

# BIBLIOTECA

RODRIGUEZ ALONZO

VICTOR EDUARDO JOSÉ

```
        'role_id'      => $role_details['id'],
        'resource_id' => $resource_details['id']
    );
    >rule_exists( $resource_details['id'], $role_d
    ccess == false ) {
        Remove the rule as there is currently no need
        tails['access'] = !$access;
        $is->_sql->delete( 'acl_rules', $details );
        [
        Update the rule with the new access value
        $is->_sql->update( 'acl_rules', array( 'access'

    [ $this->rules as $key=>$rule ) {
    [ $details['role_id'] == $rule['role_id'] && $
    if ( $access == false ) {
        unset( $this->rules[ $key ] );
    } else {
        $this->rules[ $key ]['access'] = $access;
    }
}
```

PROYECTO 1

201900018

MS©



# BIBLIOTECA MS©

## INTRODUCCIÓN.

En esta manual es para facilitar a un programador el código necesario para hacer o mejorar el sistema de Biblioteca MSC, en el cual veremos la funcionalidad del mismo con su código y la lógica que se le dio a este proyecto, veremos programación orientada a objetos como lo principal, pero enseñaremos mucho conocimiento sobre interfaz Gráfica, como generación de reportes, carga masiva etc, esperando que este manual sea de mucha ayuda para usted que lee el mismo.

## OBJETIVOS.

El objetivo general de este Manual es que el programador pueda encontrar el código necesario para realizar el mismo o mejorar el sistema de biblioteca y adaptarlo para cualquiera.

## ESPECIFICACIÓN TÉCNICA.

### 1. Requisitos de Hardware

- Computadora de escritorio o portátil.
- Mínimo 4GB de Memoria RAM.
- 250GB de Disco Duro o Superior.
- Procesador Inter Core i3 o superior.
- Resolución gráfica mínimo 1024 x 768 pixeles.

### 2. Requisitos de Software

- Sistema Operativo Windows 10 o superior
- Tener instalado Java Runtime Enviroment (JRE ) versión 15.2.
- Tener instalado Java Development Kit (JDK) versión 15.2.
- Lenguaje de Programación JAVA.
- NetBeans IDE 12.6
- Librería externa itextpdf
- Librería externa gson
- Librería externa jfreechart
- Librería interna Java.AWT

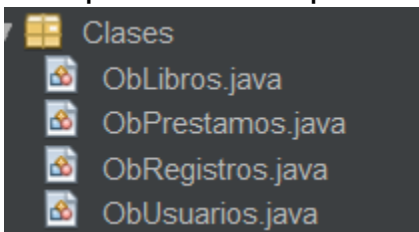


# BIBLIOTECA MS©

- Librería interna Javax.SWING
- Librería interna IO
- Acrobat Reader DC o algún lector de PDF compatible.

## 1. CLASES.

En esta sección se mostrará la creación de las clases utilizadas en este proyecto, una clase podríamos decir que es donde se coloca los atributos, constructor y encapsulamiento que lo veremos uno a uno.



Veremos las 4 clases que creamos que son:

- ObLibros
- ObPrestamos
- ObRegistros
- ObUsuarios

### 1.1. ATRIBUTOS.

Los atributos son las partes con las que se crea un objeto en este caso los atributos son una parte fundamental para la creación de un objeto, ahora mostraremos los Atributos de nuestras 4 clases.



# BIBLIOTECA MS©

## ➤ ObLibros

```
public class ObLibros {  
    //Creamos los Atributos del objeto de libros  
    private int IDlibro;  
    private String Titulo;  
    private String Autor;  
    private int Tipos;  
    private int Copias;  
    private int Disponibles;  
    private int Ocupados;
```

## ➤ ObPrestamos

```
public class ObPrestamos {  
    //Creamos los Atributos del objeto Prestamos  
    private int IDlibro;  
    private int IDusuario;  
    private String fechasinda;  
    private String status;
```

## ➤ ObRegistros

```
public class ObRegistros {  
    //Atributos para el objeto registros  
    private String ReporteUsuarios;  
    private String ReporteLibrosReg;  
    private String ReportesPrestrealizados;
```





# BIBLIOTECA MS©

## ➤ ObUsuarios.

```
//*****  
//ATRIBUTOS PARA EL OBJETO USUARIOS  
private int ID;  
private String Usuario;  
private String Password;  
private String Facultad;  
private String Carrera;  
private int Tipo;
```

