GUÍA DE LABORATORIO N° 01

Escuela Profesional: Ingeniería de Sistemas. **Asignatura:** Metodología de la Programación.
Ciclo y Turno: IV – Mañana **Semestre Académico:** 2017 - II
Docente: Ing.Ivan Petrlik / Ing. Gustavo Coronel **Fecha:** 20 de Agosto al 01 de Setiembre del 2017

Serialización de Objetos

INTRODUCCION

Una de las grandes utilidades de de la serialización de objetos con el manejo de arraylist y manejo de archivos binarios es su capacidad de guardar cualquier tipo de datos y su manipulación está sujeta al uso de una clase principal con el manejo de un arreglo de objetos para manipular la información a través de posiciones.
Se implementaran aplicaciones en entorno visual con JFrame, utilizando como herramienta de desarrollo NetBeans 6.8 y además para demostrar los el uso de archivos binarios y arreglos de objetos.

I. OBJETIVOS

1. Desarrollo de ejercicios de Manejo de Cadenas usando Swing WT.
2. Codificar, compilar y ejecutar los ejercicios de aplicación.
3. Conocer la estructura de un programa visual en Java.
4. Se espera que el estudiante asocie los conocimientos nuevos con la "nueva plantilla" que se propone.

II. EQUIPOS Y MATERIALES

- Computadora personal.
- Programa NetBeans IDE 6.7.1 o 6.8 correctamente instalado.
- Notas de los ejercicios resueltos en la clase.

III. METODOLOGIA Y ACTIVIDADES

- Codificar los ejercicios desarrollados en el aula.
- Presentar avances y ejecución de cada uno de los ejercicios al docente o jefe de práctica encargado para la calificación correspondiente.
- Guardar la carpeta de sus archivos a sus memorias.
- Apagar el computador y dejarla en buen estado al retirarse del laboratorio dejar todo en orden.

IV. OBSERVACION

- El estudiante deberá crear una carpeta de trabajo con el nombre "Laboratorio_12", a donde deberá direccional su proyecto a crear.
- Se recomienda que el estudiante haya planteado los ejercicios de la Guía de ejercicios 12 para que pueda comprender las soluciones que se proponen.

V. NUEVAS LIBRERIAS

Ver diapositiva GUI en Java-AWT & SWING.pdf

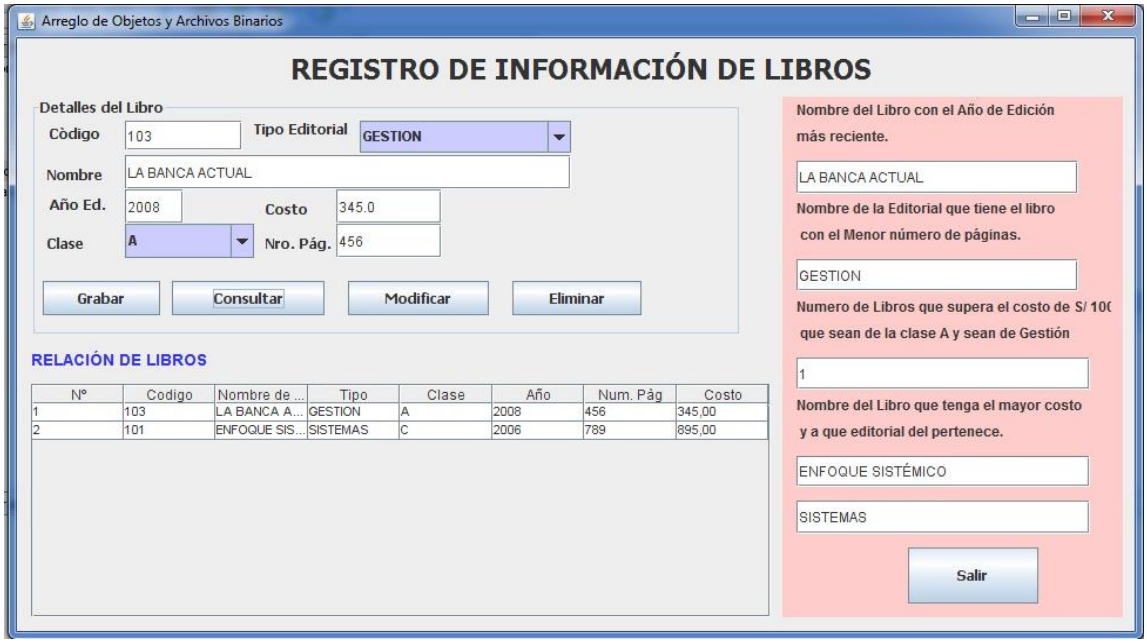
VI. EJERCICIOS.

Ejemplo 01: Construya un programa que guarde la siguiente información en un archivo de binario llamado "Libros.bin" de los libros de una Librería como: Código del libro (código único), Nombre del Libro, Tipo de Editorial (A, B, y C), Clase de Libro (Programación, Gestión y Sistemas), Año de Edición, Nro. de Páginas y Costo del Libro. Su programa deberá mostrar los siguientes datos:

a) Nombre del Libro con el Año de Edición más reciente.
b) Nombre de la Editorial que tiene el libro con el Menor número de páginas.
c) Numero de Libros que supera el costo de S/ 100 que sean de la Editorial A y sean de Gestión
d) Nombre del Libro que tenga el mayor costo y a que editorial del pertenece.

SOLUCION:

Paso 1: Construya el siguiente diseño:

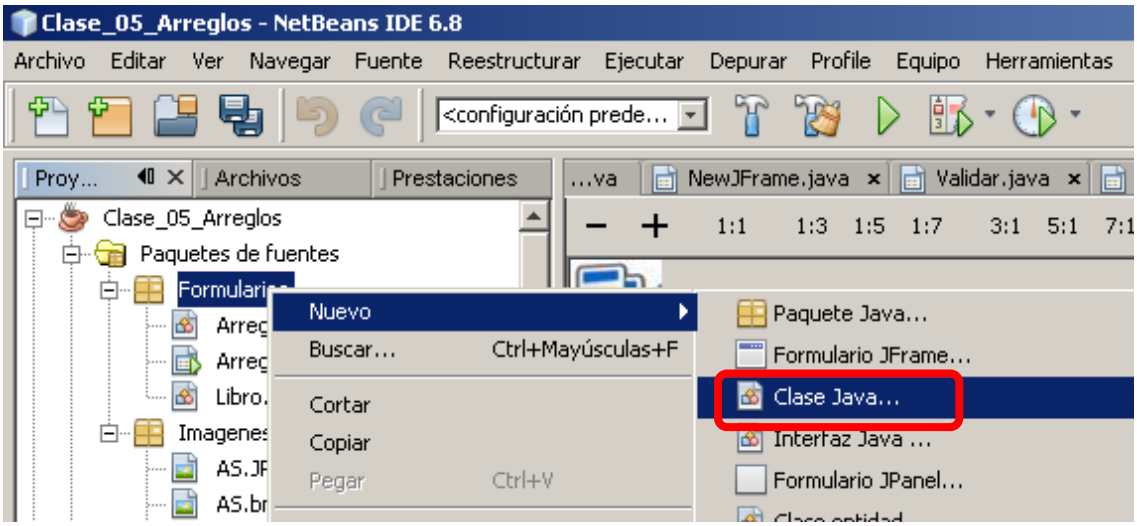


El paquete contiene lo siguiente

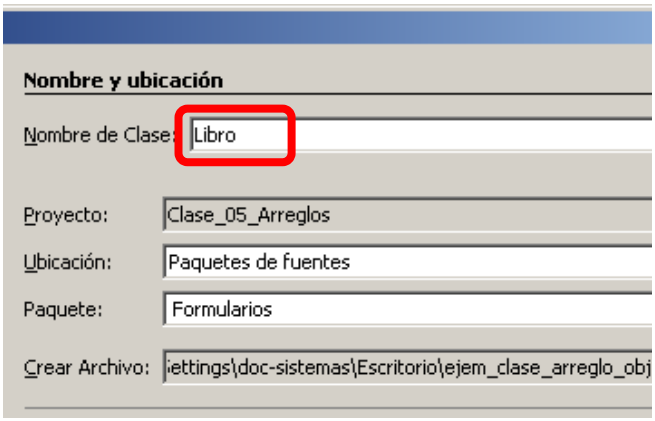
a) Una clase **Libro** que permitirá guardar la información referente a los libros el cual tiene que ser Serializada para utilizar manejo de archivos binarios.
b) Una clase **ArregloLibros** que permitirá utilizar un arreglo de objetos (ArrayList) que contendrá la información de todos los objetos instanciados de la clase Libro, así como todos los métodos para manejar dichos objetos.
c) Un JFrame **Archivos_Binarios** que utilizara las dos clases anteriores donde se interactuara para guardar información, consultas, búsquedas, entre otras.

LA CLASE Libro

Dentro del Paquete Formulario Para la creación de la **clase Libro** haga lo siguiente: sobre el paquete Formulario, botón derecho selecciones Nuevo y luego Clase Java tal y como se ve en la figura:



A continuación aparcera la siguiente ventana y cree su clase y coloque el nombre de la clase como: **Libro** y luego presiones el botón **Aceptar**



Ahora complete las líneas de código que faltan tal y como se muestra en el siguiente grafico

```
import java.io.Serializable;

public class Libro implements Serializable {
    // atributos
    private String codigo;
    private String nombre;
    private String tipo;
    private String clase;
    private int anio;
    private int num_pag;
    private double costo;

// constructor vacio
public Libro(){        }
// constructor con parametros
public Libro(String codigo,String nombre,String tipo,String clase,int anio,int num_pag,double costo){
         this.codigo=codigo;
        this.nombre = nombre;
         this.tipo = tipo;
         this.clase = clase;
        this.anio=anio;
        this.num_pag=num_pag;
        this.costo=costo;
 }
```

```
    public String getCodigo() { return codigo; }
    public void setCodigo(String codigo) { this.codigo = codigo;}
    public String getNombre() { return nombre; }
    public void setNombre(String nombre) { this.nombre = nombre; }
    public String getTipo() { return tipo; }
    public void setTipo(String tipo) { this.tipo = tipo; }
    public String getClase() { return clase; }
    public void setClase(String clase) { this.clase = clase;}
    public int getAnio() { return anio; }
    public void setAnio(int anio) { this.anio = anio; }
    public int getNum_pag() {  return num_pag;   }
    public void setNum_pag(int num_pag) { this.num_pag = num_pag; }
    public double getCosto() { return costo; }
    public void setCosto(double costo) { this.costo = costo; }

}
```

LA CLASE ArregloLibros

Siga los mismos pasos para crear una clase y cree la clase **ArregloLibros** y complete las siguientes instrucciones como se detalla a continuación:

```
//Declaracion de librerias adicionales
import java.io.Serializable;
import java.util.ArrayList;
public class ArregloLibros implements Serializable {
private ArrayList <Libro>  a;  // arreglo de objetos para libros
//Generacion del Constructor
      public ArregloLibros(){
                a = new ArrayList();// crea el objeto
        }
// metodos de administracion del arreglo
        // adiciona un nuevo Libro
        public void agrega(Libro nuevo){  a.add(nuevo); }
        // obtiene un Libro
        public Libro getLibro(int i){ return a.get(i) ;       }
        // reemplaza un informacion de un libro
        public void reemplaza(int i, Libro actualizado){
          a.set(i, actualizado);
        }
        // elimina un libro
        public void elimina(int i){  a.remove(i);  }
        // elimina a todos los libros
        public void elimina(){
          for (int i=0; i<numeroLibros(); i++)
                a.remove(0);
      }
        // obtiene número de libros registrado
        public int numeroLibros(){  return a.size();   }

        // busca un libro por codigo
        public int busca(String codigo){
          for (int i=0; i<numeroLibros(); i++){
                if (codigo.equalsIgnoreCase(getLibro(i).getCodigo()))
                return i; // retorna indice
          }
          return -1; // significa que no lo encontró
        }
}
```

DECLARACION DE LIBRERIAS ADICIONALES, VARIABLES GLOBALES Y INICIALIZACION DE METODOS EN EL ENCABEZADO DE LA CLASE DEL EDITOR DE CODIGO

//Declaracion de Librerias

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.text.DecimalFormat;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
public class Archivos_Binarios extends javax.swing.JFrame {
   // Declarando las clases
   Libro  objLibro;            // un Libro
   ArregloLibros objArreglo;  // todos los Libros
   DefaultTableModel miModelo;
   String[] cabecera={"Nº","Codigo","Nombre de Libro","Tipo","Clase","Año","Num.
Pàg","Costo"};
   String[][] data={};
    int num = 0;

public Archivos_Binarios() {
        initComponents();
        // Creando el modelo de la tabla
        miModelo=new DefaultTableModel(data,cabecera);
        tblRegistros.setModel(miModelo);
        // objeto que administra la información de los Libros
          objArreglo = new ArregloLibros();
        // proceso de carga de data del archivo al arreglo de objetos
          cargaData();
        //actualizando la tabla
        actualizar_tabla();
        //llamando al método resumen para mostrar los datos de salida
        resumen();
        //colocando el cursor en el text Código
        jtxtCodigo.requestFocus();
    }
```

A CONTINUACION DE DEFINEN LOS METODOS UTILIZADOS PARA LA MANIPULACION DE LA INFORMACION DE ARCHIVOS BINARIOS Y EL ARREGLO DE OBJETOS

//método para extraer la informacion del archivo binario al arreglo de objetos

```
public void cargaData(){
        // lee la data del objeto serializado
        try {
           FileInputStream fis = new FileInputStream("Libros.bin");
           ObjectInputStream in = new ObjectInputStream(fis);
           if (in != null){
                 objArreglo = (ArregloLibros)in.readObject();
                 in.close();
           }
        }catch(Exception ex){
          System.out.println(ex);
```

```
        JOptionPane.showMessageDialog(null,"Error al cargar el archivo binario.."+ex);
      }// fin del catch
}// fin de cargaData

//método para grabar la informacion del arreglo de objetos al archivo binario
public void grabar(){
//guarda la data en el archivo serializado
   try{
      FileOutputStream fos = new FileOutputStream("Libros.bin");
      ObjectOutputStream out = new ObjectOutputStream(fos);

        if (out != null){
              out.writeObject(objArreglo);
              out.close();
        }// fin del if
   }catch(IOException ex){
   mensaje("Error en grabacion del archivo serializado.."+ex);
   }// fin del catch
}// fin de grabar

//limpia la tabla cada vez que esta se actualiza
public void vaciar_tabla()
{
int filas=tblRegistros.getRowCount();
for(int i=0;i<filas;i++)
   miModelo.removeRow(0); // removiendo la fila de la tabla
}

//método para mostrar los mensajes del aplicativo
public void mensaje(String texto){
        JOptionPane.showMessageDialog(this,texto);
}// fin de mensaje
```

A continuación el método Resumen que extrae la información del arreglo de Objetos y obtiene la información de los datos de salida

```
void resumen(){  //Declaración de las variables para la extracción de datos
   String sA="",sB="",sD1="",sD2="";
   int mayor=-99,menor=999999,sC=0;
   double maycos=-99;
   int n=objArreglo.numeroLibros(); //Numero de Libros
   for(int i=0;i<n;i++)
   {   // Se extrae la información de cada objeto del Arreglo
      String cod = objArreglo.getLibro(i).getCodigo();
      String nombre = objArreglo.getLibro(i).getNombre();
      String tipo =  objArreglo.getLibro(i).getTipo();
      String clase= objArreglo.getLibro(i).getClase();
      int anio=objArreglo.getLibro(i).getAnio();
      int num_pag=objArreglo.getLibro(i).getNum_pag();
      double costo=objArreglo.getLibro(i).getCosto();
      //calculando los datos de salida
     //Nombre del Libro con el Año de Edición más reciente
        if(anio>mayor)
         {  mayor=anio;
           sA=nombre; }
     //Nombre de la Editorial que tiene el libro con el Menor número de páginas.
        if(num_pag<menor)
         {  menor=num_pag;
           sB=tipo; }
     //Numero de Libros que supera el costo de S/ 100 que sean de la Editorial A
     //y sean de Gestión
```

```
        if(costo>100 && tipo.equalsIgnoreCase("GESTION") && clase.equalsIgnoreCase("A"))
           sC++;
      //Nombre del Libro que tenga el mayor costo y a que editorial del pertenece.
        if(costo>maycos)
         {  maycos=costo;
           sD1=nombre;
           sD2=tipo; }
       }
         //colocando la información en los TextField
         jtxtsA.setText(sA);
         jtxtsB.setText(sB);
         jtxtsC.setText(String.valueOf(sC));
         jtxtsD1.setText(sD1);
         jtxtsD2.setText(sD2);
 }
```

A continuación el método que actualiza la tabla con la información que se guarda en el Arreglo de Objetos.

```
public void actualizar_tabla()
{   vaciar_tabla(); // Vaciando la información de la tabla
    //capturando el tamaño del arreglo
    int n=objArreglo.numeroLibros();
    for(int i=0;i<n;i++)
    {   // Se extrae la información de cada objeto del Arreglo
        String cod = objArreglo.getLibro(i).getCodigo();
        String nom = objArreglo.getLibro(i).getNombre();
        String ti =  objArreglo.getLibro(i).getTipo();
        String cla= objArreglo.getLibro(i).getClase();
        int an=objArreglo.getLibro(i).getAnio();
        int np=objArreglo.getLibro(i).getNum_pag();
        double c=objArreglo.getLibro(i).getCosto();
        //Insertando la información en el Tabla
        insertar(i+1,cod,nom,ti,cla,an,np,c);
    }

}
```

A CONTINUACION DE DEFINEN LOS METODOS UTILIZADOS PARA LAS CONSULTAS, ELIMINACION Y ACTUALIZACION DE DATOS

```
public void eliminar(){
    // se llamada método consulta para ver los datos a borrar
    consulta();
   //Se llama al método busca en el arreglo que devuelve la posición del código buscado
    int p = objArreglo.busca(jtxtCodigo.getText().toUpperCase());
  if (p!=-1){
  int r=JOptionPane.showConfirmDialog(this,"Esta seguro de eliminar a éste registro ?","Reponder",0);
       if (r==0)
       {   objArreglo.elimina(p); // eliminado el objeto en la posición p
           limpiar_entradas(); //Limpiando las entradas
           grabar(); // grabamos la información en el archivo binario
           actualizar_tabla(); //actualizando la tabla
           //llamando al método resumen para mostrar los datos de salida
           resumen();
          jtxtCodigo.requestFocus(); //colocando el cursor en el text Codigo
       }
    }// fin del else
}// fin de elimina
```

```
public void modifica(){
            String cod = jtxtCodigo.getText().toUpperCase();
            // se vuelve a buscar el codigo para no repetir el mismo
               int p=objArreglo.busca(cod);
             //Se leen los datos de entrada de los TextField
             String nom = jtxtNombre.getText().toUpperCase();
             String ti =  jCbxTipo.getSelectedItem().toString();
             String cla=jCbxClase.getSelectedItem().toString();
             int an=Integer.parseInt(jtxtAnio.getText());
             int np=Integer.parseInt(jtxtNroPag.getText());
             double c=Double.parseDouble(jtxtCosto.getText());

             //Generando la clase para manejar un libro
            objLibro = new Libro(cod,nom,ti,cla,an,np,c);
            // Verificando si el código existe dentro del arreglo
               if (p==-1) // código nuevo
                     objArreglo.agrega(objLibro);
               else    // codigo ya existente
                        objArreglo.reemplaza(p,objLibro);
            limpiar_entradas(); //Limpiando las entradas
            grabar(); // grabamos la información en el archivo binario
              actualizar_tabla(); //actualizando la tabla
            //llamando al método resumen para mostrar los datos de salida
            resumen();
           //colocando el cursor en el text Codigo
            jtxtCodigo.requestFocus();
}// fin de modifica

public void consulta(){
   String cod = jtxtCodigo.getText().toUpperCase();
   //Se llama al metodo busca en el arreglo que devuelve la posición
   // del código buscado
   int p = objArreglo.busca(cod);
        if (p==-1){
         mensaje("Codigo no existe");
         limpiar_entradas();    }
        else{
                // se extrae todo el objeto con toda la información
            objLibro = objArreglo.getLibro(p);
            // se extrae la informacion de los campos del objeto
            String nombre = objLibro.getNombre();
            String tipo =  objLibro.getTipo();
            String clase= objLibro.getClase();
            int anio=objLibro.getAnio();
            int num_pag=objLibro.getNum_pag();
            double costo=objLibro.getCosto();
            //colocando la información en los objetos
            jtxtNombre.setText(nombre);
            //colocando la información en el combobox tipo
            if(tipo.equalsIgnoreCase("PROGRAMACION"))
               jCbxTipo.setSelectedIndex(1);
            else if(tipo.equalsIgnoreCase("GESTION"))
                  jCbxTipo.setSelectedIndex(2);
            else if(tipo.equalsIgnoreCase("SISTEMAS"))
                    jCbxTipo.setSelectedIndex(3);
            //colocando la información en el combobox clase
            if(clase.equalsIgnoreCase("A"))
               jCbxClase.setSelectedIndex(1);
            else if(clase.equalsIgnoreCase("B"))
                  jCbxClase.setSelectedIndex(2);
            else if(clase.equalsIgnoreCase("C"))
                    jCbxClase.setSelectedIndex(3);
```

```
            // Información para los text field
            jtxtNombre.setText(nombre);
            jtxtAnio.setText(String.valueOf(anio));
            jtxtNroPag.setText(String.valueOf(num_pag));
            jtxtCosto.setText(String.valueOf(costo));
        }
    }
```

LOS METODOS UTILIZADOS LIMPIAR LAS ENTRADAS Y INSERTAR LA INFORMACION EN LA TABLA

```
void limpiar_entradas() {
        jtxtCodigo.setText("");
        jtxtNombre.setText("");
        jtxtAnio.setText("");
        jtxtNroPag.setText("");
        jtxtCosto.setText("");
        jCbxTipo.setSelectedIndex(0);
        jCbxClase.setSelectedIndex(0);
        jtxtCodigo.requestFocus(true);
    }

void insertar(int num,String codigo,String nombre,String tipo,String clase,int anio,int
num_pag,double costo){
        String co;
        //dando Formato al sueldo
        DecimalFormat df2 = new DecimalFormat("####.00");
       co=df2.format(costo);
Object[] fila={num,codigo,nombre,tipo,clase,String.valueOf(anio),String.valueOf(num_pag),co};
        miModelo.addRow(fila);
}
```

A CONTINUACION SE PRESENTAN LOS BOTONES

```
private void jbtnGrabarActionPerformed(java.awt.event.ActionEvent evt) {
       //Se leen los datos de entrada de los TextField
        String cod = jtxtCodigo.getText().toUpperCase();
        String nom = jtxtNombre.getText().toUpperCase();
        String ti =  jCbxTipo.getSelectedItem().toString();
        String cla=jCbxClase.getSelectedItem().toString();
        int an=Integer.parseInt(jtxtAnio.getText());
        int np=Integer.parseInt(jtxtNroPag.getText());
        double c=Double.parseDouble(jtxtCosto.getText());

        //Generando la clase para manejar un libro
        objLibro = new Libro(cod,nom,ti,cla,an,np,c);
          // Verificando si el código existe dentro del arreglo
        if (objArreglo.busca(objLibro.getCodigo())!=-1)
              mensaje("Codigo Repetido"); // se muestra mensaje
          else
        {   objArreglo.agrega(objLibro); // se agrega el objeto al arreglo
            //Insertando la información en el Tabla
            insertar(0,cod,nom,ti,cla,an,np,c);
           //Limpiando las entradas
            limpiar_entradas();
           //Grabando la información en el archivo binario
            grabar();
           //actualizando la tabla
            actualizar_tabla();
           // llamada al método resumen para ver los datos de salida
```

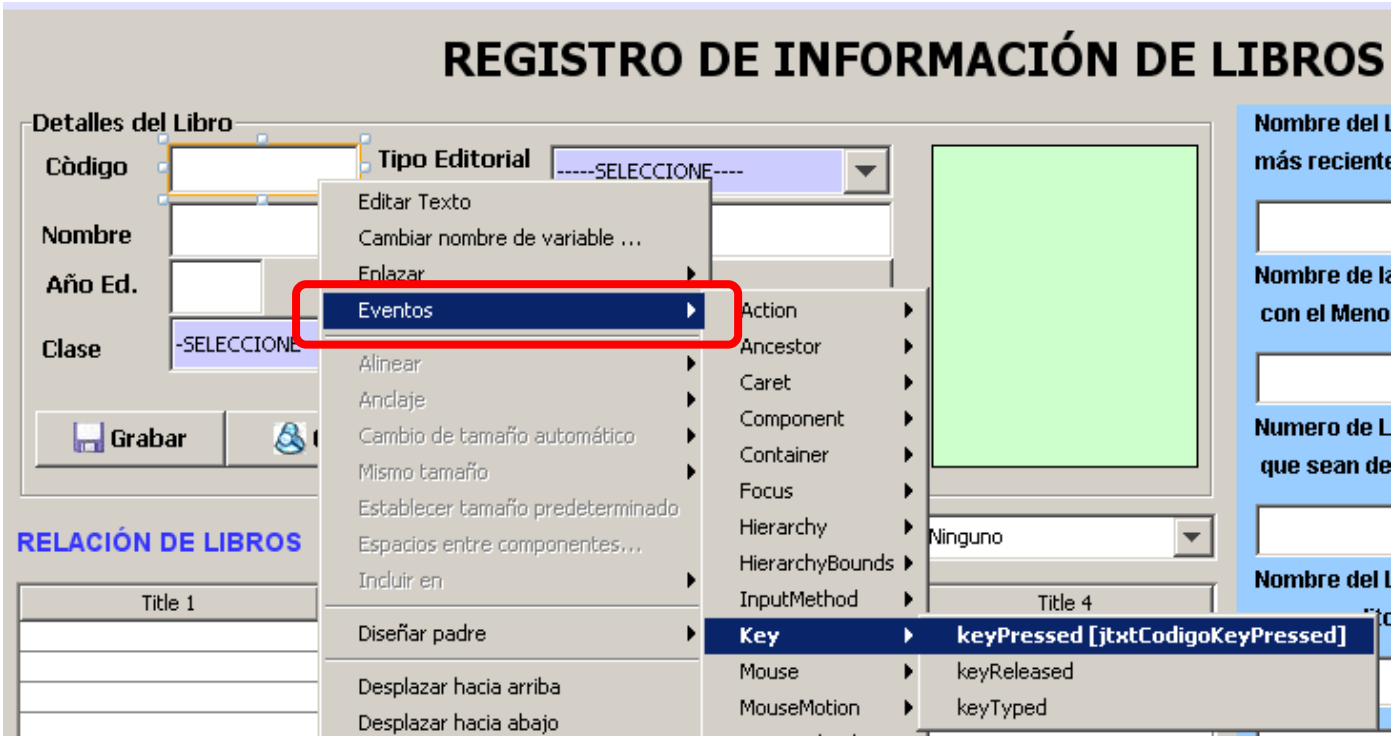
```
        resumen();
        }
    }

private void btnConsultarActionPerformed(java.awt.event.ActionEvent evt) {
            consulta();
}

private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {
        modifica();
}

private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
        eliminar();
}
```

PARA LOS EVENTOS DE TECLADO PARA EVALUAR LA TECLA ENTER SELECCIONE CUALQUIER OBJETO Y HAGA LO SIGUIENTE Y COMPLETE LOS CODIGOS




```
private void jtxtNombreKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==evt.VK_ENTER)
        jCbxTipo.requestFocus();
    }

    private void jCbxTipoKeyPressed(java.awt.event.KeyEvent evt) {
      if(evt.getKeyCode()==evt.VK_ENTER)
        jtxtAnio.requestFocus();
    }

    private void jtxtNroPagKeyPressed(java.awt.event.KeyEvent evt) {
        if(evt.getKeyCode()==evt.VK_ENTER)
            jtxtCosto.requestFocus();
    }

    private void jtxtAnioKeyPressed(java.awt.event.KeyEvent evt) {
```

```
    if(evt.getKeyCode()==evt.VK_ENTER)
        jtxtNroPag.requestFocus();
}

private void jtxtCostoKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==evt.VK_ENTER)
       jCbxClase.requestFocus();
}

private void jCbxClaseKeyPressed(java.awt.event.KeyEvent evt) {
   if(evt.getKeyCode()==evt.VK_ENTER)
        jbtnGrabar.requestFocus();
}
```

Guía Práctica 12

Resolver todos los aplicativos con las opciones de actualizar, eliminar y consultar registros

1. Construya un programa que guarde la siguiente información en un archivo binario llamado "Competencia.bin" de los competidores de lanzamiento de jabalina Código, Nombre del participante, Nacionalidad, Categoría (infantil, juvenil, adulto), edad y distancia del lanzamiento. Su programa deberá mostrar los siguientes datos:

   a) Nombre del participante más joven.
   b) Nombre de los ganadores en cada una de las categorías.
   c) Nacionalidad del participante que tuvo el menor lanzamiento.
   d) Buscar, Modificar y Eliminar un registro en base al código.

2. La Oficina de Servicios Académicos de la UCV cuenta con la siguiente información: código, nombres del alumno, su foto, nota de práctica, control de lectura, nota de investigación, promedio (dato calculado), sexo, semestre (2009-II, 2010-I y 2010-II) y escuela (Sistemas, Industrial y Ambiental). La Universidad desea los siguiente datos estadísticos:

   - Número de estudiantes de cuyo promedio no pasa de la nota aprobatoria (10.5)
   - La escuela que donde se encuentra el alumno con el menor promedio.
   - Número de Estudiantes de la Escuela de Ingeniería de Escuela de Ingeniería de Sistemas en el semestre 2010-I y 2010-II aprobaron la nota de investigación.
   - Porcentaje de estudiantes que desaprobaron las notas de control de lectura y de practica de la Escuela de Ingeniería de Ambiental en el semestre 2009-II.

3. Construir un programa que permita calcular la planilla de la Empresa Boxers S.A. que tiene N empleados. La empresa cuenta con la siguiente información: Nombres del empleado y horas trabajadas, y el pago por hora y las bonificaciones está de acuerdo a las siguientes tablas:

Puesto	Pago Hora
Capataz	55.00
Maestro	42.00
Operario	25.00

Modalidad	% de Bonificación
Estable	25%
Contratado	10%

Además al empleado se le puede hacer a lo mucho 3 descuentos por planilla, y de acuerdo a la siguiente tabla:

Descuento	% de Descuento
AFP	3.5%
SEGURO	3.0%
FONAVI	2.5%

Determinar el monto total que la empresa gastara en pagar a sus empleados y cuanto pagara según el Puesto.

4. Una empresa tiene N empleados para cada uno de los cuales registra los siguientes datos: Nombre, Sueldo base, Ventas realizadas, Estado civil, Número de Hijos y Sueldo Neto.

