

UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE



DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN



TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN (TICS)

PROGRAMACIÓN ORIENTADA A OBJETOS

NRC: 1323

TEMA: Diseño y Modelado

ALUMNO:

- Gabriel Gualpa

DOCENTE:

Ing. Luis Jaramillo

Sangolquí, Diciembre 08, 2024

1. Introducción	2
2. Objetivo General	2
2.1. Objetivos Específicos	2
3. Desarrollo	3
3.1 Diagrama UML	3
3.1.1 Explicación de las Clases	3
3.1.1 Explicación de las Relaciones	4
3.2 Implementación Código Java	4
3.3. Resultados	16
4. Conclusiones	16
5. Recomendaciones	17

1. Introducción

En el desarrollo de sistemas de software, es importante comprender cómo modelar y organizar los objetos y las relaciones entre ellos. La actividad de modelado en UML (Lenguaje Unificado de Modelado) nos permite representar visualmente las clases, atributos, métodos y las interacciones entre los objetos que componen el sistema. El objetivo de esta actividad fue diseñar un sistema de gestión de una biblioteca utilizando diagramas UML y luego implementar ese diseño en código Java. El modelo incluye clases como Libro, Usuario, Biblioteca, Empleado y Transaccion, que interactúan entre sí para simular el préstamo, devolución y gestión de libros dentro de una biblioteca.

2. Objetivo General

Diseñar y modelar un sistema de gestión de biblioteca utilizando diagramas UML, y posteriormente implementarlo en Java, estableciendo las relaciones y comportamientos entre las clases para simular operaciones como el préstamo y devolución de libros.

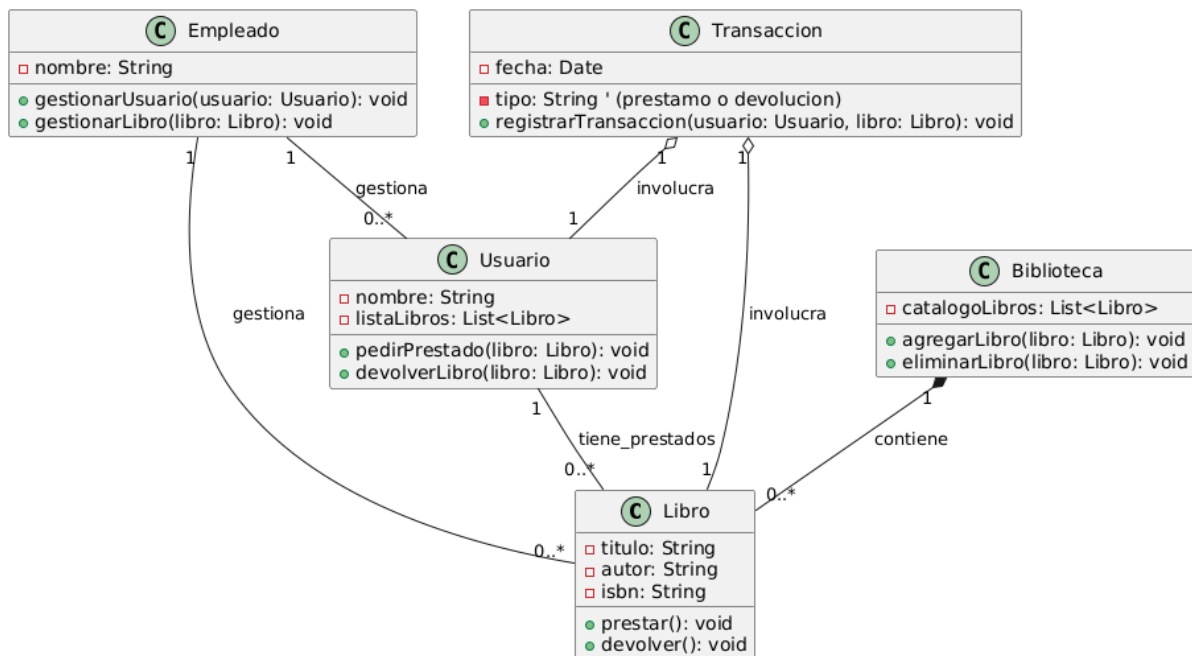
2.1. Objetivos Específicos

- Diseñar un sistema utilizando diagramas UML que describan el comportamiento y la estructura de las clases involucradas en el sistema de biblioteca.
- Implementar las clases en Java basadas en los diagramas UML diseñados.