## 1.2. CONSTRUCTOR.

Un constructor es un elemento de una clase cuyo identificador coincide con el de la clase correspondiente y que tiene como objetivo obligar y controlar como se inicializa una instancia de una determinada clase. Por fines del manual mostraremos el constructor de una sola clase que es ObLibros.

```
//Realizamos el constructor para este objeto  
public ObLibros(String Titulo,int IDlibro,String Autor, int Tipos, int Copias, int Disponibles, int ocupados) {  
    this.IDlibro = IDlibro;  
    this.Titulo = Titulo;  
    this.Autor = Autor;  
    this.Tipos = Tipos;  
    this.Copias = Copias;  
    this.Disponibles = Disponibles;  
    this.Ocupados = ocupados;  
}
```

En esta imagen podemos observar que el constructor tiene como parámetro los atributos que mencionamos anteriormente y así mismo se les hace mención en el objeto actual con la declaración (this).

## 1.3. ENCAPSULAMIENTO.

Este mismo consiste en ocultar los atributos de un objeto de manera que solo se pueda cambiar mediante operaciones definidas, en las cuales se utiliza el get y set.



# BIBLIOTECA MS©

```
//Encapsulamos con get y set todos los parametros
/**
 * @return the IDlibro
 */
public int getIDlibro() {
    return IDlibro;
}

/**
 * @param IDlibro the IDlibro to set
 */
public void setIDlibro(int IDlibro) {
    this.IDlibro = IDlibro;
}
```

En esta imagen podemos observar un solo encapsulamiento, ya que se tiene que hacer por cada parámetro de nuestro constructor pero solamente dejamos la idea de como se realiza cada uno el get indica lo que esta en el objeto guardado y el set indica la actualización del objeto.

## 2. VARIABLES GLOBALES Y CONTADORES.

Las variables Globales se colocan en la clase principal donde esta nuestro main para ser llamadas a cualquier otra clase, en la siguiente imagen se muestra como se realiza las variables globales.

```
//Creando los arreglos para las consideraciones que nos indicaron en el enunciado
public static ObLibros[] oblibros = new ObLibros[100];
public static ObUsuarios[] obuser = new ObUsuarios[50];
public static ObPrestamos[] obpres = new ObPrestamos[200];
public static ObRegistros[] obreg = new ObRegistros[100];
```

Los contadores son variables globales que también necesitamos para poder construir un método de agregar cualquier variable global creada como objeto en la imagen anterior, por lo cual también pueden ser llamadas a cualquier parte del proyecto.

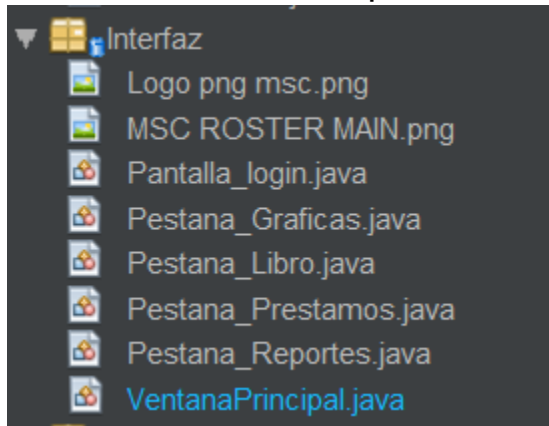


# BIBLIOTECA MS©

```
//CONTADORES PARA AGREGAR A SUS OBJETOS CORRESPONDIENTES  
public static int contusuario = 0, contlibros = 0, contprestamos = 0, contregistro = 0; //Iniciando contador
```

## 3. INTERFAZ GRÁFICA.

Es aquella que nos ayuda a que el usuario no vea toda la codificación al momento de querer correr el programa, mas bien ayuda a que el usuario trabaje directamente solo con el manual usuario sin tener que preocuparse por la codificación, ahora especificaremos que se realizo en este apartado.



Creamos una carpeta dedicada solamente para la interfaz, lógicamente dentro de cada una se trabajo toda la lógica que lleva, pero eso será para mas adelante.

- Pantalla\_login
- Pestana\_Graficas
- Pestana\_Libro
- Pestana\_Prestamos
- Pestana\_Reportes

En cada una de estas se trabajo lo que fueron JTextField, JBotton, JTable, JFrame, JPanel etc. Se mostrará como funciona cada uno, pero especificaremos que se tiene que realizar en todas según lo solicitado en Enunciado.