   El programa debe calcular para cada empleado el sueldo neto mediante la relación siguiente:

   Sueldo Neto = Sueldo Base + Comisión por Ventas – Descuento por Impuesto – Descuento por Seguro.

   Donde: la comisión por ventas es el 5% de las ventas realizadas
   El descuento por seguro se obtiene como sigue:
   - Si el empleado es soltero el descuento es de S/. 100.
   - Si el empleado es casado sin hijos el descuento es de S/. 120.
   - Si el empleado es casado con hijos el descuento es de S/. 50 + S/. 70 por cada hijo.

   Suponga que llamamos TA a la suma del Sueldo base + comisión por ventas, entonces el Descuento por el Impuesto (DI) se obtiene como sigue:

Rango del TA	Monto de DI
0 a 1500	0
1500 a 2300	3% del TA
2301 a 3000	4% del TA
3001 a mas	6% del TA

   El programa debe mostrar el monto total que la empresa debe pagar por concepto de sueldos, el monto total por comisiones de ventas, el monto total de los descuentos por impuesto y por seguro.

5. Construya un programa que guarde la siguiente información de los registros de llamada: Tipo de Plan (A o B), tipo de llamada (local, celular A, celular B y celular C), horario de llamada (mañana-tarde, noche, madrugada), minutos de llamada, costo por minuto, costo por llamada (calculado de la multiplicación de minutos de llamada por costo por minuto).
   Las tarifas de las llamadas se dan de la siguiente manera:

TIPO PLAN	TIPO DE LLAMADA (Costo x minuto)			
	Local	Celular A	Celular B	Celular C
Tipo A	0.20	0.40	0.45	0.70
Tipo B	0.15	0.30	0.35	0.60

   Por otro lado el costo por minuto se reducirá en un 35% para llamadas locales y 25% para celulares si es que se da en el horario de la madrugada
   Su programa deberá mostrar los siguientes datos:

   a) Número de llamadas con más de 5 minutos y menos de 20 en horario de la noche.
   b) Costos totales de llamada según Tipo de Plan.
   c) Total de minutos realizados en el turno de la noche y madrugada a teléfonos locales con costo de llamada superiores a 10 soles.
   d) En que horario y tipo de Plan se realizo la llamada con mayor costo de llamada
   e) Total de llamadas en minutos realizadas al celular C en horario de mañana-tarde.

6. Una empresa guarda la siguiente información de sus empleados: Nombres, Apellido Paterno, Apellido Materno, Área al que pertenece (Planificación, Ventas y Marketing), Tiempo de Servicio (en años), Condición (Contratado o Estable), Sueldo Base, Asignación de Movilidad, Nº de Hijos, descuento de AFP (17.5% del sueldo base) y Sueldo Final.
   Para la asignación movilidad es el porcentaje del sueldo base y se asigna de acuerdo al área al que pertenece y la condición del empleado tal y como se muestra en la siguiente tabla:

Condición	Área		
	Planificación	Ventas	Marketing

Contratado	7%	8%	12%
Estable	9%	12%	16%

Además por cada hijo del empleado la empresa le otorga 55 soles. El sueldo final se calcula de la siguiente fórmula:

Sueldo Final = Sueldo Base + Asignación de Movilidad + Nº de Hijos * 55 - 17.5% del Sueldo Base

Construir un programa que pueda proporcionar la siguiente información:

a) Nombre del empleado que tenga el menor sueldo final que sea del área de Ventas que tengan entre 2 y 10 años de Servicio.
b) Promedio de Salarios de los empleados contratados del área de Marketing.
c) Número de empleados estables que ganan más de S/.2500 y menos de S/ 3500 con más de 5 años de servicio y con más de 3 hijos.
d) Porcentaje de empleados estables del área de Marketing con sueldo menores a S/. 1200 y asignación de movilidad superior a 60 soles
e) Área que tiene el empleado estable con el mayor tiempo de servicio sin hijos.