	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Septiembre 2020 – Febrero 2021



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

CARRERA: Computación

ASIGNATURA: Programación Aplicada

NRO. PRÁCTICA:

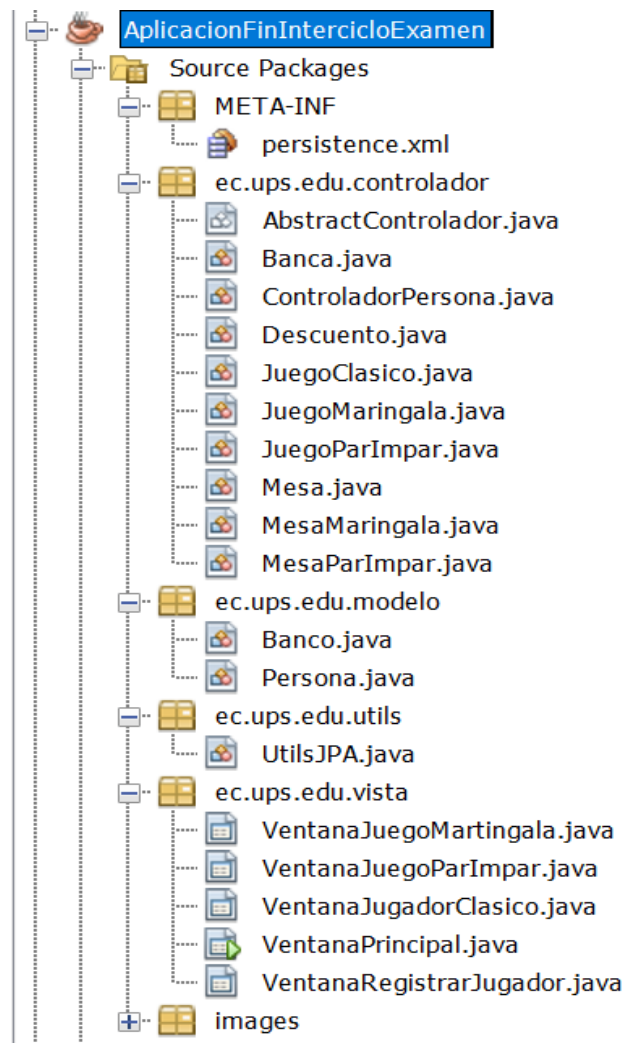
TÍTULO PRÁCTICA: Examen Final

OBJETIVO ALCANZADO:

Utilizar JPA para la creación de bases de datos.
Utilizar Hilos para la creación de diferentes juegos de un casino

ACTIVIDADES DESARROLLADAS

1. Creación de paquetes y clases

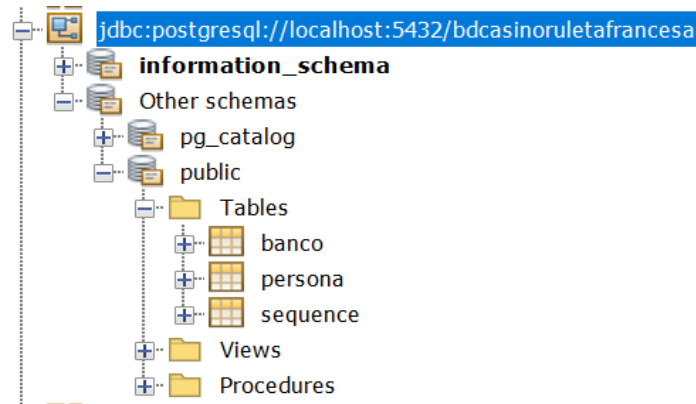


2. Creación del paquete modelo

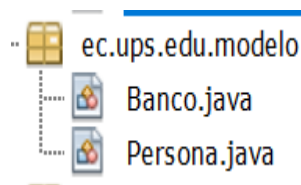
Se han creado dos clases, Una clase Banco y una clase Persona.

En la clase Persona es la encargada de crear los clientes que se van a relacionar con los hilos. Y la clase Banco es donde se guardarán los datos de los hilos de la Persona.

Se ha utilizado JPA para crear la base de datos en postgres



Modelo



Cabe aclarar que se realizó una relación entre estas dos clases de **ManyToOne** y **OneToMany**

Clase Banco

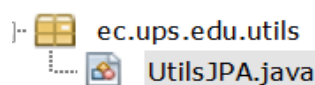
```
@ManyToOne
@JoinColumn(name = "fk_persona")
private Persona persona;
```


Clase Persona

```
@OneToMany(mappedBy = "persona", cascade = CascadeType.ALL)
List<Banco> listaBanco;
```

Creación de la clase Utils

Esta clase nos ayudara a hacer la relación con persistence



	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Septiembre 2020 – Febrero 2021



3. Creación de la clase AbstractFactory

En esta clase se crea el CRUD en donde realizaremos el ingreso de los jugadores

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

import ec.ups.edu.utils.UtillsJPA;
import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;
import java.util.ArrayList;
import java.util.List;
import javax.persistence.EntityManager;

/**
 *
 * @author santi
 */
public abstract class AbstractControlador<T> {

    private List<T> lista;
    private Class<T> clase;
    private EntityManager em;

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

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

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

    public T read(Object id) {
```

```

        return (T) em.find(clase, id);
    }

    public boolean update(T objeto) {
        em.getTransaction().begin();
        objeto = em.merge(objeto);
        em.getTransaction().commit();
        lista = findAll();
        return true;
    }

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

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

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

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

    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;
    }
}

```

Creación de la clase Persona

Heredamos de la clase AbstractFactory y creamos un método buscar con el cual la utilizaremos mas adelante

```

package ec.ups.edu.controlador;

import ec.ups.edu.modelo.Persona;
import static ec.ups.edu.utils.UtilsJPA.getEntityManager;
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;

```

```
/**
 *
 * @author santi
 */
public class ControladorPersona extends AbstractControlador<Persona> {

    public Persona buscar(String cedula) {

        EntityManager em = getEntityManager();

        try {

            String jpql = "Select p from Persona p where p.cedula='" + cedula + "'";
            Persona p = (Persona) em.createQuery(jpql).getSingleResult();

            return p;

        } catch (NoResultException e) {

            System.out.println("Error " + e);

        }

        return null;

    }

}
```

Creación de la clase Banca

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

import ec.ups.edu.modelo.Banco;
import ec.ups.edu.modelo.Persona;
import java.util.List;

/**
 *
 * @author santi
 */
public class Banca extends AbstractControlador<Banco>{

    public long dineroBanco;
    ControladorPersona controladorPersona;
    Persona persona;

    public Banca() {

        setDineroBanco(50000);
        controladorPersona = new ControladorPersona();
        buscar();

    }

    public synchronized int getDineroGanado(Thread tread) {

        if (tread.getName().contains(" numeroConcreto ")) {

            if (getDineroBanco() > 360) {
```

```
        return 360;
    } else {
        return 0;
    }
} else if (tread.getName().contains(" ParImpar ")) {

    if (getDineroBanco() > 20) {

        return 20;

    } else {

        return 0;

    }

}
return 0;
}

public int getDineroGanado(Thread currentTread, int dineroApuesto) {

    if (getDineroBanco() > dineroApuesto) {

        return dineroApuesto * 36;

    } else {

        return 0;

    }

}

public synchronized void retirarDinero(int valor, String modalidad) {

    List<Banco> listaB = persona.getListaNbanco();
    listaB.add(new Banco(0, valor, "retiro", modalidad, persona));
    persona.setListaBanco(listaB);
    long var = persona.getCuenta() - valor;
    persona.setCuenta(var);
    controladorPersona.update(persona);

}

public synchronized void ingresoDinero(String modalidad) {

    System.out.println(modalidad);
    List<Banco> listaB = persona.getListaNbanco();
    listaB.add(new Banco(0, 10, "ingreso", modalidad, persona));
    persona.setListaBanco(listaB);
    dineroBanco = persona.getCuenta() + 10;
    persona.setCuenta(dineroBanco);
    controladorPersona.update(persona);

}

public void buscar() {
```

```

    Persona persona = controladorPersona.buscar("123");
    this.persona = persona;
    dineroBanco = persona.getCuenta();

}

public synchronized int dineroGanado(Thread tread) {

    if (tread.getName().contains("juegoClasico")) {

        if (getDineroBanco() > 360) {

            retirarDinero(360, "juego clasico");
            return 360;

        } else {

            return 0;

        }

    }

    return 0;
}

public synchronized long getDineroBanco() {

    return dineroBanco;

}

public synchronized void setDineroBanco(long dineroBanco) {

    this.dineroBanco = dineroBanco;

}

}

```

Creación de la clase Controlador Persona

```

package ec.ups.edu.controlador;

import ec.ups.edu.modelo.Persona;
import static ec.ups.edu.utils.UtilsJPA.getEntityManager;
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;

/**
 *
 * @author santi
 */
public class ControladorPersona extends AbstractControlador<Persona> {

    public Persona buscar(String cedula) {

        EntityManager em = getEntityManager();

        try {

            String jpql = "Select p from Persona p where p.cedula='" + cedula + "'";
            Persona p = (Persona) em.createQuery(jpql).getSingleResult();

```

```
        return p;

    } catch (NoResultException e) {

        System.out.println("Error " + e);

    }

    return null;

}

}
```

Creación de la clase Descuento

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

/**
 *
 * @author santi
 */
public class Descuento {

    private int dinero;

    public Descuento() {
        setDinero(0);
    }

    public synchronized void setAumento(long dinero) {
        this.dinero += dinero;
    }

    public int getDinero() {
        return dinero;
    }

    public void setDinero(int dinero) {
        this.dinero += dinero;
    }

}
```

Creación de la clase Juego Clásico

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

import ec.ups.edu.modelo.Banco;
import ec.ups.edu.modelo.Persona;
import java.util.List;
import javax.swing.JTextArea;
import javax.swing.JTextField;

/**
 *
 * @author santi
 */
```



```
public class JuegoClasico implements Runnable {

    private Banca banco;
    private int numeroAzar;
    private long saldoInicial;
    String texto = "";
    JTextArea txtArea;
    Descuento descuento;
    ControladorPersona controladorPesona;
    Persona persona;
    JTextField txtCasa;

    public JuegoClasico(Banca banca, Descuento descuento, JTextArea txtArea, Persona per-
sona, JTextField txtCasa) {
        this.banco = banca;
        this.saldoInicial = persona.getCuenta();
        this.descuento = descuento;
        this.txtArea = txtArea;
        this.persona = persona;
        this.txtCasa = txtCasa;
        controladorPesona = new ControladorPersona();
    }

    public JuegoClasico(JTextArea txtArea) {
        this.txtArea = txtArea;
    }

    public boolean tieneDinero() {

        if (getSaldoInicial() > 0) {

            return true;

        } else {

            return false;

        }

    }

    public int apostar() {

        int numero = (int) (Math.random() * 36 + 1);
        quitarDinero(numero);
        List<Banca> lista = persona.getListaNúmero();
        lista.add(new Banca(numero, 10, "retiro", "juego clasico", persona));
        long var = persona.getCuenta() - 10;
        persona.setListaNúmero(lista);
        persona.setCuenta(var);
        banco.ingresoDinero(" juego clasico ");
        controladorPesona.update(persona);
        return numero;

    }

    public void numeroGanado() {
```

```
//le quitamos el dinero al hilo
int dineroDisponibleBanco = banco.getDineroGanado(Thread.currentThread());
List<Banco> lista = persona.getListaNicho();
lista.add(new Banco(numeroAzar, dineroDisponibleBanco,"ingreso" , "juego clasico", per-
sona));
long var = persona.getCuenta() + dineroDisponibleBanco;
persona.setListaNicho(lista);
persona.setCuenta(var);
banco.ingresoDinero("juego clasico");
controladorPersona.update(persona);
//le aumentamos el dinero al hilo ganador
aumentarDinero(dineroDisponibleBanco);
//le disminuimos el dinero al banco.
banco.setDineroBanco(banco.getDineroBanco() - dineroDisponibleBanco);
descuento.setAumento(dineroDisponibleBanco);

}

public void numeroPerdido() {

    descuento.setAumento(-10);

}

public void aumentoDinero(long dinero) {

    long nuevoDinero = getSaldoInicial() + dinero;
    setSaldoInicial(nuevoDinero);

}

public void quitarDinero(long dinero) {

    long dineroPe = getSaldoInicial() - dinero;
    setSaldoInicial(dineroPe);

}

public void aumentarDinero(long dinero) {

    long dineroAu = getSaldoInicial() - dinero;
    setSaldoInicial(dineroAu);

}

@Override
public void run() {

    if (tieneDinero()) {

        int numeroAleatorio = apostar();

        txtCasa.setText(numeroAleatorio+"");

        if (getNumeroAzar() == numeroAleatorio) {

            numeroGanado();
            System.out.println(Thread.currentThread().getName() + " usted gano, saldo actual
" + getSaldoInicial());
        }
    }
}
```

```
        texto = Thread.currentThread().getName() + " usted gano, saldo actual " +
getSaldoInicial() + "\n";
        txtArea.append(texto);

    } else {

        numeroPerdido();
        texto = Thread.currentThread().getName() + " usted perdio, saldo actual " +
getSaldoInicial() + "\n";
        txtArea.append(texto);
        System.out.println(Thread.currentThread().getName() + " usted perdio, saldo ac-
tual " + getSaldoInicial());

    }

    } else {

        System.out.println("No cuenta con dinero suficiente");
        texto = "No cuenta con dinero suficiente \n";
        txtArea.append(texto);

    }

}

public Banca getBanco() {
    return banco;
}

public void setBanco(Banca banco) {
    this.banco = banco;
}

public int getNumeroAzar() {
    return this.numeroAzar;
}

public void setNumeroAzar(int numeroAzar) {
    this.numeroAzar = numeroAzar;
}

public long getSaldoInicial() {
    return saldoInicial;
}

public void setSaldoInicial(long saldoInicial) {
    this.saldoInicial = saldoInicial;
}

}
```

Creación de la clase Juego Martingala

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.ups.edu.controlador;
```

```
import ec.ups.edu.modelo.Banco;
import ec.ups.edu.modelo.Persona;
import java.util.List;
import javax.swing.JTextArea;
import javax.swing.JTextField;

/**
 *
 * @author santi
 */
public class JuegoMaringala implements Runnable {

    private Descuento descuento;
    private Banca banca;
    private int numeroAzar;
    private long saldoInicial;
    private long dineroApuesto;
    Persona persona;
    String texto = "";
    JTextArea txtArea;
    ControladorPersona controladorPersona;
    JTextField txtCasa;

    public JuegoMaringala(Banca banco, Descuento descuento, JTextArea txtArea, Persona persona,
        JTextField txtCasa) {
        this.banca = banco;
        this.descuento = descuento;
        this.saldoInicial = persona.getCuenta();
        this.dineroApuesto = 10;
        controladorPersona = new ControladorPersona();
        this.txtArea = txtArea;
        this.txtCasa = txtCasa;
        this.persona = persona;
    }

    @Override
    public void run() {
        if (bTieneDinero()) {
            int iNumeroHilo = apostar();
            txtCasa.setText(iNumeroHilo + "");
            if (getiNumeroRuleta() == iNumeroHilo) {
                miNumeroGanado();
                System.out.println(Thread.currentThread().getName() + " ha ganado! Ahora tiene "
+ getiSaldoInicial());
                texto = Thread.currentThread().getName() + " ha ganado! Ahora tiene " + getiSal-
doInicial()+"\n";
                txtArea.append(texto);
            } else {
                numeroPerdido();
                System.out.println(Thread.currentThread().getName() + " ha perdido! Ahora tiene
" + getiSaldoInicial());
                texto = Thread.currentThread().getName() + " ha perdido! Ahora tiene " + geti-
SaldoInicial()+"\n";
                txtArea.append(texto);
            }
        } else {
    }
}
```

```

        System.out.println(Thread.currentThread().getName() + " se ha quedado sin dinero
para apostar");
        texto = Thread.currentThread().getName() + " se ha quedado sin dinero para apos-
tar"+"\n";
        txtArea.append(texto);

