



# **BIBLIOTECA**

## **Proyecto 1 IPC 1**

Universidad de San Carlos de Guatemala  
Facultad de ingeniería  
Escuela de Ciencias y Sistemas  
Laboratorio de Introducción a la Programación 1  
Sección D  
Aux. José Orlando Wannan Escobar  
Manual de Usuario

# Indice

Objetivo.....	4
Requerimientos.....	4
Estructura.....	5
1. Paquete Clases.....	6
2. Paquete Interfazg.....	6
2.1 Login.....	7
2.2 CargadeUsuario.....	8
2.3 PantallaPrincipal.....	10
2.4 PestañaLibros.....	11
2.5 PestañaPrestamos.....	12
2.6 PestañaReportes.....	13
2.7 PestañaGraficos.....	15
2.8 CargaLibros.....	17
2.9 CargaPrestamos.....	18
3. Paquete ipc1.proyecto.pkg1_201902259.....	19
3.1 PantallaPrincipal.....	19
Conclusiones.....	22

# Introducción

En el presente manual se pretende explicar de manera detallada la funcionalidad del programa de Biblioteca, especificando cada función de botones, paneles, entradas de texto, entre otras cosas.

En la implementación del programa de Biblioteca se busca llevar un control de datos de usuario para que estos puedan acceder al programa de una manera sencilla y realizar sus préstamos eficazmente. En complemento a esto se aplica el almacenamiento de datos de libros y préstamos del usuario correspondiente.

# Aplicación Biblioteca

## Objetivo

Permitir a las personas que inician en el ámbito de la programación en Java, como estudiantes de Ciencias y Sistemas o auxiliares de cátedra, puedan visualizar el presente manual para comprender la funcionalidad del programa, así como su estructura.

## Requerimientos

CPU:	Intel de 64 bits, Recomendado: Intel Core i3 o superior
RAM:	4GB, Recomendado: 8 GB o más
Espacio disponible en disco:	8GB
Sistema Operativo:	Windows 10
Resolución de Pantalla:	Básico: pantalla de 1280 x 800
Programas Adicionales	Apache Netbeans IDE 12.6



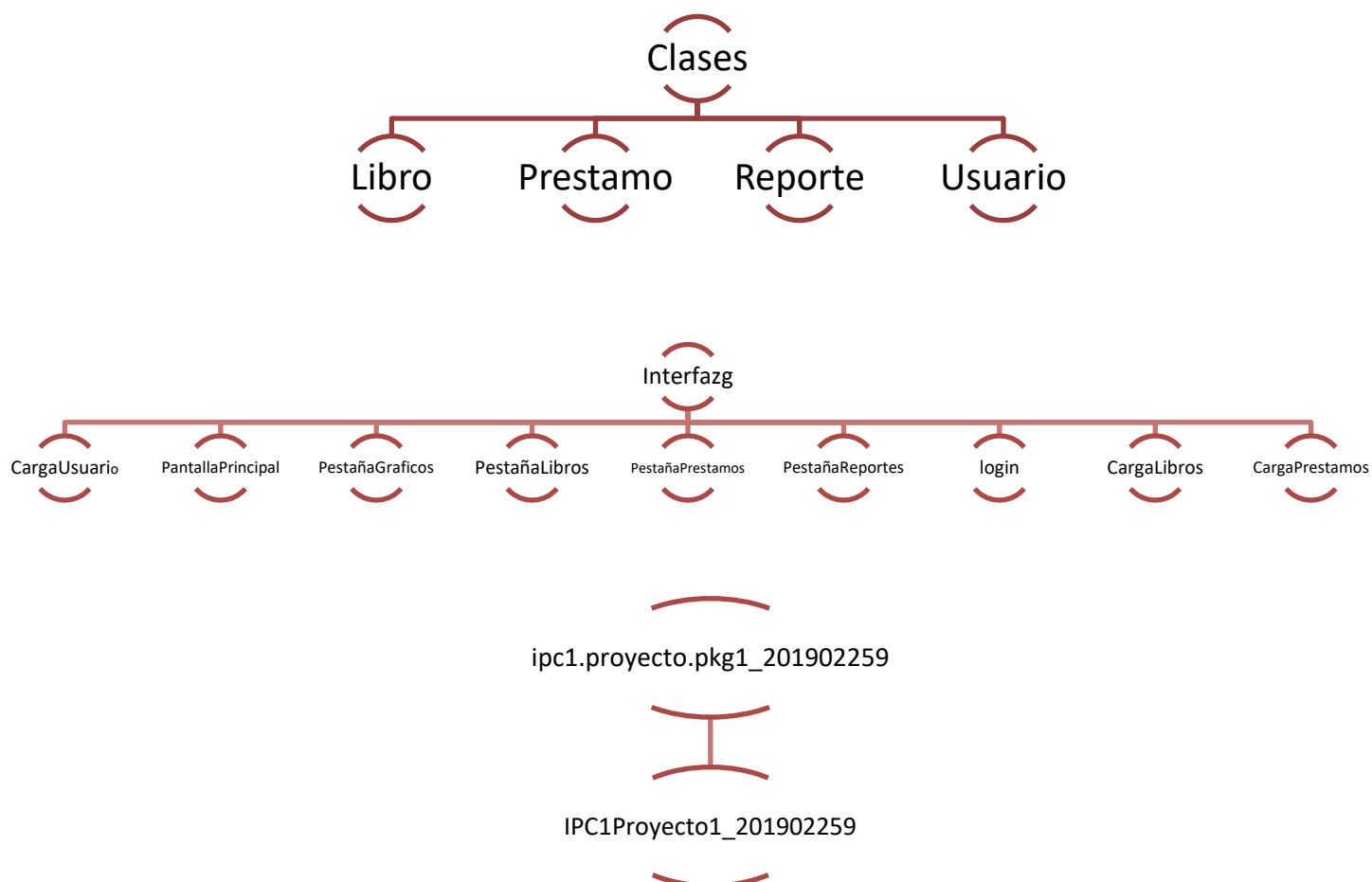


# Estructura

El programa está dividido en los siguientes paquetes:



Cada paquete se subdivide en clases:



# 1. Paquete Clases

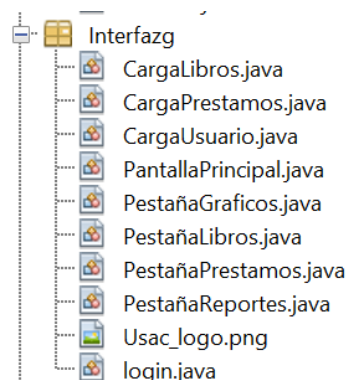
**Libro:** se creó una clase que hace referencia a la creación del Objeto Libro  
**Prestamo:** clase creada para hacer referencia a la creación del Objeto Préstamo.  
**Reporte:** clase creada para hacer referencia a la creación del Objeto Reporte.  
**Usuario:** clase creada para hacer referencia a la creación del Objeto Usuario.

En estas clases se han implementado dos de los pilares de POO, abstracción y encapsulamiento.

```
public class Libro {  
    private int id;  
    private String titulo;  
    private String autor;  
    private int tipo;  
    private int copias;  
    private int disponibles;  
    private int ocupados;  
  
    public Libro(int id, String titulo, String autor, int tipo, int copias, int disponibles, int ocupados) {  
        this.id = id;  
        this.titulo = titulo;  
        this.autor = autor;  
        this.tipo = tipo;  
        this.copias = copias;  
        this.disponibles = disponibles;  
        this.ocupados = ocupados;  
    }  
}
```

## 2. Paquete Interfazg

Dentro de este paquete, se encuentran las clases para almacenar los datos de los componentes de las pestañas, tales como botones, tablas, graficas, entre otras cosas.

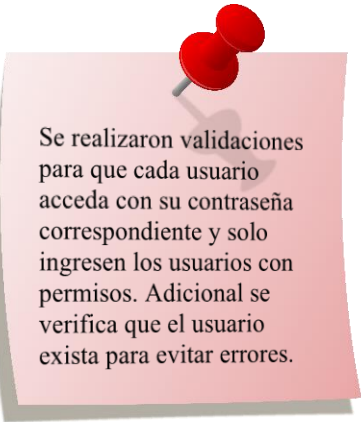


## 2.1 Login

Dentro de esta clase se creó la ventana del login, acompañada de sus respectivos componentes, el botón de carga masiva, el botón de ingresar, y los espacios de texto donde se leerán las credenciales.

```
public class login extends JFrame implements ActionListener {  
  
    Color rosita = new Color(250, 219, 216 );  
    Color moradito = new Color(235, 222, 240);  
    Color celestito = new Color(174, 214, 241);  
  
    //Label  
    JLabel titulo, use, pass, imagen, i1, i2;  
    JButton botlogin, botcarga;  
    JTextField texuser;  
    JPasswordField texpass;  
    static String user;  
  
    public login() {
```

Adicional se generaron los eventos de acción (ActionEvent) en los elementos botones para invocar el método actionPerformed(ActionEvent e) que realiza las acciones programadas para estos eventos.



Se realizaron validaciones para que cada usuario acceda con su contraseña correspondiente y solo ingresen los usuarios con permisos. Adicional se verifica que el usuario exista para evitar errores.

```
@Override  
public void actionPerformed(ActionEvent e) {  
    if (e.getSource() == botlogin) {  
  
        user = texuser.getText();  
        String pass = texpass.getText();  
        if (IPC1Proyecto1_201902259.entrusuario(user) == true) {  
            if (IPC1Proyecto1_201902259.retornus(user).getPassword().equals(pass)) {  
                if (IPC1Proyecto1_201902259.retornus(user).getTipo() == 1) {  
                    PantallaPrincipal p = new PantallaPrincipal();  
                    this.dispose();  
                } else {  
                    JOptionPane.showMessageDialog(this, "Usuario sin permisos para acceder al sistema");  
                }  
            } else {  
                JOptionPane.showMessageDialog(this, "Contraseña Inválida");  
            }  
        } else {  
            JOptionPane.showMessageDialog(this, "Usuario no válido");  
        }  
    } else if (e.getSource() == botcarga) {  
        CargaUsuario c = new CargaUsuario();  
        this.dispose();  
    }  
}
```

## 2.2 CargaUsuario

En esta clase se creó una ventana, dicha ventana es para realizar la carga masiva, para esto están contenidos en ella; el botón de carga masiva, el espacio de texto de carga masiva, el botón para regresar al login, y adicional un mensaje de éxito al realizar correctamente la carga masiva.

```
public class CargaUsuario extends JFrame implements ActionListener {

    JButton regresar, botcarga;
    TextArea cuad;
    Color rosita = new Color(250, 219, 216 );
    Color moradito = new Color(235, 222, 240);
    Color celestito = new Color(174, 214, 241);


    public CargaUsuario() {
```

A continuación se muestran las acciones realizadas por el botón de carga, por medio de un `ActionPerformed(ActionEvent ae)`

```
if (ae.getSource() == botcarga) {
    try {

        String car = cuad.getText();
        //System.out.println(car);
        if (!car.equals("") || car != null) {
            JsonParser JSONValue = new JsonParser();
            Object objeto = JSONValue.parse(car);
            JsonObject ob = (JsonObject) objeto;
            Object arregl1 = ob.get("Usuarios");
            JSONArray arreglo = (JSONArray) arregl1;
            for (int i = 0; i < arreglo.size(); i++) {
                JsonObject li = arreglo.get(i).getAsJsonObject();
                int Id = li.get("ID").getAsInt();
                String Usuario = li.get("Usuario").getString();
                String Password = li.get("Password").getString();
                int Tipo = li.get("Tipo").getAsInt();
                String Facultad = li.get("Facultad").getString();
                String Carrera = li.get("Carrera").getString();
                Usuario nuevo = new Usuario(Id, Usuario, Password, Facultad, Carrera, Tipo);
                IPCLProyecto1_201902259.crearuser(nuevo);
            }
        } else {
            JOptionPane.showMessageDialog(this, "Cuadro de texto vacío, por favor ingrese el texto");
        }
        IPCLProyecto1_201902259.verusuarios();
        JOptionPane.showMessageDialog(this, "Usuarios Cargados Exitosamente c:");
    } catch (Exception e) {
```





Se condiciona: cuando se tenga texto dentro del cuadro, este texto se almacena en una variable de tipo String, que posteriormente evaluará el tamaño del arreglo y obtendrá cada uno de los atributos del usuario, para luego crear el nuevo usuario llamando al método crearuser. El método crearuser, se encuentra en la clase IPC1Proyecto1\_201902259, donde creamos todas las funciones y métodos orientados a la lógica del programa.

```
//Metodo para guardar los usuarios en el arreglo de usuarios
public static void crearuser(Usuario nuevo) {
    if (contusers < usuarios.length) {
        usuarios[contusers] = nuevo;
        contusers++;
    }
}
```

Si la condición inicial no se cumple, es decir no se presiona el botón de carga, sino se presiona el botón de regresar, se abre nuevamente a la ventana login y se cierra automáticamente la ventana de carga.

Si la segunda condición no se cumple, se pedirá al usuario ingresar los datos al sistema para poder continuar con el proceso.

## 2.3 PantallaPrincipal

La pantalla principal consiste en una ventana que contiene las 4 pestañas de funcionalidad del programa y adicional un botón de actualizar.

```
public class PantallaPrincipal extends JFrame implements ActionListener {
    private JPanel panelpestañas;
    JButton close, actualizar;
    JLabel imagen;
    Color rosadito = new Color(255,183,227);
    Color rosita = new Color(250, 219, 216 );
    Color moradito = new Color(235, 222, 240);
    Color celestito = new Color(214, 234, 248);
    Color verdi = new Color(189, 238, 194);

    public PantallaPrincipal(){

        //Panel de pestañas
        panelpestañas = new JPanel();
        panelpestañas.setBorder(new EmptyBorder(5,25,5,5));
        setContentPane(panelpestañas);
        panelpestañas.setLayout(null);

        //Creación de Pestañas
        JTabbedPane conjpest = new JTabbedPane(JTabbedPane.TOP);
        conjpest.setBounds(10, 11, 960, 450);
        conjpest.setFont(new Font("Century Gothic",Font.PLAIN,12));
        panelpestañas.add(conjpest);
    }
}
```

El botón de actualizar permite visualizar los cambios en las tablas de datos, es importante presionarlo luego de cada cambio, ya que este vuelve a llamar a la Pantalla Principal, en caso contrario no podrá ver los datos obtenidos.

```
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource()==actualizar) {
        PantallaPrincipal p = new PantallaPrincipal();
        this.dispose();
    }
}
```

## 2.4 PestañaLibros

En esta clase se creó una ventana, dicha ventana es para realizar la carga masiva, para esto están contenidos en ella; el botón de carga masiva, el espacio de texto de carga masiva, el botón para regresar al login, y adicional un mensaje de éxito al realizar correctamente la carga masiva.

```
public class PestañaLibros extends JPanel implements ActionListener {

    Color rosadito = new Color(255, 183, 227);
    Color amarelo = new Color(252, 253, 206);
    Color moradito = new Color(235, 222, 240);
    Color celestito = new Color(214, 234, 248);

    JLabel ID, libro, autor, cop, tip;
    JTextField numID, Nlibro, Aut, Copi;
    JButton regis, botcarga;
    JComboBox tipo;
    static JTable tablita;
    static Object[][] datos;

    public PestañaLibros() {
```

### Registro manual de Libros

```
String car = cuad.getText();
//System.out.println(car);
if (!car.equals("") || car != null) {
    JsonParser JSONValue = new JsonParser();
    Object objeto = JSONValue.parse(car);
    JsonObject ob = (JsonObject) objeto;
    Object arreglo1 = ob.get("Libros");
    JSONArray arreglo = (JSONArray) arreglo1;
    for (int i = 0; i < arreglo.size(); i++) {
        JsonObject li = arreglo.get(i).getAsJsonObject();
        String Titulo = li.get("Titulo").getString();
        int Id = li.get("ID").getAsInt();
        String Autor = li.get("Autor").getString();
        int Tipo = li.get("Tipo").getAsInt();
        int Copias = li.get("Copias").getAsInt();
        int Disponibles = li.get("Disponibles").getAsInt();
        int Ocupados = li.get("Ocupados").getAsInt();
        Libro nuevo = new Libro(Id, Titulo, Autor, Tipo, Copias, Disponibles, Ocupados);
        IPC1Proyecto1_201902259.crearlibro(nuevo);
    }
}
```

## 2.5 Pestaña Prestamos

Esta clase contiene espacios de texto para realizar la carga de prestamos manualmente. También contiene 2 botones, prestar libro: registra el préstamo manual, carga masiva: envía a la ventana para realizar la carga masiva con el formato específico. Adicional muestra la tabla de préstamos.

```
public class PestañaPrestamos extends JPanel implements ActionListener {

    JLabel IDusu, librrro, fecha;
    JTextField IDusu, IDlibro, date;
    JButton preslib, cargass;
    static JTable tablita2;
    static Object[][] datos2;
    Color rosadito = new Color(255,183,227);
    Color rosita = new Color(250, 219, 216 );
    Color moradito = new Color(235, 222, 240);
    Color celestito = new Color(214, 234, 248);

    public PestañaPrestamos() {
```

Se implementaron las condiciones necesarias para saber si el libro está ocupado o disponible por medio de la fecha actual, así como las condiciones que permitan ofrecer la cantidad de libros disponibles o bien que no existan libros disponibles.

```
if (ae.getSource() == cargass) {
    cargamassiva();
} else if (ae.getSource() == preslib) {
    int ide = Integer.parseInt(IDusu.getText());
    int idl = Integer.parseInt(IDlibro.getText());
    String fech = date.getText();
    DateTimeFormatter dtf5 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
    String hoy = dtf5.format(LocalDate.now());
    SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
    try {
        Date f1 = dateFormat.parse(fech);
        Date f2 = dateFormat.parse(hoy);
        if (IPCIProyecto1_201902259.verfus(ide) == true && IPCIProyecto1_201902259.verlb(idl) == true) {
            if (IPCIProyecto1_201902259.obtenerlibro(idl).getDisponibles() != 0) {
                if (f1.before(f2)) {
                    Prestamo nuevo = new Prestamo(ide, idl, fech, "Entregado");
                    IPCIProyecto1_201902259.crearprestamo(nuevo);
                    IPCIProyecto1_201902259.obtenerlibro(idl).setDisponibles(IPCIProyecto1_201902259.obtenerlibro(idl).getDisponibles() + 1);
                    IPCIProyecto1_201902259.obtenerlibro(idl).setOcupados(IPCIProyecto1_201902259.obtenerlibro(idl).getOcupados() - 1);
                } else {
                    Prestamo nuevo = new Prestamo(ide, idl, fech, "Ocupado");
                    IPCIProyecto1_201902259.crearprestamo(nuevo);
                    IPCIProyecto1_201902259.obtenerlibro(idl).setDisponibles(IPCIProyecto1_201902259.obtenerlibro(idl).getDisponibles() - 1);
                    IPCIProyecto1_201902259.obtenerlibro(idl).setOcupados(IPCIProyecto1_201902259.obtenerlibro(idl).getOcupados() + 1);
                }
            } else {
                JOptionPane.showMessageDialog(this, "No existen copias disponibles");
            }
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, e.getMessage());
    }
}
```



## 2.6 PestañaReportes

En esta clase se incluye un botón de generar reporte y un ComboBox que permite seleccionar el tipo de Reporte, para posteriormente generarlo en formato PDF y almacenarlo dentro de la carpeta del programa.

```
public class PestañaReportes extends JPanel implements ActionListener {

    JLabel tiprep;
    JButton gen;
    JComboBox tipor;
    static JTable tablita3;
    static Object[][] datos3;
    Color rosadito = new Color(255,183,227);
    Color rosita = new Color(250, 219, 216 );
    Color moradito = new Color(235, 222, 240);
    Color celestito = new Color(214, 234, 248);

    public PestañaReportes() {
```

### Función para el Reporte de Usuario

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

    try {
        nombreReporte = "Usuarios.html";
        reporte = new File(nombreReporte);
        fw = new FileWriter(reporte);
        br = new BufferedWriter(fw);

        cadenaHTML = "<html>"
            + "    <head>"
            + "    <body>"
            + "        <table border = 1>"
            + "            <tr>"
            + "                <td>ID</td>"
            + "                <td>Usuario</td>"
            + "                <td>Password</td>"
            + "                <td>Facultad</td>"
            + "                <td>Carrera</td>"
            + "                <td>Tipo</td>"
            + "            </tr>";
```

```

for (int x = 0; x < IPC1Proyecto1_201902259.contadoru(); x++) {
    if (IPC1Proyecto1_201902259.arreglou()[x] != null) {
        cadenaHTML += "        <tr>"
        + "            <td>" + IPC1Proyecto1_201902259.arreglou()[x].getId() + "</td>"
        + "            <td>" + IPC1Proyecto1_201902259.arreglou()[x].getUser() + "</td>"
        + "            <td>" + IPC1Proyecto1_201902259.arreglou()[x].getPassword() + "</td>"
        + "            <td>" + IPC1Proyecto1_201902259.arreglou()[x].getFacultad() + "</td>"
        + "            <td>" + IPC1Proyecto1_201902259.arreglou()[x].getCarrera() + "</td>"
        + "            <td>" + IPC1Proyecto1_201902259.arreglou()[x].getTipo() + "</td>"
        + "        </tr>";
    }
}

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

br.write(cadenaHTML);

br.close();
fw.close();
DateTimeFormatter dtf3 = DateTimeFormatter.ofPattern("ddMMyyyyHHmmss");
String fecha = "reporteUsuarios_" + dtf3.format(LocalDate.now());
crearPdf(cadenaHTML, fecha);
} catch (IOException ex) {

```

## 2.7 Pestaña Graficos

En esta clase se muestran 3 graficas.

Dos gráficas de pastel:

- Tipo de Usuarios Registrados en el Sistema
- Tipos de Libros Registrados en el Sistema


Una gráfica de Barras:

- Cantidad de Libros Prestados por Fecha

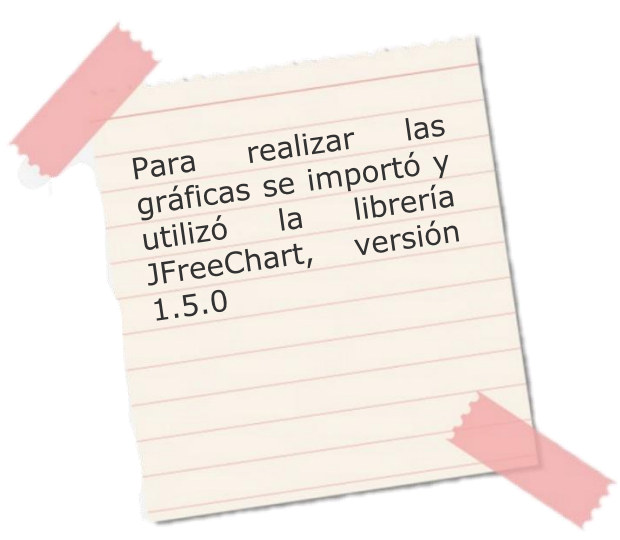
```
public PestañaGraficos() {
    //grafica pastel de usuarios
    DefaultPieDataset datos1 = new DefaultPieDataset();
    datos1.setValue("Usuarios Administradores", IPC1Proyecto1_201902259.ust1());
    datos1.setValue("Usuarios Estudiantes", IPC1Proyecto1_201902259.ust2());
    JFreeChart graficaus = ChartFactory.createPieChart3D("Tipo de Usuarios Registrados en el Sistema", datos1);
    PiePlot3D pie1 = (PiePlot3D) graficaus.getPlot();
    pie1.setForegroundAlpha(0.40f);
    pie1.setInteriorGap(0.05);
    pie1.setBackgroundPaint(pie1);
    ChartPanel p1 = new ChartPanel(graficaus);
    p1.setBounds(20, 50, 290, 290);
    this.add(p1);
}
```

Para la grafica de barras se obtienen las funciones de los meses y se convierten en String sus valores retornados.

```
//grafica de barras de prestamos
DefaultCategoryDataset datos2 = new DefaultCategoryDataset();
datos2.setValue(IPC1Proyecto1_201902259.enero(), "Enero", String.valueOf(IPC1Proyecto1_201902259.enero()));
datos2.setValue(IPC1Proyecto1_201902259.febrero(), "Febrero", String.valueOf(IPC1Proyecto1_201902259.febrero()));
datos2.setValue(IPC1Proyecto1_201902259.marzo(), "Marzo", String.valueOf(IPC1Proyecto1_201902259.marzo()));
datos2.setValue(IPC1Proyecto1_201902259.abril(), "Abril", String.valueOf(IPC1Proyecto1_201902259.abril()));
datos2.setValue(IPC1Proyecto1_201902259.mayo(), "Mayo", String.valueOf(IPC1Proyecto1_201902259.mayo()));
datos2.setValue(IPC1Proyecto1_201902259.junio(), "Junio", String.valueOf(IPC1Proyecto1_201902259.junio()));
datos2.setValue(IPC1Proyecto1_201902259.julio(), "Julio", String.valueOf(IPC1Proyecto1_201902259.julio()));
datos2.setValue(IPC1Proyecto1_201902259.agosto(), "Agosto", String.valueOf(IPC1Proyecto1_201902259.agosto()));
datos2.setValue(IPC1Proyecto1_201902259.septiembre(), "Septiembre", String.valueOf(IPC1Proyecto1_201902259.septiembre()));
datos2.setValue(IPC1Proyecto1_201902259.octubre(), "Octubre", String.valueOf(IPC1Proyecto1_201902259.octubre()));
datos2.setValue(IPC1Proyecto1_201902259.noviembre(), "Noviembre", String.valueOf(IPC1Proyecto1_201902259.noviembre()));
datos2.setValue(IPC1Proyecto1_201902259.diciembre(), "Diciembre", String.valueOf(IPC1Proyecto1_201902259.diciembre()));
JFreeChart barras = ChartFactory.createBarChart("Cantidad de Libros Prestados por Fecha", "mes", "cantidad de libros", datos2, P
barras.setBackgroundPaint(Color.PINK);
barras.getTitle().setPaint(Color.BLACK);
barras.getTitle().setFont(new Font("Century Gothic", Font.PLAIN, 15));
ChartPanel p2 = new ChartPanel(barras);
p2.setBounds(330, 50, 290, 290);
this.add(p2);
}
```



```
// grafica pastel de tipos de libros
DefaultPieDataset datos3 = new DefaultPieDataset();
datos3.setValue("Libro", IPC1Proyecto1_201902259.tl1());
datos3.setValue("Revista", IPC1Proyecto1_201902259.tl2());
datos3.setValue("Libro Electrónico", IPC1Proyecto1_201902259.tl3());
JFreeChart graficalb = ChartFactory.createPieChart3D("Tipos de Libros Registrados en el Sistema", datos3);
PiePlot3D pie2 = (PiePlot3D) graficalb.getPlot();
pie2.setForegroundAlpha(0.40f);
pie2.setInteriorGap(0.05);
pie2.setBackgroundPaint(pie1);
ChartPanel p3 = new ChartPanel(graficalb);
p3.setBounds(640, 50, 290, 290);
this.add(p3);
```



Para realizar las  
gráficas se importó y  
utilizó la librería  
JFreeChart, versión  
1.5.0



## 2.8 CargaLibros

Esta clase se creó con el fin de almacenar los datos obtenidos partiendo desde una nueva ventana de carga masiva. La ventana contiene dos botones, "Carga masiva" y "Regresar".

```
public class CargaLibros extends JFrame implements ActionListener {  
  
    JButton regresar, botcarga;  
    TextArea cuad;  
    Color rosita = new Color(250, 219, 216);  
    Color moradito = new Color(235, 222, 240);  
    Color celestito = new Color(174, 214, 241);  
  
    public CargaLibros() {
```

Código para obtener el texto ingresado en el cuadro de texto y almacenarlo en la tabla de datos del sistema, por medio del botón de carga masiva.

```
@Override  
public void actionPerformed(ActionEvent ae) {  
    if (ae.getSource() == botcarga) {  
        try {  
  
            String car = cuad.getText();  
            System.out.println(car);  
            if (!car.equals("") || car != null) {  
                JsonParser JSONValue = new JsonParser();  
                Object objeto = JSONValue.parse(car);  
                JsonObject ob = (JsonObject) objeto;  
                Object arreglo1 = ob.get("Libros");  
                JsonArray arreglo = (JsonArray) arreglo1;  
                for (int i = 0; i < arreglo.size(); i++) {  
                    JsonObject li = arreglo.get(i).getAsJsonObject();  
                    String Titulo = li.get("Titulo").getAsString();  
                    int Id = li.get("ID").getAsInt();  
                    String Autor = li.get("Autor").getAsString();  
                    int Tipo = li.get("Tipo").getAsInt();  
                    int Copias = li.get("Copias").getAsInt();  
                    int Disponibles = li.get("Disponibles").getAsInt();  
                    int Ocupados = li.get("Ocupados").getAsInt();  
                    Libro nuevo = new Libro(Id, Titulo, Autor, Tipo, Copias, Disponibles, Ocupados);  
                    IPC1Proyecto1_201902259.crearlibro(nuevo);  
                }  
            }  
        }  
    }  
}
```

Adicional a eso se evitaron errores con un try catch.

## 2.9 CargaPrestamos

En esta clase se creó con el fin de almacenar los datos de préstamos, obtenidos partiendo desde una nueva ventana de carga masiva. La ventana contiene dos botones, "Carga masiva" y "Regresar".

```
public class CargaPrestamos extends JFrame implements ActionListener {

    JButton regresar, botcarga;
    TextArea cuad;
    Color rosita = new Color(250, 219, 216);
    Color moradito = new Color(235, 222, 240);
    Color celestito = new Color(174, 214, 241);

    public CargaPrestamos() {
```

En el siguiente fragmento del código se obtienen los datos de la carga masiva, muy similar a la anterior, con la variante de los libros entregados y ocupados, para esto se realizó una comparación de fechas que indica si el libro actualmente está ocupado o disponible.

```
if (!car.equals("") || car != null) {
    JsonParser JSONValue = new JsonParser();
    Object objeto = JSONValue.parse(car);

    JsonObject ob = (JsonObject) objeto;
    Object arreglo1 = ob.get("Prestamos");
    JSONArray arreglo = (JSONArray) arreglo1;
    for (int i = 0; i < arreglo.size(); i++) {
        JsonObject li = arreglo.get(i).getAsJsonObject();
        int ide = li.get("IDUsuario").getAsInt();
        String Fecha = li.get("FechaEntrega").getString();
        int idel = li.get("IDLibro").getAsInt();
        DateTimeFormatter dtf5 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        String hoy = dtf5.format(LocalDate.now());
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
        try {
            Date f1 = dateFormat.parse(Fecha);
            Date f2 = dateFormat.parse(hoy);
            if (IPC1Proyecto1_201902259.verfus(ide) == true && IPC1Proyecto1_201902259.verlb(idel) == true) {
                if (f1.before(f2)) {
                    Prestamo nuevo = new Prestamo(ide, idel, Fecha, "Entregado");
                    IPC1Proyecto1_201902259.crearprestamo(nuevo);
                } else {
                    Prestamo nuevo = new Prestamo(ide, idel, Fecha, "Ocupado");
                    IPC1Proyecto1_201902259.crearprestamo(nuevo);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Librerías importadas:  
ParseException  
SimpleDateFormat  
time.LocalDateTime  
time.format.DateTime  
Formatter  
Date