# BIBLIOTECA MS©

## 3.1. Creación Boton.

```
masival = new JButton("Carga Masiva");  
masival.setBounds(10, 580, 150, 25);  
masival.setFont(new Font("Franklin Gothic Medium", Font.BOLD, 14));  
masival.setBackground(azulejo);  
masival.setForeground(Color.white);  
masival.setVisible(true);  
masival.addActionListener(this);  
this.add(masival);
```

- Primera línea especifica la variable que se le asigna al botón y el título que lleva.
- Segunda línea son las dimensionales del botón.
- Tercera línea es el tipo de letra y tamaño
- Cuarta línea es el color que se le asigna
- Quinta línea muestra si es visible en la pantalla
- Sexta línea lo adhiere a la ventana.

## 3.2. Creación escoger una opción.

```
setLayout(null);  
tip = new JComboBox<String>();  
tip.setBounds(70, 170, 200, 25);  
tip.setFont(new Font("Verdana", Font.BOLD, 12));  
add(tip);  
tip.addItem("Libro");  
tip.addItem("Revista");  
tip.addItem("Libro Electrónico");  
tip.addActionListener(this);
```

Para Creación de un JComboBox lo único que cambia es que se necesita un setlayout null y que se agreguen las opciones que el usuario pueda escoger.





# BIBLIOTECA MS©

### 3.3. Creación del Cuadro de Texto.

```
//Cuadro Texto para ingresar codigo
cop = new JTextField();
cop.setBounds(70, 130, 200, 25);
cop.setFont(new Font("Verdana", Font.BOLD, 12));
cop.setVisible(true);
this.add(cop);
```

Este es el mismo formato solamente que acá se esta realizando un cuadro de texto donde el usuario va a poder ingresar lo solicitado en el título.

### 3.4. Creación de un texto multilinea.

```
JTextArea info = new JTextArea( "{\n" +
    "\nUsuarios\":[\n" +
    "    {\n" +
    "        \nID\": 1,\n" +
    "        \nUsuario\": \"user\",\n" +
    "        \nPassword\": \"123456\",\n" +
    "        \nFacultad\": \"Administrador\",\n" +
    "        \nCarrera\": \"Administrador\",\n" +
    "        \nTipo\": 1\n" +
    "    },\n" +
    "    {\n" +
    "        \nID\": 2,\n" +
    "        \nUsuario\": \"user1\",\n" +
    "        \nPassword\": \"123456\",\n" +
    "        \nFacultad\": \"Ingenieria\",\n" +
    "        \nCarrera\": \"Ingenieria en ciencias y sistemas\",\n" +
    "        \nTipo\": 2\n" +
    "    }\n" +
    "]\n" +
    "}" );
// info.setBounds(200, 350, 450, 250);
info.setVisible(true);
info.setEditable(true);
JScrollPane js = new JScrollPane(info);
js.setBounds(200, 350, 450, 250);
info.setFont(new Font("Verdana", Font.BOLD, 8));
this.add(js);
```



# BIBLIOTECA MS©

Esto nos sirve para poder agregar mas usuarios en este caso en formato json y nos permite editarlo a nuestro gusto.

## 3.5. Creación de un Título.

```
tipo = new JLabel("Tipo:");  
tipo.setBounds(10, 170, 80, 20);  
tipo.setFont(new Font("Arial Narrow", Font.BOLD, 16));  
tipo.setForeground(blanco);  
tipo.setVisible(true);  
this.add(tipo);
```

Esto nos sirve para colocar un titulo de lo que necesitemos en la ventana.

## 3.6. Creación de un Panel para agregarlo a una ventana

```
//*****  
//CREACIÓN DE LA PESTAÑA LIBRO  
//Diseño de Jpanel  
this.setBackground(plateado);  
this.setLayout(null);
```

Esto nos sirve únicamente para crear toda esta clase como una pestaña



# BIBLIOTECA MS©

## 3.7. Creación de una Ventana.

```
//*****  
//CREACIÓN DEL DISEÑO DE LA VENTANA  
//diseño de la ventana  
this.setTitle("Proyecto 1");  
//QUITAR MARGENES  
this.setLayout(null);  
//TAMAÑO DE MI VENTANA  
//POSICION X, POSICION Y, TAMAÑO X, TAMAÑO Y  
this.setBounds(300, 100, 700, 700);  
//QUITAR EL CAMBIO DE TAMAÑO  
this.setResizable(false);  
//CERRAR Y TERMINAR EL PROCESO  
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
//HACER VISIBLE LA VENTANA  
this.setVisible(true);
```

Está será una de varias ventanas que podemos tener y su creación es la misma para todas.