    }

}

private boolean bTieneDinero() {
    if (getiSaldoInicial() < 10) {
        return false;
    } else {
        return true;
    }
}

private int getiNumeroRuleta() {
    return this.numeroAzar;
}

public void setiNumeroRuleta(int iiNumeroRuleta) {
    this.numeroAzar = iiNumeroRuleta;
}

public int apostar() {

    System.out.println(persona);
    int numero = (int) (Math.random() * 36 + 1);
    disminuirDinero(getiDineroApuesta());
    List<Banco> lista = persona.getListaNúmero();
    lista.add(new Banco(numero, (int) getiDineroApuesta(), "retiro", "juego Martingala",
persona));
    long var = persona.getCuenta() - getiDineroApuesta();
    persona.setListaBanco(lista);
    persona.setCuenta(var);
    banca.ingresoDinero("juego Martingala");
    controladorPersona.update(persona);
    //banca.setDineroBanco(banca.getDineroBanco() + getiDineroApuesta());
    return numero;
}

public void miNumeroGanado() {

    int dineroDisponibleBanco = banca.getDineroGanado(Thread.currentThread(), (int) getiDine-
roApuesta());
    List<Banco> lista = persona.getListaNúmero();
    lista.add(new Banco(numeroAzar, dineroDisponibleBanco, "ingreso", "juego Martingala",
persona));
    long var = persona.getCuenta() + dineroDisponibleBanco;
    persona.setListaBanco(lista);
    persona.setCuenta(var);
    banca.ingresoDinero("juego Martingala");
    controladorPersona.update(persona);
    aumetnarDinero(dineroDisponibleBanco);
    banca.setDineroBanco(banca.getDineroBanco() - dineroDisponibleBanco);
    descuento.setAumento(dineroDisponibleBanco);
    setiDineroApuesta(10);
}

public void aumetnarDinero(long iDinero) {

```

```

        long iNuevoDinero = getiSaldoInicial() + iDinero;
        setiDineroApuesta(iNuevoDinero);
    }

    public void disminuirDinero(long iDinero) {
        long iNuevoDinero = getiSaldoInicial() - iDinero;
        setSaldoInicial(iNuevoDinero);
    }

    public long getiSaldoInicial() {
        return saldoInicial;
    }

    public void setSaldoInicial(long dinero) {
        this.saldoInicial = dinero;
    }

    private void numeroPerdido() {
        descuento.setAumento(getiDineroApuesta());
        setiDineroApuesta(getiDineroApuesta() * 2);
    }

    public long getiDineroApuesta() {
        return dineroApuesto;
    }

    public void setiDineroApuesta(long iDineroApuesta) {
        this.dineroApuesto = iDineroApuesta;
    }
}

```

Creación de la clase Juego Par Impar

```

package ec.ups.edu.controlador;

import ec.ups.edu.modelo.Banco;
import ec.ups.edu.modelo.Persona;
import java.util.List;
import java.util.Random;
import javax.swing.JTextArea;
import javax.swing.JTextField;

/**
 *
 * @author santi
 */
public class JuegoParImpar implements Runnable {

    private Banca banca;
    private int numeroAzar;
    private long saldoInicial;
    private boolean esPar;
    String texto = "";
    JTextArea txtArea;
    Descuento descuento;
    ControladorPersona controladorPesona;
    Persona persona;
    JTextField txtCasa;
}

```

```
public JuegoParImpar(Banca banca, Descuento descuento, Persona persona, JTextArea txtArea,
JTextField txtCasa) {

    this.banca = banca;
    this.saldoInicial = persona.getCuenta();
    this.descuento = descuento;
    this.txtArea = txtArea;
    this.persona = persona;
    this.txtCasa = txtCasa;
    controladorPesona = new ControladorPersona();

}

public JuegoParImpar(int numeroAzar, int saldoInicial) {

    this.numeroAzar = numeroAzar;
    this.saldoInicial = saldoInicial;

}

@Override
public void run() {

    try {

        if (tieneDinero()) {

            setEsPar(apostar());

            if ((getNumeroAzar() % 2 == 0) && isEsPar()) {

                numeroGanado();
                System.out.println(Thread.currentThread().getName() + " Usted gano, Ahora
tiene " + getSaldoInicial());
                texto = Thread.currentThread().getName() + " Usted gano, Ahora tiene " +
getSaldoInicial() + "\n";
                txtArea.append(texto);

            } else {

                numeroPerdido();
                System.out.println(Thread.currentThread().getName() + " Usted perdio, Ahora
tiene " + getSaldoInicial());
                texto = Thread.currentThread().getName() + " Usted perdio, Ahora tiene " +
getSaldoInicial() + "\n";
                txtArea.append(texto);

            }

        } else {

            System.out.println("Ya no cuenta con dinero");
            texto = "Ya no cuenta con dinero \n";
            txtArea.append(texto);

        }

    } catch (InstantiationException | IllegalAccessException e) {

        e.printStackTrace();

    }

}
```

```
}

}

public boolean tieneDinero() {

    if (saldoInicial > 0) {

        return true;

    } else {

        return false;

    }

}

public boolean apostar() throws InstantiationException, IllegalAccessException {

    boolean numeroPar = Random.class.newInstance().nextBoolean();
    disminuirDinero(10);
    List<Banco> lista = persona.getListaNicho();
    lista.add(new Banco(numeroAzar, 10, "retiro", "juego par impar", persona));
    long var = persona.getCuenta() - 10;
    persona.setListaNicho(lista);
    persona.setCuenta(var);
    banca.ingresoDinero(" juego par Impar ");
    controladorPersona.update(persona);
    //banca.setDineroBanco(banca.getDineroBanco() + 10);
    return numeroPar;

