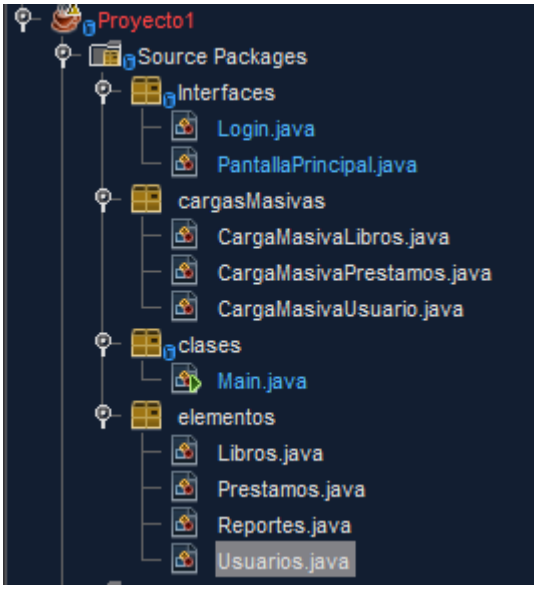
# Manual Técnico

En este manual técnico se explicará de manera resumida el funcionamiento de ciertos, métodos, variables de importancia.

Nuestro programa está divido en tres paquetes, para tener un mejor ordenamiento.




Podemos observar cuatros packages, el primero “Interfaces” el cual cuenta con las gráficas visuales que el usuario puede ver al momento de correr el programa y cuenta con las funcionalidades de los botones, donde cada botón tiene cierta función que se conecta con cada clase de nuestro programa, de segundo tenemos “CargasMasivas” donde se encuentran 3 ventanas visuales encargadas de recibir el texto en formato JSON, decodificarlo y manipular los datos que se contengan en él, luego se encuentra “clases” donde está la clase main para ejecutar nuestro programa y en donde se encuentra la mayoría de nuestros métodos y los arreglos que manejaran toda la información de nuestro programa, por ultimo esta “elementos” que cuenta con los objetos de este proyecto.

Empezaremos con los métodos de la clase main, donde se encuentra lo más importante.

```
//AGREGAR TODOS LOS ARREGLOS QUE VAMOS A UTILIZAR
public static Usuarios[] adminNow = new Usuarios[1]; //Pers

public static Usuarios[] users = new Usuarios[50]; //Arregl
public static Libros[] libros = new Libros[100]; //Arreglo
public static Prestamos[] prestamos = new Prestamos[200];//
public static Reportes[] reportes = new Reportes[100];//Arr
public static int cUsers = 0; //Contador User
public static int cLibros = 0; //Contador libros
public static int cPrestamos = 0;//Contador prestamos
public static int cReportes = 0;//Contador reportes
```

Acá se encuentran los arreglos de cada objeto creado y sus respectivos contadores que serán utilizados en los métodos de agregar. En estos se manejará toda la información de nuestro programa.

```java
//Agregar usuario
    public static void agregarUsuario(Usuarios usuario) {
        if (cUsers < users.length) {
            users[cUsers] = usuario;
            cUsers += 1;
        } else {
            JOptionPane.showMessageDialog(new Login(), "Se llego al limite");
        }
    }
    //Agregar libro
    public static void agregarLibro(Libros librosIngresados) {
        if (cLibros < libros.length) {
            libros[cLibros] = librosIngresados;
            cLibros += 1;
        } else {
            JOptionPane.showMessageDialog(new Login(), "Se llego al limite de libros");
        }
    }

    //Agregar prestamo
    public static void agregarPrestamo(Prestamos prestamosIngresados) {
        if (cPrestamos < prestamos.length) {
            prestamos[cPrestamos] = prestamosIngresados;
            cPrestamos += 1;
        }
    }
    //Agregar Reporte
    public static void agregarReporte(Reportes reportesIngresados){
        if(cReportes < reportes.length){
            reportes[cReportes] = reportesIngresados;
            cReportes +=1;
        }
    }
```

Acá se encuentran los cuatro métodos para agregar los objetos, estos métodos como parámetro reciben un objeto usuario y evalúan si es que el contador respectivo es menor que el tamaño del arreglo, y si es así, entonces se agrega el objeto a la posición del contador de cada arreglo declarado, después se le aumenta en uno a cada contador hasta llegar al valor máximo de cada arreglo y ejecutar el respectivo else.

```java
//VerificarUsuario
    public static Usuarios verificarUser(String usuario, String pass) {
        for (int i = 0; i < cUsers; i++) {
            if (users[i].getUsuario().equals(usuario) && users[i].getPassword().equals(pass)) {
                return users[i];
            }
        }
        return null;
    }
```

Este método recibe como parámetros dos string que vienen de la clase login y retornan un objeto usuario, se itera y se evalúa que los datos sean iguales a los que se encuentran en el arrelgo y se retorna el objeto en esa posición, sino retorna null.

```java
//Verificar Usuario Tipo
    public static boolean verificarUserTIpo(String tipo) {
        for (int i = 0; i < cUsers; i++) {
            if (users[i].getUsuario().equals(tipo)) {
                if (users[i].getTipo().equals(1l)) {
                    return true;
                }

            }
        }
        return false;
    }
```

Este retorna un booleano y recibe un parámetro de tipo String, el cual se evalúa que el parámetro recibido es igual al nombre del usuario en la posición i, entonces vuelve a evaluar que si en esa posición i el tipo de usuario es igual a 1 entonces retorna true, de lo contrario false.

```java
public static boolean verificarPrestamo(Long IDUsuario, Long IDLibro)
{
        boolean iduser = false;
        boolean idlibro = false;
        for (int i = 0; i < cUsers; i++) {
            if (users[i].getiD().equals(IDUsuario)) {
                iduser = true;

            }

        }
        for (int i = 0; i < cLibros; i++) {
            if (libros[i].getID().equals(IDLibro)) {
                idlibro = true;

            }

        }
        return iduser == true && idlibro == true;
    }
```

Este método retorna booleano y recibe dos tipos Long, verifica que los ID libro y libro existan en los arreglos para poder realizar un préstamo y retorna true, de lo contrario false.

```java
public static void administrarPrestamosPrestado(String prestamoLibro) {
        for (int i = 0; i < cLibros; i++) {
            if (libros[i].getTitulo().equals(prestamoLibro)) {
                //Se actualiza disponibles y ocupados
                libros[i].setDisponibles(libros[i].getDisponibles() - 1);
                libros[i].setOcupados(libros[i].getOcupados() + 1);
            }
        }
    }
```

Este método accede al nombre del usuario que se encuentra en el arreglo y modifica disponibles y ocupados si la fecha de entrega es prestada.

```java
public static Object[][] tablaPrestamos() throws ParseException {
        int filas = Main.cPrestamos;
        Object[][] arregloTabla2 = new Object[filas][4];
        for (int i = 0; i < filas; i++) {
            arregloTabla2[i][0] = Main.prestamos[i].getUsuarioID();
            arregloTabla2[i][1] = Main.prestamos[i].getLibroID();
            arregloTabla2[i][2] = Main.prestamos[i].getFechaEntrega();
            arregloTabla2[i][3] = Main.prestamos[i].getStatus();
        }
        return arregloTabla2;
    }
```

Este método retorna una matriz de los atributos del arreglo préstamo, que servirán para poner en los JTable

```java
    //Administrar que disponibles llegue a 0
    public static boolean verificarDisponibles() {
        for (int i = 0; i < Main.cLibros; i++) {
            if (Main.libros[i].getDisponibles().equals(0l)) {
                return true;
            }
        }
        return false;
    }
    //Administrar que ocupados no se pase de copias que hay
    public static boolean verificarOcupadosMax(){
        for (int i = 0; i < Main.cLibros; i++) {
            if(Main.libros[i].getOcupados() > Main.libros[i].getCopias()){
                return true;
            }
        }
        return false;
    }
    //Administrar que ocupados este en 0, porque sino no se puede entragr un
libro si no esta ocupado
    public static boolean verificarOcupados() {
        for (int i = 0; i < Main.cLibros; i++) {
            if (Main.libros[i].getOcupados().equals(0l)) {
                return true;
            }
        }
        return false;
    }
    }
```

Estos tres métodos retornan valores booleanos si recorrer y evaluar el arreglo libro los atributos de disponibles, ocupados son iguales a 0.

```java
public static void PDFUsuarios(String fechaActual) throws DocumentException,
FileNotFoundException{
        Document documento = new Document(PageSize.LETTER);

        OutputStream archivo = new
FileOutputStream("E:\\ReportesProyecto1\\Reportes_Usuarios\\reporteUsuarios_"+fech
aActual + ".pdf");
        PdfWriter.getInstance(documento, archivo);
        documento.open();
        Paragraph p = new Paragraph();
        Paragraph p2 = new Paragraph();
        p.add("REPORTES BIBLIOTECA USAC");
        p.setAlignment(1);
        p2.add("Registro de usuarios en el sistema");
        p2.setAlignment(1);
        //Creamos la tabla registro de libros
        PdfPTable tabla = new PdfPTable(5);
        tabla.setHorizontalAlignment(1);
        tabla.addCell("ID");
        tabla.addCell("Usuario");
        tabla.addCell("Facultad");
        tabla.addCell("Carrera");
        tabla.addCell("Tipo");
        for (int i = 0; i < Main.cUsers; i++){
            tabla.addCell(Main.users[i].getiD().toString());
            tabla.addCell(Main.users[i].getUsuario());
            tabla.addCell(Main.users[i].getFacultad());
            tabla.addCell(Main.users[i].getCarrera());
            tabla.addCell(Main.users[i].getTipo().toString());
        }
        //Agregar
        documento.add(p);
        documento.add(Chunk.NEWLINE);
        documento.add(p2);
        documento.add(Chunk.NEWLINE);
        documento.add(tabla);
        documento.close();
    }
```

Existen tres métodos los cuales generan los tres PDF´s distintos, este método crea un PDF de usuarios usando la librería iText 5, el cual trae métodos para crear tablas y mediante un for pasarle los datos guardados en nuestro arreglo.

```java
public static void tipoMes(int numeroMes){
        switch(numeroMes){
            case 1:
                Main.enero ++;
                break;
            case 2:
                Main.febrero ++;
                break;
            case 3:
                Main.marzo ++;
                break;
            case 4:
                Main.abril ++;
                break;
            case 5:
                Main.mayo ++;
                break;
            case 6:
                Main.junio ++;
                break;
            case 7:
                Main.julio ++;
                break;
            case 8:
                Main.agosto ++;
                break;
            case 9:
                Main.septiembre ++;
                break;
            case 10:
                Main.octubre ++;
                break;
            case 11:
                Main.noviembre ++;
                break;
            case 12:
                Main.diciembre ++;
                break;
            default:
                System.out.println("Error");

        }
    }
```

```java
public static void tipoLibros(String tipo){
        switch(tipo){
            case "Libro":
                Main.ctipoLibros ++;
                break;
            case "Revista":
                Main.ctipoRevista ++;
                break;
            case "Libro electronico":
                Main.ctipoLibroElec ++;
                break;
            default:
                System.out.println("Error");
        }
    }
```