- Establecer las relaciones entre las clases (composición, agregación, asociación) y cómo estas reflejan las interacciones del sistema en el código.
- Demostrar cómo el modelado UML facilita la comprensión y organización de un sistema antes de su implementación.

3. Desarrollo

3.1 Diagrama UML



3.1.1 Explicación de las Clases

Las clases involucradas son las siguientes:

Libro: Representa los libros disponibles en la biblioteca, con atributos como título, autor, ISBN, y métodos como prestar() y devolver(). Este objeto se utiliza para simular la disponibilidad de un libro y las acciones de préstamo y devolución.

Usuario: Representa a los usuarios que interactúan con la biblioteca. Cada usuario tiene una lista de libros prestados. En términos de relaciones UML, se usa composición con la clase Libro, ya que un usuario puede tener varios libros prestados, y la vida de los libros prestados depende de la existencia del usuario.

Biblioteca: Es el centro de la gestión de libros en el sistema. Contiene un catálogo de libros que se puede gestionar a través de métodos como agregarLibro() y eliminarLibro(). Se relaciona con Libro por composición, ya que la biblioteca posee y controla la vida de los libros.

Empleado: Gestiona tanto a los usuarios como los libros. Los empleados tienen métodos como gestionarUsuario() y gestionarLibro(), que permiten realizar diversas operaciones en los objetos de tipo Usuario y Libro. Estas relaciones se representan mediante asociación, ya que los empleados trabajan con los usuarios y libros, pero no dependen completamente de ellos para existir.

Transacción: Registra los eventos de préstamo y devolución de libros. Se utiliza agregación para modelar la relación con Usuario y Libro, ya que una transacción involucra un usuario y un libro, pero no existe de manera independiente sin esos objetos.

3.1.1 Explicación de las Relaciones

El diagrama muestra las relaciones entre las clases:

Composición entre Biblioteca y Libro, indicando que los libros son parte fundamental del catálogo de la biblioteca y no existen sin ella.

Asociación entre Empleado y las clases Usuario y Libro, porque los empleados gestionan a los usuarios y los libros.

Agregación entre Transaccion y las clases Usuario y Libro, ya que una transacción involucra estas clases, pero los objetos pueden existir por separado sin la transacción.

3.2 Implementación Código Java

A continuación, se creó un código en Java basado en el modelo UML diseñado previamente. Las clases definidas en el diagrama UML fueron implementadas como clases Java con los siguientes detalles:

Clase Libro

Contiene atributos como el título, autor y ISBN, junto con los métodos prestar() y devolver(), que simulan el préstamo y la devolución de libros.

```
Main.java : Libro.java : Usuario.java : Biblioteca.java : Empleado.java : Transaccion.java :
1 public class Libro {
2     private String titulo;
3     private String autor;
4     private String isbn;
5
6     // Constructor
7     public Libro(String titulo, String autor, String isbn) {
8         this.titulo = titulo;
9         this.autor = autor;
10        this.isbn = isbn;
11    }
12
13    // Métodos
14    public void prestar() {
15        System.out.println("El libro '" + titulo + "' ha sido prestado.");
16    }
17
18    public void devolver() {
19        System.out.println("El libro '" + titulo + "' ha sido devuelto.");
20    }
21
22    // Getters
23    public String getTitulo() {
24        return titulo;
25    }
26
27    public String getAutor() {
28        return autor;
29    }
30
31    public String getIsbn() {
32        return isbn;
33    }
34 }
35
```

```
public class Libro {
    private String titulo;
    private String autor;
    private String isbn;

    // Constructor
    public Libro(String titulo, String autor, String isbn) {
        this.titulo = titulo;
        this.autor = autor;
        this.isbn = isbn;
    }

    // Métodos
    public void prestar() {
        System.out.println("El libro '" + titulo + "' ha sido prestado.");
    }

    public void devolver() {
        System.out.println("El libro '" + titulo + "' ha sido devuelto.");
    }
}
```

```

// Getters
public String getTitulo() {
    return titulo;
}

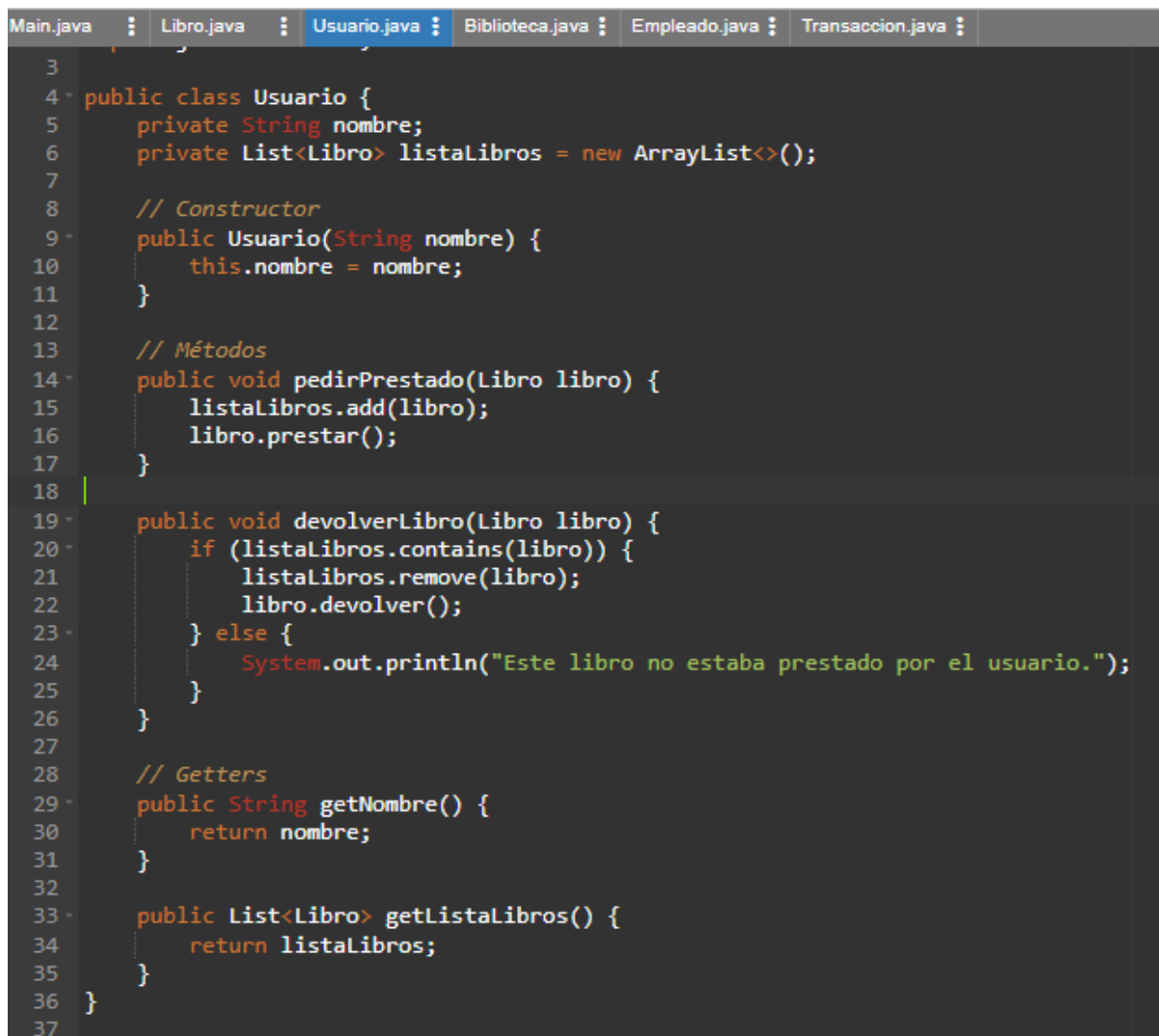
public String getAutor() {
    return autor;
}

public String getIsbn() {
    return isbn;
}
}

