

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 Anahi
 */
@Entity
public class JugadorRuleta implements Serializable {

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

    public Integer getId() {
        return 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")
```

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

```

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 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;
}

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;
}

```

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

```

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;
}

```

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

```

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

}

```

ec.edu.ups.controlador

Controlador Abstracto

```

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

    private List<E> lista;
    private Class<E> clase;
    private 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();
        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;
```

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

```

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;
}

}

```

Controlador Jugador

```

import ec.edu.ups.modelo.JugadorRuleta;

/**
 *
 * @author Anahi
 */
public class controladorJugador extends AbstractControlador<JugadorRuleta> {

    @Override
    public boolean validar(JugadorRuleta objeto) throws Exception {
        return true;
    }

}

```

ec.edu.ups.hilos

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;
import java.util.concurrent.ThreadLocalRandom;
import javax.swing.JLabel;
/**
 *
 * @author Anahi
 */
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 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()){
            JugadorRuleta 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));
        control.actualizar(j);
        tiempoEsperar (segundos);
    }
    return jugador;
}).filter(jugador -> (numeroBanca!=jugador.getNumero())).map(jugador
-> {

    JugadorRuleta j = jugador;
    j.setnLost(jugador.getnLost()+1);
    descripcion.setText(jugador.getNombre()+ "Pierde");
    control.actualizar(j);
    return jugador;
}).forEachOrdered((JugadorRuleta _item) -> {
    tiempoEsperar (segundos);
    iterar=false;
});

break;


case "parImpar":
    jugadores.forEach(jugador -> {
        if(parImpar(numeroBanca)==jugador.getIsPar()){
            JugadorRuleta 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));
            control.actualizar(j);
            tiempoEsperar (segundos);
        }else{
            JugadorRuleta j = jugador;
            j.setnLost(jugador.getnLost()+1);
            descripcion.setText(jugador.getNombre()+ "Pierde");
            tiempoEsperar (segundos);
            iterar=false;
        }
    });

break;

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

```

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

```

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

}

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;
}
}
```

JPA

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

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

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

```

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

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

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

}

```


Base de Datos

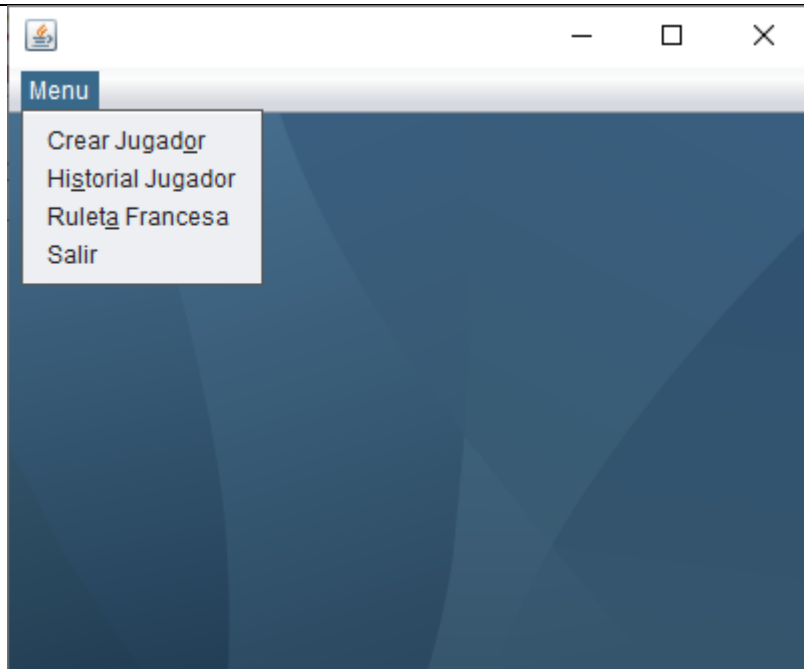
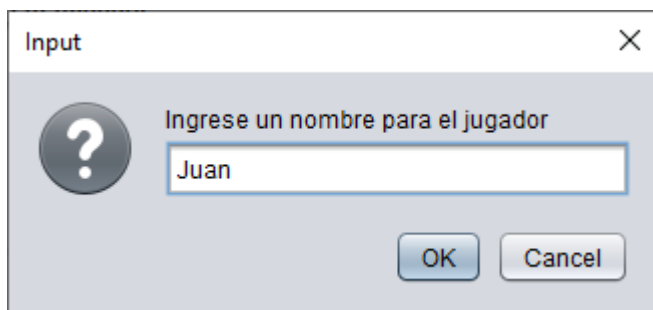
SELECT * FROM "public".ju... X