## 3.8. Añadir una Pestaña a una Ventana.

```
//Mandamos a llamar la pestaña Libros  
Pestana_Libro pl = new Pestana_Libro();  
xx.addTab("Libros", null, pl, null);
```

Estas dos líneas son las únicas que necesitamos donde creamos la ventana que unirá las pestañas para mandarla a llamar y es crear un objeto de la pestaña que creamos y mandarlo a llamar por un addTab.



# BIBLIOTECA MS©

## 4. FUNCIONALIDAD Y CREACIÓN DE UNA TABLA

### 4.1. Crear una Tabla.

```
//CREACION DE TABLA
String[] cabeza = {"Nombre Libro", "ID Libro", "Autor", "tipo", "Copias", "Disponibles", "Ocupados"}; //Arreglo del encabezado
datos = mlibro();
tablalibros = new JTable(datos, cabeza);
JScrollPane js1 = new JScrollPane(tablalibros);
js1.setBounds(300, 10, 950, 330);
DefaultTableModel modelo = new DefaultTableModel(datos, cabeza) {
    public boolean isCellEditable(int row, int column) {
        return false;
    }
};
this.add(js1);
```

La creación de la tabla es algo sencillo, solamente se hace un string para colocar el encabezado, los datos lo veremos en el siguiente punto y el boolean que le ingresamos es para que la tabla no sea editable.

### 4.2. Método para añadir a cualquier tabla (Utilizando contadores de variables globales)

```
public static void crearlib(ObLibros nuevoL) {
    if (contlibros < oblibros.length) {
        oblibros[contlibros] = nuevoL;
        contlibros++;
    }
}
```

Este método únicamente nos sirve para ir añadiendo mas libros, usuarios, prestamos etc, pero al llegar al límite ya no ingresara mas (Los límites son las variables globales)





# BIBLIOTECA MS©




## 4.3. Método para añadir datos a la Tabla sirve para manualmente y tiene que estar para carga masiva.

```
//FUNCION PARA AÑADIR LOS LIBROS A LA TABLA
public static Object[][] mlibro() {
    Object[][] libros = new Object[contlibros][7];
    for (int i = 0; i < contlibros; i++) {
        if (oblibros[i] != null) {
            libros[i][0] = oblibros[i].getTitulo();
            libros[i][1] = oblibros[i].getIDlibro();
            libros[i][2] = oblibros[i].getAutor();
            if (oblibros[i].getTipos() == 1) {
                libros[i][3] = "Libro";
            } else if (oblibros[i].getTipos() == 2) {
                libros[i][3] = "Revista";
            } else if (oblibros[i].getTipos() == 3) {
                libros[i][3] = "Libro Electronico";
            }
            libros[i][4] = oblibros[i].getCopias();
            libros[i][5] = oblibros[i].getDisponibles();
            libros[i][6] = oblibros[i].getOcupados();
        }
    }
    return libros;
}
```

Este método funciona para que los datos ingresados por el usuario manualmente vayan directamente a la tabla.

## 5. CARGA MASIVA.

La carga masiva sirve para que al momento de tener que ingresar datos no lo hagamos de uno en uno si no que se puedan hacer cargando un archivo en nuestro caso es un archivo en formato json.

 libros	8/03/2022 13:01	Archivo JSON	3 KB
 prestamos	20/02/2022 17:19	Archivo JSON	1 KB
 usuarios	7/03/2022 16:17	Archivo JSON	1 KB



# BIBLIOTECA MS©

## 5.1. Método para abrir una Ventana Emergente, seleccionar archivo json y leerlo.

```
public void leerarchivo() {  
    try {  
        JFileChooser fc = new JFileChooser();  
        int op = fc.showOpenDialog(this);  
        if (op == JFileChooser.APPROVE_OPTION) {  
            json = fc.getSelectedFile();  
        } else {  
            System.out.println("No abriste nada");  
        }  
        lectura = new FileReader(json); //Obtendremos el texto  
        buff = new BufferedReader(lectura); //leera el texto  
        String casilla;  
        //Lo siguiente se leera linea a linea  
        while ((casilla = buff.readLine()) != null) {  
            contenido += casilla;  
        }  
        cargamaslibros(contenido);  
    } catch (Exception e) {  
        e.printStackTrace();  
    } finally {  
        try {  
            if (null != lectura) {  
                lectura.close();  
            }  
        } catch (Exception e2) {  
            e2.printStackTrace();  
        }  
    }  
}
```

Este método lo que hace es primero llamar a una ventana emergente donde abriremos el archivo tipo json, luego obtiene el texto y lo lee, por ultimo llamamos a otro método que mostraremos abajo.



