	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Marzo 2020 – Julio 2020



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

**CARRERA:** Ing Computation

**ASIGNATURA:** Programación Aplicada

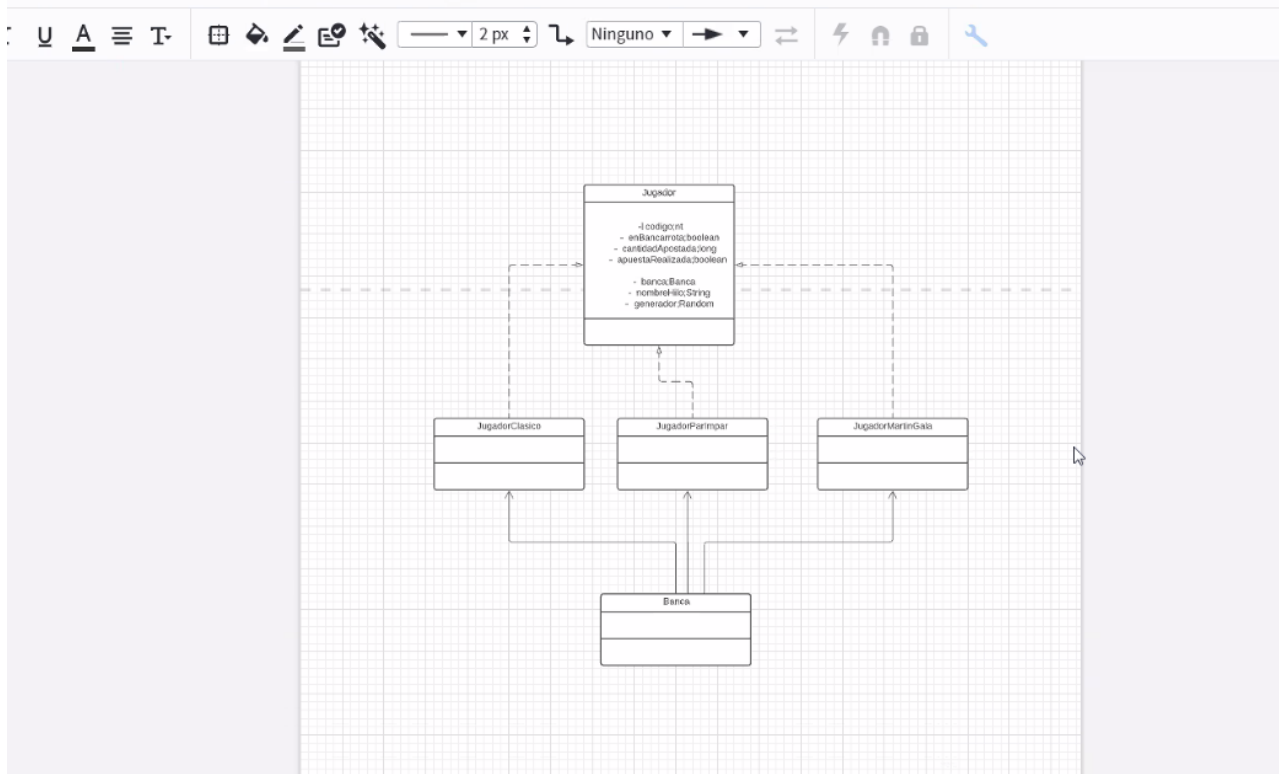
**NRO. PRÁCTICA:** 8 **TÍTULO PRÁCTICA:** Examen final

### OBJETIVO ALCANZADO:

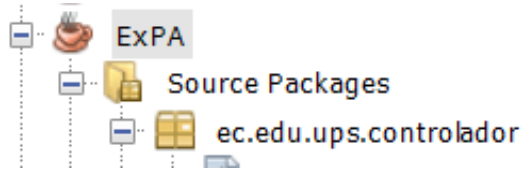
- Implementar los hilos además del JPA con
- Buenas prácticas de programación aplicada

### ACTIVIDADES DESARROLLADAS

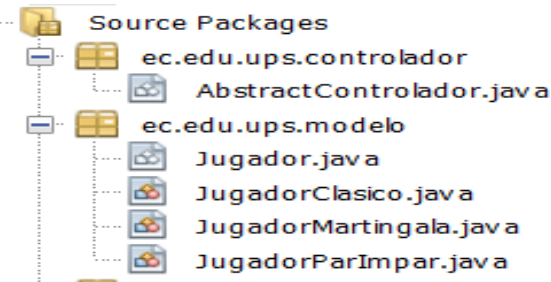
- ⑩ 1. Se creó también clases que definían los tipos de jugador y que heredaban de la clase jugador.
- ⑩ Adicionalmente se creó la clase banca la cual sería nuestra clase principal que gestionaría las acciones.
- ⑩ Los hilos se implementaron en la clase jugador-abstracta y que la clase banca hacía uso de esos hilos para iniciar con las apuestas.
- ⑩ Las apuestas se hacen en un tiempo aleatorio y los juegos son hechos por el sistema de tal manera que este seguirá siempre que la banca cuente con el dinero suficiente y en caso de que este en bancarrota se terminara el juego
- ⑩ Diagrama UML



- 10 Se procedió a crear un paquete de nombre ExPa que significa examen de programación aplicada



- 10 Se creo también clases que definían los tipos de jugador y que heredaban de la clase jugador.



- 10 A continuación se demuestra las clases dentro del paquete modelo  
Clase para el jugador clásico

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package ec.edu.ups.modelo;
7
8  import ec.edu.ups.vista.Banca;
9  import java.io.Serializable;
10 import javax.persistence.Column;
11 import javax.persistence.Entity;
12 import javax.persistence.Inheritance;
13 import javax.persistence.InheritanceType;
14 import javax.persistence.NamedQueries;
15 import javax.persistence.NamedQuery;
16 import javax.persistence.Table;
17
18 /**
19 *
20 * @author ASUS
21 */
22 @Entity
23 @Table(name = "JugadorClasicos")
24 @NamedQueries({
25     @NamedQuery(name = "JugadorClasico.findAll", query = "SELECT p FROM JugadorClasico p")
26 })
27 public class JugadorClasico extends Jugador {
28     @Column(name = "numeroElegido")
29     int numeroElegido;
30     public JugadorClasico(long saldoInicial, Banca b) {
31         super(saldoInicial, b);
32     }
33
34     @Override
35     public void comunicarNumero(int numero) {
36         /* No hace falta comprobar si el numero es
37         * 0, ya que este jugador no lo elige nunca */
38         if (numero==numeroElegido){
39             System.out.println(
40                 "JugadorClasico: (Saldo: 25 pesos la apuesta: 250 pesos)");
41         }
42     }
43 }
```

## Jugador

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.edu.ups.modelo;

import ec.edu.ups.vista.Banca;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

/**
 *
 * @author ASUS
 */

@Entity
@Table(name = "JugadorMartingalas")
@NamedQueries({
    @NamedQuery(name = "JugadorMartingala.findAll", query = "SELECT p FROM JugadorMartingala p")
})
public class JugadorMartingala extends Jugador {
    @Column(name = "cantidadAApostar")
    private int cantidadAApostar;
    @Column(name = "numeroElegido")
    private int numeroElegido;
    public JugadorMartingala(long saldoInicial, Banca b) {
        super(saldoInicial, b);
        cantidadAApostar=1;
    }

    @Override
    public void comunicarNumero(int numero) {
        if (numero==0){
            System.out.println(nombreHilo + " pierde "+cantidadAApostar);
            cantidadAApostar=cantidadAApostar*2;
        }
    }
}

```

clase jugador par/impar

```

    */
    package ec.edu.ups.modelo;

    import ec.edu.ups.vista.Banca;
    import java.util.ArrayList;
    import java.util.Random;
    import javax.persistence.Column;
    import javax.persistence.Entity;
    import javax.persistence.NamedQueries;
    import javax.persistence.NamedQuery;
    import javax.persistence.Table;

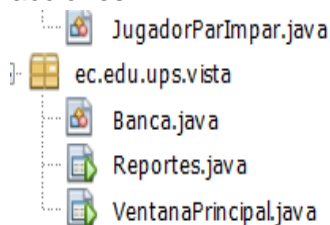
    @Entity
    @Table(name = "JugadorParImpares")
    @NamedQueries({
        @NamedQuery(name = "JugadorParImpar.findAll", query = "SELECT p FROM JugadorParImpar p")
    })
    public class JugadorParImpar extends Jugador {
        public JugadorParImpar(long saldoInicial, Banca b) {
            super(saldoInicial, b);
        }

        @Column(name = "jugamosAPares")
        protected boolean jugamosAPares;

        @Override
        public void hacerApuesta() {

```

- ⑩ Adicionalmente se creó la clase banca la cual sería nuestra clase principal que gestionaría las acciones.



#### Test Packages

Los atributos del Abstrac Jugador serían los siguientes

```

import org.eclipse.persistence.annotations.RangePartition;

@Entity
@Table(name = "Jugadores")
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
@NamedQueries({
    @NamedQuery(name = "Jugador.findAll", query = "SELECT p FROM Jugador p"),
    @NamedQuery(name = "Jugador.findByCedula", query = "SELECT p FROM Jugador p WHERE p.codigo = :codigo"),
})

public abstract class Jugador implements Runnable, Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "codigo")
    private int codigo;
    @Column(name = "saldo")
    protected long saldo;
    @Column(name = "enBancarrota")
    protected boolean enBancarrota;
    @Column(name = "codigo")
    protected long cantidadApuesta;
    @Column(name = "cantidadApuesta")
    protected boolean apuestaRealizada;
    @OneToOne
    @JoinColumn(name = "banca_id", nullable = false)
    protected Banca banca;
    @Column(name = "nombreHilo")
    protected String nombreHilo;
    @Column(name = "generador")

```

- ⑩ Los hilos se implementaron en la clase jugador-abstracta y que la clase banca hacia uso de esos hilos para iniciar con las apuestas.

```
@Override
public void run() {
    nombreHilo = Thread.currentThread().getName();
    while (!enBancarrota() && !banca.enBancarrota()) {
        int msAzar;
        /* Mientras la ruleta no acepte apuestas, dormimos un
         * periodo al azar */
        while (!banca.aceptaApuestas()) {
            msAzar = this.generador.nextInt(500);
            try {
                //System.out.println(nombreHilo+":banca ocupada, durmiendo...");
                Thread.sleep(msAzar);
            } catch (InterruptedException e) {
                return;
            }
            if (banca.enBancarrota()) {
                return;
            }
        }
        hacerApuesta();
    }
    String nombre = Thread.currentThread().getName();
    if (enBancarrota()) {
        System.out.println(nombre + ": ¡Me arruiné!!");
        return;
    }
    if (banca.enBancarrota()) {
        System.out.println(nombre + " hizo saltar la banca");
    }
}
```

10

10 Los atributos que le dimos a la banca con los siguientes

```
14
15
16 public class Banca {
17     protected long saldo;
18     protected boolean enBancarrota;
19     protected Random generador;
20     protected boolean sePuedenHacerApuestas;
21     protected int numeroGanador;
22     public enum Estado {
23         INICIO, ACEPTANDO APUESTAS,
24         RULETA_GIRANDO, PAGANDO_APUESTAS,
25         EN_BANCARROTA
26     };
27 }
```

10

10 Las apuestas se hacen en un tiempo aleatorio y los juegos son hechos por el sistema de tal manera que este seguirá siempre que la banca cuente con el dinero suficiente y en caso de que este en bancarrota se terminara el juego

```
public void simular(int jugadoresPar, int jugadoresMartingala,
    int jugadoresClasicos) throws InterruptedException {
    Thread[] hilosJugadoresPares = new Thread[jugadoresPar];
    for (int i=0; i<jugadoresPar; i++){
        JugadorParImpar jugador = new JugadorParImpar(1000, this);
        hilosJugadoresPares[i] = new Thread ( jugador );
        hilosJugadoresPares[i].setName("Apostador par/impar "+i);
        hilosJugadoresPares[i].start();
    }

    Thread[] hilosJugadoresMartingala = new Thread[jugadoresMartingala];
    for (int i=0; i<jugadoresMartingala; i++){
        JugadorMartingala jugador = new JugadorMartingala(1000, this);
        hilosJugadoresMartingala[i] = new Thread ( jugador );
        hilosJugadoresMartingala[i].setName("Apostador martingala "+i);
        hilosJugadoresMartingala[i].start();
    }

    Thread[] hilosJugadoresClasico = new Thread[jugadoresClasicos];
    for (int i=0; i<jugadoresClasicos; i++){
        JugadorClasico jugador = new JugadorClasico(1000, this);
        hilosJugadoresClasico[i] = new Thread ( jugador );
        hilosJugadoresClasico[i].setName("Apostador clasico "+i);
        hilosJugadoresClasico[i].start();
    }
    this.girarRuleta();
}
```

10

10 Los métodos para que se inicie las apuestas serian los siguientes

```

    }

    public void girarRuleta() throws InterruptedException{
        int segundosAzar;
        System.out.println("Empieza el juego!");
        while (estadoRuleta!=Estado.EN_BANCARROTA){
            estadoRuleta=Estado.ACEPTANDO_APUESTAS;
            /* Se eligen unos milisegundos al azar para que los jugadores
             * elijan, aunque quizá no todos puedan llegar a apostar
             */
            segundosAzar=1+generador.nextInt(3);
            System.out.println("Hagan juego, tienen Vds "+segundosAzar+" segundos");
            Thread.sleep(1000*segundosAzar);

            System.out.println("Ya no va más, señores. ¡Girando!");
            estadoRuleta=Estado.RULETA_GIRANDO;
            Thread.sleep(3000);
            numeroGanador=generador.nextInt(37);
            System.out.println("El número ganador es el : "+numeroGanador);
            estadoRuleta=Estado.PAGANDO_APUESTAS;
            this.comunicarNumeroGanador(numeroGanador);
            System.out.println("El saldo de la banca es ahora:"+saldo);
        }
    }

    public synchronized boolean enBancarrota() {
        return enBancarrota;
    }

    public synchronized void sumarSaldo(long cantidad){
        saldo = saldo + cantidad;
    }

    public synchronized void restarSaldo(long cantidad){
        if (saldo - cantidad <= 0){
            saldo=0;
            estadoRuleta=Estado.EN_BANCARROTA;
            return ;
        }
        saldo = saldo - cantidad;
    }

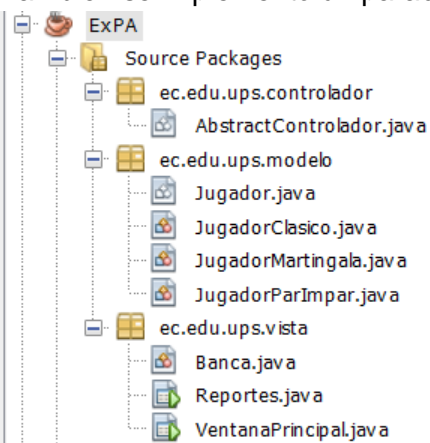
    public synchronized void aceptarApuesta(Jugador jugador){
        if (estadoRuleta == Estado.ACEPTANDO_APUESTAS ) {
            apostadores.add(jugador);
        }
    }

    public synchronized boolean aceptaApuestas(){
        if (estadoRuleta == Estado.ACEPTANDO_APUESTAS ) {
            return true;
        }
        return false;
    }

    public void comunicarNumeroGanador (int numero){
        /* Al pasar el número a los jugadores, ellos nos
         * irán restando el saldo que les corresponda por haber ganado */
        int numApostadores=apostadores.size();
    }

```

## 10 También se implementó un paradigma MVC



- 10 En este caso se procedió a hacer pruebas previas a las interfaces para corroborar la funcionalidad del sistema imprimiendo los datos en la RAM obteniendo los siguientes resultados.

...java | Reportes.java | VentanaPrincipal.java | Banca.java | Jugador.java | JugadorParImpar.java | JugadorMartingala.java...

Source | Design | History

The Tools>Palette>Swing/AWT Components menu item allows you to modify the content of the Palette.

Palette


- Swing Containers
  - Panel
  - Split Pane
  - ToolBar
  - Internal Frame
- Swing Controls
  - Label
  - Toggle Button
  - Tabbed Pane
  - Scroll Pane
  - Desktop Pane
  - Layered Pane
  - Button
  - Check Box

Output

```

run:
Empieza el juego!
Hagan juego, tienen Vds 2 segundos
Ya no va más, señores. ¡Girando!
    
```

To select multiple components in an area hold Shift and drag mouse over the components.



Output - ExPA (run)

```

run:
Empieza el juego!
Hagan juego, tienen Vds 3 segundos
Ya no va más, señores. ¡Girando!
El número ganador es el :31
Apostador martingala 4 queda con un saldo de 999
Apostador clasico 3 queda con un saldo de 990
Apostador par/impar 1 se queda con un saldo de 990
Apostador clasico 1 queda con un saldo de 990
Apostador clasico 4 queda con un saldo de 990
Apostador clasico 0 queda con un saldo de 990
Apostador martingala 2 queda con un saldo de 999
Apostador martingala 1 queda con un saldo de 999
Apostador par/impar 0 se queda con un saldo de 990
Apostador martingala 0 queda con un saldo de 999
Apostador clasico 2 queda con un saldo de 990
Apostador par/impar 3 gana 20 euros por acertar impar
    
```



Source
Design
History

To select multiple components in an area hold Shift and drag mouse over the components.

Output - ExPA (run)

```

Apostador martingala 1 queda con un saldo de 745
Apostador par/impar 1 se queda con un saldo de 960
Apostador martingala 3 queda con un saldo de 745
Apostador clasico 1 queda con un saldo de 920
Apostador par/impar 3 se queda con un saldo de 980
Apostador par/impar 2 se queda con un saldo de 1000
Apostador martingala 0 queda con un saldo de 745
Apostador par/impar 0 gana 20 euros por acertar impar
Apostador par/impar 0 se queda con un saldo de 1040
Apostador par/impar 4 se queda con un saldo de 980
Apostador martingala 2 queda con un saldo de 745
Apostador clasico 2 queda con un saldo de 1280
Apostador clasico 0 queda con un saldo de 920
Apostador clasico 3 queda con un saldo de 920
Apostador martingala 4 queda con un saldo de 745
El saldo de la banca es ahora:51355
Hagan juego, tienen Vds 1 segundos

```

ExPA (run)

### RESULTADO(S) OBTENIDO(S):


- Interpreta de forma correcta los algoritmos de programación y su aplicabilidad.
- Identifica correctamente qué herramientas de programación se pueden aplicar.

### CONCLUSIONES:

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos.
- Los estudiantes implementan soluciones gráficas en sistemas.
- El estudiante aplica JPA

**Nombre de estudiante:** John Farez



 <b>UNIVERSIDAD POLITÉCNICA SALESIANA</b> ECUADOR	<b>Computación</b>	<b>Docente: Diego Quisi Peralta</b>
	Programacion Aplicada	<b>Período Lectivo:</b> Marzo 2020 – Julio 2020

***Firma de estudiante:***