Estos dos métodos tienen la funcionalidad de saber qué tipo de mes son para sumarle uno al contador de cada mes y el tipo de libro que es para sumarle a cada contador respectivamente, estos métodos servirán para las gráficas.

```java
if (e.getSource() == b1) {

            String texto = ta1.getText();

            Object jsonObteto = JSONValue.parse(texto);
            JSONObject obteto = (JSONObject) jsonObteto;

            Object jsonarrayoLibro = obteto.get("Libros");
            JSONArray arrayobjetoLibro = (JSONArray) jsonarrayoLibro;

            for (Object objeto_inarray : arrayobjetoLibro) {
                JSONObject objeto_value = (JSONObject) objeto_inarray;
                titulo = (String) objeto_value.get("Titulo");
                ID = (Long) objeto_value.get("ID");
                autor = (String) objeto_value.get("Autor");
                esTipo = (Long) objeto_value.get("Tipo");
                if(esTipo == 1l){
                    tipo = "Libro";
                }else if(esTipo == 2l){
                    tipo = "Revista";
                }else if(esTipo == 3l){
                    tipo = "Libro electronico";
                }
                copias = (Long) objeto_value.get("Copias");
                disponibles = (Long) objeto_value.get("Disponibles");
                ocupados = (Long) objeto_value.get("Ocupados");

                //Se crea un nuevo libro
                Libros nuevoLibro = new Libros(ID, titulo, autor, tipo, copias,
disponibles, ocupados);
                //Agregar libro
                Main.agregarLibro(nuevoLibro);
                //Cargar tablero
                //Guardar que tiplo de libro es
                Main.tipoLibros(tipo);

            }
            JOptionPane.showMessageDialog(this, "Se ha agregado la carga masiva
de libros");
            ta1.setText("");
            Main.leerLibros();

        }
```

En los botones de las ventanas de las clases donde se pueden hacer las cargas masivas de los usuarios, préstamos y libros, utilizan la librería JSON para decodificar el formato String recibido del textArea. Primero se parsea a un objeto, luego a un objeto JSON, luego este objeto lo hace un JSONArray y por último se recorre con un ForEach para obtener los valores de cada JSON, se crea el nuevo objeto y se agrega mediante el método de agregar.

```java
public static String verificarFecha(String fecha) throws ParseException {
        try
        {
            SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");
            Calendar calendario = Calendar.getInstance();
            Date FechaActual = calendario.getTime();

            Date fechaIngresada = formatoFecha.parse(fecha);
            if (FechaActual.before(fechaIngresada))
            {
                return "Prestado";
            } else
            {
                return "Entregado";
            }
        } catch (ParseException ex)
        {
            System.out.println("ERORRRRRRESRJWEUJR");
        }
        return null;
    }
```

Este método verifica la fecha ingresada manualmente en nuestra pestaña préstamos y la evalúa con la fecha del sistema para retornar el status de libro. Se obtiene un String y este se parsea a Date.

```java
public static int obtenerMes(String fecha) throws ParseException {
        SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");
        Date fechaActual = formatoFecha.parse(fecha);
        SimpleDateFormat formatoMes= new SimpleDateFormat("MM");
        String mes = formatoMes.format(fechaActual);
        int numeroMes = Integer.parseInt(mes);
        return numeroMes;
    }
```

Con este método usando nuevamente simpleDateFormat parseamos un String a un objeto de tipo Date y obtenemos el número de mes mediante un método especial de format para luego retornar el numero obtenido de ese mes del año 2022.

```java
//Generar la fecha y hora de hoy
    public static String generarFecha() {
        String fechaActual = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss").format(Calendar.getInstance().getTime());
        return fechaActual;
    }

    //Generar fecha para los reportes
    public static String fechaReportes() {
        String fechaActual = new
SimpleDateFormat("ddMMyyyyHHmmss").format(Calendar.getInstance().getTime());
        return fechaActual;
    }
```

Estos dos métodos tienen casi la misma funcionalidad, que es generar la fecha actual del sistema y retornarla como String para ser utilizadas como variables en otros métodos.

```java
public static void tableroLibros() {

        librosDatos = Main.tablaLibros();
        String[] columnas1 =
        {
            "ID libro", "Nombre libro", "Autor", "Tipo", "Copias", "Disponibles", "Ocupados"
        };
        table1 = new JTable(librosDatos, columnas1);
        DefaultTableModel modelo = new DefaultTableModel(librosDatos,columnas1){
            @Override
            public boolean isCellEditable(int row, int column){
                return false;
            }
        };
        table1.setModel(modelo);
        scrolltable = new JScrollPane(table1);

        scrolltable.setBounds(0, 0, 950, 550);
        scrolltable.setVisible(true);
        p2.add(scrolltable);

    }
```