# BIBLIOTECA MS©

## 5.2. Método para carga masiva usando gson-2.2.

```
public void cargamaslibros(String content) {  
    JsonParser parser = new JsonParser();  
    Object contenido = parser.parse(content);  
    JsonObject objetito = (JsonObject) contenido;  
    Object jsonarrayobyeto = objetito.get("Libros");  
    JSONArray arreglo = (JSONArray) jsonarrayobyeto;  
    System.out.println("Cantidad Objetos: " + arreglo.size());  
    for (int i = 0; i < arreglo.size(); i++) {  
        JsonObject objeto = arreglo.get(i).getAsJsonObject();  
        String Titulo = objeto.get("Titulo").getString();  
        int ID = objeto.get("ID").getAsInt();  
        String Autor = objeto.get("Autor").getString();  
        int Tipo = objeto.get("Tipo").getAsInt();  
        int Copias = objeto.get("Copias").getAsInt();  
        int Disponibles = objeto.get("Disponibles").getAsInt();  
        int Ocupados = objeto.get("Ocupados").getAsInt();  
        ObLibros nuevo = new ObLibros(Titulo, ID, Autor, Tipo, Copias, Disponibles, Ocupados);  
        crearlib(nuevo);  
    }  
}
```

Este método es utilizado con una librería externa, la cual nos ayudara a poder obtener los datos del archivo json y conforme un for agregarlo a la tabla, como se podrán dar cuenta al final del método llama a nuestro método de contar libros en este caso, para que sepa que no se debe pasar del límite y este es el método llamado en el punto 5.1.



# BIBLIOTECA MS©

## 6. REPORTES.

### 6.1. Método para crear el reporte en html.

```
public void reporteU() {
    String nombreReporte;
    File reporte;
    FileWriter fw;
    BufferedWriter br;
    String cadenaHTML;

    try {
        nombreReporte = "RU.html"; //Nombre del archivo html
        reporte = new File(nombreReporte);
        fw = new FileWriter(reporte);
        br = new BufferedWriter(fw);

        cadenaHTML = "<html>"
            + "    <head>"
            + "    <body>"
            + "        <table border = 2>"
            + "            <tr>"
            + "                <td>ID</td>" //Esto es para el encabezado
            + "                <td>Usuario</td>"
            + "                <td>Password</td>"
            + "                <td>Facultad</td>"
            + "                <td>Carrera</td>"
            + "                <td>Tipo</td>"
            + "            </tr>";

        for(int i = 0; i < contusuario; i++){ //MI CANTIDAD DE ARREGLO MI CONTADOR DE USUARIO
            if(obuser[i] != null){ //Llamar arreglo
                cadenaHTML += "            <tr>"
                    + "                <td>" + obuser[i].getID() + "</td>" //llamamos lo que contiene la tabla
                    + "                <td>" + obuser[i].getUsuario() + "</td>"
                    + "                <td>" + obuser[i].getPassword() + "</td>"
                    + "                <td>" + obuser[i].getFacultad() + "</td>"
                    + "                <td>" + obuser[i].getCarrera() + "</td>"
                    + "                <td>" + obuser[i].getTipo() + "</td>"
                    + "            </tr>";
            }
        }

        cadenaHTML += "        </table>"
            + "    </body>"
            + "</html>";

        br.write(cadenaHTML);

        br.close();
        fw.close();

        pdfusuario(cadenaHTML);

    } catch (IOException ex) {
        System.out.println("error escribiendo el reporte. Detalles " + ex.getMessage());
    }
}
```





# BIBLIOTECA MS©

Este método es utilizado para crear el reporte en HTML para luego crear un método mas sencillo para la creación del PDF.

