Documentación Proyecto final

Enunciado del Proyecto

Creación del software para una agencia de alquiler de vehículos aplicando la Programación orientada a objetos.

Competencias a evaluar

Modelación y creación de clases, asignación de atributos y responsabilidades. Herencia y Polimorfismo. Interfaces. Vista/swing. Archivos.

Enunciado

Una Agencia de alquiler de vehículos dispone de Camiones y de vehículos tipo turismo. Todos los vehículos tienen una matrícula que los identifica y un atributo alquilado que indica si el vehículo está o no alquilado (verdadero o falso). Además, sobre ellos se realizan las siguientes acciones: alquilar, la cual informa si se pudo o no realizar el alquiler; devolver, la cual informa el coste que debe recibir la agencia por el alquiler; por último mostrar Información, como su nombre lo indica, muestra la información del vehículo respectivo.

- El alquiler de vehículos tipo turismo se factura por kilómetros, por esta razón, para éste tipo de vehículo se maneja un precio por kilómetro y se tiene en cuenta el kilometraje del auto al momento de alquilarlo y al momento de la devolución.
- El alquiler de Camiones se factura por días, por ello, para éste tipo de vehículo se tiene en cuenta el precio por día, la fecha en la que se alquila y la fecha en la cual se devuelve.

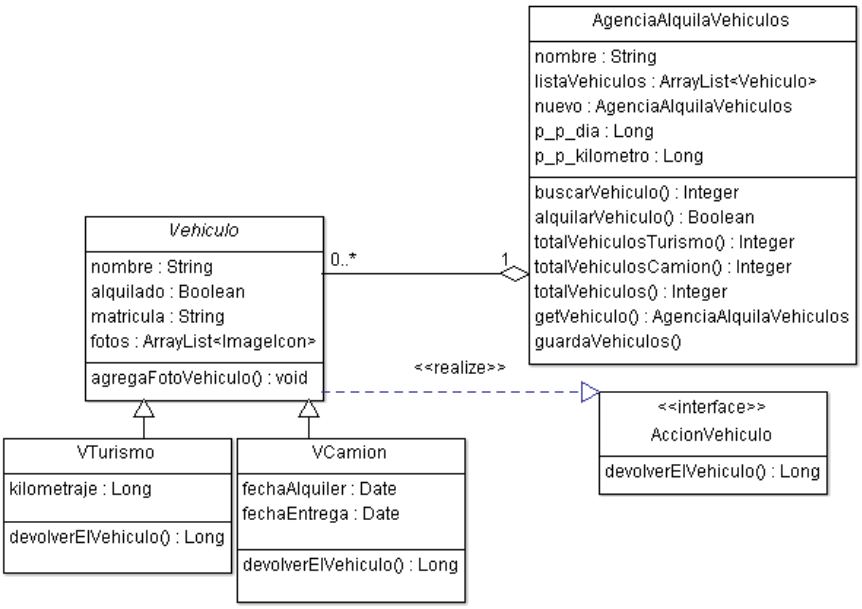
La agencia cuenta con procedimientos que le permiten saber en cualquier momento el número total de cada tipo de vehículo que posee, así como el número y listado de alquilados y disponibles.

Notas:

- Para los camiones maneje todos los meses de 30 días. Además, los días se cuentan como números enteros a partir del 1 de Enero de 2000.
- Para mostrar Información tenga en cuenta: el tipo de vehículo, la matrícula, si el vehículo está alquilado o no, el precio por día o por km.

Universidad de Cartagena, Programa de Ingeniería de Sistemas, P.O.O. Estudiante[1], Docente[2].
Padilla F. David[1], Rodríguez V. Andrés[1], Moya V. Yasmín[2].

Para comenzar a escribir el código, antes es necesario tener el diagrama de clases desarrollado en UML, el cual debe contemplar todos los requisitos funcionales del sistema. Para realizar el diagrama puede usar ArgoUML (Que fue el utilizado para este proyecto), disponible en el laboratorio donde se desarrollan las clases y de uso gratuito en idioma español. Disponible en: http://argouml.tigris.org/



En el diagrama se puede apreciar lo siguiente:

Existe una clase AgenciaAlquilaVehiculos, la cual administra la Agencia, contiene su nombre y el ArrayList de vehículos que administra.

Existe la clase Vehiculo que implementa la interfaz AccionVehiculo.

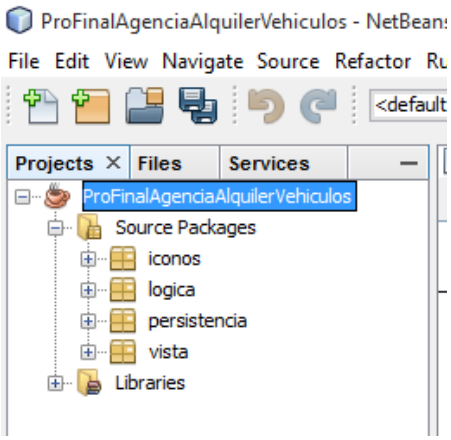
Están las clases Vturismo y VCamion que heredan de Vehiculo (o extienden).

¿Por qué se implementó una interfaz?

R/ El taller decía la implementación de una o varias interfaces.

Después de haber realizado el diagrama se procede al desarrollo del mismo en un lenguaje de programación orientado a objetos, en este caso se utilizará el entorno de desarrollo NetBeans IDE 8.0, con el JDK 1.8 y el JRE 1.8.

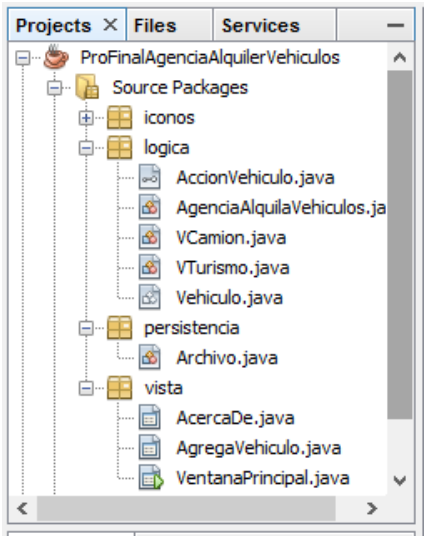
Al final de este capítulo, la estructura debe quedar así como se muestra en esta imagen.



El primer paso es crear el proyecto y ubicarlo en una carpeta con su mismo nombre así: proFinalAgenciaAlquilerVehiculos. Luego se crean distintos paquetes para mantener separada la parte lógica de la vista y de la persistencia así:

- Lógica: donde se deben crear todas las clases del diagrama de clases, incluyendo la interfaz.
- vista: donde quedarán todas las clases que conforman la vista, incluyendo el main.
- persistencia: aquí van todas las clases que funcionan como interfaz entre el software y el sistema operativo para almacenar o hacer persistir los atributos de los objetos creados a través del software. En este caso, la información de los objetos se almacena en archivos.
- Iconos (opcional): es donde almacenamos las imágenes que utilizaremos en los botones y labels como iconos o simples imágenes.

Luego de creados los paquetes y sus respectivas clases, la estructura debe quedar así:



Respecto a la codificación

Se agregan los atributos de la clase AgenciaAlquilaVehiculos. Se crea el atributo nuevo que es donde se almacenará toda la agencia y se declara **public static** para su manejo dentro del proyecto.

```
public class AgenciaAlquilaVehiculos implements Serializable{
private String nombre;
private ArrayList<Vehiculo> listaVehiculos;
private long p_p_dia;
private long p_p_kilometro;
public static AgenciaAlquilaVehiculos nuevo = null;
```

Serializar un objeto es el proceso de convertirlo a bytes, para poder enviarlo por una red o guardarlo en un archivo/fichero, y reconstruirlo luego a partir de esos bytes.

Para que un objeto sea serializable basta con que implemente la interfaz Serializable.

Si dentro de la clase hay atributos que son otras clases, éstos a su vez también deben ser Serializable. Con los tipos de java/clases envolventes no hay problema porque lo son. Si ponemos como atributos nuestras propias clases, éstas a su vez deben implementar Serializable. Por ejemplo:

```
public class Info implements Serializable{
public int x;
public String y;
public boolean z;
}
```

La clase InfoAdd es Serializable porque implementa Serializable y todos sus atributos son Serializable, incluido "Info f;"

```
public class InfoAdd implements Serializable{
public int a;
public Integer i;
Info f;
}
```

Es posible que sea necesario hacer algo especial en el momento de serializar el objeto, bien al construirlo a bytes, bien al recibirlo. Por ello java permite hacerlo. Basta con hacer que el objeto defina uno o los dos métodos siguientes:

```
private void readObject(java.io.ObjectInputStream stream)
 throws IOException, ClassNotFoundException{
// Aqui debemos leer los bytes de stream y reconstruir el objeto
}
```

```
private void writeObject(java.io.ObjectOutputStream stream)
throws IOException{
// Aquí escribimos en stream los bytes que queramos que se envíen por red.
// o que se almacenen en un archivo
 }
```

Java llamará a estos métodos cuando necesite convertir el objeto a bytes para enviarlo por red/almacenarlo en un archivo o cuando necesite reconstruirlo a partir de los bytes en red/recuperarlo del archivo.

Convertir un serializable a byte[] y viceversa Es posible convertir cualquier objeto Serializable a un array de byte y viceversa. Normalmente esto no es necesario hacerlo explícitamente en el código para enviar el objeto por un socket o escribirlo en un archivo puesto que existen las clases ObjectInputStream y ObjectOutputStream que se encargan de ello. Para hacer esta conversión, se puede usar este código:

De objeto a byte[]

```
ByteArrayOutputStream bs= new ByteArrayOutputStream();
ObjectOutputStream os = new ObjectOutputStream (bs);
os.writeObject(unObjetoSerializable);
os.close();
byte[] bytes =  bs.toByteArray(); // devuelve byte[]
```

De byte[] a objeto

```
ByteArrayInputStream bs= new ByteArrayInputStream(bytes); // bytes es el
// byte[]
ObjectInputStream is = new ObjectInputStream(bs);
ClaseSerializable unObjetoSerializable = (ClaseSerializable)is.readObject();
is.close();
```

Una utilidad de estos dos códigos es el realizar copias "profundas" de un objeto, es decir, se obtiene copia del objeto, copias de los atributos y copias de los atributos de los atributos. Basta convertir el objeto a byte[] y luego reconstruirlo en otra variable. (Fuente: http://chuwiki.chuidiang.org/index.php?title=Serializaci%C3%B3n_de_objetos_en_java).

Continúa codificación

Universidad de Cartagena, Programa de Ingeniería de Sistemas, P.O.O. Estudiante[1], Docente[2].
Padilla F. David[1], Rodríguez V. Andrés[1], Moya V. Yasmín[2].

