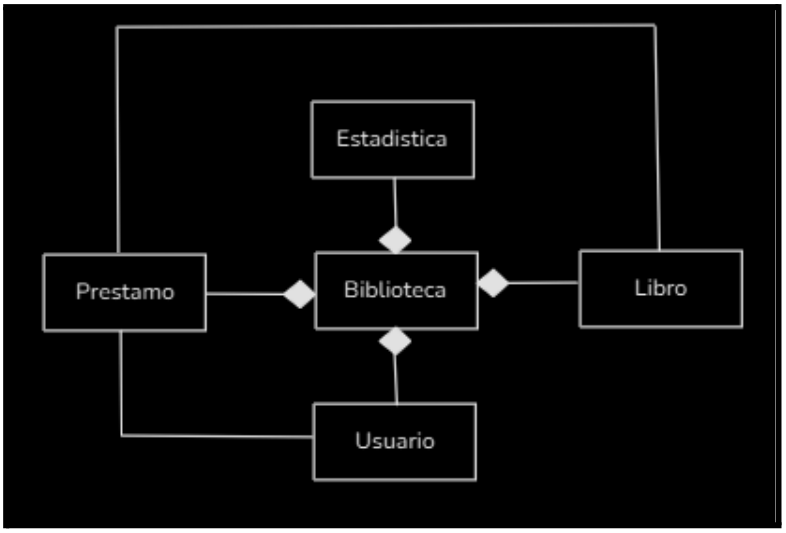
Biblioteca manual técnico

-**Por Daniel Álvarez Morales.**

ÍNDICE

1.-Esquema/UML

-El programa en general se basa en diversas clases que interactúan entre sí, aunque la mayoría de funciones y métodos las lleva a cabo la clase biblioteca.



(UML implementación de las clases)

-Serían todas las clases una composición para biblioteca debido a que si borras la biblioteca borras todo los objetos relacionados a él. Y los préstamos representan una relación de un libro y usuario por lo tanto están asociados con líneas simples.

2.-El tratamiento de null exception de los arrays.

-En vez de que se itere con lenght hasta el límite del array, solamente se itera hasta el número de elementos que tenga, así se evita acceder a atributos o métodos para objetos vacíos que no existen,

```
for (int i = 0; i < numLibros && noExiste; i++) {
    if (arrayLibros[i].getIsbn() == (isbn)) {
        noExiste = false;

    }
}
```

(Recorrido de iteraciones de libros)

3.-Estadísticas de la biblioteca

-Las estadísticas con la ayuda de la clase estadística se hace un contador de préstamos activos y otro de préstamos totales. Que van incrementando o reduciendo según las devoluciones o préstamos exitosos.

```
private int nPrestamosActivos;
private int nPrestamosTotales;
```

(Atributos de la clase estadística)

- Y se agrega a la clase usuario y libro contadores de libros prestados para el libro **(a)** y libro contadores de préstamos activos para usuario **(b)** .

a)

```
private int contadorLibrosPrestados;
```

b)

```
private int contadoPrestamosActivos;
```

4.-Las implementaciones entre usuario y libro (Gestion prestamos).

-La relación entre el usuario y el libro se da a través de un un objeto préstamo que es una clase en la que sus atributos son un libro y un usuario.

```
public Prestamo(Usuario usuario, Libro libro) {
    this.usuario = usuario;
    this.libro = libro;
}
```

(Constructo de la clase Préstamo)

-Por lo tanto cada vez que un usuario realiza un préstamo se crea el objeto préstamo con el usuario y el libro en cuestión almacenandose en un array de tipos préstamo.

```
private Prestamo[] arrayPrestamos;
```

(El atributo de la biblioteca que almacenará préstamos)

5.-El formato de los arrays.

-El formato de los arrayss son estático más no dinámico quizás en una futura versión del proyecto se pueda escalar con *copy of* o con *arraylists*. Tienen 15 posiciones de momento.

6.-La creación de objetos por el programa.

-La creación de usuarios o libros se lleva a cabo con scanners introducidos por el usuario administrador que está iniciado en la sesión en ese momento.

```
System.out.println("Introduce el nombre de usuario a registrar");
sc.nextLine(); // limpiar el buffer
String nombreUsuario = sc.nextLine();
System.out.println("Introduce el dni de usuario a registrar");
String dni = sc.nextLine();
System.out.println("Introduce la contraseña de usuario a registrar");
String contrasenia = sc.nextLine();
```

(Scanners para registrar nuevo usuario)

-Luego, se pasan por los parámetros del método y lo implementamos con los setters de los atributos.

```
public void registrarUsuario(String nombreUsuario, String dni, String contrasenia, boolean admin) {
    Usuario usuario = new Usuario(null, null, null, false);
    usuario.setNombreUsuario(nombreUsuario);
    usuario.setContrasenia(contrasenia);
    usuario.setDni(dni);
    usuario.setAdmin(admin);
    agregarUsuario(usuario);

}
```

(Setters para registrar nuevo usuario)

6.1.-La creación del isbn.

-Para que cada libro sea único y no haya repetidos se implementa el ISBN, una variable tipo int que se incrementa cada vez que el programa crea un libro.

```
private int isbn;
```

(el atributo isbn)

```java
public Libro(String titulo, String autor, String categoria) {
    this.titulo = titulo;
    this.autor = autor;
    this.categoria = categoria;
    this.isbn = contadorIsbn;
    contadorIsbn++;
    this.contadorLibrosPrestados = 0;

}
```

(El constructor de libro)

7.-Función prestar libro.

-El usuario que está iniciado sesión debe introducir el isbn del libro que quiere.

```java
public Prestamo tomarLibroPrestado(Usuario usuario, int isbn) {
```

(Parámetros a introducir de la función)

-Y con una función independiente se chequea si el libro existe.

```java
boolean noExiste = checkeadorArraysLibros(isbn);
```

(La variable que almacenará el booleano de la función chequeadora)

-En caso de que no exista el libro imprime “no existe” y devuelve null y en caso de que si existiese:

1.- Se crea un libro vacío que servirá para recoger el libro con el isbn del parámetro.

2, 3 y 4.- Recorremos el array con un booleano como break y cuando detecte el libro de la biblioteca lo coja la variable “libroDetectado”.

5.- El array de libro de la biblioteca que contenía el libro que se buscaba le asignamos un null, ya que ya no estará en la biblioteca sino que lo tendrá el usuario.

6 y 7.- Este bucle es importante debido a que ordena los elementos de derecha a izquierda desde el elemento que se ha detectado es decir borrado es decir convertido en null hasta el número de libros actual que se tienen, evitando así exitosamente nulls exceptions.

8.- Es un disminuidor del número de libros, es similar a “*numlibros–;*”.

1 y 2.- Estas líneas no tienen que ver con el método en sí, sino que son para estadísticas.

9.- Creamos el nuevo préstamo con el usuario pasado por el parámetro y el libro detectado.

10.- Devolvemos el nuevo préstamo.

```
Libro libroDetectado = null; 1

boolean encontrado = true;

for (int i = 0; i < getNumLibros() && encontrado; i++) { 2
    if (arrayLibros[i].getIsbn() == (isbn)) { 3
        libroDetectado = arrayLibros[i]; 4
        encontrado = false;
        arrayLibros[i] = null; 5

        for (int j = i; j < numLibros - 1; j++) { 6
            arrayLibros[j] = arrayLibros[j + 1]; 7
        }

        setNumLibros(getNumLibros() - 1); 8
    }
}

libroDetectado.getContadorLibrosPrestados(); 1
libroDetectado.setContadorLibrosPrestados(libroDetectado.getContadorLibrosPrestados() + 1);2
Prestamo prestamos = new Prestamo(usuario, libroDetectado); 9
return prestamos; 10
```

(Imagen de la función “realizar préstamo” en la clase biblioteca)

-En el main si detecta que es nulo porque no lo encuentra imprime “Error” y si no, lo agrega al array de préstamos de la biblioteca.

```
System.out.println("Has elegido realizar prestamo de libro ");
System.out.println("Introduce el isbn del libro que quieres tomar prestado ");
sc.nextLine(); // limpiar el buffer
int isbnPrestamo = sc.nextInt();
Prestamo a = new Prestamo(null, null);
a = biblioteca.tomarLibroPrestado(sesionUsuario, isbnPrestamo);

if (a == null) {
    System.out.println("Error");
} else {

    arrayPrestamos = biblioteca.getArrayPrestamos();
    int numPrestamos = biblioteca.getNumPrestamos();

    if (numPrestamos < arrayPrestamos.length) {
        arrayPrestamos[numPrestamos] = a;
        biblioteca.setNumPrestamos(numPrestamos + 1);
    } else {
        System.out.println("Error: No hay más espacio para registrar préstamos.");
    }

    biblioteca.setArrayPrestamos(arrayPrestamos);

}
```

(Imagen de la función "realizar préstamo" en la clase App)

8.-Función devolver libro.