}

public void aumentarDinero(long dinero) {

    long nuevoValor = saldoInicial + dinero;
    setSaldoInicial(nuevoValor);

}

public void disminuirDinero(long dinero) {

    long nuevoValor = saldoInicial - dinero;
    setSaldoInicial(nuevoValor);

}

public void numeroGanado() {

    int numeroGana = banca.getDineroGanado(Thread.currentThread());
    List<Banco> lista = persona.getListaNicho();
    lista.add(new Banco(numeroAzar, numeroGana, "ingreso", "juego par Impar", persona));
    long var = persona.getCuenta() + numeroGana;
    persona.setListaNicho(lista);
    persona.setCuenta(var);
    banca.ingresoDinero("juego par Impar");
    controladorPersona.update(persona);
    //le aumentamos el dinero al hilo ganador
    aumentarDinero(numeroGana);
    //le disminuimos el dinero al banco.
    banca.setDineroBanco(banca.getDineroBanco() - numeroGana);
}
```



```

        descuento.setAumento(numeroGana);

    }

    private void numeroPerdido() {
        descuento.setAumento(-10);
    } //mvNumeroPerdido()

    public Banca getBanca() {
        return banca;
    }

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

    public int getNumeroAzar() {
        return numeroAzar;
    }

    public void setNumeroAzar(int numeroAzar) {
        this.numeroAzar = numeroAzar;
    }

    public long getSaldoInicial() {
        return saldoInicial;
    }

    public void setSaldoInicial(long saldoInicial) {
        this.saldoInicial = saldoInicial;
    }

    public boolean isEsPar() {
        return esPar;
    }

    public void setEsPar(boolean esPar) {
        this.esPar = esPar;
    }
}

```

Creación de la clase Mesa

```

package ec.ups.edu.controlador;

import java.util.List;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JTextField;

/**
 *
 * @author santi
 */
public class Mesa extends Thread {

    //Sirve para recibir los hilos.
    List<JuegoClasico> listaJuegoClasico;
    Random numeroApuesta;
}

```

```
JTextField txtNumeroMesa;  
boolean bandera = true;  
  
public Mesa(List<JuegoClasico> lista, JTextField txtNumeroMesa) {  
    //resive los hilos.  
    this.listaJuegoClasico = lista;  
    this.txtNumeroMesa = txtNumeroMesa;  
    //a esta clase has hilo viejo meco  
    Thread hilo = new Thread(this);  
  
    hilo.start();  
  
    System.out.println("hola");  
  
}  
  
@Override  
public void run() {  
    while (bandera) {  
        int numeroAzar = ((int) (Math.random() * 36));  
        txtNumeroMesa.setText(numeroAzar + "");  
        for (JuegoClasico juegoClasico : listaJuegoClasico) {  
            if (juegoClasico != null) {  
                juegoClasico.setNumeroAzar(numeroAzar);  
                Thread hilo1 = new Thread(juegoClasico);  
                hilo1.start();  
                System.out.println("esta mamada ke");  
            }  
        }  
        try {  
            Thread.sleep(2000);  
        } catch (InterruptedException ex) {  
            Logger.getLogger(Mesa.class.getName()).log(Level.SEVERE, null, ex);  
        }  
    }  
}  
  
public boolean isBandera() {  
    return bandera;  
}  
  
public void setBandera(boolean bandera) {  
    this.bandera = bandera;  
}  
}
```

Creación de la clase Mesa Maringala

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

import java.util.List;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JTextField;

/**
 *
 * @author santi
 */
public class MesaMaringala extends Thread {

    List<JuegoMaringala> lista;
    Random numeroAleatorio;
    JTextField txtMesa;
    boolean bandera;

    public MesaMaringala(List<JuegoMaringala> lista, JTextField txtMesa) {

        this.lista = lista;
        this.txtMesa = txtMesa;
        //a esta clase has hilo
        Thread hilo = new Thread(this);
        hilo.start();
        System.out.println("hola");

    }

    @Override
    public void run() {

        bandera = true;

        while (bandera) {

            int numeroAzar = ((int) (Math.random() * 36));

            txtMesa.setText(numeroAzar + "");

            for (JuegoMaringala juegoMaringala : lista) {

                if (juegoMaringala != null) {

                    juegoMaringala.setNumeroRuleta(numeroAzar);

                    Thread hilo1 = new Thread(juegoMaringala);

                    hilo1.start();

                    System.out.println("--");

                }

            }

        }

        try {
```

```
        Thread.sleep(2000);
    } catch (InterruptedException ex) {
        Logger.getLogger(Mesa.class.getName()).log(Level.SEVERE, null, ex);
    }

}

}

public boolean isBandera() {
    return bandera;
}

public void setBandera(boolean bandera) {
    this.bandera = bandera;
}

}
```

Creación de la clase Mesa Par Impar

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.ups.edu.controlador;

import java.util.List;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JTextArea;
import javax.swing.JTextField;

/**
 *
 * @author santi
 */
public class MesaParImpar extends Thread{

    List<JuegoParImpar> listaJuegoParImpar;
    Random numeroAleatorio;
    JTextField txtMesa;
    boolean bandera;

    public MesaParImpar(List<JuegoParImpar> lista, JTextField txtMesa) {

        this.listaJuegoParImpar = lista;
        this.txtMesa = txtMesa;
        //a esta clase has hilo
        Thread hilo = new Thread(this);

        hilo.start();

        System.out.println("hola");

    }

    @Override
```

```
public void run() {

    bandera = true;

    while (bandera) {

        int numeroAzar = ((int) (Math.random() * 36));

        txtMesa.setText(numeroAzar + "");

        for (JuegoParImpar juegoParImpar : listaJuegoParImpar) {

            if (juegoParImpar != null) {

                juegoParImpar.setNumeroAzar(numeroAzar);

                Thread hilo1 = new Thread(juegoParImpar);

                hilo1.start();

                System.out.println("esta mamada ke");

            }

        }

        try {
            Thread.sleep(2000);
        } catch (InterruptedException ex) {
            Logger.getLogger(Mesa.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

}

public boolean isBandera() {
    return bandera;
}

public void setBandera(boolean bandera) {
    this.bandera = bandera;
}

}
```

Creación de la clase JPA

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

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

/**
 *
 * @author santi
 */
```

```
*/
public class UtilsJPA {

    public static final EntityManagerFactory emf = Persistence.createEntityManagerFactory("AplicacionFinIntercicloExamenPU");

    public static EntityManager getEntityManager() {

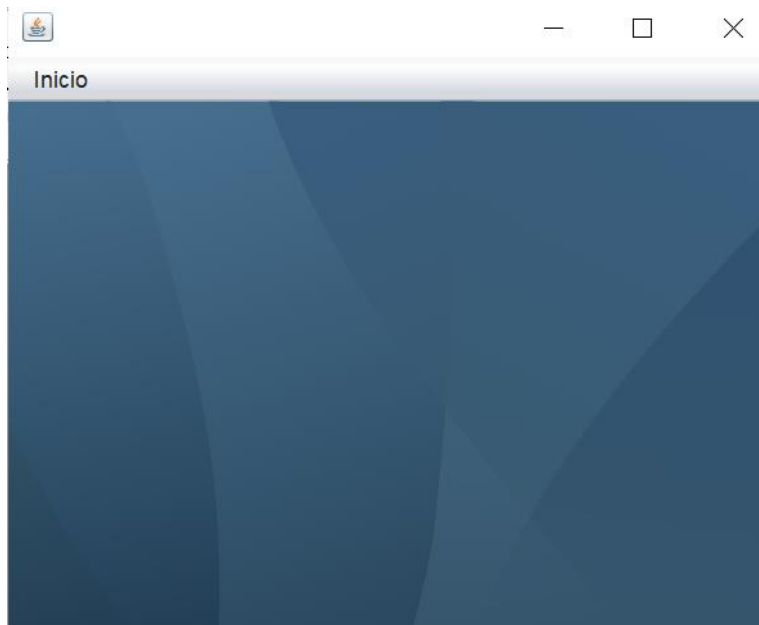
        return emf.createEntityManager();

    }

}
```

4. Funcionamiento del programa

Se crea una clase Principal donde se encontraran varias opciones



Al presionar la opción de registrar jugadores nos mostrara la siguiente pestaña en donde podremos registrar, actualizar, eliminar y buscar los diferentes usuarios que se hayan registrado

REGISTRAR JUGADORES

Id	Cedula	Nombre	Apellido	Cuenta
----	--------	--------	----------	--------

Registrar

Actualizar

Eliminar

Listar todo

Id

Cedula

Nombre

Apellido

Cuenta

1000

Limpiar

Al momento de registrar nos mostrara el siguiente mensaje indicándonos que se ha creado con éxito.

REGISTRAR JUGADORES

Id	Cedula	Nombre	Apellido	Cuenta
----	--------	--------	----------	--------

Registrar

Actualizar

Eliminar

Listar todo

Id

Cedula

Nombre


Apellido

Cuenta

1000

Limpiar

Message

 Se ha registrado con éxito.

OK

#	id	apellido	cedula	cuenta	nombre
1	3501	Cabrera	0150583746	1000	Santiago
2	2201	sad	66	640	sad
3	9	BANCO	123	52120	BANCO
4	4951	Cabrera	0100513589	-30	Nicolas
5	5551	Cabrera	0150683746	1000	Nicolas
6	51	xv xv	987	780	sdfdgvx
7	1	adadad	45	-10	asdsad

REGISTRAR JUGADORES

Id	Cedula	Nombre	Apellido	Cuenta
3501	0150583746	Santiago	Cabrera	1000
2201	66	sad	sad	640
9	123	BANCO	BANCO	52120
4951	0100513589	Nicolas	Cabrera	-30
5551	0150683746	Nicolas	Cabrera	1000
51	987	sdfdgvx	xv xv	780
1	45	asdsad	adadad	-10

Id

Cedula

0150683746

Nombre

Nicolas

Apellido

Cabrera

Cuenta

1000

Registrar

Actualizar

Eliminar

Listar todo

Limpiar

Resolución CS N° 076-04-2016-04-20

Existen tres tipos de juego

Juego Clásico

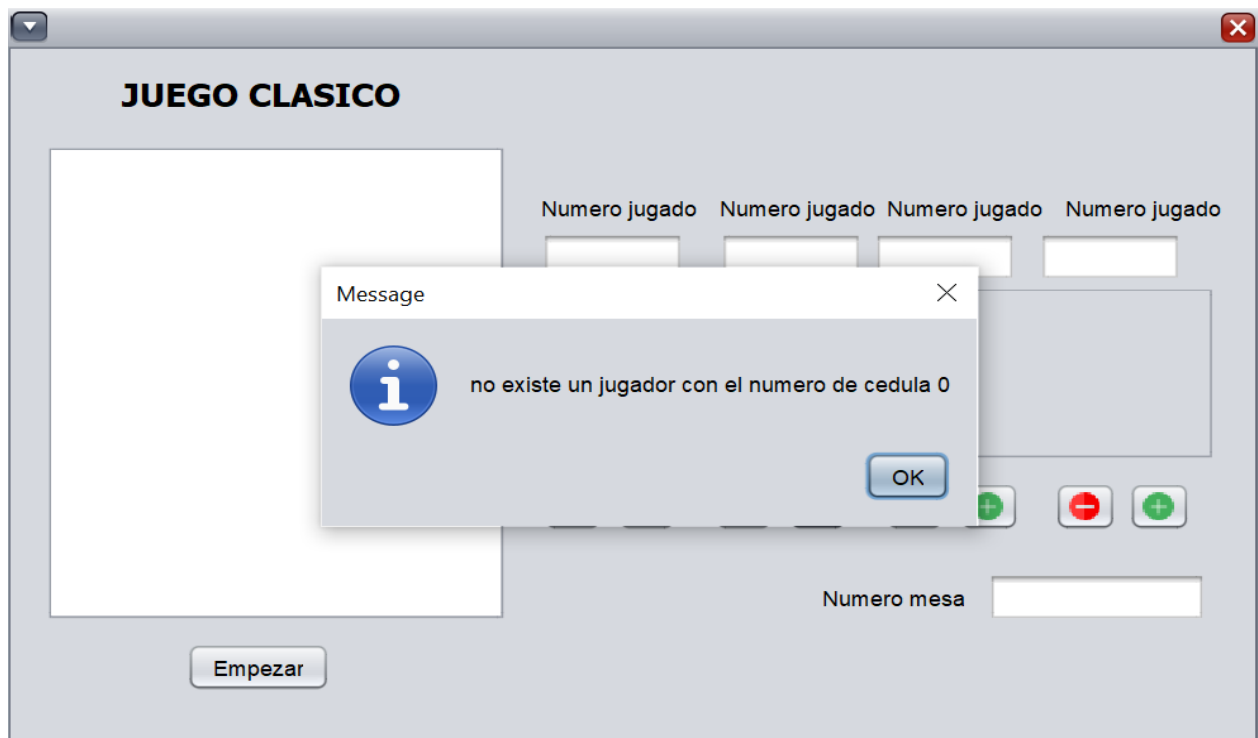
En esta ventana podremos agregar a los jugadores que deseemos mandándolos a buscar por la cedula

The screenshot shows a software application titled "JUEGO CLASICO". An "Input" dialog box is open in the center, asking the user to "Ingrese el numero de cedula de la puerca" (Enter the pig's ID number). The dialog box has a question mark icon and "OK" and "Cancel" buttons. In the background, there is a large empty rectangular area on the left, a "Empezar" button at the bottom left, and a row of eight buttons with red minus and green plus signs. To the right of these buttons, there are labels "Numero jugado" and "Numero jugado" above input fields, and "Numero mesa" above another input field.

Al agregar el número de cedula se pondrá un label en el espacio



Pero si el label no es igual nos mostrara el siguiente mensaje.



Al presionar el botón empezar este iniciará los hilos agregados.

Y como vemos los números aleatorios que cada cliente saca aparece encima de sus nombres y el número que saca la mesa en el txt que dice número de mesa.

JUEGO CLASICO

Thread-2 usted perdio, saldo actual 996

Thread-5 usted perdio, saldo actual 982

Thread-4 usted perdio, saldo actual 974

Thread-3 usted perdio, saldo actual 984

Thread-6 usted perdio, saldo actual 963

Thread-9 usted perdio, saldo actual 976

Thread-8 usted perdio, saldo actual 969

Thread-7 usted perdio, saldo actual 958

Thread-10 usted perdio, saldo actual 929

Thread-13 usted perdio, saldo actual 949

Thread-11 usted perdio, saldo actual 938

Thread-12 usted perdio, saldo actual 964

Thread-14 usted perdio, saldo actual 895

Thread-17 usted perdio, saldo actual 920

Thread-16 usted perdio, saldo actual 954

Numero jugado

3

Numero jugado





35



Numero jugado



7



Numero jugado



18

Numero mesa

2

Empezar

Juego Par Impar

Es la misma estructura para los demás juegos lo que cambia son los resultados.



Inicio



JUEGO PAR IMPAR

Thread-2 Usted perdio, Ahora tiene 95
Thread-3 Usted perdio, Ahora tiene 98
Thread-4 Usted perdio, Ahora tiene 97
Thread-5 Usted gano, Ahora tiene 960
Thread-6 Usted perdio, Ahora tiene 95
Thread-7 Usted perdio, Ahora tiene 94
Thread-8 Usted gano, Ahora tiene 930
Thread-9 Usted perdio, Ahora tiene 92
Thread-10 Usted perdio, Ahora tiene 9
Thread-11 Usted perdio, Ahora tiene 9
Thread-12 Usted perdio, Ahora tiene 8
Thread-13 Usted perdio, Ahora tiene 8
Thread-14 Usted perdio, Ahora tiene 8
Thread-15 Usted perdio, Ahora tiene 8
Thread-16 Usted perdio, Ahora tiene 8
Thread-17 Usted perdio, Ahora tiene 8
Thread-18 Usted perdio, Ahora tiene 8
Thread-19 Usted perdio, Ahora tiene 8