Max. rows: 100 | Fetched Rows: 5 |

#	id	cantidadapuesta	daupuesta	ispar	napuestas	nperdidas	nganadas	nombre
1	3	0	<input type="checkbox"/>		0	0	0	0 Meli
2	2	20	<input checked="" type="checkbox"/>		0	2	2	0 Paul
3	52	0	<input type="checkbox"/>		0	0	0	0 Juan
4	1	320	<input checked="" type="checkbox"/>		0	6	5	0 Ani
5	51	80	<input checked="" type="checkbox"/>		0	4	3	0 Anahi

RESULTADO(S) OBTENIDO(S):

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





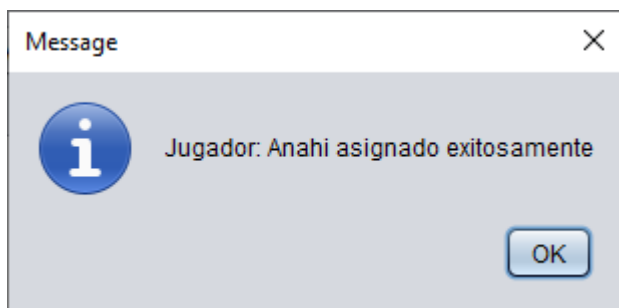
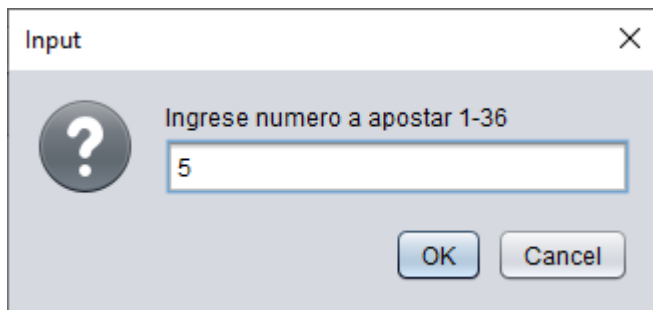
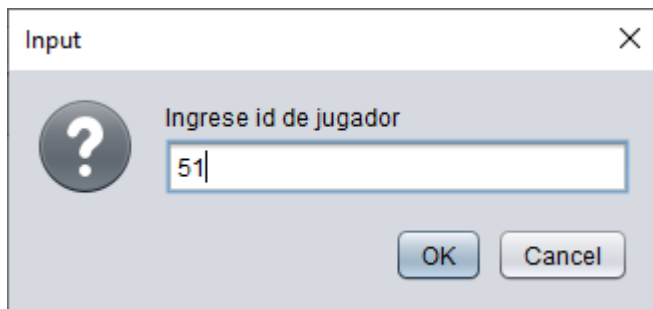
The screenshot shows an 'Input' dialog box with a question mark icon. The text inside says 'Ingrese un nombre para el jugador'. A text field contains the name 'Juan'. At the bottom are 'OK' and 'Cancel' buttons.



The screenshot shows a window titled 'Menu' with a sub-window titled 'Historial de jugadores'. It contains a search bar and buttons 'Buscar', 'Mostrar Todos', and 'Limpiar'. Below is a table with player statistics.

id	Nombre	numero apuest...	Saldo	veces ganadas	veces perdidas
3	Meli	0	1.000	0	0
1	Ani	2	980	0	2
2	Paul	2	980	0	2
51	Anahi	0	1.000	0	0
52	Juan	0	1.000	0	0

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





CONCLUSIONES:


Como estudiantes pudimos aplicar los conocimientos en la creación del juego de la ruleta francesa para poder demostrar lo aprendido.

RECOMENDACIONES:

Aplicar buenas prácticas de programación

Revisar el contenido de la materia

Nombre de estudiante: Edith Anahí Cabrera Bermeo

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



Firma de estudiante: