



**Universidad de las Fuerzas Armadas - ESPE**

**Pregrado SII OCT24 – MAR25**

**Programación Orientada a Objetos – Nrc: 1323**

**Actividad Autónoma No. 1: Sistema de Gestión de Biblioteca**

**Nombre del estudiante:**

David Iván Granada Pachacama

**Nombre del docente:**

Mgtr. Luis Enrique Jaramillo Montaña

**Fecha de entrega:** 15 de diciembre del 2024

## Contenido

Introducción.....	3
1. Objetivos.....	4
1.1. Objetivos General .....	4
1.2. Objetivos Específicos .....	4
2. Diagrama UML.....	5
3. Codificación .....	6
4. Resultados.....	15
5. Conclusiones .....	20
6. Recomendaciones .....	20
7. Referencias Bibliográficas .....	21
8. Apéndice.....	22

## Tabla de figuras

<b>Figura 1. UML gestión Biblioteca .....</b>	<b>5</b>
<b>Figura 2. Corrida del programa .....</b>	<b>17</b>

## Introducción

Según (*Libro - 15 Ejercicio Complementario*, s. f.) concluye que:

Aplicar los conceptos básicos de Programación Orientada a Objetos (POO) para gestionar una Biblioteca siendo el estilo y diseño el primer paso para codificarlo es muy importante ya que podemos realizarlo mediante un modelado unificado de objetos (UML).

Además, se podrá visualizar los requerimientos como modelado de clases y objetos, así como también la estructura del programa, encapsulamiento, constructores, gestión de errores (excepciones), persistencia de datos así como también colección y arreglos.

## **1. Objetivos**

### **1.1. Objetivos General**

- Diseñar y desarrollar un programa en Java que represente un Sistema de Gestión de Biblioteca, con su correspondiente diagrama UML y código mediante el uso de la IDE Apache NetBeans para mostrar las relaciones que existe entre las clases y objetos.

### **1.2. Objetivos Específicos**

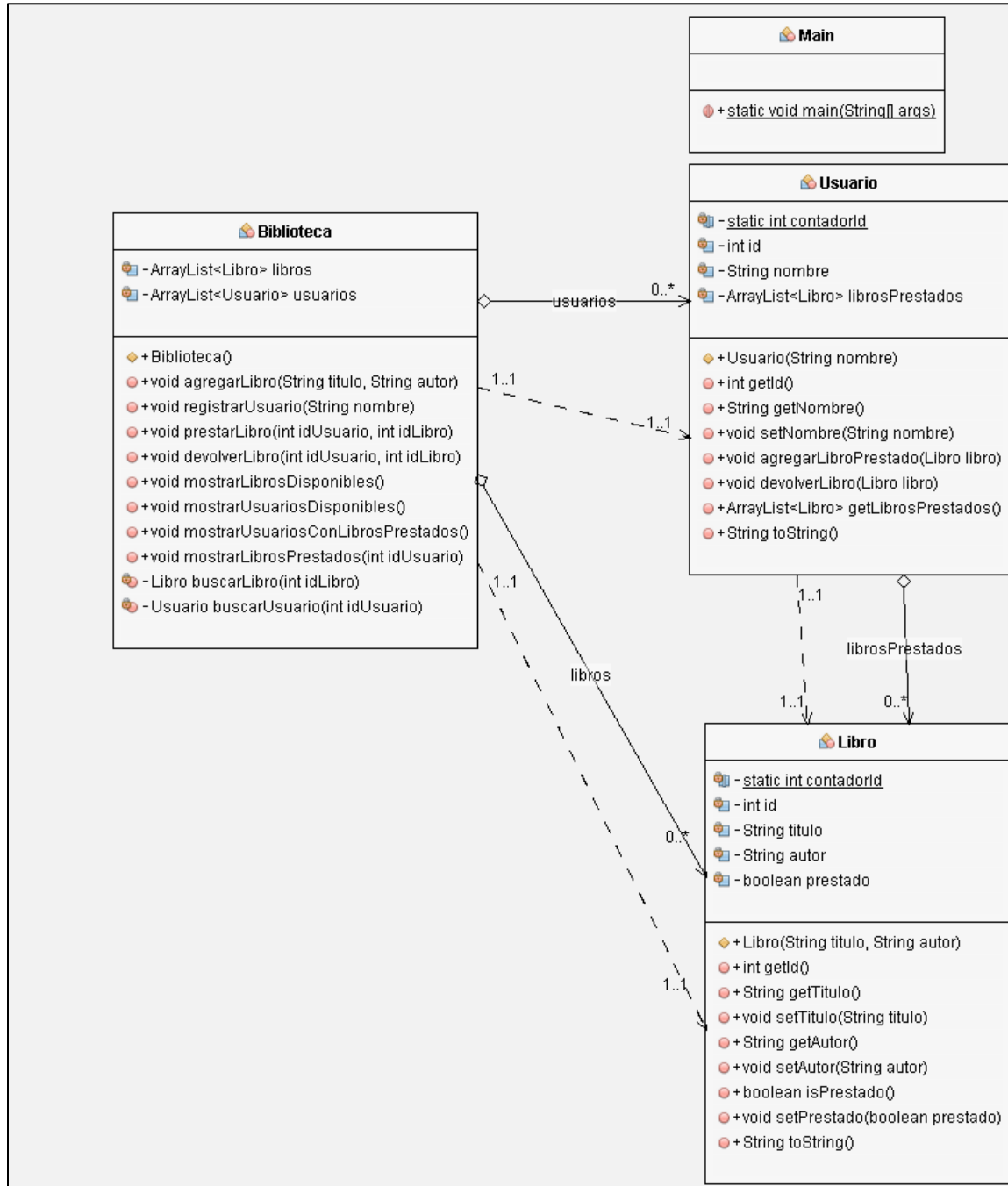
- Registrar libros y usuarios que se visualicen en pantalla.
- Gestionar prestamos que se visualicen en pantalla.
- Aplicar los conceptos básicos de Programación Orientada a Objetos (POO) aprendido en clase.