Numero jugado Numero jugado Numero jugado Numero jugado



Numero mesa 11

Empezar

Juego Maringala



JUEGO MARTINGALA

Thread-2 ha perdido! Ahora tiene 990
Thread-3 ha perdido! Ahora tiene 970
Thread-4 ha perdido! Ahora tiene 930
Thread-5 ha perdido! Ahora tiene 850
Thread-6 ha perdido! Ahora tiene 690
Thread-7 ha ganado! Ahora tiene 370
Thread-8 ha perdido! Ahora tiene 360
Thread-9 ha perdido! Ahora tiene 340
Thread-10 ha perdido! Ahora tiene 300
Thread-11 ha perdido! Ahora tiene 220
Thread-12 ha perdido! Ahora tiene 60
Thread-13 ha perdido! Ahora tiene -26
Thread-14 se ha quedado sin dinero p
Thread-15 se ha quedado sin dinero p
Thread-16 se ha quedado sin dinero p
Thread-17 se ha quedado sin dinero p

Numero jugado Numero jugado Numero jugado Numero jugado



Numero mesa 7

Empezar

Podemos ver los todos los registros

Se creo una gran cantidad de registro debido a todas las pruebas realizadas en el programa

Id	Modalidad	Numero Azar	Tipo	Valor
101	juego clasico	0	ingreso	10
102	juego clasico	12	retiro	10
103	juego clasico	0	ingreso	10
104	juego clasico	0	ingreso	10
105	juego clasico	12	retiro	10
106	juego clasico	8	retiro	10
109	juego clasico	0	ingreso	10
107	juego clasico	0	ingreso	10
108	juego clasico	0	ingreso	10
112	juego clasico	24	retiro	10
111	juego clasico	8	retiro	10
110	juego clasico	12	retiro	10
114	juego clasico	0	ingreso	10
113	juego clasico	0	ingreso	10
115	juego clasico	0	ingreso	10
116	juego clasico	0	ingreso	10
119	juego clasico	24	retiro	10
117	juego clasico	12	retiro	10
120	juego clasico	31	retiro	10
118	juego clasico	8	retiro	10

Listar todo

Id	Modalidad	Numero Azar	Tipo	Valor
2805	juego par impar	17	retiro	10
2808	juego par Impar	0	ingreso	10
2807	juego par Impar	0	ingreso	10
2809	juego par Impar	0	ingreso	10
2811	juego par impar	8	retiro	10
2812	juego par impar	33	retiro	10
2810	juego par impar	17	retiro	10
2816	juego par Impar	0	ingreso	10
2813	juego par Impar	0	ingreso	10
2815	juego par Impar	0	ingreso	10
2814	juego par Impar	0	ingreso	10
2817	juego par impar	17	retiro	10
2819	juego par impar	33	retiro	10
2820	juego par impar	15	retiro	10
2818	juego par impar	8	retiro	10
2823	juego par Impar	0	ingreso	10
2825	juego par Impar	0	ingreso	10
2824	juego par Impar	0	ingreso	10
2822	juego par Impar	0	ingreso	10
2821	juego par Impar	0	ingreso	10


Listar todo

Id	Modalidad	Numero Azar	Tipo	Valor
5347	juego Martingala	0	ingreso	10
5348	juego Martingala	0	ingreso	10
5344	juego Martingala	0	ingreso	10
5343	juego Martingala	0	ingreso	10
5349	juego Martingala	0	ingreso	10
5345	juego Martingala	0	ingreso	10
5346	juego Martingala	0	ingreso	10
5354	juego Martingala	11	retiro	10
5351	juego Martingala	28	retiro	10
5352	juego Martingala	12	retiro	10
5355	juego Martingala	9	retiro	10
5353	juego Martingala	36	retiro	10
5356	juego Martingala	12	retiro	10
5350	juego Martingala	8	retiro	10
5357	juego Martingala	0	ingreso	10
5359	juego Martingala	0	ingreso	10
5361	juego Martingala	0	ingreso	10
5363	juego Martingala	0	ingreso	10
5358	juego Martingala	0	ingreso	10
5360	juego Martingala	0	ingreso	10

Listar todo

Registro en postgres

	id [PK] bigint	modalidad character varying (255)	numeroazar integer	tipo character varying (255)	valor integer	fk_persona bigint
1	101	juego clasico	0	ingreso	10	9
2	102	juego clasico	12	retiro	10	1
3	103	juego clasico	0	ingreso	10	9
4	104	juego clasico	0	ingreso	10	9
5	105	juego clasico	12	retiro	10	1
6	106	juego clasico	8	retiro	10	1
7	107	juego clasico	0	ingreso	10	9
8	108	juego clasico	0	ingreso	10	9
9	109	juego clasico	0	ingreso	10	9
10	110	juego clasico	12	retiro	10	1

	Computación	Docente: Diego Quisi Peralta
	Programacion Aplicada	Período Lectivo: Septiembre 2020 – Febero 2021

RESULTADO(S) OBTENIDO(S):


- Utilización de JPA para crear base de datos.
- Reforzar conocimientos de hilos.
- Creación de base de datos
- Utilización de Postgres
- Implementar las diferentes librerías.

CONCLUSIONES:

Con esta práctica hemos afianzado nuestros conocimientos sobre hilos y JPA en base de datos, Trabajar con JPA nos ayuda mucho en la creación de código es mucho mas dinámico y se utiliza menos código, tener un buen entendimientos de hilos(Thread) Es muy importante para la realización de esta práctica.

RECOMENDACIONES:

Como recomendación puedo decir que haber asistido a las clases y ver todo el material proporcionado por el docente ayuda a la realización de este proyecto.

 UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Septiembre 2020 – Febrero 2021

Nombre de estudiante: Jorge Santiago Cabrera Arias

Firma de estudiante:

