

Criterio C: Desarrollo

Técnicas Utilizadas

Para el desarrollo de este programa se utilizaron dos IDE (Integrated Development Environment), en un principio Dr Java y posteriormente se terminó usando Net Beans. Ya que a mi parecer el segundo demuestra una mayor facilidad para el ordenamiento dentro del programa, ya que permita la separación por capas MVC (Modelo Vista Controlador). Además que señala los errores y advertencias inmediatamente, sin necesidad de presionar un botón de compilación, y muestra maneras de solucionar cada uno de estos.

1-. Clases

Dentro del proyecto se realizaron nueve clases, las cuales son mostradas en la figura 1. Dentro de estas clases dos extienden de JFrame ya que hacen referencia a las pantallas de la interfaz gráfica, estas son "MainFrame1" y "UserFrame1", mostradas en las figuras 2 y 3. Dentro de la creación de clases, se hizo uso de herencia, en las clases "Cliente" y "Empleado" mostradas en las figuras 8 y 9, las cuales extienden de la clase "Persona" (Figura 6), de este forma se reutiliza código (el de la clase Persona) y facilita la complejidad del código. También se llegó a establecer una relación entre las clases de acuerdo a los atributos que tenían, esta se da en la clase "Evento" (Figura 7), la cual tiene como atributo un objeto de la clase "Cliente" (Figura 8), de esta manera se puede hacer referencia a que cada objeto de tipo evento tiene un objeto de tipo cliente que extiende de la clase persona, logrando así una mejor conexión en la información. Por ultimo existen dos clases que son independientes a las demás la clase "n2t" (Figura 5) y la clase "Usuario" (Figura 4), la primera tiene métodos que convierten un número en las palabras del mismo, la segunda clase tiene como fin almacenar los usuarios de acceso de manera ordenada junto con sus atributos. Es importante mencionar que en cuenta a las clases de las figuras 4, 6 7 8 y 9, contienen un área de "gets" y "sets" de sus atributos respectivos, tal y como se muestra en la Figura 10.

2-.Archivos

En vista de que al cerrar el programa se necesita que guarde información el programa para seguir utilizándola posteriormente cuando se reanude el programa. Se optó por la implementación de archivos binarios, exactamente 6 archivos binarios (Figura 11). Se decidieron usar este tipo de archivos ya que cada uno de estos guarda información del área que su nombre indica, de manera segura y ligera. Para su implementación se usó la clase "RandomAccessFile" para leer y escribir en los archivos.

1. Código leer archivo

- a. No se realizaron métodos para leer, ya que solo se hace una vez, al empezar el programa. Al leer el archivo se basa en obtener primero la cantidad de registros que se obtienen y posteriormente se pasan un arreglo del tipo de la clase que se esté hablando, creando objetos de este tipo, un ejemplo se encuentra en la figura 12. Este proceso se sigue al leer los archivos "InfoEvento", "InfoComidas", "CuentaLogIn" y "Empleados". Como en el ejemplo de la Figura 12.
- b. Los dos restantes se basan en leer una secuencia simple que no se necesita revisar el número de registros que existen. Como en el ejemplo de la Figura 14.

2. Método de guardar archivo

- a. Cada archivo cuenta con un método de guardar. Al igual que al leer el archivo el proceso de 4 archivos binarios es diferente al de los dos restantes. En los archivos "InfoEvento", "InfoComidas", "CuentaLogin" y "Empleados", se guarda el número de registros primero y posteriormente se recorre todo el arreglo del tipo del objeto respectivo usando los métodos "gets" guardando cada uno de los registros. Como en el ejemplo de la Figura 13.
- b. En los dos restantes simplemente se guarda la información tal cual, del mismo orden que se leyó. Como en el ejemplo de la Figura 15.

3-. Librerías

Para la realización de este proyecto se descargaron varias librerías de internet, encontradas en la figura 16. Se basaron para dos funciones principalmente, la primera para la implementación de un calendario (org.jdatepicker) y la segunda fue el API "APACHE POI" para manipular información dentro de archivos de Word (org.apache.poi). La primera se implementó con el fin de facilitar al cliente el ingreso de fechas y la segunda con el fin de facilitar al cliente la creación de contratos.

1. Para la implementación del calendario se importó-"org.jdatepicker.impl.*". Se crearon objetos de tipo UtilDateModel y Properties se inicializo el segundo adecuadamente y se mandó junto con el primero como parámetros al constructor del nuevo objeto tipo JDatePanellImpl. También se creó un objeto de tipo AbstractFormatter. Por último se creó el componente JDatePickerImpl mandando como parámetros el JDatePanellImpl y el AbstractFormatter. Tal como se muestra en la Figura 17.
2. Para la implementación a la modificación de archivos de Word se importaron los "imports" encontrados en la Figura 18. Para su desarrollo se creó un método que recibía una matriz bidimensional que contenía en la primera columna la palabra que se encontraba actualmente en el archivo y en la segunda columna la palabra nueva, además el método recibía el número de contrato para establecer un nombre. Este método crea una copia del template y posteriormente empieza a llenarlo palabra por palabra recorriendo la matriz como se aprecia en la Figura 20 de la línea 282 a 287, ejemplo de inicialización de la matriz y llamado de método en la figura 19. En el método se creó un objeto de tipo XWPFDocument que hace referencia al documento, el método analiza todo su contenido y posteriormente inserta la información deseada con el método ".replace(x[i] [j], x[i] [j+1]);" recibiendo como parámetro la palabra vieja (Columna 0)-"x[i] [j]" y la palabra nueva (Columna 1)-"x[i] [j+1]". Tal como se muestra en la Figura 20 en la línea 285.

4-. Menús y Sub Menús

Para la implementación de menús y sub menú se usó un panel con un "CardLayout". Se usó de esta forma con el fin que el menú principal siempre este presente, abriendo la posibilidad de cambiar a los submenús y a otros paneles. En la figura 21 se encuentra circulado en rojo el Menú principal siempre presente y en verde un ejemplo de un submenú, el lugar del submenú es donde se muestran todos los paneles incluidos en el "CardLayout" con su respectiva referencia, como se puede observar

al comparar las figuras 21 y 22. Por ultimo en la figura 23, se muestra como se agrega cada panel con su referencia y cuantos paneles en totalidad tiene el panel con “CardLayout”.

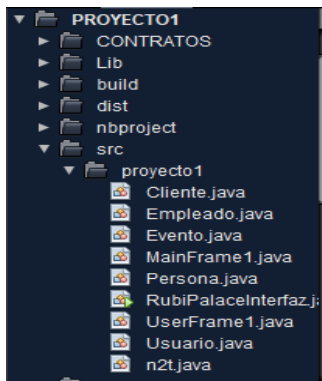


FIGURA 1

```
1 package proyecto1;
2
3 //OBJETO DEL TIPO EVENTO
4 public class Evento {
5     //SE DECLARAN LOS ATRIBUTOS
6     public Cliente clientes;
7     private String nombre;
8     private String fecha;
9     private String primerT;
10    private String segundoT1;
11    private String segundoT2;
12    private String segundoT3;
13    private String tercerT;
14    private int cantidadAdt;
15    private String comidaNiño;
16    private int cantidadNiño;
17    private String horaInicial;
18    private String horaFinal;
19    private int duracion;
20    private String salon;
21    private boolean musica;
22    private boolean arreglo;
23    private int total;
24    private String path;
25    private String descuento;
```

FIGURA 7

```
public class MainFrame1 extends JFrame {
```

FIGURA 2

```
public class UserFrame1 extends JFrame {
```

FIGURA 3

```
1 package proyecto1;
2
3 //OBJETO DEL TIPO USUARIO
4 public class Usuario {
5     //SE DECLARAN LOS ATRIBUTOS
6     private String cuenta;
7     private String password;
8     private boolean restriccion1; //RESTRICCION CONTRATOS1
9     private boolean restriccion2; //RESTRICCION CONTRATOS2
10    private boolean restriccion3; //RESTRICCION COMIDAS1
11    private boolean restriccion4; //RESTRICCION COMIDAS2
12    private boolean restriccion5; //RESTRICCION EMPLEADOS1
13    private boolean restriccion6; //RESTRICCION EMPLEADOS2
14    private boolean restriccion7; //RESTRICCION SEGURIDAD
```

FIGURA 4

```
1 package proyecto1;
2 public class n2t {
3     //SE DECLARAN ATRIBUTOS
4     private int flag;
5     public int numero;
6     public String importe_parcial;
7     public String num;
8     public String num_letra;
9     public String num_letras;
10    public String num_letracm;
11    public String num_letradm;
12    public String num_letracm;
13    public String num_letramm;
14    public String num_letradm;
```

FIGURA 5

```
1 package proyecto1;
2
3 //OBJETO DEL TIPO PERSONA
4
5 public class Persona {
6     //SE DECLARAN LOS ATRIBUTOS
7     private String nombre;
8     private String apellido;
9     private char genero;
10    private String telefono;
11    private String domicilio;
12    private String correo;
```

FIGURA 6

```
1 package proyecto1;
2
3 //OBJETO DEL TIPO CLIENTE, EXTIENDE EL OBJETO DE TIPO PERSONA
4
5 public class Cliente extends Persona{
6     //DECLARACIÓN DE VARIABLES
7     private String nreferente;
8     private String treferente;
9     private String rfc;
```







FIGURA 8

```
39 //AREA DE SETS
40 public void setCuenta(String var){cuenta=var;}
41 public void setPassword(String var){password=var;}
42 public void setRes1(boolean var){restriccion1=var;}
43 public void setRes2(boolean var){restriccion2=var;}
44 public void setRes3(boolean var){restriccion3=var;}
45 public void setRes4(boolean var){restriccion4=var;}
46 public void setRes5(boolean var){restriccion5=var;}
47 public void setRes6(boolean var){restriccion6=var;}
48 public void setRes7(boolean var){restriccion7=var;}
49 //AREA DE GETS
50 public String getCuenta(){return cuenta;}
51 public String getPassword(){return password;}
52 public boolean getRes1(){return restriccion1;}
53 public boolean getRes2(){return restriccion2;}
54 public boolean getRes3(){return restriccion3;}
55 public boolean getRes4(){return restriccion4;}
56 public boolean getRes5(){return restriccion5;}
57 public boolean getRes6(){return restriccion6;}
58 public boolean getRes7(){return restriccion7;}
```

```
1 package proyecto1;
2
3 //OBJETO DEL TIPO EMPLEADO, EXTIENDE EL OBJETO DE TIPO PERSONA
4
5 public class Empleado extends Persona{
6     //DECLARACIÓN DE VARIABLES
7     private String nomina;
8     private String sueldo;
9     private Double sueldo;
```

FIGURA 9

FIGURA 10

 CuentaLogin	19/11/2017 11:32 ...	Archivo DAT	1 KB
 Empleados	22/10/2017 08:41 ...	Archivo DAT	1 KB
 InfoComidas	19/11/2017 11:39 ...	Archivo DAT	1 KB
 InfoEvento	19/11/2017 11:39 ...	Archivo DAT	5 KB
 Nevento	19/11/2017 11:32 ...	Archivo DAT	1 KB
 preferencias	19/11/2017 11:39 ...	Archivo DAT	1 KB

```
541 //LEER EMPLEADOS
542 arch3 = new RandomAccessFile ("Empleados.dat","rws");
543 empleados= new Empleado[100];
544 int numempleados= arch3.readInt();
545 x=0;
546
547 while(numempleados>0){
548     empleados[x]= new Empleado(arch3.readUTF(),arch3.readUTF(),arch3.readChar(),arch3.readUTF(),arch3.readUTF(),arch3.readUTF(),arch3.readUTF(),arch3.readUTF());
549     x++;
550     numempleados--;
551 }
```

```

236 //METODO GUARDAR EMPLEADO
237 //DECLARACION DE VARIABLES QUE SE USARAN EN ESTE METODO Y O FUERA TAMBIEN
238 public Empleado[] empleados;
239 public void guardarEmpleados() {
240     try{
241         arch3 = new RandomAccessFile ("Empleados.dat","rws");
242         int utilizados=0;
243         for(int i=0;i<empleados.length;i++){
244             if(empleados[i]!=null){utilizados++;}
245         }
246         arch3.writeInt(utilizados);
247         for(int x=0;x<empleados.length;x++){
248             if(empleados[x]!=null){
249                 arch3.writeUTF(empleados[x].getNombre());
250                 arch3.writeUTF(empleados[x].getApellido());
251                 arch3.writeChar(empleados[x].getGenero());
252                 arch3.writeUTF(empleados[x].getTelefono());
253                 arch3.writeUTF(empleados[x].getDomicilio());
254                 arch3.writeUTF(empleados[x].getCorreo());
255                 arch3.writeUTF(empleados[x].getNomina());
256                 arch3.writeUTF(empleados[x].getPuesto());
257                 arch3.writeDouble(empleados[x].getSueldo());
258             }
259         }
260     }catch(Exception ex){}
261 }
262 //TERMINA METODO GUARDAR EMPLEADO

```

```
552 //LEER NUMERO EVENTOS
553 archivo5 = new RandomAccessFile ("Nevento.dat","rws");
554 Nevento=archivo5.readInt();
555 //DECLARACION DE EVENTOS
```

```
//METODO PARA GUARDAR NUMERO EVENTO
//DECALARACION DE VARIABLES QUE SE USARAN EN ESTE METODO Y O FUERA TAMBIEN
public int Nevento=0;
RandomAccessFile archivo5;
public void guardarNumero(){
    try{
        archivo5 = new RandomAccessFile ("Nevento.dat","rws");
        archivo5.writeInt(Nevento);
    }catch(Exception e){}
}
//TERMINA METODO QUE GUARDA EL NUMERO
```



```
import org.apache.poi.xwpf.usermodel.XWPFDocument;
import org.apache.poi.xwpf.usermodel.XWPFParagraph;
import org.apache.poi.xwpf.usermodel.XWPFRun;
```

FIGURA 18

FIGURA 16

```
656 //CALENDARIO
657 UtilDateModel model = new UtilDateModel();
658 Properties p = new Properties();
659 p.put("text.today", "Today");
660 p.put("text.month", "Month");
661 p.put("text.year", "Year");
662 JDatePanelImpl datePanel = new JDatePanelImpl(model, p);
663 AbstractFormatter formatter = new AbstractFormatter() {
664     private static final long serialVersionUID = -8693300026142745158L;
665     private String datePattern = "dd.MM.yyyy";
666     private SimpleDateFormat dateFormatter = new SimpleDateFormat(datePattern);
667     @Override
668     public Object stringToValue(String text) throws ParseException {
669         return dateFormatter.parseObject(text);
670     }
671     @Override
672     public String valueToString(Object value) throws ParseException {
673         try {
674             if (value != null) {
675                 Calendar cal = (Calendar) value;
676                 return dateFormatter.format(cal.getTime());
677             } catch (Exception e) { alert("Error revise los datos ingresados"); }
678             return "";
679         }
680     }
681 };
682 JDatePickerImpl datePicker = new JDatePickerImpl(datePanel, formatter);
```

FIGURA 17

```
1872 String[][] cambios={{ "VAR$", (String.valueOf(Nevento)) }, {"VAR1$", eventos[x].clientes
1873 , {"VAR2$", eventos[x].clientes.getDireccion() }, {"VAR35", eventos[x].clientes.getRfc() },
1874 , {"VAR6", String.valueOf(eventos[x].getCantidadAdt() + eventos[x].getCantidadNiño() ) },
1875 , {"VAR9", eventos[x].getFecha().substring(6) }, {"VAR10", String.valueOf(boxscalendario
1876 , {"VAR12", eventos[x].getHoraFinal() }, {"VAR13", String.valueOf(eventos[x].getTotal()
1877 { "VAR15", "20" }, {"VAR16", String.valueOf(eventos[x].getTotal() * 8) }, {"VAR17", numer
1878 { "VAR19", String.valueOf(dia) }, {"VAR20", String.valueOf(mes) }, {"VAR21", String.valueO
1879
1880 escribir2(cambios, String.valueOf(Nevento));
```

```
263 //METODO PARA ESCRIBIR EN WORD
264 //DECLARACION DE VARIABLES QUE SE USARAN EN ESTE METODO Y O FUERA TAMBIEN
265 public static XWPFDocument doc;
266 public void escribir2(String[][] x, String w) throws IOException {
267     String filepath = ".\\CONTRATOS\\CONTRATOBASE.docx";
268     String outpath = ".\\CONTRATOS\\CONTRATOBASE"+w+".docx";
269     doc = new XWPFDocument(new FileInputStream(filepath));
270     for (XWPFParagraph p : doc.getParagraphs()) {
271         StringBuilder sb = new StringBuilder();
272         for (XWPFRun r : p.getRuns()) {
273             int pos = r.getTextPosition();
274             if (r.getText(pos) != null) {
275                 sb.append(r.getText(pos));
276             }
277         }
278         List<XWPFRun> runs = p.getRuns();
279         if (runs != null) {
280             for (XWPFRun r : runs) {
281                 String text = r.getText(0);
282                 for (int i = 0; i < 22; i++) {
283                     int j = 0;
284                     if (text != null && text.contains(x[i][j])) {
285                         text = text.replace(x[i][j], x[i][j+1]);
286                         r.setText(text, 0);
287                     }
288                 }
289             }
290         }
291         doc.write(new FileOutputStream(outpath));
292     }
293     //TERMINA METODO PARA ESCRIBIR EN WORD
```

FIGURA 19

FIGURA 20



FIGURA 21



FIGURA 22

```
//SE CREAN EL PANEL DE CARTAS

final JPanel cards = new JPanel(new CardLayout());
//SE AÑADEN LOS PANELES AL PANEL DE CARTAS AÑADIENDO REFERENCIAS RESPECTIVAS
cards.add(main10, "CALENDARIO");
cards.add(main3, "COMIDASMENU");
cards.add(main2, "COMIDASVER");
cards.add(main4, "COMIDASMODIFICAR");
cards.add(main5, "COMIDASAGREGAR");
cards.add(main6, "COMIDASQUITAR");
cards.add(main, "CONTRATOSAGREGAR");
cards.add(main7, "CONTRATOSMENU");
cards.add(main8, "CONTRATOSBUSCAR");
cards.add(main11, "EMPLEADOS");
cards.add(main12, "EMPLEADOSVER");
cards.add(main13, "EMPLEADOSAGREGAR");
cards.add(main14, "EMPLEADOSQUITAR");
cards.add(main15, "SEGURIDAD");
cards.add(main16, "SEGURIDADAGREGAR");
cards.add(main17, "SEGURIDADMODIFICAR");
cards.add(main18, "SEGURIDADQUITAR");
cards.add(tablapanel, "PANELTABLA");
```

FIGURA 23

Referencias:

Total de Palabras: 1093

CaveofProgramming. (2012, January 28). Advanced Java: Swing (GUI) Programming. Retrieved November 20, 2017, from <https://www.youtube.com/watch?v=DJqIT1d67jI&t=281s>

Font (Java Platform SE 7). Retrieved July 9, 2016, from <https://docs.oracle.com/javase/7/docs/api/java/awt/Font.html>

JComboBox (Java Platform SE 7). Retrieved August 6, 2016, from <https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html>

Creating a JTable : JTable « Swing « Java Tutorial. Retrieved August 9, 2016, from http://www.java2s.com/Tutorial/Java/0240_Swing/CreatingJTable.htm

JTable (Java Platform SE 7). Retrieved August 9, 2016, from <https://docs.oracle.com/javase/7/docs/api/javax/swing/JTable.html>

API for Microsoft Documents. (n.d.). Retrieved October 9, 2016, from <https://poi.apache.org/>

Stack Overflow - Where Developers Learn, Share, & Build Careers. (n.d.). Retrieved November 20, 2017, from <https://stackoverflow.com/>

How to use JDatePicker to display calendar component. (n.d.). Retrieved November 20, 2017, from <http://www.codejava.net/java-se/swing/how-to-use-jdatepicker-to-display-calendar-component>

JarDownload (n.d.). Retrieved November 20, 2017, from <http://www.java2s.com/Code/Jar/>

La web del programador (n.d.). Retrieved November 20, 2017, from <https://www.lawebdelprogramador.com/codigo/Java/338-Convertir-numeros-a-letras.html>