Este método se manda a llamar en el constructor de pantallaPrincipal el cual genera la tabla que se ve en el GUI, se crea una tabla y se le manda la matriz de los datos, y los nombres de las columnas, luego sigue un método para que no se pueda editar dentro de las celdas, por último, se añade al panel declarado inicialmente. Este mismo código se repite con ciertas modificaciones para las otras dos tablas del GUI.

```java
public boolean verificarIDLibro() {
        for (int i = 0; i < Main.cLibros; i++)
        {
            if (Main.libros[i].getID() == ID)
            {
                return true;
            }

        }
        return false;
    }
```

Este método evalúa que el ID de un libro no este repetido, con un for se recorre el arreglo de libros y si el ID ya se encuentra, entonces retorna un true, sino false.

```java
public static void grafica1() {
        DefaultPieDataset datos = new DefaultPieDataset();

        datos.setValue("Administradores", Main.ctipoAdmin);
        datos.setValue("Estudiantes", Main.ctipoEstudiante);

        JFreeChart grafica1 = ChartFactory.createPieChart("Gráfica de usuarios registrados", datos, true, true, false);

        ChartPanel panel = new ChartPanel(grafica1);
        panel.setBounds(10, 10, 450, 500);
        panelGraficos.add(panel);
    }
```

```java
public static void grafica2(){
        DefaultCategoryDataset datos = new DefaultCategoryDataset();
        datos.addValue(Main.enero, "Libros Prestados", "Ene");
        datos.addValue(Main.febrero, "Libros Prestados", "Feb");
        datos.addValue(Main.marzo, "Libros Prestados", "Mar");
        datos.addValue(Main.abril, "Libros Prestados", "Abril");
        datos.addValue(Main.mayo, "Libros Prestados", "May");
        datos.addValue(Main.junio, "Libros Prestados", "Jun");
        datos.addValue(Main.julio, "Libros Prestados", "Jul");
        datos.addValue(Main.agosto, "Libros Prestados", "Ago");
        datos.addValue(Main.septiembre, "Libros Prestados", "Sep");
        datos.addValue(Main.octubre, "Libros Prestados", "Oct");
        datos.addValue(Main.noviembre, "Libros Prestados", "Nov");
        datos.addValue(Main.diciembre, "Libros Prestados", "Dic");

        JFreeChart graficoBarras = ChartFactory.createBarChart3D("Libros prestados 2022", "Meses", "Total de libros
prestados", datos,PlotOrientation.VERTICAL,true,true,false);
        ChartPanel panel = new ChartPanel(graficoBarras);
        panel.setBounds(450, 10, 470, 500);
        panelGraficos.add(panel);

    }
```

```java
public static void grafica3() {
        DefaultPieDataset datos = new DefaultPieDataset();

        datos.setValue("Libros", Main.ctipoLibros);
        datos.setValue("Revistas", Main.ctipoRevista);
        datos.setValue("Libros electronicos", Main.ctipoLibroElec);

        JFreeChart grafica1 = ChartFactory.createPieChart("Gráfica tipos de libros", datos, true, true, false);

        ChartPanel panel = new ChartPanel(grafica1);
        panel.setBounds(920, 10, 450, 500);
        panelGraficos.add(panel);
    }
```

Existen tres métodos para las tres graficas distintas que hay, donde las tres tienen parentesco, mediante el uso de dos librerías externas JFreechart y JCommon se declaran los valores que queremos poner dentro de nuestra gráfica y luego instanseamos un objeto para la gráfica y se añade a un panel especial, luego este panel se agrega a nuestro panel de gráficas.

Luego seguimos con los botones de la pantalla principal:

El botón de registrar libro y cargar libros

```java
if (e.getSource() == b1)
        {

            ID = Long.parseLong(tf1.getText());
            boolean validarID = verificarIDLibro();
            if (validarID)
            {
                JOptionPane.showMessageDialog(this, "EL ID del libro ya existe \n Ingrese nuevamente");
                tf1.setText("");

            } else
            {
                libro = tf2.getText();
                autor = tf3.getText();
                copias = Long.parseLong(tf4.getText());
                ID = Long.parseLong(tf1.getText());
                if (com1.getSelectedItem().toString() == "1")
                {
                    tipo = "Libro";
                } else if (com1.getSelectedItem().toString() == "2")
                {
                    tipo = "Revista";
                } else if (com1.getSelectedItem().toString() == "3")
                {
                    tipo = "Libro electronico";
                }


                //Creando el objeto libro
                Libros libroNuevo = new Libros(ID, libro, autor, tipo, copias, copias, 0l);
                Main.agregarLibro(libroNuevo);
                Main.leerLibros();
                //Guardar que tiplo de libro es
                Main.tipoLibros(tipo);
                //Se actualiza la tabla cada vez que se agrega un libro
                this.dispose();
                try
                {
                    PantallaPrincipal principal = new PantallaPrincipal();
                } catch (ParseException ex)
                {

                }

                JOptionPane.showMessageDialog(this, "Se ha agreado el libro con exito");

                tf1.setText("");
                tf2.setText("");
                tf3.setText("");
                tf4.setText("");
                Main.leerUsuarios();
                Main.leerLibros();
                Main.leerPrestamo();
            }

        }
```

```java
if (e.getSource() == b2)
        {
            //Creando la ventana
            CargaMasivaLibros cargaLibros = new CargaMasivaLibros();
            this.dispose();
        }
```

Obteniendo los valores de los textField y el uso de los métodos descritos anteriormente, se evalúa y verifica para poder crear un objeto de tipo libro para luego ser almacenado en el arreglo correspondiente. Luego el botón de carga masiva de libros que fue descrito su funcionalidad con anterioridad.

El botón de hacer un préstamo y cargar un préstamo.

```
if (e.getSource() == b2_1)
        {
            IDUsuario = Long.parseLong(tf2_1.getText());
            IDLibro = Long.parseLong(tf2_2.getText());
            fechaEntrega = tf2_3.getText();

            try
            {
                if (PantallaPrincipal.verificarFecha(fechaEntrega).equals("Prestado"))
                {
                    //Metodo de contadores del mes
                    int numeroMes = PantallaPrincipal.obtenerMes(fechaEntrega);
                    Main.tipoMes(numeroMes);
                    System.out.println("Mes no. " + numeroMes);
                }

            } catch (ParseException ex)
            {
                Logger.getLogger(CargaMasivaPrestamos.class.getName()).log(Level.SEVERE, null, ex);
            }
            //Verificar que este en el sistema el libro y usuario
            boolean prestamoEs = Main.verificarPrestamo(IDUsuario, IDLibro);
            //HACER MEJOR UN SWITCH JUSTO ACA

            if (prestamoEs)
            {
                try
                {
                    //Transformar ese Long a el nombre que pertenece el ID
                    for (int i = 0; i < Main.cUsers; i++)
                    {
                        if (Main.users[i].getiD().equals(IDUsuario))
                        {
                            prestamoUsuario = Main.users[i].getUsuario();
                        }

                    }
                    for (int i = 0; i < Main.cLibros; i++)
                    {
                        if (Main.libros[i].getID().equals(IDLibro))
                        {
                            prestamoLibro = Main.libros[i].getTitulo();
                        }

                    }
                    //Si disponibles y ocupados
                    if (Main.verificarDisponibles())
                    {
                        JOptionPane.showMessageDialog(this, "Ya no hay libros disponibles para prestar");
                    } //Si ocupados es lo max de copias ya no se puede hacer prestados
                    else if (Main.verificarOcupadosMax())
                    {
                        JOptionPane.showMessageDialog(this, "Ya estan todas las copias ocupadas");
                    } //Si disponibles es diferente de cero, ocupados es igual a cero y la fecha es entregado, entonces solo
se agrega el prestamo
                    else if (Main.verificarDisponibles() != true && Main.verificarOcupados() != true &&
verificarFecha(fechaEntrega) == "Entregado")
```

```
                else if (Main.verificarDisponibles() && Main.verificarOcupados() != true && verificarFecha(fechaEntrega)
== "Entregado")
                {
                    Prestamos prestamoNuevo = null;
                    try
                    {
                        prestamoNuevo = new Prestamos(prestamoUsuario, prestamoLibro, fechaEntrega,
verificarFecha(fechaEntrega));
                    } catch (ParseException ex)
                    {

                    }
                    Main.agregarPrestamo(prestamoNuevo);
                    //Main.administrarPrestamosEntregado(prestamoLibro);
                    JOptionPane.showMessageDialog(this, "Se ha hecho un prestamo correctamente");
                    tf2_1.setText("");
                    tf2_2.setText("");
                    tf2_3.setText("");

                    //Actualizar
                    prestamoUsuario = null;
                    prestamoLibro = null;

                    //Leer
                    Main.leerUsuarios();
                    Main.leerLibros();
                    Main.leerPrestamo();
                    this.dispose();
                    try
                    {
                        PantallaPrincipal principal = new PantallaPrincipal();
                    } catch (ParseException ex)
                    {

                    }
                }

            } catch (ParseException ex)
            {

            }

        } else
        {
            JOptionPane.showMessageDialog(this, "Algún ID no existe");
            tf2_1.setText("");
            tf2_2.setText("");
        }

    }
```

```
if (e.getSource() == b2_2)
        {
            System.out.println("Click carga masiva de prestamos");
            //Creando la nueva ventana
            CargaMasivaPrestamos cargaPrestamos = new CargaMasivaPrestamos();
            this.dispose();
        }
```