Constructores: el método constructor está sobrecargado. Dos formas distintas para la instanciación de la clase AgenciaAlquilaVehiculos.

```
public AgenciaAlquilaVehiculos (){
      this.listaVehiculos = new ArrayList<Vehiculo> ();
   }

   public AgenciaAlquilaVehiculos(String nombre) {
      this.nombre = nombre;
      this.listaVehiculos = new ArrayList<Vehiculo> ();
   }
```

Getter's y Setter's

```
public String getNombre() {
      return nombre;
   }

   public void setNombre(String nombre) {
      this.nombre = nombre;
   }

   public ArrayList<Vehiculo> getListaVehiculos() {
      return listaVehiculos;
   }

   public void setListaVehiculos(ArrayList<Vehiculo> listaVehiculos) {
      this.listaVehiculos = listaVehiculos;
   }
   public long getP_p_dia() {
      return p_p_dia;
   }

   public void setP_p_dia(long p_p_dia) {
      this.p_p_dia = p_p_dia;
   }

   public long getP_p_kilometro() {
      return p_p_kilometro;
   }

   public void setP_p_kilometro(long p_p_kilometro) {
      this.p_p_kilometro = p_p_kilometro;
   }
```

Métodos que resuelven requisitos funcionales para en la clase AgenciaAlquilaVehiculos

Método para buscar vehículos en el **ArrayList**, recibe la matrícula y el ArrayList como parámetros y buscamos si el Vehiculo existe retorna la posición, caso contrario retorna el valor -1.

```
public int buscarVehiculo (String mat, ArrayList<Vehiculo> alv){
    for (int i=0; i<alv.size(); i++){
        if (alv.get(i).getMatricula().equals(mat.toUpperCase())){
            return i;
        }
    }
    return -1;
  }
```

Método para alquilar un Vehiculo, recibe un entero como parámetro que es la posición en el ArrayList del Vehiculo, comprueba de que esté disponible y realiza la acción, retorna **true** si realizo la operación con éxito o **false** si no.

```
public boolean alquilarVehiculo (int x){
    if (listaVehiculos.get(x).alquilado == true){
        //invoco un error
        return false;
    }else{
        listaVehiculos.get(x).alquilado = true;
        if (listaVehiculos.get(x) instanceof VCamion){
            VCamion aux = new VCamion (new Date(), null,
listaVehiculos.get(x).getNombre(), true, listaVehiculos.get(x).getMatricula());
            listaVehiculos.set(x, aux);
        }
        return true;
    }
  }
```

Método para calcular el total de vehículos de tipo turismo que están almacenados en la agencia, retorna la cantidad de vehículos de turismo.

```
public int totalVehiculosTurismo (){
    int cont=0;
    for (int i=0; i<listaVehiculos.size(); i++){
        if (listaVehiculos.get(i) instanceof VTurismo) cont++;
```

```
        }
        return cont;
    }
```

Método para calcular el total de vehículos de tipo camión que están almacenados en la agencia, retorna la cantidad de vehículos de camión.

```
public int totalVehiculosCamion (){
        int cont=0;
        for (int i=0; i<listaVehiculos.size(); i++){
            if (listaVehiculos.get(i) instanceof VCamion) cont++;
        }
        return cont;
    }
```

Método para calcular el total de vehículos que están almacenados en la agencia, retorna la cantidad de vehículos.

```
public int totalVehiculos (){
        return listaVehiculos.size();
    }
```

Método que devuelve la variable donde está almacenada toda la agencia para su uso en todo el programa.

```
public static AgenciaAlquilaVehiculos getVehiculo (){
        if (nuevo == null){
            nuevo = new AgenciaAlquilaVehiculos();
        }
        return nuevo;
    }
```

Método para agregar vehículos al ArrayList de la agencia.

```
public void guardaVehiculos (Vehiculo vh){
        ArrayList<Vehiculo> alv = nuevo.getListaVehiculos();
        alv.add(vh);
        nuevo.setListaVehiculos(alv);
    }
} //Fin clase AgenciaAlquilaVehiculo
```

Atributos, métodos constructores, analizadores y modificadores de la clase Vehiculo

La clase Vehiculo será extendida por las clases VTurismo y VCamion, por ello sus atributos tendrán un nivel de **protected,** al mismo tiempo que implementa Serializable por los motivos anteriormente señalados y AccionVehiculo que es la interfaz que sobrescribiremos su método.

```
public abstract class Vehiculo implements AccionVehiculo, Serializable {
    protected String nombre;
    protected boolean alquilado;
    protected String matricula;
    protected ArrayList<ImageIcon> fotos;
```

Métodos constructores de la clase Vehiculo

```
public Vehiculo (){
        this.fotos = new ArrayList<ImageIcon> ();
    }

    public Vehiculo(String nombre, boolean alquilado, String matricula) {
        this.nombre = nombre;
        this.alquilado = alquilado;
        this.matricula = matricula;
        this.fotos = new ArrayList<ImageIcon> ();
    }
```

Analizadores y modificadores

```
public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public boolean isAlquilado() {
        return alquilado;
    }

    public void setAlquilado(boolean alquilado) {
        this.alquilado = alquilado;
    }

    public String getMatricula() {
        return matricula;
```

```
    }

    public void setMatricula(String matricula) {
        this.matricula = matricula;
    }
    public ArrayList<ImageIcon> getFotos() {
        return fotos;
    }

    public void setFotos(ArrayList<ImageIcon> fotos) {
        this.fotos = fotos;
    }

    public void agregaFotoVehiculo (ImageIcon img, int i){
          ArrayList<ImageIcon> ali =
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(i).fotos;
        ali.add(img);
        AgenciaAlquilaVehiculos.nuevo.getListaVehiculos().get(i).setFotos(ali);
    }
} //Fin clase Vehiculo
```

Atributos, métodos constructores, analizadores y modificadores de la clase VCamion

La clase VCamion hereda de Vehiculo.

```
public class VCamion extends Vehiculo {
    private Date fechaAlquiler;
    private Date fechaEntrega;
```

Continuando con el código

Constructores

```
public VCamion (){
        super();
    }


    public VCamion(Date fechaAlquiler, Date fechaEntrega, String nombre,
boolean alquilado, String matricula) {
        super(nombre, alquilado, matricula);
        this.fechaAlquiler = fechaAlquiler;
        this.fechaEntrega = fechaEntrega;
```

```
    }
```

Analizadores y modificadores

```
public Date getFechaAlquiler() {
        return fechaAlquiler;
    }

    public void setFechaAlquiler(Date fechaAlquiler) {
        this.fechaAlquiler = fechaAlquiler;
    }

    public Date getFechaEntrega() {
        return fechaEntrega;
    }

    public void setFechaEntrega(Date fechaEntrega) {
        this.fechaEntrega = fechaEntrega;
    }
```

Métodos de la clase VCamion

Métodos con @Override que son sobrescritos uno de la clase String de Java y el otro de la interfaz AccionVehiculo.

```
@Override
    public long devolverElVehiculo (ArrayList<Vehiculo> al, int x, long nk){
        VCamion aux = (VCamion)al.get(x);
        aux.fechaEntrega = new Date ();
        aux.alquilado = false;
        AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().add(x, aux);
        return (aux.fechaAlquiler.getTime() - aux.fechaEntrega.getTime() + 1) *
AgenciaAlquilaVehiculos.getVehiculo().getP_p_dia();
    }
    @Override
    public String toString() {
        return "Vehículo tipo: Camion" + "   Precio por día: " +
AgenciaAlquilaVehiculos.getVehiculo().getP_p_dia();
    }
} //Fin clase VCamion
```

Atributos, métodos constructores, analizadores y modificadores de la clase VTurismo

```
public class VTurismo extends Vehiculo {
   private long kilometraje;
```

Constructores

```
public VTurismo (){
      super();
   }

   public VTurismo(long kilometraje, String nombre, boolean alquilado, String
matricula) {
      super(nombre, alquilado, matricula);
      this.kilometraje = kilometraje;
   }
```

Analizadores y modificadores

```
public long getKilometraje() {
      return kilometraje;
   }

   public void setKilometraje(long kilometraje) {
      this.kilometraje = kilometraje;
   }
```

Métodos de la clase VTurismo

```
@Override
   public long devolverElVehiculo (ArrayList<Vehiculo> al, int x, long nk){
      VTurismo aux = (VTurismo)al.get(x);
      return (nk - aux.getKilometraje()) *
AgenciaAlquilaVehiculos.getVehiculo().getP_p_kilometro();
   }

   @Override
   public String toString() {
      return "Vehículo tipo: Turismo" + "     Precio por Kilometro: " +
AgenciaAlquilaVehiculos.getVehiculo().getP_p_kilometro();
   }
}
```

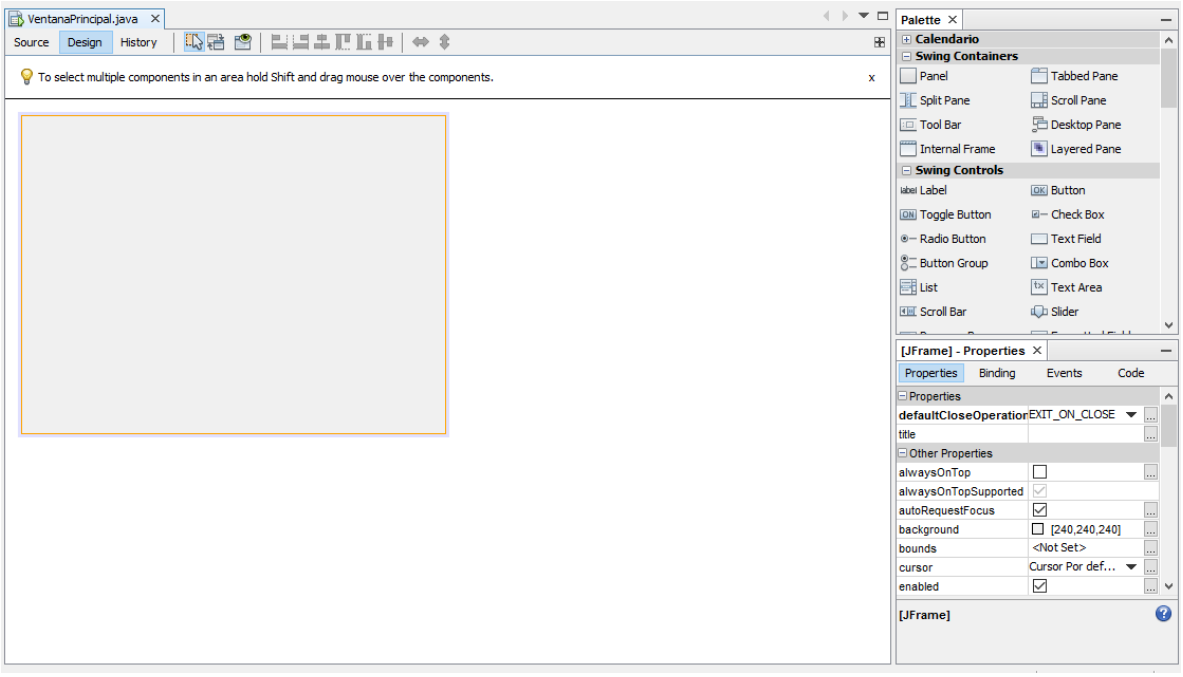
La interfaz AccionVehiculo

```
public interface AccionVehiculo {
```

```
    long devolverElVehiculo (ArrayList<Vehiculo> al, int x, long nk);
}
```

Creando la vista del software

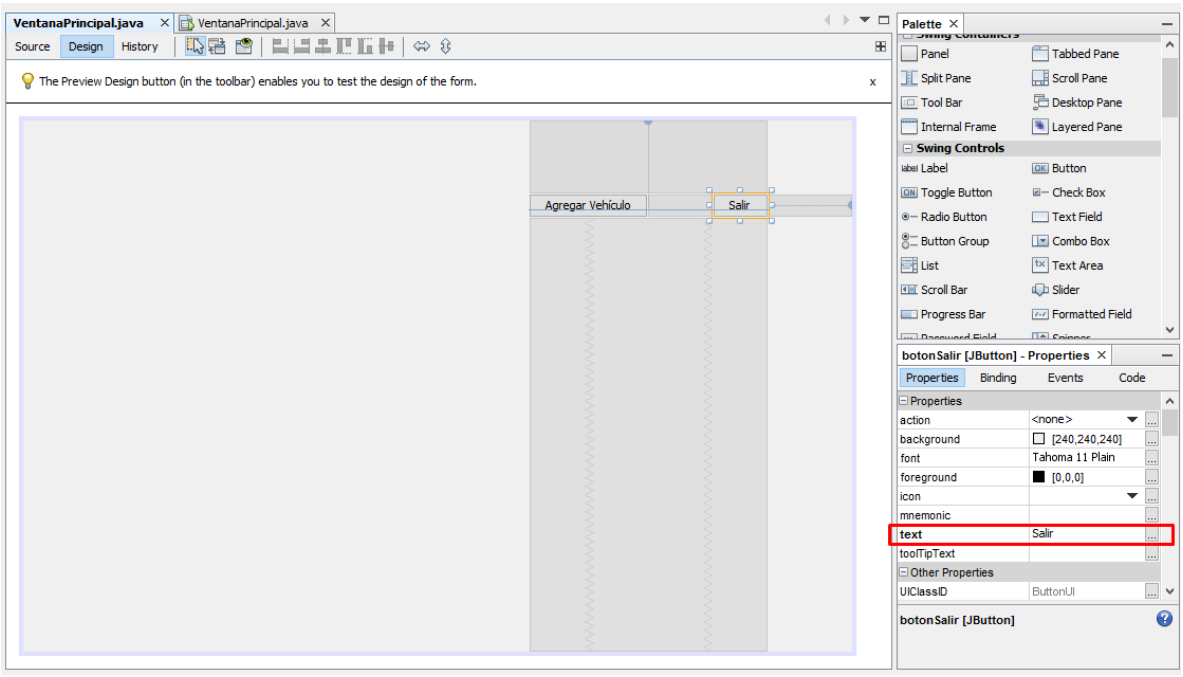
- Crear el paquete vista
- Crear un Jframe llamado VentanaPrincipal