## 2. Diagrama UML

A continuación, se puede observar en la Figura 1 el diagrama de clases en UML generado por NetBeans.

**Figura 1.**

*UML gestión Biblioteca*



### 3. Codificación

A continuación, se puede observar la codificación en lenguaje Java.

#### 3.1. Clase Libro.java

```
public class Libro { //declara una clase pública llamada Libro

    private static int contadorId = 1; // declara una variable estática contadorId que se inicializa en 1.

    private final int id; // declara una variable de instancia de tipo int.

    private String titulo; // una cadena de texto para almacenar el título del libro

    private String autor; // una cadena de texto para almacenar el autor del libro

    private boolean prestado; // una variable booleana para indicar si el libro está prestado o no.


    public Libro(String titulo, String autor) { //declara un constructor público para la clase Libro que toma dos parámetros (titulo y
autor) para inicializar un nuevo objeto de tipo Libro.

        this.id = contadorId++; //asigna a la variable de instancia id el valor actual de contadorId y luego incrementa contadorId en 1.

        this.titulo = titulo;// inicializa los atributos titulo, autor y prestado

        this.autor = autor;

        this.prestado = false;

    }

    public int getId() { // define un método público que devuelve el valor del atributo id

        return id;

    }

    public String getTitulo() { // define un método público que devuelve el valor del atributo titulo

        return titulo;

    }

    public void setTitulo(String titulo) { // define un método público que establece el valor del atributo titulo

        this.titulo = titulo;

    }

    public String getAutor() { // define un método público que devuelve el valor del atributo autor.

        return autor; // Permite modificar el título del libro.

    }

    public void setAutor(String autor) { // define un método público que establece el valor del atributo autor.

        this.autor = autor; // Permite acceder al nombre del autor del libro.

    }

    public boolean isPrestado() { // define un método público que devuelve el valor del atributo prestado

        return prestado; // permite saber si el libro está prestado (true) o no (false).

    }

    public void setPrestado(boolean prestado) { // define un método público que establece el valor del atributo prestado

        this.prestado = prestado; // permite cambiar el estado del libro a prestado o no prestado

    }

    @Override // sobrescribe el método toString() de la clase Object, devuelve una cadena de texto con el ID, titulo, autor y si está
prestado o no

    public String toString() {

        return "ID: " + id + ", Título: " + titulo + ", Autor: " + autor + ", Prestado: " + (prestado ? "Sí" : "No");

    }

}
```