-El procedimiento se desarrolla de forma parecida a tomar un libro prestado.

```
public void devolverLibroTomado(Usuario usuario, int isbn) {
    Libro libroDetectado = null;
```

(Imagen de la función "devolver libros prestados" en la clase biblioteca)

-Sin embargo, en esta función hace falta además de verificar los libros del array préstamo también tiene que verificar los libros que ya están prestados de los otros usuario es decir, para que no devuelva a la biblioteca un libro tomado por otro usuario.

```java
boolean noExiste = true;

for (int i = 0; i < this.numPrestamos; i++) {
    if (this.arrayPrestamos[i].getLibro().getIsbn() == isbn) {
        noExiste = false;

    }

    if (!arrayPrestamos[i].getUsuario().getNombreUsuario().equals(usuario.getNombreUsuario())) {
        noExiste = true;

    }

}
```

(Imagen del código que verifica los libros tomados)

1.- “Cancelación de la devolución”, esta variable sirve después por si está lleno la biblioteca y se quiere devolver un libro.

Es decir es una copia de libro a devolver si la transacción no se completa y vuelva al usuario en vez de que se pierda el libro.

1.- Se sustituye el último préstamo por el préstamo expirado es decir el a eliminar.

2.- El último préstamo se le asigna un null ya que sino quedaría duplicado y esto sirve para ordenar.

3.- Y finalmente se disminuye el número de préstamos ya que ha finalizado un préstamo.

```
Prestamo cancelacionDevolucion = null; 1

System.out.println("Devolución realizado con exito");
estadisticas.setnPrestamosActivos(estadisticas.getnPrestamosActivos() - 1);
usuario.setContadoPrestamosActivos(usuario.getContadoPrestamosActivos() - 1);

boolean encontrado = true;
for (int i = 0; i < numPrestamos && encontrado; i++) {
    if (arrayPrestamos[i].getLibro().getIsbn() == isbn) {

        libroDetectado = arrayPrestamos[i].getLibro();
        cancelacionDevolucion = arrayPrestamos[i];

        encontrado = false;

        arrayPrestamos[i] = arrayPrestamos[numPrestamos - 1]; 1

        arrayPrestamos[numPrestamos - 1] = null; 2

        setNumPrestamos(getNumPrestamos() - 1); 3
    }
}
```

(Imagen de la función "devolver libro" en la clase biblioteca)

-Si cabe el libro a devolver se mete con los demás libros de la biblioteca sino se cancela.

```
if (this.numLibros < this.arrayLibros.length) {
    this.arrayLibros[numLibros] = libroDetectado;
    this.numLibros++;

} else {
    System.out.println("No caben mas libros (Está lleno la biblioteca de libros)");
    System.out.println("Cancelamiento de la transacción de libro");
    this.arrayPrestamos[numPrestamos] = cancelacionDevolucion;

    setNumPrestamos(getNumPrestamos() + 1);

}
```

(Imagen de la función "devolver libro" en la clase biblioteca)

-El main goza de ser más sencillo que la anterior función ("realizar préstamo").

```
System.out.println("Has elegido devolver libros prestados ");
System.out.println("Introduce el isbn del libro que quieres devolver ");
sc.nextLine(); // limpiar el buffer
int isbnDevolver = sc.nextInt();
biblioteca.devolverLibroTomado(sesionUsuario, isbnDevolver

);
```

(Imagen de la función "devolver libro" en la clase App)

9.-El log-in.

-El inicio de sesión se compone de una variable vital, que designa al usuario que está iniciando sesión. (Eludiendo así la necesidad de crear de un objeto cuenta).

-Obtiene el objeto usuario cada vez que inicia sesión el usuario correspondiente.

```
// SesionUsuario indica el usuario que ha iniciado sesion en la biblioteca.
sesionUsuario = biblioteca.getArrayUsuarios()[posPassword];
```

(Imagen de la variable "sesionUsuario" en la clase App)

-Cada vez que un procedimiento solicita un parámetro de usuario, se le pasa el usuario que ha iniciado sesión es decir el usuario que contiene "sesionUsuario".

```
biblioteca.tomarLibroPrestado(sesionUsuario, isbnPrestamo);
```

(Imagen de la función tomarLibroPrestado con "sesionUsuario" e ISBN en la clase App)

9.1.-Roles administrador.

-Los roles de los usuarios se definieron en el programa implementado un booleano en la clase usuario. El primer usuario creado en el programa es administrador.

```
private boolean admin;
```

(Imagen del atributo admin en la clase "usuario")

-Dependiendo del estado del booleano asignado el usuario podrá hacer ciertas acciones adicionales o estarían visibles pero restringidas.