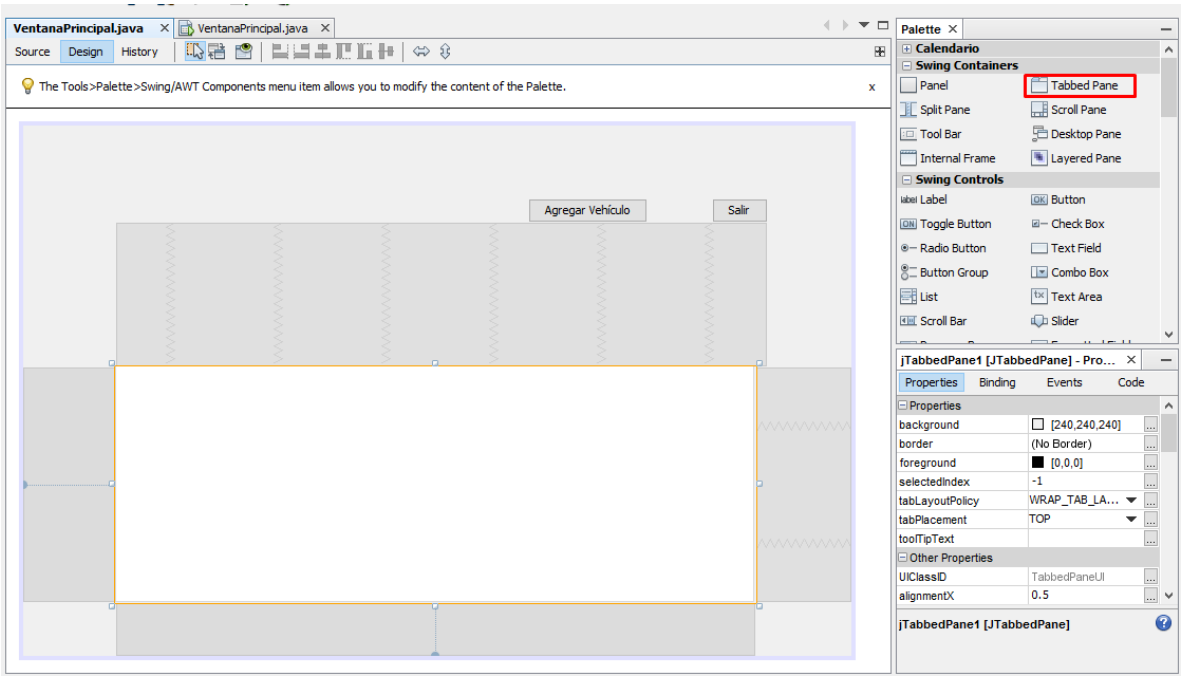
Luego de crear el JFrame agregamos dos botones, arrastramos desde la pestaña que tenemos a la derecha llamada “Palette” en la parte de “Swing Controls” y “Button”. Podemos modificar el texto de los botones y cambiarles el nombre de la variable en la parte de debajo de Palette, donde está ubicada la pestaña de “Propiedades”, buscamos la parte donde dice “text” y procedemos a ponerle el nombre deseado, en nuestro caso le pondremos a un botón “Salir” y al otro “Agregar Vehículo”. La parte del nombre de la variable para cambiarle el que trae por defecto, le hacemos clic derecho en el botón y seleccionamos la opción “Change Variable Name…”, nos saltará una pequeña ventana e introduciremos el nombre que le queramos dar a la variable, en nuestro caso usaremos “botonSalir” y “botonAgregaVehiculo”.

Nota1: Si quieren agregarle un icono al botón ver video tutorial: https://www.youtube.com/watch?v=i9ryooREVz8

Nota2: Si quieren agregarle una imagen al software ver video tutorial: https://www.youtube.com/watch?v=oWwEZxXxHBs

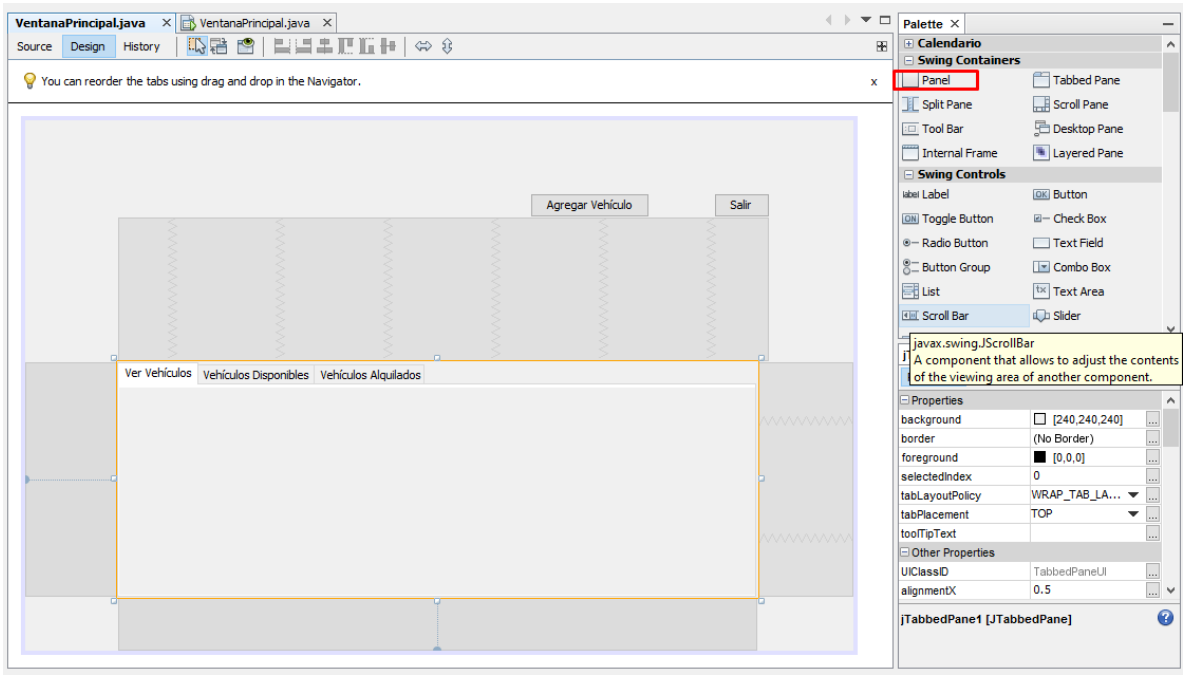


Ahora necesitamos tres paneles en la parte media hacia abajo donde ubicaremos las tres pestañas, nos vamos nuevamente a la Palette de Swing Containers y buscamos donde dice “Tabbed Pane” y arrastramos, esto para poder agregarle más pestañas (o paneles).

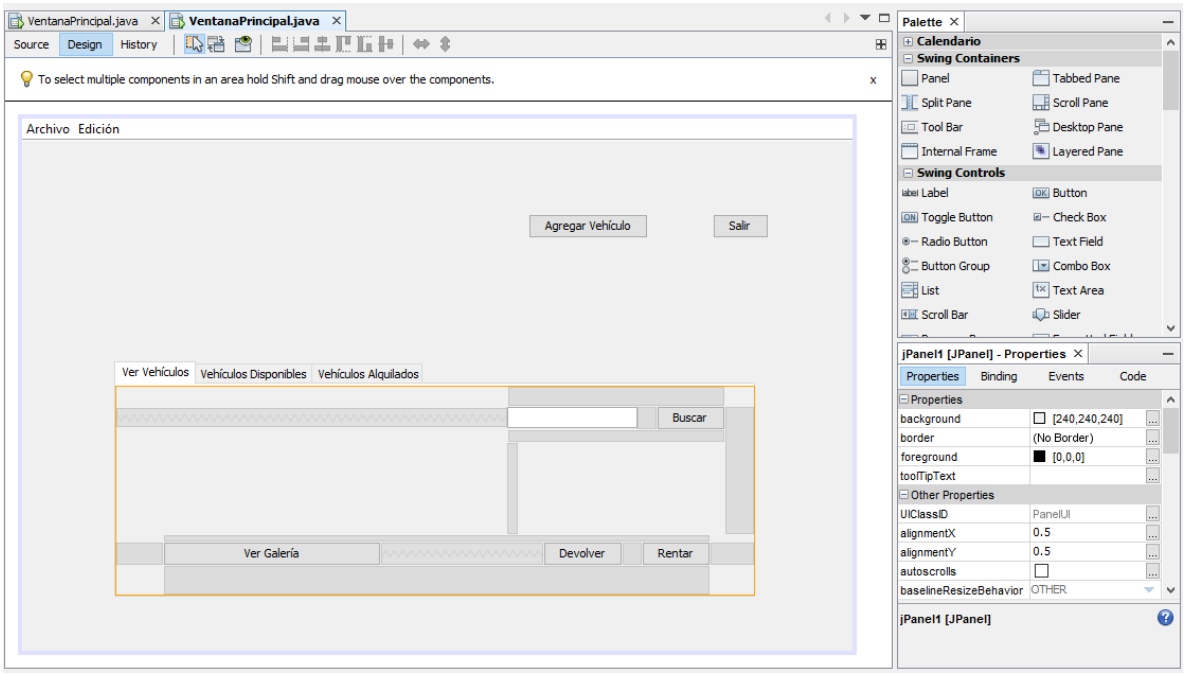


Procedemos con los siguientes paneles, agregamos tres más desde la misma Palette Swing Containers(Arrastramos y los dejamos caer sobre la línea de arriba del Tabbed Pane para que quede a modo de pestaña), que completarían las tres

pestañas que necesitamos. También podemos cambiarles su texto seleccionándolas y haciendo un clic en donde aparece “tab1”, “tab2” y “tab3”, en la primera pondremos “Ver Vehículo”, en la segunda “Vehículos disponibles” y en la tercera “Vehículos Alquilados”.

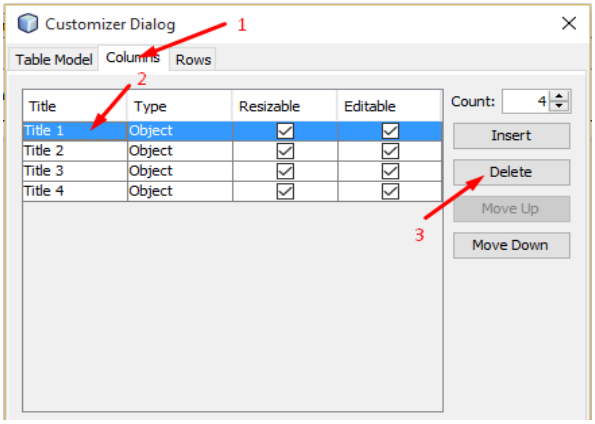


Seguimos poniéndole desde la Palette de Swing Controls dos “Text Field” y cuatro botones más para la pestaña de “Ver Vehículos” un botón lo llamaremos en su texto “Buscar” otro “Rentar”, el tercero “Devolver” y el ultimo “Ver Galería”, hacemos lo mismo que en la parte anterior y modificamos sus nombres de variables, quitándoles las que traen por defecto y asignándoles “botonBuscar”, “botonAgregaRenta”, “botonDevolverVehiculo” y “botonVerGaleria” respectivamente y a los Text Field lo nombramos con el nombre de variable “textMtricula” y “labelImage”. Los ubicamos de esta forma.



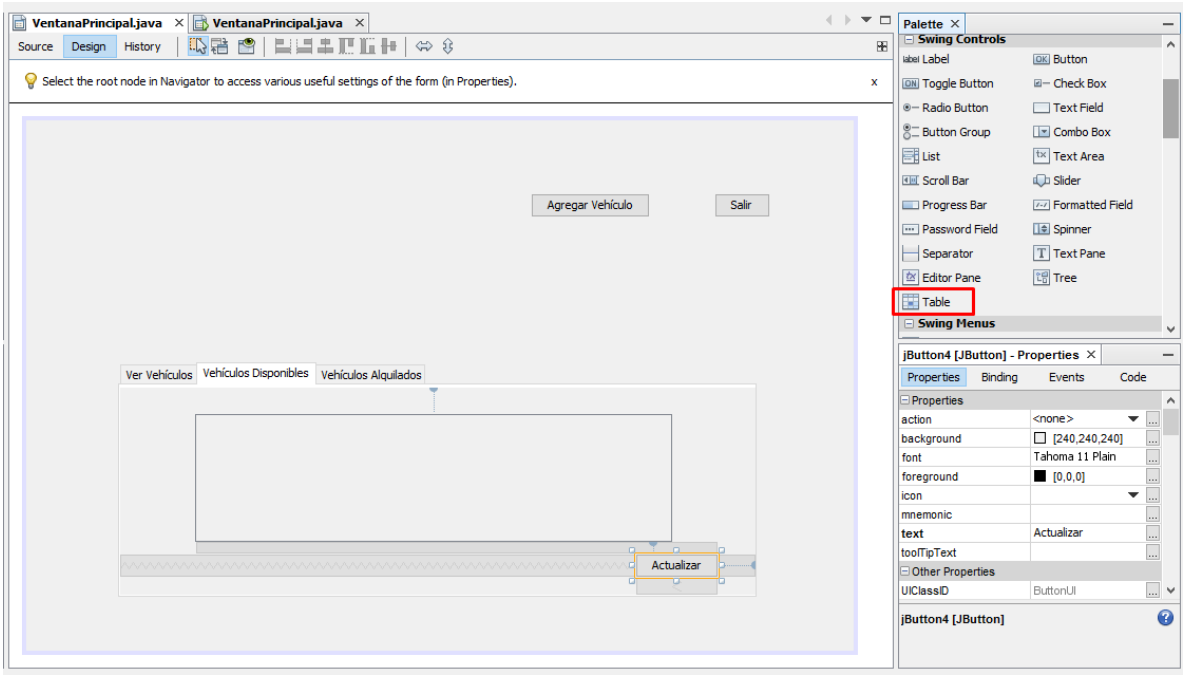
Luego de esto agregamos en la parte izquierda de la misma pestaña “Ver Vehículos” cuatro Labels y los dejamos vacíos, es donde visualizaremos información de los vehículos. Les cambiamos el nombre a cada Label y usamos “muestraNombre”, “muestraMatricula”, “muestraTipo” y “muestraEstado”.

Avanzamos a la siguiente pestaña de “Vehículos Disponibles” seleccionándola y nuevamente en la Paletta de Swing Controls buscamos al final donde está “Table” y agregamos una, la restauramos al tamaño necesario para que quede dentro de la pestaña. Como necesitamos llenaremos esa tabla en tiempo de ejecución del programa la necesitamos que esté vacía (Sin filas y sin columnas), así que le damos clic derecho y seleccionamos la primera opción que es “Table Contents…”, buscamos “Columns” y las eliminamos una por una “Delete”.



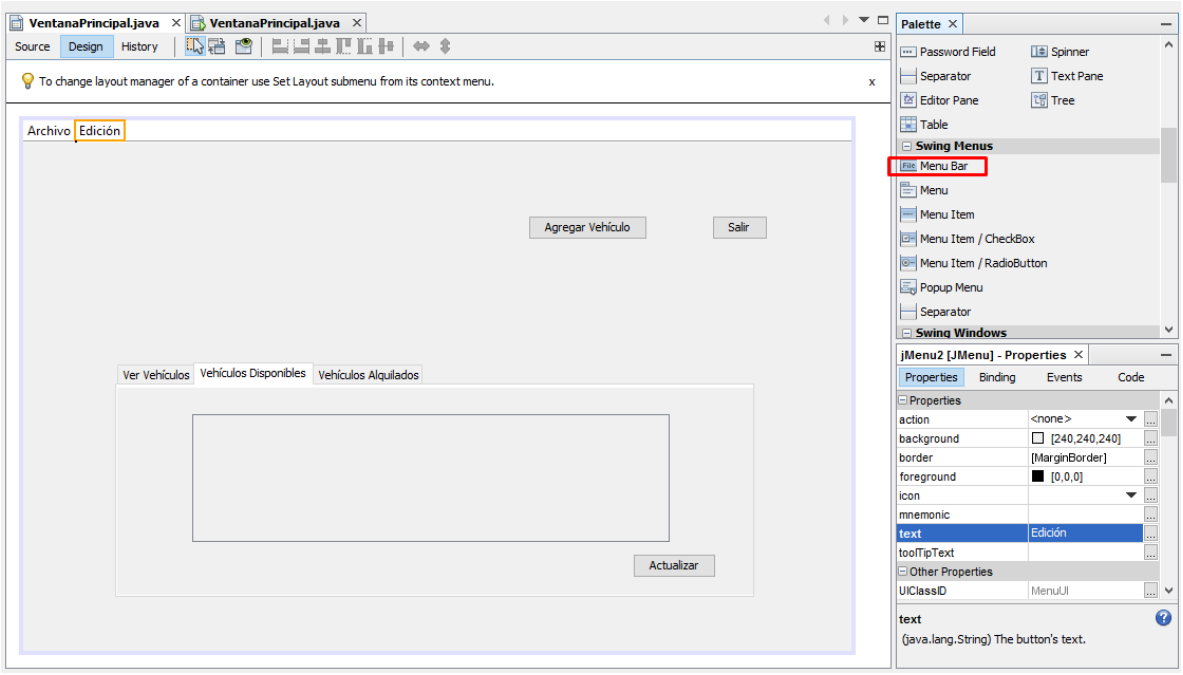
Lo otro que necesitamos para esta pestaña es un botón, así que incluimos uno lo nombramos “Actualizar” y lo ubicamos en la parte de abajo, procedemos a cambiar

el nombre de sus variables y a la tabla la llamamos “tablaDispoibles” y al botón “botonActualizar1”. Quedará de la siguiente forma.

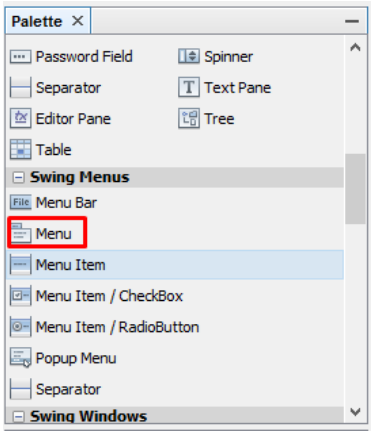


Seguimos y hacemos lo mismo en la siguiente pestañas “Vehículos Alquilados”, la tabla y el botón actualizar, solo que a estos al momento de cambiarles el nombre de la variable les ponemos “tablaAlquilados” y “botonActualizar2”.

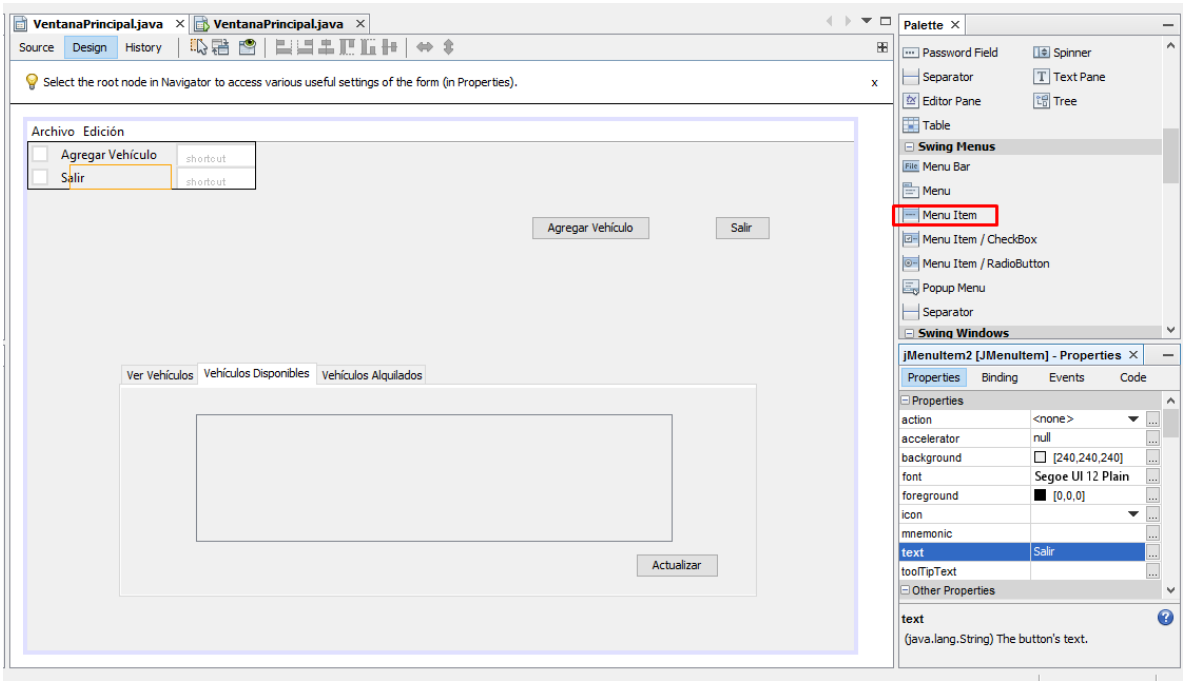
Lo siguiente que hacemos es agregarle una barra de menú a la ventana, volvemos a la Palette, esta vez a la parte de “Swing Menus”, cogemos el primero que es “Menu Bar”, arrastramos y dejamos en la parte superior del programa. Podemos también cambiarle su nombre desde la parte de propiedades a las opciones del menú, quitándoles las que trae por defecto y poniéndoles “Archivo” y “Edición” respectivamente. Se verá de la siguiente manera.

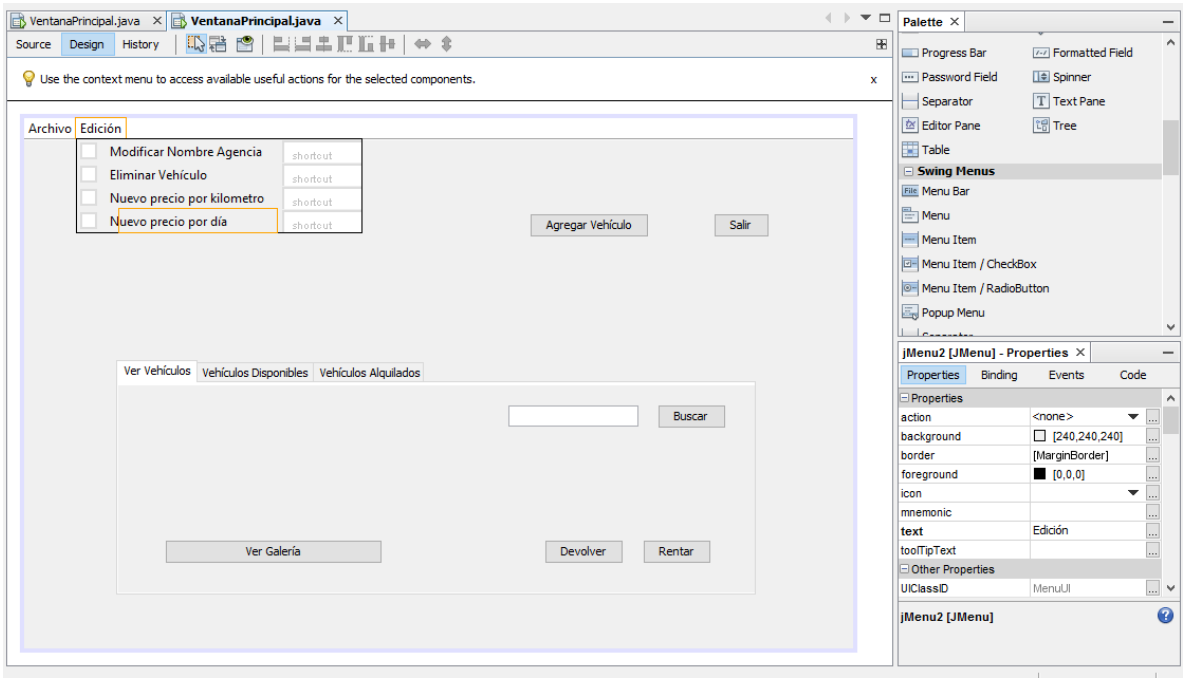


Podemos agregar más opciones a la barra de menú, así como está en este proyecto la opción “Ayuda” que en este caso la obviare ya que allí solo manejamos la parte de la documentación e información nuestra. Pero si queremos agregar más opciones podemos arrastrar “Menu” tantos como deseen añadir.

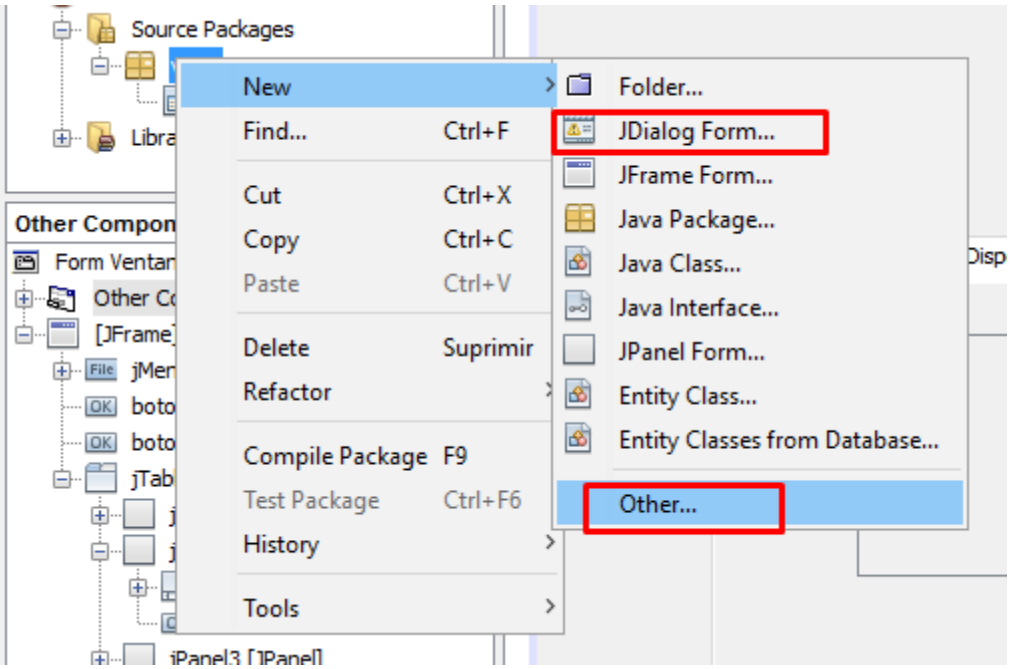


Ahora para agregarle opciones a los menús sólo debemos arrastrar desde la Palette Swing Menus “Menu Item” los que necesitemos y dejarlos caer encima del menú donde lo queremos agregar, en nuestro caso agregamos dos en “Archivo” a los cuales modificamos su texto a “Agregar Vehículo” y “Salir” respectivamente. Los dos de “Edición” les ponemos “Modificar Nombre Agencia”, “Eliminar Vehículo”, “”. Quedará de esta forma:

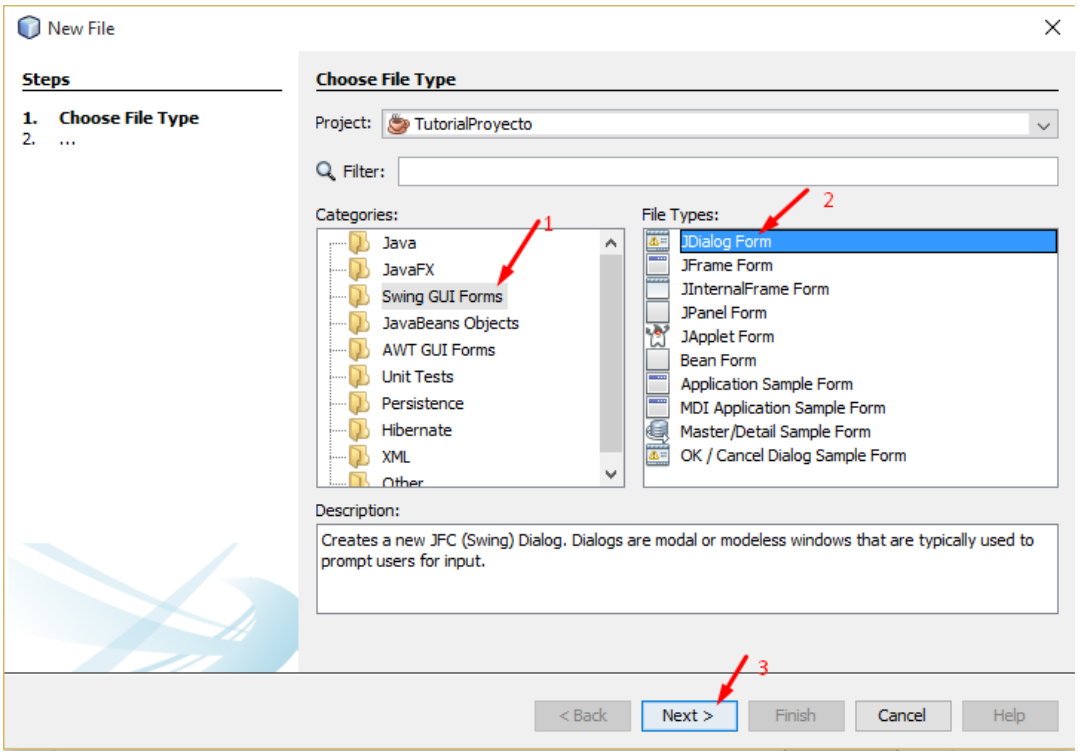




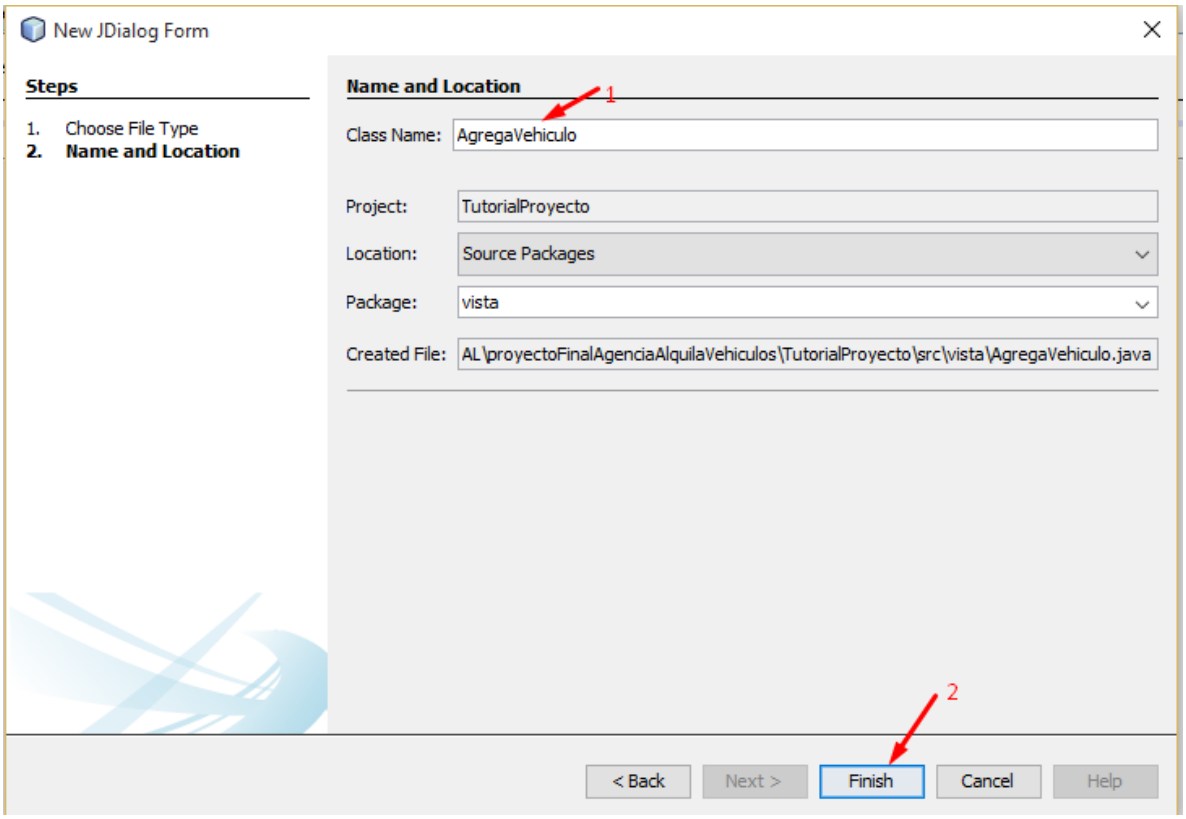
Ahora hacemos una ventana externa para agregar nuevos vehículos, usaremos un JDialog, lo creamos dándole clic derecho en el paquete vista, seleccionamos “new” y si no nos aparece JDialog en las opciones seleccionamos la última “other…”.



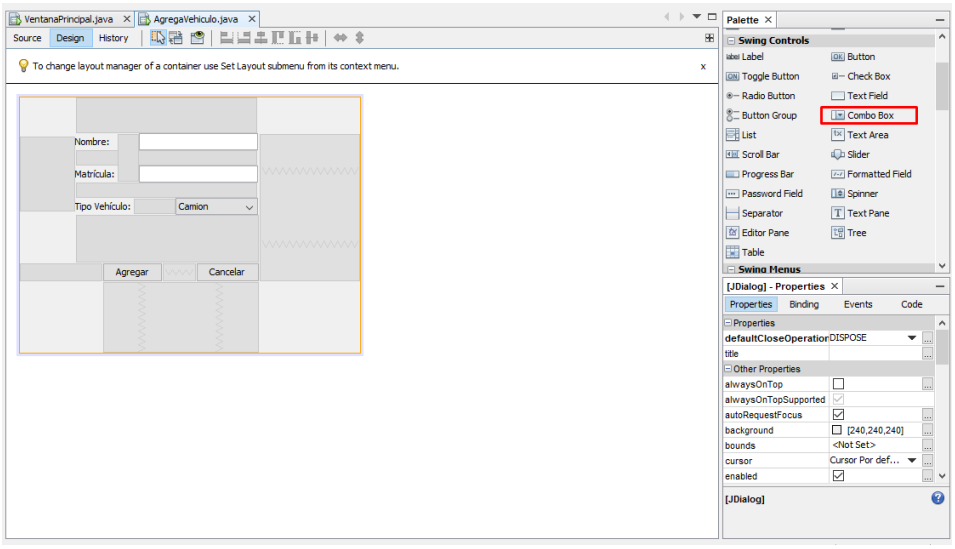
Allí nos saltara una ventana, seleccionamos en las categorías “Swing GUI Forms” y luego “JDialog Form”.



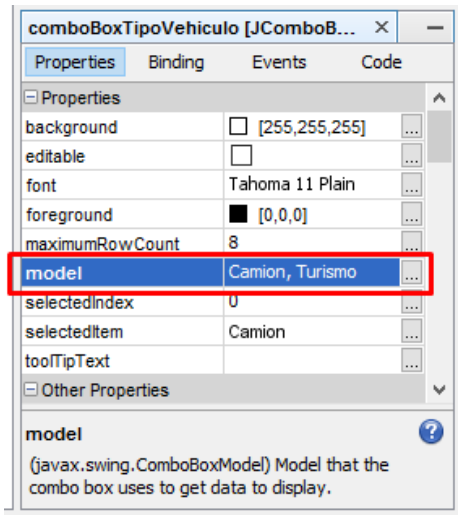
Para continuar ingresamos el nombre del JDialog, en nuestro caso será “AgregaVehiculo” y “Finish”.



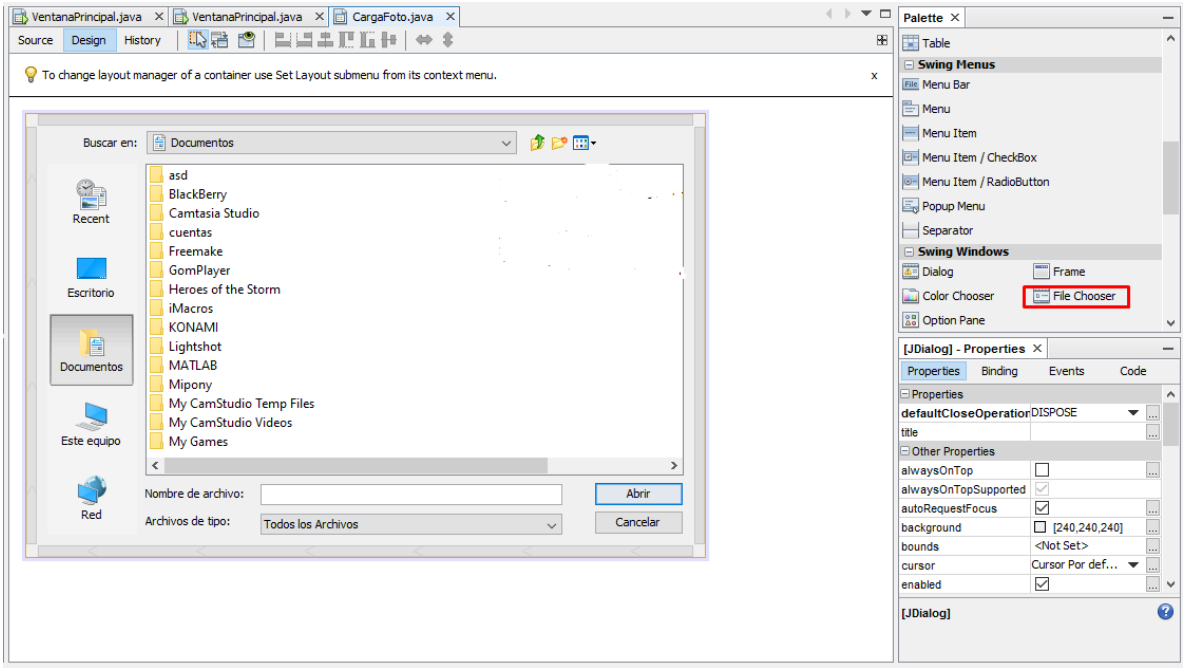
Finalmente nos queda el JDialog, para ingresarle como a una ventana normal todo lo que requiera nuestro software, para nuestro proyecto es necesario agregarle tres labels con los siguientes textos “Nombre:”, “Matrícula:” y “Tipo Vehículo:”, luego agregamos dos “Text Fiedl” y los ubicaremos al lado de los labels “Nombre:” y “Matrícula:” respectivamente. Al lado de “Tipo Vehículo:” ponemos un “Combo Box”, que encontramos en la Palette Swing Controls. Debajo de todo añadimos dos botones uno con el texto “Agregar” y el otro con “Cancelar”. Lo veremos de esta forma.



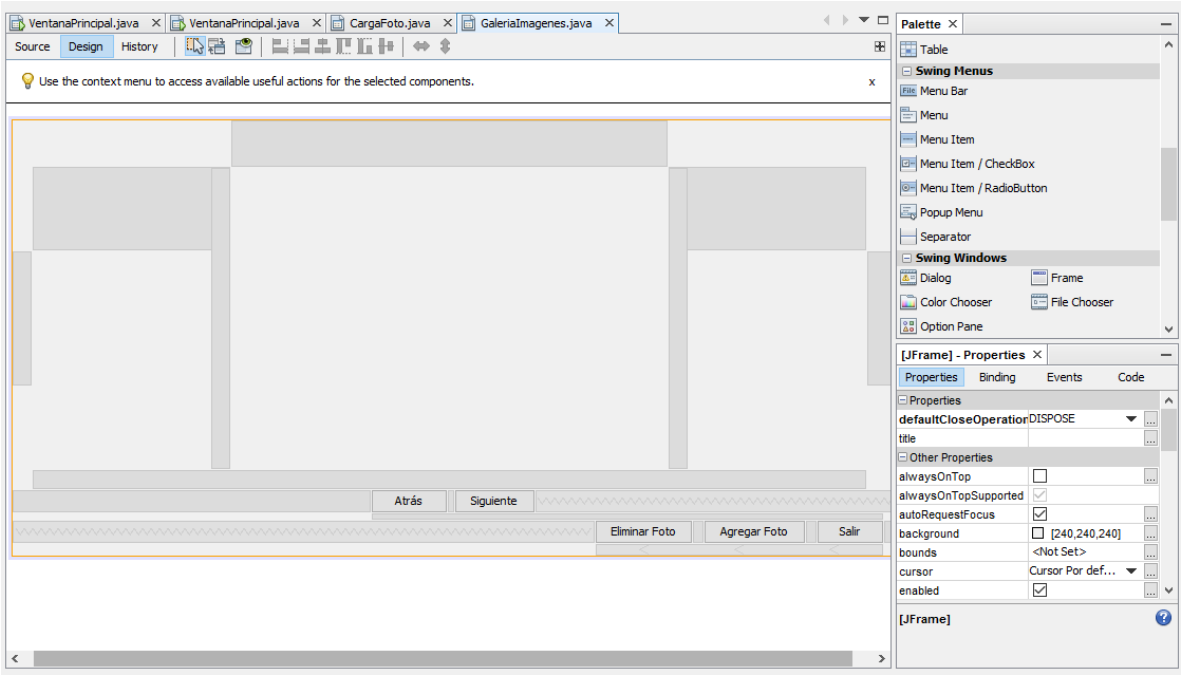
Para agregarle opciones al Combo Box y en nuestro caso que solamente tenemos vehículos de tipo camión y turismo, seleccionamos el Combo Box, nos vamos a la ventanita que está ubicada en la parte derecha inferior “Propiedades” y buscamos la opción “model”, e ingresamos las opciones que queramos separadas por coma (,), en nuestro caso son “Camión” y “Turismo”.



Luego necesitamos dos ventanas más, una para visualizar las fotos de los vehículos llamada “GaleriaImagenes” y otra con solo un “File Chooser” para subir nuestras fotos a la galería. Esta ultima la llamaremos “CargaFoto”, Ahora si creamos un nuevo JDialog y agregamos en toda su extensión un “File Chooser” desde la palette Swing Windows, quedando de esta forma.



Finalmente Creamos un nuevo JFrame para la galería de fotos, arrastramos tres labels, uno grande en el centro y dos más pequeños ubicados en los lados para así poder visualizar las fotos, también necesitamos cinco botones, dos de ellos son para pasar las imágenes y los llamaremos “Atrás” y “Siguiente” en su texto. El resto son opciones como la de “Eliminar foto”, “Agregar Foto” y “salir”. La ventana debería quedar de esta forma.



Integración de la vista con la lógica del Software

Iniciamos con la ventana principal, los atributos aquí declarados son para las funcionalidades de algunas acciones que se producen.

```
public class VentanaPrincipal extends javax.swing.JFrame {
    DefaultTableModel tabla1;
    DefaultTableModel tabla2;
```

En el constructor de la ventana lo primero es cargar el archivo, si no existe lo crea en el método **loading ()**, usamos un **try catch** por posibles errores en el archivo y actualizamos las tablas donde se visualizan los vehículos disponibles y alquilados con **agregaTablas().**

```
public VentanaPrincipal() {
        try {
            AgenciaAlquilaVehiculos.nuevo = (AgenciaAlquilaVehiculos)
Archivo.loading("ArchivoAgencia.dat");
        }
```

```
        catch (IOException e) {}
        catch (ClassNotFoundException e) {}
        initComponents();
        this.setResizable(false); //Deshabilita la opción de maximizar la
ventana.
        this.setLocationRelativeTo(null); //Centra la ventana
        agregaTablas();
    }
```

Programándole la acción a los botones.

Visualiza el JDialog que agrega vehículos, y actualiza las tablas.

```
private void
botonAgregaVehiculoActionPerformed(java.awt.event.ActionEvent evt) {
        AgregaVehiculo avv = new AgregaVehiculo(this, true);
        avv.setVisible(true);
        agregaTablas ();
        jLabel2.setText(infoAgencia ());
    }
```

Al hacer clic para escribir una nueva matricula, ésta limpia el jText.

```
private void textMatriculaMouseClicked(java.awt.event.MouseEvent evt) {
        textMatricula.setText("");
    }
```

Realiza la búsqueda de la matrícula ingresada en el ArrayList de Vehiculo. Igual siempre verificando la existencia de la matrícula, luego si existe toma el resultado del método usado que sería la posición en el array del vehículo para visualizarlo en los labels de la pestaña.

```
private void botonBuscarActionPerformed(java.awt.event.ActionEvent evt) {
        if
(AgenciaAlquilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getTe
xt(), AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos()) == -1){
            JOptionPane.showMessageDialog(null, "METRÍCULA INGRESADA
INEXISTENTE", "ERROR DE ENTRADA",
JOptionPane.WARNING_MESSAGE);
        }else{
            muestraNombre.setText("Nombre:
"+AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(AgenciaAl
quilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText(),
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos())).getNombre()) ;
```

```
        muestraMatricula.setText("Matrícula:
"+AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(AgenciaAl
quilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText(),
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos())).getMatricula()) ;

muestraTipo.setText(AgenciaAlquilaVehiculos.getVehiculo().getListaVehicul
os().get(AgenciaAlquilaVehiculos.getVehiculo().buscarVehiculo(textMatricul
a.getText(),
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos())).toString()) ;
        if
(AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(AgenciaAlqu
ilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText(),
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos())).isAlquilado() ==
true)
            muestraEstado.setText("Estado: Alquilado");
        else muestraEstado.setText("Estado: Disponible");
        if
(AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(AgenciaAlqu
ilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText(),
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos())).getFotos().size(
) != 0){
        Icon icono = new
ImageIcon(AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(A
genciaAlquilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText()
,
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos())).getFotos().get(
0).getImage().getScaledInstance(labelImage.getWidth(),
labelImage.getHeight(), Image.SCALE_DEFAULT));
        labelImage.setIcon( icono );
        }else{
            Icon icono = new ImageIcon(getClass().getResource("/iconos/no-
foto.gif"));
            labelImage.setIcon(icono);
        }
    }
  }
```

Renta el vehículo si está disponible o si existe. Primero revisa la existencia del vehículo que se le ingresó su matrícula en el jText, luego revisa que el vehículo se encuentre disponible y si esto es cierto lo alquila, al final actualiza las tablas.

```
private void botonAgregaRentaActionPerformed(java.awt.event.ActionEvent
evt) {
```

```
        if
(AgenciaAlquilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getTe
xt(), AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos()) == -1){
            JOptionPane.showMessageDialog(null, "METRÍCULA INGRESADA
INEXISTENTE", "ERROR DE ENTRADA",
JOptionPane.WARNING_MESSAGE);
        }else{
        if
(AgenciaAlquilaVehiculos.getVehiculo().alquilarVehiculo(AgenciaAlquilaVehi
culos.getVehiculo().buscarVehiculo(textMatricula.getText(),
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos())) == true){
            JOptionPane.showMessageDialog(null, "Confirmacion Vehículo
alquilado exitosamente!");
            agregaTablas ();
        }else {
            JOptionPane.showMessageDialog(null, "EL VEHÍCULO NO SE
PUDO RENTAR", "ACCIÓN INVALIDADA",
JOptionPane.WARNING_MESSAGE);
        }
        }
    }
```

Devuelve un vehículo si este se encuentra alquilado. Primero verifica que la matrícula existe, luego revisa que el vehículo se encuentre alquilado y continúa verificando que tipo de vehículo es para que si es camión pida lo necesario para este o si es turismo también. Al final salta el coste total del alquiler.

```
private void
botonDevolverVehiculoActionPerformed(java.awt.event.ActionEvent evt) {
        if
(AgenciaAlquilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getTe
xt(), AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos()) == -1){
            JOptionPane.showMessageDialog(null, "METRÍCULA INGRESADA
INEXISTENTE", "ERROR DE ENTRADA",
JOptionPane.WARNING_MESSAGE);
        }else{
        if
(AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(AgenciaAlqu
ilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText(),
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos())).isAlquilado() ==
true){
            if
(AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(AgenciaAlqu
```

```
ilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText(),
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos())) instanceof
VTurismo){
            VTurismo vt = new VTurismo ();
            vt =
(VTurismo)AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(A
genciaAlquilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText()
, AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos()));
            try{
            long nuevoKilometraje =
Long.parseLong(JOptionPane.showInputDialog(null, "Ingrese nuevo registro
del kilometraje"));
            if (nuevoKilometraje > vt.getKilometraje()){
            JOptionPane.showMessageDialog(null, "Total a Pagar:
"+AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(AgenciaAl
quilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText(),
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos())).devolverElVehi
culo(AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos(),
AgenciaAlquilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText
(), AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos()),
nuevoKilometraje));
            vt.setKilometraje(nuevoKilometraje);
            vt.setAlquilado(false);

AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().set(AgenciaAlqui
laVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText(),
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos()), vt);
            }else{
               JOptionPane.showMessageDialog(null, "Cifra ingresada
incorrecta debe ser mayor a "+vt.getKilometraje() );
            }
            }catch (Exception asd){
               JOptionPane.showMessageDialog(null, "Cifra ingresada
incorrecta" );
            }

         }else{
            JOptionPane.showMessageDialog(null, "Total a Pagar:
"+AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(AgenciaAl
quilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText(),
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos())).devolverElVehi
culo(AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos(),
```

```
AgenciaAlquilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getText
(), AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos()), 0));
            }
            agregaTablas ();
        }else{
            JOptionPane.showMessageDialog(null, "EL VEHÍCULO NO SE
PUDO DEVOLVER", "ACCIÓN INVALIDADA",
JOptionPane.WARNING_MESSAGE);
        }
        }
    }
```

Botón salir, guarda todo en el archivo y cierra el programa. Primero salta un JOptionPane de “YES” o “NO” preguntando si quiere salir y realiza la acción correspondiente, si es afirmativa la acción usa el método de la persistencia que guarda en el archivo.

```
private void botonSalirActionPerformed(java.awt.event.ActionEvent evt) {
        if( JOptionPane.showConfirmDialog( this, "seguro que quiere salir?",
"Confirmacion", JOptionPane.YES_NO_OPTION)==
JOptionPane.YES_OPTION){
        try {

Archivo.saving(AgenciaAlquilaVehiculos.nuevo,"ArchivoAgencia.dat");
            this.dispose();
        }
        catch (ClassNotFoundException e){}
        catch (IOException e) {}
        catch(NullPointerException e){}
    }
    }
```

Botones que están en las pestañas junto a las tablas y que actualizan el estado de los vehículos almacenados en el ArrayList.. Invoca los métodos respectivos.

```
private void botonActualizar1ActionPerformed(java.awt.event.ActionEvent
evt) {
        agregaTablas ();
        jLabel2.setText(infoAgencia ());
    }

    private void botonActualizar2ActionPerformed(java.awt.event.ActionEvent
evt) {
        agregaTablas ();
```

```
        jLabel2.setText(infoAgencia ());
    }
```

Botón “Ver Galería” que muestra en pantalla la ventana para manipular las fotos de los vehículos.

```
private void botonVerGaleriaActionPerformed(java.awt.event.ActionEvent
evt) {
        if
(AgenciaAlquilaVehiculos.getVehiculo().buscarVehiculo(textMatricula.getTe
xt(), AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos()) == -1){
            JOptionPane.showMessageDialog(null, "METRÍCULA INGRESADA
INEXISTENTE", "ERROR DE ENTRADA",
JOptionPane.WARNING_MESSAGE);
        }else{
                GaleriaImagenes gi = new
GaleriaImagenes(AgenciaAlquilaVehiculos.getVehiculo().buscarVehiculo(tex
tMatricula.getText(),
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos()));
                gi.setVisible(true);
        }
    }
```

Acción de las opciones en la barra de menú, “Archivo”. El primero es “Agregar un Vehículo” y el otro es “Salir”. Estas opciones son las mismas que las del botón “Agregar Vehículo” y “Salir”.

```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        AgregaVehiculo avv = new AgregaVehiculo(this, true);
        avv.setVisible(true);
        agregaTablas ();
        jLabel2.setText(infoAgencia ());
    }

    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt)
{
        if( JOptionPane.showConfirmDialog( this, "seguro que quiere salir?",
"Confirmacion", JOptionPane.YES_NO_OPTION)==
JOptionPane.YES_OPTION){
        try {

Archivo.saving(AgenciaAlquilaVehiculos.nuevo,"ArchivoAgencia.dat");
            this.dispose();
        }
```

```
        catch (ClassNotFoundException e){}
        catch (IOException e) {}
        catch(NullPointerException e){}
    }
    }
```

Acción de las opciones en la barra de menú, “Edición”. El primero es “Modificar nombre de Agencia” y el otro es “Eliminar Vehículo”. “Modificar nombre de Agencia” Visualiza un JOptionPane que recibe el nuevo nombre de la agencia y “Eliminar Vehiculo” también muestra un JOptionPane que recibe la matrícula del vehículo a eliminar, este comprueba si existe y realiza la acción si esto es verdadero. También añadimos la opción “Nuevo precio por kilómetro” y “Nuevo precio por día” para que el administrador tenga el poder de cambiar en cualquier momento estas variables.

```
private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {

AgenciaAlquilaVehiculos.nuevo.setNombre(JOptionPane.showInputDialog(n
ull, "Ingrese nuevo nombre a la Agencia"));
    }

    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt)
{
        String placa;
        placa = JOptionPane.showInputDialog(null, "Ingrese Matrícula de
Vehículo a eliminar");
         if (AgenciaAlquilaVehiculos.getVehiculo().buscarVehiculo(placa,
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos()) == -1){
           JOptionPane.showMessageDialog(null, "METRÍCULA INGRESADA
INEXISTENTE", "ERROR DE ENTRADA",
JOptionPane.WARNING_MESSAGE);
        }else{
            ArrayList<Vehiculo> aux =
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos();

aux.remove(AgenciaAlquilaVehiculos.getVehiculo().buscarVehiculo(placa,
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos()));
            AgenciaAlquilaVehiculos.nuevo.setListaVehiculos(aux);
            JOptionPane.showMessageDialog(null, "VEHÍCULO ELIMINADO!",
"INFORMACIÓN", JOptionPane.WARNING_MESSAGE);
            agregaTablas ();
            jLabel2.setText(infoAgencia ());
         }
    }
```

```
private void jMenuItem11ActionPerformed(java.awt.event.ActionEvent evt) {
        try{

AgenciaAlquilaVehiculos.getVehiculo().setP_p_kilometro(Integer.parseInt(J
OptionPane.showInputDialog(null, "Ingrese nuevo precio por kilometro para
vehículos de turismo")));
        }catch (Exception ex){
          JOptionPane.showMessageDialog(null, "CIFRA INVÁLIDA", "ERROR
DE ENTRADA", JOptionPane.WARNING_MESSAGE);
        }
    }

    private void jMenuItem12ActionPerformed(java.awt.event.ActionEvent
evt) {
        try{

AgenciaAlquilaVehiculos.getVehiculo().setP_p_dia(Integer.parseInt(JOption
Pane.showInputDialog(null, "Ingrese nuevo precio por día para vehículos
camión")));
        }catch (Exception ex){
          JOptionPane.showMessageDialog(null, "CIFRA INVÁLIDA", "ERROR
DE ENTRADA", JOptionPane.WARNING_MESSAGE);
        }
    }
```

Extras

Como poner un link en una opción en Java, existen varias formas pero para este caso usamos esta, así mismo en la segunda parte vemos como abrir un archivo también desde un opción del programa. Tomado de internet desde el sitio: https://franciscoguemes.wordpress.com/2011/08/03/abrir-documentos-pdf-word-etc-desde-java/

```
if(java.awt.Desktop.isDesktopSupported()){
          try{
            Desktop dk = Desktop.getDesktop();
            dk.browse(new URI("https://www.youtube.com/watch?v=y9V7Mf6H-
DQ&feature=youtu.be"));
           }catch(Exception e){
            System.out.println("Error al abrir URL: "+e.getMessage());
            }
        }

try {
```

```
File path = new File ("Diagramadeclase.png");
Desktop.getDesktop().open(path);
}catch (IOException ex) {
ex.printStackTrace();
 }
```

Métodos de la ventana principal

Método main para que corra el programa desde la ventana principal

```
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
/* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new VentanaPrincipal().setVisible(true);
        }
    });
  }
```

Método que carga y actualiza las tablas de vehículos disponibles y alquilados, usa las variables que declaramos en la parte de arriba, agrega la primera fila que contiene los nombres de las columnas, luego recorre el ArrayList de Vehículo y organiza según disponibilidad de estos en las tablas, agregando así la información de cada uno.

```
public void agregaTablas (){
    tabla1 = new DefaultTableModel();
    tabla2 = new DefaultTableModel();
    tabla1.addColumn("Nombre"); tabla2.addColumn("Nombre");
    tabla1.addColumn("Matrícula"); tabla2.addColumn("Matrícula");
    tabla1.addColumn("Tipo"); tabla2.addColumn("Tipo");
    this.tablaDisponibles.setModel(tabla1);
    this.tablaAlquilados.setModel(tabla2);
    for (int i=0;
i<AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().size(); i++){
        String[] Datos = new String [3];
        Datos[0] =
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(i).getNombre
();
        Datos[1] =
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(i).getMatricul
a();
```

```
            if
(AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(i) instanceof
VTurismo)
                Datos[2] = "Vehículo Turismo";
            else Datos[2] = "Vehículo Camion";
          if
(AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(i).isAlquilad
o() == false){
            tabla1.addRow(Datos);
          }else {
            tabla2.addRow(Datos);
          }
       }
    }
```

Método que actualiza la información sobre la cantidad de vehículos camión, turismo y total que aparece en la parte inferior del software. Retorna una cadena **String**

```
public String infoAgencia (){
      return "Vehículos Turismo:
"+AgenciaAlquilaVehiculos.getVehiculo().totalVehiculosTurismo()+"
Vehículos Camion:
"+AgenciaAlquilaVehiculos.getVehiculo().totalVehiculosCamion()+"   Total
Vehículos: "+AgenciaAlquilaVehiculos.getVehiculo().totalVehiculos();
    }
} //Fin clase VentanaPrincipal
```

Descripción JDialog AgregaVehiculo

```
public class AgregaVehiculo extends javax.swing.JDialog {
    VCamion vc = null; //Variables que sirven como auxiliares
    VTurismo vt = null;
```

Constructor recibe dos parámetros.

```
public AgregaVehiculo(java.awt.Frame parent, boolean modal) {
      super(parent, modal);
      initComponents();
      this.setResizable(false); //Anula maximizar
      this.setLocationRelativeTo(null); //Centra la ventana
      this.textNombre.requestFocus(); //Coloca el foco en el primer jText
    }
```