### 3.2. Clase Usuario.java

```
import java.util.ArrayList; //importa la clase ArrayList y almacén una lista dinámica de objetos

public class Usuario { //declara una clase pública que será accesible desde otras clases.

    private static int contadorId = 1; // Atributo estático que se inicializa en 1.

    private final int id; // declara una variable de instancia de tipo int que no puede cambiar después de ser asignada.

    private String nombre; //declara una variable de tipo String para almacenar el nombre del usuario.

    private ArrayList<Libro> librosPrestados; // declara una variable de instancia librosPrestados que será utilizada para almacenar
una lista dinámica de objetos Libro

    public Usuario(String nombre) { //declara el constructor público de la clase Usuario, para inicializar un nuevo objeto Usuario.

        this.id = contadorId++; //asigna a la variable id el valor actual de contadorId y luego incrementa contadorId en 1.

        this.nombre = nombre; //inicializa la variable nombre con el valor del argumento en el constructor.

        this.librosPrestados = new ArrayList<>(); // inicializa la lista librosPrestados que almacenará los libros prestados por el
usuario.

    }

    public int getId() { // define un método público que devuelve el valor del atributo id

        return id;

    }

    public String getNombre() { // define un método público que devuelve el valor del atributo nombre

        return nombre;

    }

    public void setNombre(String nombre) { // define un método público que permite modificar el nombre del usuario

        this.nombre = nombre;

    }

    public void agregarLibroPrestado(Libro libro) { // define un método público que agrega un libro a la lista de librosPrestados del
usuario

        librosPrestados.add(libro);

    }

    public void devolverLibro(Libro libro) { // define un método público que elimina un libro de la lista librosPrestados

        librosPrestados.remove(libro);

    }

    public ArrayList<Libro> getLibrosPrestados() { // define un método público que devuelve la lista de libros prestados por el
usuario.

        return librosPrestados;

    }

    @Override // sobrescribe el método toString() de la clase Object, devuelve una cadena de texto con el ID y nombre.

    public String toString() {

        return "ID: " + id + ", Nombre: " + nombre;

    }

}
```

### 3.3. Clase Biblioteca.java

```
import java.util.ArrayList; //importa la clase ArrayList y almacena una lista dinámica de objetos

public class Biblioteca { //declaración de la clase Biblioteca

    private ArrayList<Libro> libros; //declara una lista para almacenar libros

    private ArrayList<Usuario> usuarios; // declara una lista para almacenar usuarios

    public Biblioteca() { // Inicializa las listas libros y usuarios como nuevas instancias de ArrayList

        this.libros = new ArrayList<>();

        this.usuarios = new ArrayList<>();

    }

    public void agregarLibro(String titulo, String autor) {

        Libro libro = new Libro(titulo, autor); //crea un nuevo objeto Libro con el título y autor

        libros.add(libro); // agrega a la lista de libros

        System.out.println("Libro agregado: " + libro.getTitulo() + " con ID: " + libro.getId()); //imprime un mensaje
        confirmando que el libro ha sido agregado

    }

    public void registrarUsuario(String nombre) {

        Usuario usuario = new Usuario(nombre); // crea un nuevo objeto Usuario con el nombre del parámetro

        usuarios.add(usuario); // agrega a la lista de usuarios

        System.out.println("Usuario registrado: " + usuario.getNombre() + " con ID: " + usuario.getId()); //imprime un mensaje
        confirmando que el usuario ha sido registrado

    }

    public void prestarLibro(int idUsuario, int idLibro) {

        Usuario usuario = buscarUsuario(idUsuario); // busca al usuario y al libro mediante sus ID

        Libro libro = buscarLibro(idLibro);

        if (usuario != null && libro != null) { // Si ambos existen, verifica si el libro no está prestado.

            if (!libro.isPrestado()) { // Si no lo está prestado

                libro.setPrestado(true); // lo marca como prestado

                usuario.agregarLibroPrestado(libro); // agrega el libro a la lista de libros prestados del usuario.

                System.out.println("El libro '" + libro.getTitulo() + "' ha sido prestado a " + usuario.getNombre());

            } else {

                System.out.println("El libro ya está prestado."); // si el libro ya está prestado, muestra un mensaje
                indicando que no se puede prestar.

            }

        } else {

            System.out.println("Usuario o libro no encontrado."); // si no se encuentra al usuario o al libro, muestra un
            mensaje de error.

        }

    }

}
```



```

public void devolverLibro(int idUsuario, int idLibro) {

    Usuario usuario = buscarUsuario(idUsuario);// se buscan el usuario y el libro por sus ID

    Libro libro = buscarLibro(idLibro);

    if (usuario != null && libro != null) { // Si ambos existen

        if (libro.isPrestado()) { // y el libro está prestado

            libro.setPrestado(false); // se marca como no prestado

            usuario.devolverLibro(libro); // se elimina de la lista de libros prestados del usuario.

            System.out.println("El libro '" + libro.getTitulo() + "' ha sido devuelto por " + usuario.getNombre());

        } else {

            System.out.println("El libro no estaba prestado."); // Si el libro no estaba prestado, muestra un mensaje
indicando que no se puede devolver

        }

    } else {

        System.out.println("Usuario o libro no encontrado."); // Si no se encuentra el usuario o el libro, muestra un
mensaje de error

    }

}

public void mostrarLibrosDisponibles() { // Muestra todos los libros que no están prestados.

    System.out.println("\n--- Lista de libros disponibles ---");

    boolean hayLibros = false;

    for (Libro libro : libros) { // Recorre la lista de libros

        if (!libro.isPrestado()) { // si el libro no está prestado, lo muestra con su ID, título y autor.

            System.out.println("ID: " + libro.getId() + " - Título: " + libro.getTitulo() + " - Autor: " +
libro.getAutor());

            hayLibros = true;

        }

    }

    if (!hayLibros) { // Si no hay libros disponibles

        System.out.println("No hay libros disponibles."); // imprime un mensaje indicando que no hay libros disponibles.

    }

}

public void mostrarUsuariosDisponibles() { // muestra todos los usuarios registrados en la biblioteca.

    System.out.println("\n--- Lista de usuarios disponibles ---");

    if (usuarios.isEmpty()) {

        System.out.println("No hay usuarios registrados."); // Si no hay usuarios registrados, imprime un mensaje
indicando que no hay usuarios

    } else {

        for (Usuario usuario : usuarios) { // de lo contrario, imprime la ID y el nombre de cada usuario

            System.out.println("ID: " + usuario.getId() + " - Nombre: " + usuario.getNombre());

        }

    }

}

}

```

```

public void mostrarUsuariosConLibrosPrestados() { // muestra los usuarios que tienen libros prestados.

    System.out.println("\n--- Lista de usuarios con libros prestados ---");

    boolean hayUsuariosConLibros = false;

    for (Usuario usuario : usuarios) { // recorre la lista de usuarios
        if (!usuario.getLibrosPrestados().isEmpty()) { // si un usuario tiene libros prestados
            System.out.println(" ID: " + usuario.getId() + " - Nombre: " + usuario.getNombre() +
                " - Libros prestados: " + usuario.getLibrosPrestados().size()); // muestra su ID, nombre y
la cantidad de libros prestados.

            hayUsuariosConLibros = true;
        }
    }

    if (!hayUsuariosConLibros) { // si no hay usuarios con libros prestados, imprime un mensaje indicándolo
        System.out.println("No hay usuarios con libros prestados.");
    }
}

public void mostrarLibrosPrestados(int idUsuario) { // muestra los libros prestados a un usuario específico.
    Usuario usuario = buscarUsuario(idUsuario); // busca al usuario por su ID

    if (usuario != null) { // si lo encuentra, muestra los libros que tiene prestados.
        System.out.println("\n--- Libros prestados al usuario " + usuario.getNombre() + " ---");
        if (!usuario.getLibrosPrestados().isEmpty()) { // si el usuario no tiene libros prestados, muestra un mensaje
indicándolo
            for (Libro libro : usuario.getLibrosPrestados()) {
                System.out.println("ID: " + libro.getId() + " - Título: " + libro.getTitulo() + " - Autor: " +
libro.getAutor());
            }
        } else {
            System.out.println(" El usuario no tiene libros prestados.");
        }
    } else {
        System.out.println("Usuario no encontrado.");
    }
}

private Libro buscarLibro(int idLibro) { // busca un libro en la lista de libros por su ID
    for (Libro libro : libros) { // si lo encuentra, lo devuelve
        if (libro.getId() == idLibro) {
            return libro;
        }
    }
}

// Si no, devuelve null.
return null;
}

```



```
private Usuario buscarUsuario(int idUsuario) { //busca un usuario en la lista de usuarios por su ID

    for (Usuario usuario : usuarios) { // Si lo encuentra, lo devuelve

        if (usuario.getId() == idUsuario) {

            return usuario;

        }

    }

    // Si no, devuelve null.

    return null;

}
```