## 6.2. Método para crear PDF.

```
//MÉTODO PARA CREAR PDF DE USUARIOS
public void pdfusuario(String html){
    try{ //E métodos para generar 3 reportes de arriba y abajo
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("ddMMyyyyHHmmss");
        String nombre = "reporteUsuarios_"+dtf.format(LocalDateTime.now());

        Document document = new Document(PageSize.LETTER);
        PdfWriter.getInstance(document, new FileOutputStream(nombre+".pdf"));

        document.open();
        document.addAuthor("Victor Rodriguez");
        document.addCreator("Victor Rodriguez");
        document.addSubject("reporteUsuarios");
        document.addCreationDate();
        document.addTitle("ReporteUsuario");

        HTMLWorker htmlWorker = new HTMLWorker(document);
        htmlWorker.parse(new StringReader(html));

        document.close();
        DateTimeFormatter dtf1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        String FECHA = dtf1.format(LocalDateTime.now());
        ObRegistros or = new ObRegistros(FECHA, ReporteU, "Reporte 1");
        crearreporte(or);

    }catch(Exception e){
        e.printStackTrace();
    }
}
```

Este método es el que nos sirve y es llamado en el anterior para poder crear el pdf que ayudara a tener los reportes según el tipo que necesitamos.



# BIBLIOTECA MS©

## 7. VALIDACIÓN USUARIOS.

### 7.1. Función para validación de usuario.

```
public boolean verificarusu(String user) {  
    for (int i = 0; i < contusuario; i++) {  
        if (obuser[i].getUsuario().equals(user)) {  
            return true;  
        }  
    }  
    return false;  
}
```

Esta función nos ayuda a verificar que de nuestro objeto usuarios tengamos la validación de que el Usuario será el mismo user que la carga masiva.

### 7.2. Función para validación de contraseña.

```
public boolean verificarcontra(String user, String contra) {  
    for (int i = 0; i < contusuario; i++) {  
        if (obuser[i].getUsuario().equals(user)) {  
            if (obuser[i].getPassword().equals(contra)) {  
                return true;  
            }  
        }  
    }  
    return false;  
}
```

Esta función nos ayuda a verificar el usuario de la carga masiva, con el nombre de usuario y así mismo su contraseña.



# BIBLIOTECA MS©

## 7.3. Función para retornar un Usuario.

```
public ObUsuarios retornarusuario(String user) {  
    for (int i = 0; i < contusuario; i++) {  
        if (obuser[i].getUsuario().equals(user)) {  
            return obuser[i];  
        }  
    }  
    return null;  
}  
  
public static String ReporteU = "";
```

Esta función nos ayuda a retornar un usuario de nuestro arreglo de usuarios en json.

## 7.4. Método para la validación de usuario, contraseña y tipo.

```
public void ingresar() {  
    String usul, pass;  
    usul = inusuario.getText();  
    pass = incontra.getText();  
    ReporteU = usul;  
    if (verificarusu(usul) == true) {  
        if (verificarcontra(usul, pass) == true) {  
            if (retornarusuario(usul).getTipo() == 1) {  
                JOptionPane.showMessageDialog(this, "Bienvenido Jefe");  
                this.setVisible(false);  
                VentanaPrincipal vp = new VentanaPrincipal();  
            } else if (retornarusuario(usul).getTipo() == 2) {  
                JOptionPane.showMessageDialog(this, "Usted no cuenta con los permisos para acceder al sistema");  
            }  
        } else {  
            JOptionPane.showMessageDialog(this, "Contraseña Invalida");  
        }  
    } else {  
        JOptionPane.showMessageDialog(this, "Usuario Invalido");  
    }  
}
```

En este método hacemos la validación para decir que si es el usuario administrador pueda entrar y si es el usuario normal no pueda mediante el tipo 1 o 2.

## 8. DARLE VIDA A LOS BOTONES.



# BIBLIOTECA MS©

En este caso al momento de crear un botón se crea un método automáticamente donde tendremos que colocar los métodos si es que hacemos alguno para la funcionalidad del mismo o ahí mismo crear las cosas, en la siguiente imagen se muestra como se le dio vida a los botones de nuestro login.

```
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == masiva) {
        System.out.println("Carga de archivo Json");
        leerarchivos();
        System.out.println(contenido);
    }
    if (e.getSource() == login) {
        ingresar();
    }
}
```

En ambos se les manda a llamar un método que utilizamos, tanto en carga masiva como en la validación.