Acción de los botones

Botón “Aceptar” que agrega un nuevo vehículo, primero revisa la opción del Combo Box seleccionada, y de acuerdo a esta crea un vehículo turismo o camión. También verifica que el nombre no este vacío, que la matrícula no contenga más de 6 caracteres y de que estos caracteres los tres primeros sean solo letras y los tres últimos solo números.

```
private void botonAgregarActionPerformed(java.awt.event.ActionEvent evt) {
      AgenciaAlquilaVehiculos aux = new AgenciaAlquilaVehiculos();
      if (comboBoxTipoVehiculo.getSelectedIndex() == 0){
         vc = new VCamion ();
         if (this.textNombre.getText().equals("")){
            JOptionPane.showMessageDialog( this, "Por Favor Digite
Nombre", "Registro", JOptionPane.INFORMATION_MESSAGE);
            this.textNombre.requestFocus();
         }else{
            if (textMatricula.getText().length() != 6){
               JOptionPane.showMessageDialog( this, "Caracteres de la
matrícula incompletos", "Registro",
JOptionPane.INFORMATION_MESSAGE);
               this.textMatricula.requestFocus();
            }else{
               if (isNumeric(textMatricula.getText().substring(3)) &&
isLetter(textMatricula.getText().substring(0,3))) {
                  vc.setNombre(textNombre.getText());
                  vc.setMatricula(textMatricula.getText().toUpperCase());
                  aux.guardaVehiculos (vc);
                  dispose();
               }else{
                  JOptionPane.showMessageDialog( this, "Formato de la
matrícula incorrecto (AAA000)", "Registro",
JOptionPane.INFORMATION_MESSAGE);
                  this.textMatricula.requestFocus();
               }
            }}
         }else{
            if (comboBoxTipoVehiculo.getSelectedIndex() == 1){
               vt = new VTurismo ();
               if (this.textNombre.getText().equals("")){
                  JOptionPane.showMessageDialog( this, "Por Favor Digite
Nombre", "Registro", JOptionPane.INFORMATION_MESSAGE);
                  this.textNombre.requestFocus();
               }else{
                  if (textMatricula.getText().length() != 6){
```

```
                    JOptionPane.showMessageDialog( this, "Caracteres de la
matrícula incompletos", "Registro",
JOptionPane.INFORMATION_MESSAGE);
                    this.textMatricula.requestFocus();
                }else{
                    if (isNumeric(textMatricula.getText().substring(3)) &&
isLetter(textMatricula.getText().substring(0,3))) {
                        vt.setNombre(textNombre.getText());
                        vt.setMatricula(textMatricula.getText().toUpperCase());
                        aux.guardaVehiculos (vt);
                        dispose();
                    }else{
                        JOptionPane.showMessageDialog( this, "Formato de la
matrícula incorrecto (AAA000)", "Registro",
JOptionPane.INFORMATION_MESSAGE);
                        this.textMatricula.requestFocus();
                    }
                }}
            }
        }
    }
```

Botón que cierra la ventana sin agregar vehículo

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    dispose ();
}
```

Métodos del JDialog AgregaVehiculo

Verifica si la cadena del parámetro contiene solo números y retorna la respuesta true o false, usando un parseInt, esta si no tiene números arrojara un error que lo tomará el catch y enviará un false. Método tomado de: http://lineadecodigo.com/java/validar-si-un-dato-es-numerico-en-java/

```
    private static boolean isNumeric(String cadena){
        try {
            Integer.parseInt(cadena);
            return true;
        }catch (NumberFormatException nfe){
            return false;
        }
}
```

Revisa si la cadena del parámetro contiene solo letras y retorna la respuesta true o false, primero convierte la cadena en minúsculas, luego recorre la cadena y si esta encuentra algo diferente a una letra retorna false.

```
    private static boolean isLetter (String cadena){
        cadena = cadena.toLowerCase();
   for(int i=0; i<cadena.length(); i++){
      if (!(cadena.charAt(i) >= 'a' && cadena.charAt(i) <= 'z') ){
         return false;
      }
   }
   return true;
}
} //Fin JDialog AgregaVehiculo
```

Descripción de las ventanas GaleriaImagenes y CargaFoto

Para CargaFoto prácticamente no se hace nada novedoso puesto que con este JDialog solo tenemos un File Chooser para cargar los archivos desde el pc, lo único es nombrar al FileChooser un nombre de variable en este caso le pusimos “carga”.

Para la ventana de GaleriaImagenes.

```
public class GaleriaImagenes extends javax.swing.JFrame {
//atributos necesrios para realizar algunas opciones
    File fichero;
    int ref;
    int cont;
    Icon icono = new ImageIcon(getClass().getResource("/iconos/no-
foto.gif")); //carga una imagen que la usamos para cuando no exista una
foto
```

Constructor

```
public GaleriaImagenes(int i) {
        initComponents();
        this.setResizable(false); //Anula maximizar
        this.setLocationRelativeTo(null); //Centra la ventana
        ref = i;
        cont = 0; // para recorrer el ArrayList de imagenes
        ImprimeFoto();
    }
```

Programándoles acción a los botones “Salir”, “Siguiente” y “Atrás” respectivamente.

```
//Boton salir
```

```
    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        dispose ();
    }
//Boton siguiente
    private void botonSiguienteActionPerformed(java.awt.event.ActionEvent
evt) {
        cont++;
        ImprimeFoto();
    }
//Boton Atrás
    private void botonAtrasActionPerformed(java.awt.event.ActionEvent evt) {
        cont--;
        ImprimeFoto();
    }
```

Botón “Agrega Foto”, realiza dicha acción cargando la ventana con el File Chooser y así copiando un archivo que anteriormente filtra y solo acepta que sean de tipo “jpg” y “png”. Para posteriormente guardarlo en el ArrayList de fotos. Este código fue copiado y modificado por mí para lo que necesitaba el programa desde: http://uh-tis.blogspot.com.co/2014/03/Curso-de-JAVA-Como-cargar-una-imagen-y-o-foto-en-un-JLabel-de-un-formulario-utilizando-JFileChooser.html

```
private void botonAgregaFotoActionPerformed(java.awt.event.ActionEvent
evt) {
        int resultado;
        CargaFoto ventana = new CargaFoto(this, true);
        FileNameExtensionFilter filtro = new FileNameExtensionFilter("JPG y
PNG","jpg","png");
        ventana.carga.setFileFilter(filtro);
        resultado= ventana.carga.showOpenDialog(null);
        if (JFileChooser.APPROVE_OPTION == resultado){
            fichero = ventana.carga.getSelectedFile();
            try{
                ImageIcon iconox = new ImageIcon(fichero.toString());

AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(ref).agregaF
otoVehiculo(iconox, ref);
            }catch(Exception ex){
                JOptionPane.showMessageDialog(null, "Error abriendo la imagen
"+ ex);
            }
        }
        ImprimeFoto ();
    }
```

Boton “Eliminar Foto”, realiza dicha acción.

```
private void botonEliminarFotoActionPerformed(java.awt.event.ActionEvent
evt) {
      ArrayList<ImageIcon> alii =
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(ref).getFotos
();
      if (!alii.isEmpty()){
         try{
      alii.remove(cont);

AgenciaAlquilaVehiculos.nuevo.getListaVehiculos().get(ref).setFotos(alii);
      JOptionPane.showMessageDialog(null, "Foto eliminada
exitosamente!");
      cont--;
      ImprimeFoto ();
         }catch (Exception exc){
            System.out.println(exc);
         }
      }
   }
```

Método usado en la ventana GaleriaImagenes para visualizar las fotos

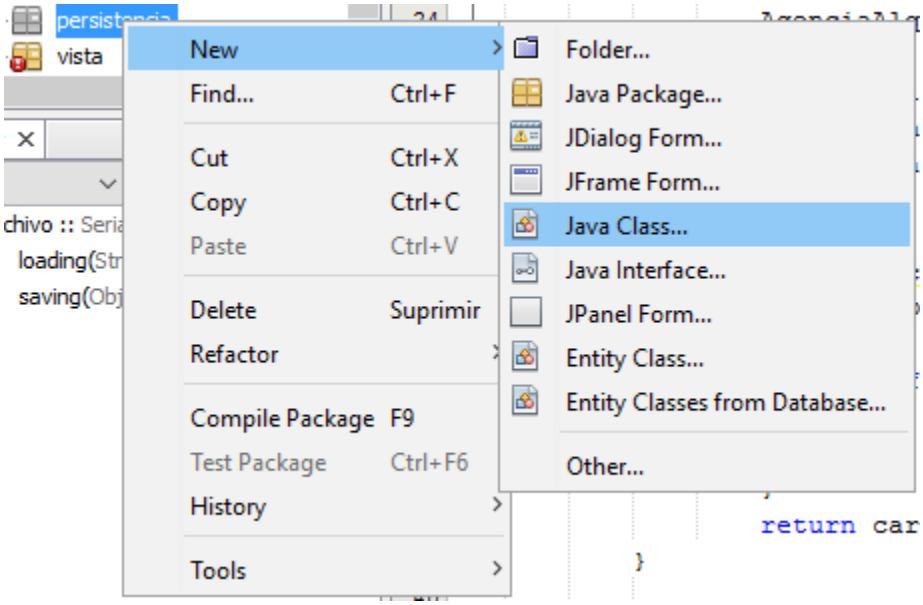
Verifica la existencia de fotos en el ArrayList, luego en el lblMain imprime si hay, la foto y si no un mensaje. En los labels de los lados imprime la foto si existe o la foto de “no-foto” si no existe, también activan o desactivan los botones de siguiente y atrás para el caso correspondiente.

```
private void ImprimeFoto (){
      ArrayList<ImageIcon> alii =
AgenciaAlquilaVehiculos.getVehiculo().getListaVehiculos().get(ref).getFotos
();
      Icon ico;
      if (!alii.isEmpty()){
      ico = new
ImageIcon(alii.get(cont).getImage().getScaledInstance(lblMain.getWidth(),
lblMain.getHeight(), Image.SCALE_DEFAULT));
      lblMain.setIcon(ico);
      }else{
         lblMain.setText("AGREGUE IMAGENES A LA GALERÍA");
      }

      if ((cont-1) < 0) { lblIzquierda.setIcon(icono);
botonAtras.setEnabled(false); }
```

```
        else {
            ico = new ImageIcon(alii.get(cont-
1).getImage().getScaledInstance(lblIzquierda.getWidth(),
lblIzquierda.getHeight(), Image.SCALE_DEFAULT));
            lblIzquierda.setIcon(ico); botonAtras.setEnabled(true); }

        if (alii.size() <= (cont+1)) { lblDerecha.setIcon(icono);
botonSiguiente.setEnabled(false); }
        else {
            ico = new
ImageIcon(alii.get(cont+1).getImage().getScaledInstance(lblDerecha.getWid
th(), lblDerecha.getHeight(), Image.SCALE_DEFAULT));
            lblDerecha.setIcon(ico); botonSiguiente.setEnabled(true); }
    }
} //Final JFrame GaleriaImagenes
```

Creando la clase Archivo

- Crear un paquete llamado archivo o persistencia
- Dentro del paquete creado, crear la clase Archivo




```
public class Archivo implements Serializable {
```

Creamos el método “loading” que es el usamos para recuperar un archivo ya existente o si no existe crea uno nuevo.

```
public static Object loading(String a) throws
ClassNotFoundException,IOException{
                File archivo=new File(a);
                ObjectInputStream entrada = null;
                AgenciaAlquilaVehiculos cargar=new
AgenciaAlquilaVehiculos();
                try{
                  if(archivo.exists()){
                        entrada=new ObjectInputStream(new
FileInputStream(archivo));
                        cargar=(AgenciaAlquilaVehiculos) entrada.readObject();
                   }

                }catch(Exception e){
                        JOptionPane.showMessageDialog(null, "EL ARCHIVO
NO EXISTE", "ERROR EN EL FICHERO",
JOptionPane.WARNING_MESSAGE);
                }finally{
                        if(entrada!=null){
                                entrada.close();
                        }
                }
                return cargar;
        }
```

El otro método que necesitamos es el de “Saving”, que guarda el archivo o lo sobrescribe si ya existe.

```
public static void saving(Object cargar, String a)throws
ClassNotFoundException,IOException{
                File archivo =new File(a);
                ObjectOutputStream salida =null;

                try{
                        salida=new ObjectOutputStream(new
FileOutputStream(archivo));
                        salida.writeObject(cargar);
                }catch(Exception e){
          System.out.println(e);
                    JOptionPane.showMessageDialog(null, "EL ARCHIVO NO
SE PUEDE ESCRIBIR", "ERROR DEL FICHERO",
JOptionPane.WARNING_MESSAGE);
                }finally{
```

```
                    salida.close();
                }
            }
} //Fin clase Archivo
```

Clase “Archivo” tomada de material didáctico compartido por Yasmín Moya.

Universidad de Cartagena
Ingeniería de Sistemas
Programación Orienta Objetos
Yasmín Moya, Docente
David Padilla Fonseca, Estudiante
Andrés Rodríguez, Estudiante