### 3.4. Clase Main.java

```
import java.util.InputMismatchException; // Maneja excepciones cuando el tipo de entrada no coincide con el tipo esperado
import java.util.Scanner; // usa para leer la entrada del usuario desde la consola

public class Main { // clase principal que ejecuta el programa

    public static void main(String[] args) {

        Biblioteca biblioteca = new Biblioteca(); // crea una instancia de la clase Biblioteca

        Scanner scanner = new Scanner(System.in); // crea un objeto Scanner llamado scanner para leer la entrada del usuario desde la
        consola.

        int opcion = 0; // guarda la opcion seleccionada por el usuario en el menu

        // se muestra un menu de opciones para que el usuario elija una acción que desea realizar en la biblioteca

        do {

            System.out.println("\n--- Menu de la Biblioteca ---");

            System.out.println("1. Agregar libro");

            System.out.println("2. Registrar usuario");

            System.out.println("3. Prestar libro");

            System.out.println("4. Devolver libro");

            System.out.println("5. Mostrar libros disponibles");

            System.out.println("6. Mostrar usuarios con libros prestados");

            System.out.println("7. Salir");

            System.out.print("Ingrese una opcion: "); // solicita al usuario que ingrese una opcion

            try {

                opcion = scanner.nextInt(); // leer un numero entero introducido por el usuario, que se asigna a la variable opcion

                scanner.nextLine(); // Limpiar el buffer de entrada

            } catch (InputMismatchException e) { // si el usuario ingresa algo diferente a un numero se captura la excepcion, muestra un
            mensaje de error y permite que el usuario ingrese nuevamente la opcion.

                System.out.println("Opcion no valida, debe ser un numero.");

                scanner.nextLine(); // Limpiar el buffer de entrada

                continue;

            }

            switch (opcion) {

                case 1: // pide que ingrese el titulo y el autor del libro

                    System.out.print("Ingrese el titulo del libro: ");

                    String titulo = scanner.nextLine();

                    System.out.print("Ingrese el autor del libro: ");

                    String autor = scanner.nextLine();

                    biblioteca.agregarLibro(titulo, autor); // llama al método agregarLibro de la clase Biblioteca

                    break;

                case 2: // pide que ingrese el nombre del usuario

                    System.out.print("Ingrese el nombre del usuario: ");

                    String nombre = scanner.nextLine();

                    biblioteca.registrarUsuario(nombre); // llama al método registrarUsuario de la clase Biblioteca para registrar al
                    nuevo usuario.

            }

        }

    }

}
```

```
System.out.println(" Usuario registrado exitosamente.");
break;
case 3:
    System.out.println("\n--- Lista de usuarios ---");
    biblioteca.mostrarUsuariosDisponibles();//muestra la lista de usuarios registrados
    try {
        System.out.print("\nIngrese el ID del usuario que tomara el libro: ");
        int idUsuarioPrestar = scanner.nextInt();
        scanner.nextLine(); // Limpiar el buffer de entrada

        System.out.println("\n--- Lista de libros disponibles ---");
        biblioteca.mostrarLibrosDisponibles(); // muestra la lista libros disponibles

        System.out.print("\nIngrese el ID del libro a prestar: ");
        int idLibroPrestar = scanner.nextInt();
        scanner.nextLine(); // Limpiar el buffer de entrada

        biblioteca.prestarLibro(idUsuarioPrestar, idLibroPrestar);// llama al metodo prestarLibro de la clase Biblioteca
con los Ids proporcionados
        System.out.println("Libro prestado exitosamente.");
    } catch (InputMismatchException e) { // Si ocurre un error se captura la excepcion y se muestra un mensaje de error
        System.out.println("Error: El ID debe ser un numero.");
        scanner.nextLine(); // Limpiar el buffer de entrada
    }
    break;
case 4:
    System.out.println("\n--- Lista de usuarios con libros prestados ---");
    biblioteca.mostrarUsuariosConLibrosPrestados();// muestra la lista de usuarios que tienen libros prestados
    try {
        System.out.print("\nIngrese el ID del usuario que devolvera el libro: ");
        int idUsuarioDevolver = scanner.nextInt();
        scanner.nextLine(); // Limpiar el buffer de entrada

        System.out.println("\n--- Lista de libros prestados ---");
        biblioteca.mostrarLibrosPrestados(idUsuarioDevolver);

        System.out.print("\nIngrese el ID del libro a devolver: ");
        int idLibroDevolver = scanner.nextInt();
        scanner.nextLine(); // Limpiar el buffer de entrada

        biblioteca.devolverLibro(idUsuarioDevolver, idLibroDevolver);// llama al metodo devolverLibro de la clase
Biblioteca con los IDs proporcionados
        System.out.println("Libro devuelto exitosamente.");
    } catch (InputMismatchException e) { // Si ocurre un error se captura la excepcion y se muestra un mensaje de error
```

```
        System.out.println("Error: El ID debe ser un numero.");
        scanner.nextLine(); // Limpiar el buffer de entrada
    }
    break;

    case 5:
        System.out.println("\n--- Lista de libros disponibles ---");
        biblioteca.mostrarLibrosDisponibles();//llama al metodo mostrarLibrosDisponibles de la clase Biblioteca, que muestra
la lista de libros que no están prestados
        break;

    case 6:
        biblioteca.mostrarUsuariosConLibrosPrestados();//llama al metodo mostrarUsuariosConLibrosPrestados de la clase
Biblioteca, que muestra la lista de usuarios que tienen libros prestados
        break;

    case 7:// imprime un mensaje de salida y se termina el ciclo do-while.
        System.out.println("Saliendo...");
        break;

    default:
        System.out.println("Opcion no valida.");// Si se ingresa una opcion que no es valida, se muestra un mensaje de error
    }
} while (opcion != 7);
scanner.close();//cierra el objeto scanner para liberar los recursos asociados.
}
}
```

