BIBLIOTECA FACULTAD DE INGENIERÍA USAC

MANUAL TÉCNICO

INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 1 C

DANIEL ESTUARDO CUQUE RUÍZ

202112145

ÍNDICE

METODO CREAR *3*
Clase Prestamo 3
Clase Usuario 3
Clase Bibliografias 4
Carga Individual 4
Carga Masiva 5
METODO ACTUALIZAR *6*
Clase Usuario 6
Clase Bibliografia 6
METODO ELIMINAR *7*
Clase Bibliografia 7
Clase Prestamos 7
Eliminar Usuario 8
Botón de eliminar 8
METODO VER *9*
Clase Prestamos 9
Cargar Datos a un JTable 10
REPORTES *11*
Reporte de Usuarios 11
Reporte de Bibliografias 12
Reporte de Prestamos 13
DIAGRAMAS UML *14*
Clase PanelAdministrador 14
Clase PanelNormal 14
Clase Alertas 15
Case Usuarios 15
Clase Bibliografías 16
Clase Prestamos 17

METODO CREAR

Se aplicó la misma lógica para la creación de objetos en las 3 clases principales, se recorre el arreglo o matriz en donde se almacenarán los objetos. Posteriormente se verifica que la posición en *i* sea *null*, lo que significa que no existe otro elemento en esa posición. Luego se indica que que poscion de arreglo[i] sea igual al objeto nuevo y se rompe el ciclo para que no siga creando objetos nuevos.

Clase Prestamo

```
public static void crearPrestamo(Prestamos nuevoPrestamo){
    for (int i = 0; i < arregloPrestamos.length; i++) {
        if(arregloPrestamos[i] == null){
            arregloPrestamos[i] = nuevoPrestamo;
            cantidadPrestamos++;
            break;
        }
    }
}
```

Clase Usuario

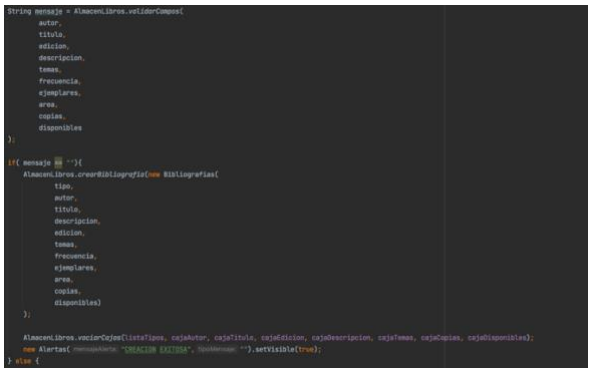
```
public String crearUsuario() {
    if (!usuarioExistente()) {
        for (int i = 0; i < matrizUsuarios.length; i++) {
            if (matrizUsuarios[i][0] == null) {
                matrizUsuarios[i][0] = getID();
                matrizUsuarios[i][1] = getNombres();
                matrizUsuarios[i][2] = getApellidos();
                matrizUsuarios[i][3] = getUser();
                matrizUsuarios[i][4] = getRol();
                matrizUsuarios[i][5] = getPassword();
                break;
            }
        }
        return "<html><p style=\"text-align:center\">¡USUARIO CREADO" +
                "<p style=\"text-align:center\">CORRECTAMENTE!<p></p><html>";
    } else if (usuarioExistente()) {
        return "¡Error!, el usuario ya existe";
    } else {
        return "Hubo un error";
    }
}
```

Clase Bibliografias

```
public static void crearBibliografia(Bibliografias nuevoLibro){
    for (int i = 0; i < arregloBibliografias.length; i++) {
        if(arregloBibliografias[i] == null){
            arregloBibliografias[i] = nuevoLibro;
            cantidadBibliografias++;
            return;
        }
    }
}
```

Carga Individual

Para la carga indivual, se necesita validar que los campos cumplen con los requisitos del formato de ingreso, se obtiene el texto dentro de la caja de texto y se usa el método de *crearBibliografia* junto con los atributos de la clase *Bibliografia*.



Carga Masiva

Para la carga individual se recibe el texto que está contenido dentro de un JTextArea, y cuando identifique un salto de línea (\n) entonces separara esa línea de texto de las demás y la añadirá en un arreglo tipo String, luego ese arreglo recorre con las líneas de texto y valida que la longitud el arreglo sea de 11 (que es la cantidad de campos que necesita la clase), y también valida que el tipo de libro esté entre 0 a 2. Posteriormente llama al método de *nuevoLibro* y así sucesivamente hasta que recorra todo el arreglo.

```
public void cargarLibros(String texto){
    String [] lineasTexto = texto.split( regex: "\n");

    for (String linea :
            lineasTexto) {
        String[] atributos = linea.split( regex: "|");
        if(atributos.length == 11 && atributos[0].matches( regex: "[0-2]") ){
            switch (atributos[0]) {
                case "0":
                    atributos[0] = "LIBRO";
                    atributos[6] = "---";
                    atributos[7] = "0";
                    atributos[8] = "---";
                    break;
                case "1":
                    atributos[0] = "REVISTA";
                    atributos[7] = atributos[7].trim();
                    atributos[8] = "---";
                    break;
                case "2":
                    atributos[0] = "TESIS";
                    atributos[6] = "---";
                    atributos[7] = "0";
                    break;
            }
            Bibliografias nuevoLibro = new Bibliografias(
                    atributos[0], //Tipo
                    atributos[1], //Autor
                    atributos[2], //Título
                    atributos[3], //Descripción
                    atributos[4].trim(), //Edición
                    atributos[5], //Temas
```

METODO ACTUALIZAR

Para el método actualizar se requiere que los parámetros del método sean los campos que se requieren actualizar dentro de la clase.

Clase Usuario

En el caso de la actualización de Usuarios, solicita el índice de la posición del Usuario dentro de la matriz, luego que encuentra el índice, se re-asigna el valor que se quiere actualizar.

```
public String actualizarUsuario(int indiceActualizar,
                                 String id,
                                 String nombre,
                                 String apellido,
                                 String user,
                                 String rol,
                                 String password) {
    matrizUsuarios[indiceActualizar][0] = id;
    matrizUsuarios[indiceActualizar][1] = nombre;
    matrizUsuarios[indiceActualizar][2] = apellido;
    matrizUsuarios[indiceActualizar][3] = user;
    matrizUsuarios[indiceActualizar][4] = rol;
    matrizUsuarios[indiceActualizar][5] = password;
    return "ACTUALIZACIÓN EXITOSA";
}
```

Clase Bibliografia

En el caso de la bibliografía, el método necesita como parámetros los campos que se desean actualizar, y va recorriendo toda la matriz, hasta que el título coincide con el libro, y reasigna los valores de los atributos de las bibliografías.

```
public static String actualizarLibro(String tipo, String autor, String titulo,
                                      String descripcion, String edicion, String temas, String frecuencia,
                                      String ejemplares, String area, String copias, String disponibles){

    String mensajeAlerta = "";
    for (int i = 0; i < (arregloBibliografias.length - 1); i++) {
        if(arregloBibliografias[i] != null){
            if(arregloBibliografias[i].getTitulo().equals(titulo)){
                arregloBibliografias[i].setTipo(tipo);
                arregloBibliografias[i].setAutor(autor);
                arregloBibliografias[i].setDescripcion(descripcion);
                //Propiedades de Edicion
                arregloBibliografias[i].setStrEdicion(edicion);
                arregloBibliografias[i].setEdicion(Integer.parseInt(edicion));
                //Propiedades de Temas
                arregloBibliografias[i].setTemas(temas.trim().split( regex: ","));
                arregloBibliografias[i].setStrTemas(temas);
                arregloBibliografias[i].setTemasConcatenados(temas);
                arregloBibliografias[i].setFrecuenciaActual(frecuencia);
                //Propiedades de Ejemplares
                arregloBibliografias[i].setStrEjemplares(ejemplares);
                arregloBibliografias[i].setEjemplares(Integer.parseInt(ejemplares));
                arregloBibliografias[i].setArea(area);
                //Propiedades de Copias
                arregloBibliografias[i].setStrCopias(copias);
                arregloBibliografias[i].setCopias(Integer.parseInt(copias));
                //Propiedades de Disponibles
                arregloBibliografias[i].setStrDisponibles(disponibles);
                arregloBibliografias[i].setDisponibles(Integer.parseInt(disponibles));
                mensajeAlerta = "ACTUALIZACION EXITOSA";
                break;
```

METODO ELIMINAR

Para el método de eliminar, se necesita como parámetro un identificador único para buscarlo dentro del arreglo o matriz. Luego de encontrar el identificador, vuelve como *null* el índice de a matriz en donde estaba contenido el objeto.

Clase Bibliografia

```
public static void eliminarLibro(String titulo){
    for (int i = 0; i < (arregloBibliografias.length - 1); i++) {
        if(arregloBibliografias[i] != null){
            if(arregloBibliografias[i].getTitulo().equals(titulo)){
                arregloBibliografias[i] = null;
                cantidadBibliografias--;
                break;
            }
        }
    }

    for (int i = 0; i < (arregloBibliografias.length - 1); i++) {
        if(arregloBibliografias[i] == null && arregloBibliografias[i+1] != null){
            arregloBibliografias[i] = arregloBibliografias[i+1];
            arregloBibliografias[i+1] = null;
        }
    }
}
```

Clase Prestamos

Para devolver los préstamos, se necesita la hora, el id del usuario y el título del libro, para buscarlo dentro del arreglo, ya que un Usuario puede obtener muchos préstamos, por lo que la hora definirá cual es el préstamo que se requiere eliminar.

```
public static void devolverLibro(String idUsuario,String tituloLibro,String hora){
    for (int i = 0; i < arregloPrestamos.length; i++) {
        if(arregloPrestamos[i] != null){
            if(idUsuario.equals(arregloPrestamos[i].getIdUsuarioPrestamo())
                    && tituloLibro.equals(arregloPrestamos[i].getTituloPrestamo())
                    && hora.equals(arregloPrestamos[i].getHora())
            ){
                arregloPrestamos[i] = null;
            }
        }
    }

    for (int i = 0; i < (arregloPrestamos.length - 1); i++) {
        if (arregloPrestamos[i] == null && arregloPrestamos[i+1] != null) {
            arregloPrestamos[i] = arregloPrestamos[i+1];
            arregloPrestamos[i+1] = null;
        }
    }
}
```

Eliminar Usuario

```java
public String eliminarUsuario(int indiceEliminar) {
    //Asignos los valores como null dentro de la fila que vamos a eliminar
    for (int j = 0; j < 6; j++) {
        matrizUsuarios[indiceEliminar][0] = "";
    }

    //Desplazamos los usuarios una casilla hacia arriba cuando un usuario sea eliminado
    for (int i = 0; i < (matrizUsuarios.length - 1); i++) {
        if(Objects.equals(matrizUsuarios[i][0], b: "") && !Objects.equals(matrizUsuarios[i + 1][0], b: "")){
            for (int j = 0; j < matrizUsuarios[i].length; j++) {
                matrizUsuarios[i][j] = matrizUsuarios[i + 1][j];
                matrizUsuarios[i + 1][j] = "";
            }
        }
    }
    return "ELIMINACIÓN EXITOSA";
}
```

Botón de eliminar

Para poder eliminar un elemento que está contenido dentro de un JTable se necesita obtener el índice de la fila, para ello se utiliza el método *getSelectedRow()* que devuelve el índice de la fila, posteriormente se hace una verificación de que el usuario sí haya seleccionado una fila; si la verificación es true, entonces obtiene el valor de la celda y se hace el llamado al método de eliminar

```java
private void btnEliminar(ActionEvent e){
    String titulo;
    int fila = tablaLibros.getSelectedRow();
    if(!(fila == - 1)){
        titulo = (String)tablaLibros.getValueAt(fila, column: 2);
        modeloTabla.removeRow(fila);
        AlmacenLibros.eliminarLibro(titulo);
        new Alertas( mensajeAlerta: "SE HA ELIMINADO CON ÉXITO", tipoMensaje: "").setVisible(true);
    } else{
        new Alertas( mensajeAlerta: "DEBE SELECCIONAR UNA FILA", tipoMensaje: "ERROR").setVisible(true);
    }
}
```

METODO VER

Para mostrar todos los datos dentro del arreglo en debemos de recorrerlo con un ciclo for, luego el método devuelve una matriz con las posiciones llenas, sin tomar en cuenta los valores que son *null*.

Clase Prestamos

Para mostrar los reportes, sí se necesita un parámetro, este será el ID del usuario que esté asociado al préstamo

```
public static String[][] mostrarPrestamos(String idUsuario){
    String [][] datos = new String[cantidadPrestamos][3];
    int pos = 0;
    for (Prestamos prestamo :
            arregloPrestamos) {
        if(prestamo != null){
            if(prestamo.getIdUsuarioPrestamo().equals(idUsuario)){
                String [] fila = {
                        prestamo.getTituloPrestamo(),
                        prestamo.getTipoPrestamo(),
                        prestamo.getHora()
                };
                datos[pos] = fila;
                pos++;
            }
        }
    }
    return datos;
}
```

```
public static String[][] obtenerLibros(){
    String[][] datos = new String[cantidadBibliografias][11];
    int pos = 0;
    for(Bibliografias bibliografias: arregloBibliografias){
        if(bibliografias != null){
            String [] fila = {
                    bibliografias.getTipo(),
                    bibliografias.getAutor(),
                    bibliografias.getTitulo(),
                    bibliografias.getDescripcion(),
                    bibliografias.getStrEdicion(),
                    bibliografias.getTemasConcatenados(),
                    bibliografias.getFrecuenciaActual(),
                    bibliografias.getStrEjemplares(),
                    bibliografias.getArea(),
                    bibliografias.getStrCopias(),
                    bibliografias.getStrDisponibles(),
            };
            datos[pos] = fila;
            pos ++;
        }
    }
    return datos;
}
```

Cargar Datos a un JTable

Para que un JTable muestre los datos necesita una matriz, es por ello que el parámetro para cargar la tabla recibe una matriz. Luego se crea un objeto de la clase DefaultTableModel, y como parámetros lleva el encabezado de la tabla, y la matriz con los datos.

```
private void cargarTabla(String[][] datos) {
    modeloTabla = new DefaultTableModel(datos, Prestamos.cabecera());
    tablaPrestamos = new JTable(modeloTabla);
    tablaPrestamos.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    tablaPrestamos.setAutoscrolls(true);
    scrollTabla.setViewportView(tablaPrestamos);
    scrollTabla.setBorder(grafica.margenPanelesAdmin);
    contenedorTabla.add(scrollTabla, BorderLayout.CENTER);
}
```

REPORTES

Para crear un reporte se necesita de un header y un footer en lenguaje de marcado HTML, esta será de tipo String. Con este método se devuelve un encabezado con el nombre del reporte a realizar, y también el footer del préstamo.

Reporte de Usuarios

Se utiliza el método para contar cuantas veces el id de algún usuario se encuentra dentro del arreglo de préstamos, y así saber cuántos préstamos ha realizado dicho usuario.

```
private String buscarCantidadPrestada(String idUsuario){
    String cantidadPresta = "";
    for (int i = 0; i < datosPrestamos.length; i++) {
        if(datosPrestamos[i][0] != null){
            if(datosPrestamos[i][0].equals(idUsuario)){
                cantidadPresta = datosPrestamos[i][1];
                break;
            }else{
                cantidadPresta = "0";
            }
        }
    }
    return cantidadPresta;
}
```

Reporte de Bibliografias

Se crea el método para verificar si un tema ya existe, si sí existe, retorna true, de lo contrario, retorna false.

```
//Verifica si el tema ya existe
private static boolean temaExiste(String tema){
    boolean validacion = false;
    for (String s : temasSinRepeticion) {
        if (s != null) {
            if (s.equals(tema)) {
                validacion = true;
                break;
            }
        }
    }
    return validacion;
}
```

Con el método de contar repetición verificamos cuantas veces el tema se encuentra dentro del arreglo de Bibliografias, si encuentra el tema, el contador suma, y cuando termine de recorrer el arreglo, retorna el valor del contador.

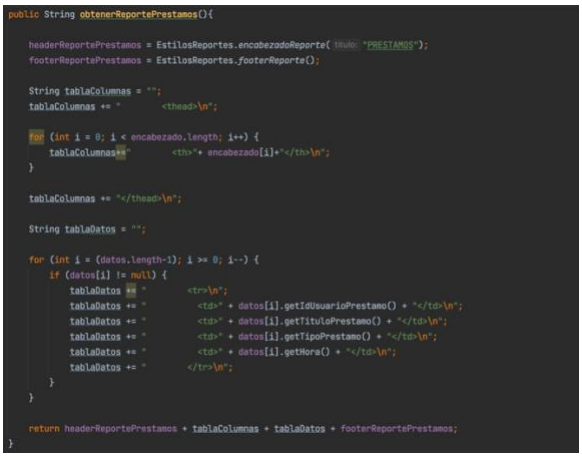
```
public static int contarRepeticionTemas(String tema){
    int contador = 0;
    for (Bibliografias libro :
            arregloBibliografias) {
        if(libro != null){
            for (String temaLibro :
                    libro.getTemas()) {
                if (temaLibro.equals(tema)) {
                    contador++;
                }
            }
        }
    }
    return contador;
}
```

Se crea un arreglo alterno que solo contenga una sola vez el tema, y se utiliza el método de temaExiste() para que devuelva un valor tipo boolean, si el tema no está dentro del arreglo alterno, entonces lo añade.

```
public static void buscarTemasRelacionados(){
    for (Bibliografias libro : arregloBibliografias) {
        if (libro != null) {
            for (String temasLibro :
                    libro.getTemas()) {
                if(!temaExiste(temasLibro)){
                    temasSinRepeticion[contadorTemas] = temasLibro;
                    contadorTemas++;
                }
            }
        }
    }
}
```

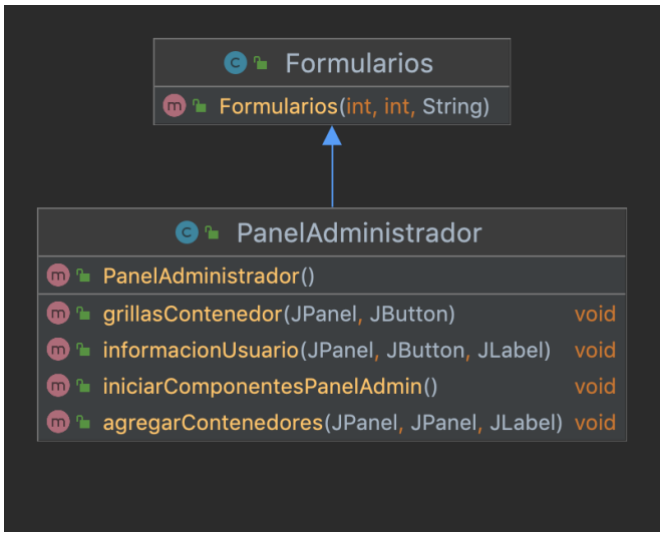
Reporte de Prestamos

Se crea el método de obtenerReportePrestamos(), para ello se necesita el header y el footer del texto en HTML, para ello se hace uso del método encabezadoReporte() y footerReporte(), posteriormente se recorre el encabezado, este servirá para la tabla, luego se recorre el arreglo en reversa para que muestre los préstamos del más reciente hasta el más antiguo, finalmente se concatena todo y se retorna para poder ser mostrado en un JTextArea.

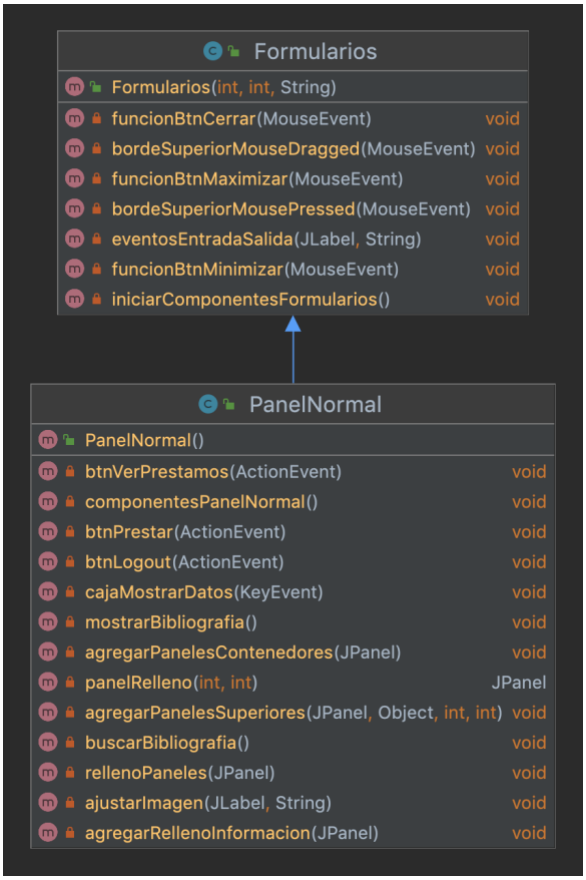


DIAGRAMAS UML

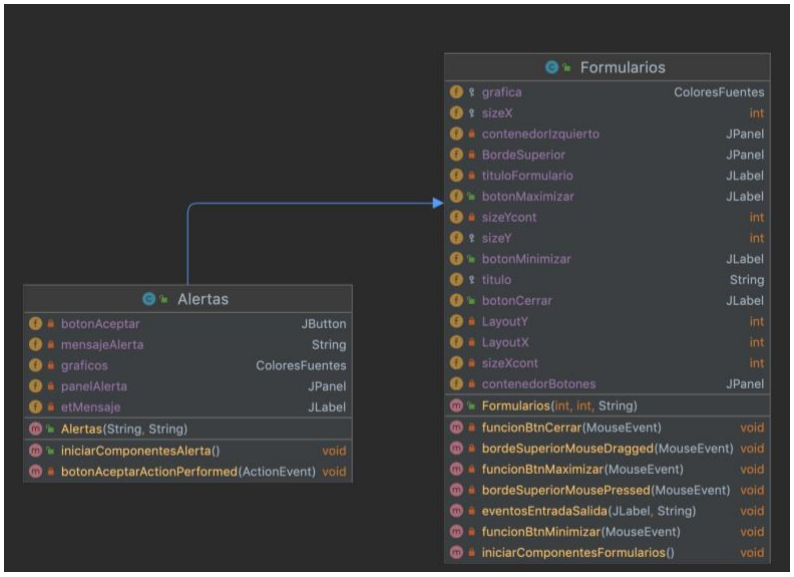
Clase PanelAdministrador



Clase PanelNormal

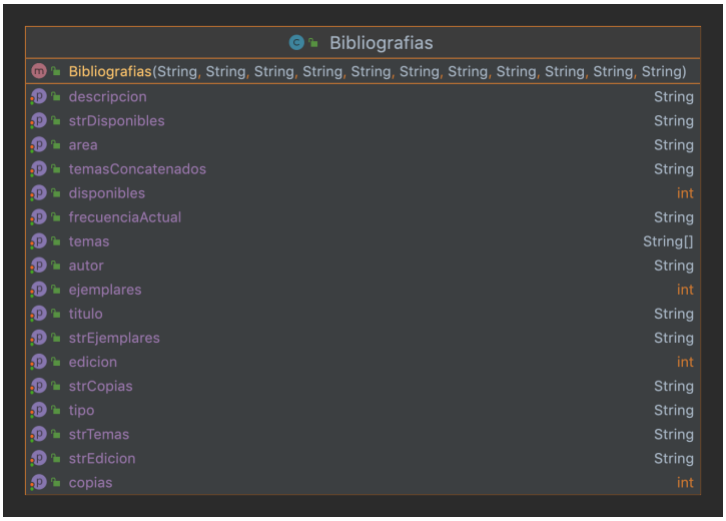


Clase Alertas



Case Usuarios

Clase Bibliografías



Clase Prestamos