Este botón es muy largo ya que contiene varias validaciones, primero se obtiene los valores de los textField y se verifica que esos ID existan en el sistema, posteriormente se evalúa el status del préstamo y dependiendo de eso diferentes condiciones se evalúan, donde cada uno contiene el mismo código que crea préstamos y los agrega al arreglo. Luego está el botón de carga masiva el cual fue explicado anteriormente.

Por último, se tiene el botón de generar reportes.

```
if (e.getSource() == b3_1)
        {
            System.out.println("Click en generar reporte");
            //Poner que tipo es y dependiendo de eso genera el reporte
            fechaDeGeneracion = generarFecha();
            usuarioEnUso = Main.adminNow[0].getUsuario();
            if (com3.getSelectedItem().toString() == "1")
            {
                tipodeReporte = "Reporte de usuarios";

            } else if (com3.getSelectedItem().toString() == "2")
            {
                tipodeReporte = "Reporte de libros";
            } else if (com3.getSelectedItem().toString() == "3")
            {
                tipodeReporte = "Reporte de prestamos";
            }
            //Se crea el objeto
            Reportes reporteNuevo = new Reportes(fechaDeGeneracion, usuarioEnUso, tipodeReporte);
            Main.agregarReporte(reporteNuevo);
            JOptionPane.showMessageDialog(this, "Se ha generado un reporte correctamente");
            Main.leerReportes();
            this.dispose();
            try
            {
                PantallaPrincipal principal = new PantallaPrincipal();
            } catch (ParseException ex)
            {

            }

            try
            {
                if (com3.getSelectedItem().toString() == "1")
                {
                    Main.PDFUsuarios(fechaReportes());
                    JOptionPane.showMessageDialog(this, "Se ha generado un PDF de usuarios");

                } else if (com3.getSelectedItem().toString() == "2")
                {
                    //Generar reporte de libros

                    Main.PDFLibro(fechaReportes());
                    JOptionPane.showMessageDialog(this, "Se ha generado un PDF de libros");
                } else if (com3.getSelectedItem().toString() == "3")
                {

                    Main.PDFPrestamo(fechaReportes());
                    JOptionPane.showMessageDialog(this, "Se ha generado un PDF de prestamos");
                }
            } catch (DocumentException ex)
            {
                Logger.getLogger(PantallaPrincipal.class.getName()).log(Level.SEVERE, null, ex);
            } catch (FileNotFoundException ex)
            {
                Logger.getLogger(PantallaPrincipal.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
```

En el cual se evalúa que tipo de reporte desea generar y dependiendo de eso llama su método respectivo el cual genera un PDF y lo guarda en la carpeta de destino. Además, se crea un objeto de tipo préstamo para mostrar en las tablas y ser agregado en nuestro arreglo.