#### 4. Resultados

Al crear un programa para gestionar una biblioteca que incluya un menú para facilitar su uso. Las funcionalidades del programa abarcarán:

- **Gestión de Usuarios:** Permitir la creación de usuarios y gestionar los libros que tienen prestados.
- **Registro de Libros:** Registrar libros en el sistema y gestionar su disponibilidad.
- **Préstamos:** Facilitar el préstamo de libros a los usuarios, mostrando alertas si un libro no está disponible.

##### Estructura del Programa

##### Clases:

Las clases deben ser privadas y contar con un constructor parametrizado solo para la creación, no para la modificación.

- **Clase Biblioteca:** Maneja arrays de objetos Libro y Usuario.

Atributos y Métodos:

- **Usuario:**

Atributos: ID, Nombre, LibroPrestado.

Métodos: getLibrosPrestados(), agregarLibro, devolverLibro. Solo se permiten getters para ID y Nombre.

- **Libro:**

Atributos: ID, Título, Autor, Prestado (booleano).

Métodos: Getters para ID, Título, Autor y Prestado; Setter para setPrestado.

- **Métodos de la Clase Biblioteca:**

registrarLibro()

registrarUsuario()

prestarLibro()

devolverLibro()

mostrarLibrosDisponibles()

El sistema debe cargar los datos al inicio y manejar adecuadamente los préstamos y devoluciones de libros.

#### 4.1. Conceptos Aplicados

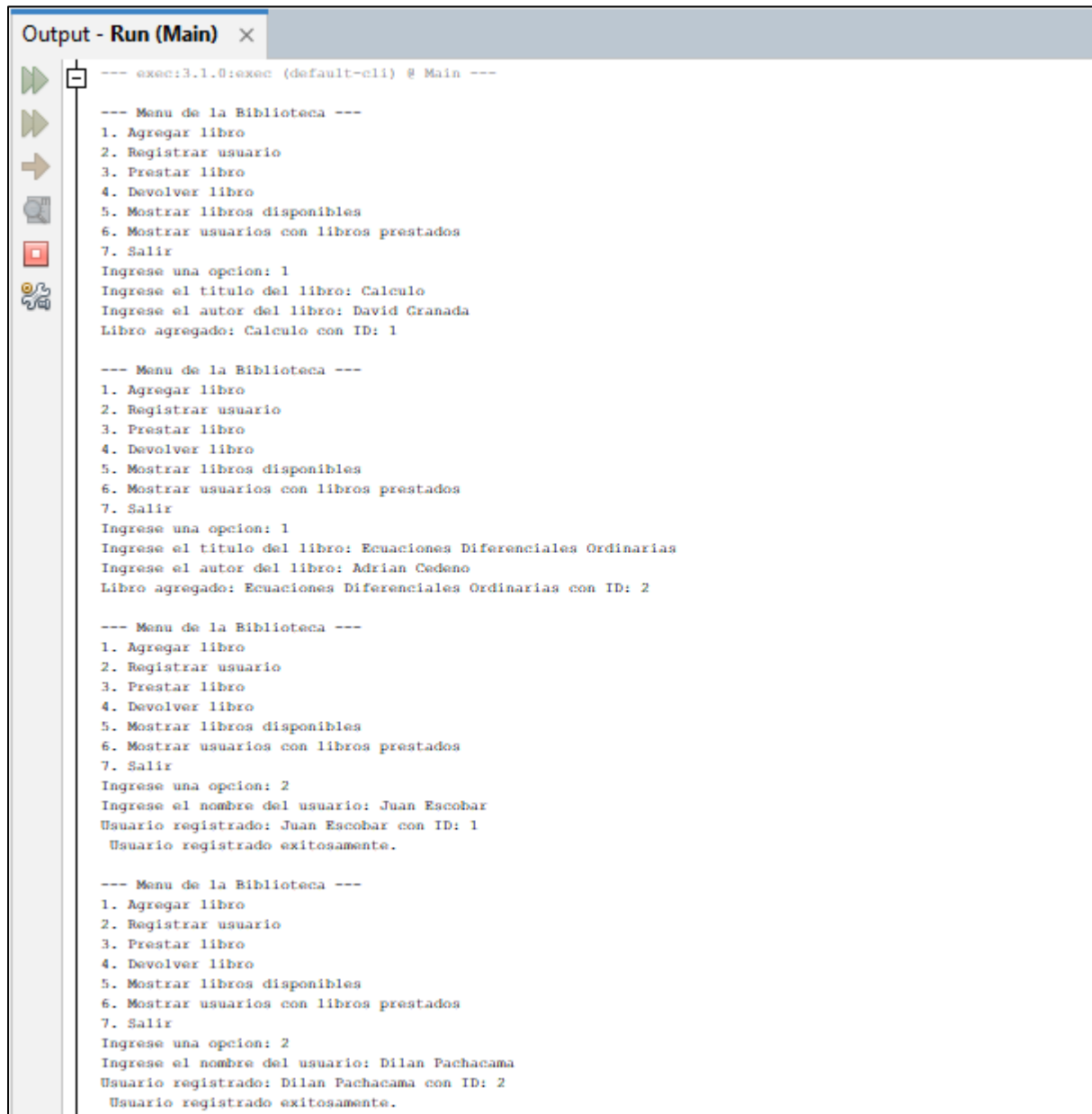
- **Modelamiento de Clases y Objetos:** Se modelaron las clases Libro, Usuario y Biblioteca.
- **Encapsulamiento:** Los atributos están **privados** y solo se acceden mediante getters y setters.
- **Constructores:** Los objetos Libro y Usuario se crean con un constructor.
- **Getters y Setters:** Se accede y modifica la información de los objetos a través de getters y setters.
- **Relaciones entre Clases:**
  - Biblioteca tiene una relación de **agregación** con Usuario y Libro.
  - Usuario tiene una **relación de composición** con la lista de libros prestados.
- **Excepciones:** Aunque no se manejan excepciones explícitas, se pueden añadir para controlar la entrada de datos.
- **Persistencia de Datos:** Se podría extender la persistencia a un archivo, pero actualmente se gestiona en memoria.
- **Arreglos y Colecciones:** Se utilizan **ArrayList** para la lista de usuarios y la lista de libros.
- **Estructura General de un Programa:** La clase Main orquesta la lógica principal de la aplicación.
- **Lectura y Escritura por Consola:** Los mensajes se muestran en la consola para facilitar la interacción con el usuario.