Librerías importadas

### 3. Paquete

ipc1.proyecto.pkg1\_201902259

Dentro de este paquete se encuentra una única clase, en la cual se almacena, en su mayoría, el código lógico de la aplicación.

#### PantallaPrincipal

Dentro de la pantalla principal podemos encontrar funciones y métodos utilizados para resolver problemas de manera eficaz y ordenada.

Arreglos de libros, prestamos, usuarios y reportes con su respectivo tamaño predefinido arreglos globales

```
//Creación del arreglo de libros
static Libro[] libros = new Libro[100];
static int contlibros = 0;
//Creación de arreglo de usuarios
static Usuario[] usuarios = new Usuario[50];
static int contusers = 0;
//Creación de arreglo de prestamos
static Prestamo[] prestamos = new Prestamo[200];
static int contpres = 0;
//Creación de arreglo de reportes
static Reporte[] reportes = new Reporte[100];
static int contrep = 0;
```

Método para guardar el libro en el arreglo de libros, de la misma manera se procede para usuarios, prestamos, reportes.

```
public static void crearlibro(Libro nuevo) {
    if (contlibros < libros.length) {
        libros[contlibros] = nuevo;
        contlibros++;
    }
}
```



Función para obtener el objeto Libro por medio del ID

```
public static Libro obtenerlibro(int id) {
    for (int i = 0; i < contlibros; i++) {
        if (id == libros[i].getId()) {
            return libros[i];
        }
    }
    return null;
}
```

Función para retornar un objeto de arreglo y utilizarlo en las tablas de las pestañas.

```
public static Object[][] tablitalibros() {
    Object[][] contenido = new Object[contlibros][7];
    for (int i = 0; i < contlibros; i++) {
        contenido[i][0] = libros[i].getId();
        contenido[i][1] = libros[i].getTitulo();
        contenido[i][2] = libros[i].getAutor();
        switch (libros[i].getTipo()) {
            case 1:
                contenido[i][3] = "Libro";
                break;
            case 2:
                contenido[i][3] = "Revista";
                break;
            case 3:
                contenido[i][3] = "Libro Electronico";
                break;
        }
        contenido[i][4] = libros[i].getCopias();
        contenido[i][5] = libros[i].getDisponibles();
        contenido[i][6] = libros[i].getOcupados();
    }
    return contenido;
}
```

Función para validar que exista el ID

```
public static boolean validid(int id) {
    for (int i = 0; i < contlibros; i++) {
        if (id == libros[i].getId()) {
            return true;
        }
    }
    return false;
}
```



Método para guardar los usuarios en un arreglo de usuarios.

```
public static void crearuser(Usuario nuevo) {  
    if (contusers < usuarios.length) {  
        usuarios[contusers] = nuevo;  
        contusers++;  
    }  
}
```

Función de validación para confirmar la existencia del usuario

```
public static boolean entrusuario(String us) {  
    for (int i = 0; i < contusers; i++) {  
        if (us.equals(usuarios[i].getUser())) {  
            return true;  
        }  
    }  
    return false;  
}
```

Función para retornar los libros prestados en ese mes

```
public static int junio() {  
    int contador = 0;  
    for (int i = 0; i < contpres; i++) {  
        DateFormat f = new SimpleDateFormat("dd/MM/yyyy");  
        try {  
            Date fecha = f.parse(prestamos[i].getFecha());  
            Calendar c = Calendar.getInstance();  
            c.setTime(fecha);  
            if (String.valueOf(c.get(Calendar.MONTH)).equals("5")) {  
                contador++;  
            }  
        } catch (Exception e) {}  
    }  
    return contador;  
}
```



## Conclusiones

Para finalizar se puede afirmar que el programa realizado ha beneficiado grandemente el conocimiento de los estudiantes en cuanto al tema de Programación Orientada a Objetos, así como ha estimulado el uso de la lógica y razón al crear y permitir interactuar con la interfaz gráfica que ofrece Java por medio de la librería AWT y SWING, las cuales permitieron la realización y funcionamiento del presente proyecto.