## 4.1. Corrida del programa

Figura 2.

*Corrida del programa*



```
Output - Run (Main) X
--- exec:3.1.0:exec (default-cli) @ Main ---

--- Menu de la Biblioteca ---
1. Agregar libro
2. Registrar usuario
3. Prestar libro
4. Devolver libro
5. Mostrar libros disponibles
6. Mostrar usuarios con libros prestados
7. Salir
Ingrese una opcion: 1
Ingrese el titulo del libro: Calculo
Ingrese el autor del libro: David Granada
Libro agregado: Calculo con ID: 1

--- Menu de la Biblioteca ---
1. Agregar libro
2. Registrar usuario
3. Prestar libro
4. Devolver libro
5. Mostrar libros disponibles
6. Mostrar usuarios con libros prestados
7. Salir
Ingrese una opcion: 1
Ingrese el titulo del libro: Ecuaciones Diferenciales Ordinarias
Ingrese el autor del libro: Adrian Cedeno
Libro agregado: Ecuaciones Diferenciales Ordinarias con ID: 2

--- Menu de la Biblioteca ---
1. Agregar libro
2. Registrar usuario
3. Prestar libro
4. Devolver libro
5. Mostrar libros disponibles
6. Mostrar usuarios con libros prestados
7. Salir
Ingrese una opcion: 2
Ingrese el nombre del usuario: Juan Escobar
Usuario registrado: Juan Escobar con ID: 1
Usuario registrado exitosamente.

--- Menu de la Biblioteca ---
1. Agregar libro
2. Registrar usuario
3. Prestar libro
4. Devolver libro
5. Mostrar libros disponibles
6. Mostrar usuarios con libros prestados
7. Salir
Ingrese una opcion: 2
Ingrese el nombre del usuario: Dilan Pachacama
Usuario registrado: Dilan Pachacama con ID: 2
Usuario registrado exitosamente.
```



```
Output - Run (Main) x
--- Menu de la Biblioteca ---
1. Agregar libro
2. Registrar usuario
3. Prestar libro
4. Devolver libro
5. Mostrar libros disponibles
6. Mostrar usuarios con libros prestados
7. Salir
Ingrese una opcion: 2
Ingrese el nombre del usuario: Dilan Pachacama
Usuario registrado: Dilan Pachacama con ID: 2
Usuario registrado exitosamente.

--- Menu de la Biblioteca ---
1. Agregar libro
2. Registrar usuario
3. Prestar libro
4. Devolver libro
5. Mostrar libros disponibles
6. Mostrar usuarios con libros prestados
7. Salir
Ingrese una opcion: 3

--- Lista de usuarios ---


--- Lista de usuarios disponibles ---
ID: 1 - Nombre: Juan Escobar
ID: 2 - Nombre: Dilan Pachacama

Ingrese el ID del usuario que tomara el libro: 2

--- Lista de libros disponibles ---

--- Lista de libros disponibles ---
ID: 1 - Titulo: Calculo - Autor: David Granada
ID: 2 - Titulo: Ecuaciones Diferenciales Ordinarias - Autor: Adrian Cedeno

Ingrese el ID del libro a prestar: 2
El libro 'Ecuaciones Diferenciales Ordinarias' ha sido prestado a Dilan Pachacama
Libro prestado exitosamente.
```



--- Menu de la Biblioteca ---

1. Agregar libro
2. Registrar usuario
3. Prestar libro
4. Devolver libro
5. Mostrar libros disponibles
6. Mostrar usuarios con libros prestados
7. Salir

Ingrese una opcion: 5

--- Lista de libros disponibles ---

--- Lista de libros disponibles ---

ID: 1 - Titulo: Calculo - Autor: David Granada

--- Menu de la Biblioteca ---

1. Agregar libro
2. Registrar usuario
3. Prestar libro
4. Devolver libro
5. Mostrar libros disponibles
6. Mostrar usuarios con libros prestados
7. Salir

Ingrese una opcion: 6

--- Lista de usuarios con libros prestados ---

ID: 2 - Nombre: Dilan Pachacama - Libros prestados: 1

--- Menu de la Biblioteca ---

1. Agregar libro
2. Registrar usuario
3. Prestar libro
4. Devolver libro
5. Mostrar libros disponibles
6. Mostrar usuarios con libros prestados
7. Salir

Ingrese una opcion: 4

--- Lista de usuarios con libros prestados ---

--- Lista de usuarios con libros prestados ---

ID: 2 - Nombre: Dilan Pachacama - Libros prestados: 1

Ingrese el ID del usuario que devolvera el libro: 2

--- Lista de libros prestados ---

--- Libros prestados al usuario Dilan Pachacama ---

ID: 2 - Titulo: Ecuaciones Diferenciales Ordinarias - Autor: Adrian Cedeno

Ingrese el ID del libro a devolver: 2

El libro 'Ecuaciones Diferenciales Ordinarias' ha sido devuelto por Dilan Pachacama

Libro devuelto exitosamente.

```
--- Lista de libros disponibles ---

--- Lista de libros disponibles ---
ID: 1 - Titulo: Calculo - Autor: David Granada
ID: 2 - Titulo: Ecuaciones Diferenciales Ordinarias - Autor: Adrian Cedeno

--- Menu de la Biblioteca ---
1. Agregar libro
2. Registrar usuario
3. Prestar libro
4. Devolver libro
5. Mostrar libros disponibles
6. Mostrar usuarios con libros prestados
7. Salir
Ingrese una opcion: 6

--- Lista de usuarios con libros prestados ---
No hay usuarios con libros prestados.

--- Menu de la Biblioteca ---
1. Agregar libro
2. Registrar usuario
3. Prestar libro
4. Devolver libro
5. Mostrar libros disponibles
6. Mostrar usuarios con libros prestados
7. Salir
```

## 5. Conclusiones

- Registrar libros y usuarios mejora la organización de la biblioteca.
- La visualización de préstamos facilita el seguimiento de libros disponibles.
- Implementar conceptos de POO refuerza el aprendizaje teórico.

## 6. Recomendaciones

- Desarrollar una interfaz intuitiva para mejorar la usabilidad.
- Implementar alertas para informar sobre la disponibilidad de libros.
- Mantener buena documentación y comentarios en el código para facilitar el mantenimiento.

## 7. Referencias Bibliográficas

*Libro—15 Ejercicio Complementario.* (s. f.). Recuperado 14 de diciembre de 2024, de  
<https://luisjaramillom.github.io/POO.io/Unidades/ejercicio1/ejercicio.html>

## 8. Apéndice

- 8.1. Link Video explicación del programa: [https://youtu.be/F6AS\\_8cUm7Y](https://youtu.be/F6AS_8cUm7Y)
- 8.2. Link Repositorio GitHub: [https://github.com/digranada/202451\\_1323.git](https://github.com/digranada/202451_1323.git)