```

Clase Usuario

Cada usuario tiene una lista de libros prestados. Los métodos pedirPrestado() y devolverLibro() permiten gestionar los libros prestados.



```

Main.java : Libro.java : Usuario.java : Biblioteca.java : Empleado.java : Transaccion.java :
3
4 public class Usuario {
5     private String nombre;
6     private List<Libro> listaLibros = new ArrayList<>();
7
8     // Constructor
9     public Usuario(String nombre) {
10         this.nombre = nombre;
11     }
12
13     // Métodos
14     public void pedirPrestado(Libro libro) {
15         listaLibros.add(libro);
16         libro.prestar();
17     }
18
19     public void devolverLibro(Libro libro) {
20         if (listaLibros.contains(libro)) {
21             listaLibros.remove(libro);
22             libro.devolver();
23         } else {
24             System.out.println("Este libro no estaba prestado por el usuario.");
25         }
26     }
27
28     // Getters
29     public String getNombre() {
30         return nombre;
31     }
32
33     public List<Libro> getListaLibros() {
34         return listaLibros;
35     }
36 }
37

```

```
import java.util.ArrayList;
import java.util.List;

public class Usuario {
    private String nombre;
    private List<Libro> listaLibros = new ArrayList<>();

    // Constructor
    public Usuario(String nombre) {
        this.nombre = nombre;
    }

    // Métodos
    public void pedirPrestado(Libro libro) {
        listaLibros.add(libro);
        libro.prestar();
    }

    public void devolverLibro(Libro libro) {
        if (listaLibros.contains(libro)) {
            listaLibros.remove(libro);
            libro.devolver();
        } else {
            System.out.println("Este libro no estaba prestado por el usuario.");
        }
    }

    // Getters
    public String getNombre() {
        return nombre;
    }

    public List<Libro> getListaLibros() {
        return listaLibros;
    }
}
```

Clase Biblioteca

Gestiona el catálogo de libros. Los métodos `agregarLibro()` y `eliminarLibro()` permiten agregar y eliminar libros del catálogo de la biblioteca.

```
Main.java : Libro.java : Usuario.java : Biblioteca.java : Empleado.java : Transaccion.java :
1 import java.util.ArrayList;
2 import java.util.List;
3
4 public class Biblioteca {
5     private List<Libro> catalogoLibros = new ArrayList<>();
6
7     // Métodos
8     public void agregarLibro(Libro libro) {
9         catalogoLibros.add(libro);
10        System.out.println("El libro '" + libro.getTitulo() + "' ha sido agregado al catálogo.");
11    }
12
13    public void eliminarLibro(Libro libro) {
14        catalogoLibros.remove(libro);
15        System.out.println("El libro '" + libro.getTitulo() + "' ha sido eliminado del catálogo.");
16    }
17
18    // Getters
19    public List<Libro> getCatalogoLibros() {
20        return catalogoLibros;
21    }
22 }
23
```

```
import java.util.ArrayList;
import java.util.List;

public class Biblioteca {
    private List<Libro> catalogoLibros = new ArrayList<>();

    // Métodos
    public void agregarLibro(Libro libro) {
        catalogoLibros.add(libro);
        System.out.println("El libro '" + libro.getTitulo() + "' ha sido agregado al catálogo.");
    }

    public void eliminarLibro(Libro libro) {
        catalogoLibros.remove(libro);
        System.out.println("El libro '" + libro.getTitulo() + "' ha sido eliminado del catálogo.");
    }

    // Getters
    public List<Libro> getCatalogoLibros() {
        return catalogoLibros;
    }
}
```

Clase Empleado

Los empleados gestionan tanto a los usuarios como a los libros. Los métodos `gestionarUsuario()` y `gestionarLibro()` permiten simular la administración de los usuarios y los libros.


```
Main.java : Libro.java : Usuario.java : Biblioteca.java : Empleado.java : Transaccion.java :
1 public class Empleado {
2     private String nombre;
3
4     // Constructor
5     public Empleado(String nombre) {
6         this.nombre = nombre;
7     }
8
9     // Métodos
10    public void gestionarUsuario(Usuario usuario) {
11        System.out.println("Empleado " + nombre + " está gestionando al usuario " + usuario.getNombre());
12    }
13
14    public void gestionarLibro(Libro libro) {
15        System.out.println("Empleado " + nombre + " está gestionando el libro '" + libro.getTitulo() + "'");
16    }
17
18    // Getter
19    public String getNombre() {
20        return nombre;
21    }
22 }
23
```

```
public class Empleado {
    private String nombre;

    // Constructor
    public Empleado(String nombre) {
        this.nombre = nombre;
    }

    // Métodos
    public void gestionarUsuario(Usuario usuario) {
        System.out.println("Empleado " + nombre + " está gestionando al usuario " +
usuario.getNombre());
    }

    public void gestionarLibro(Libro libro) {
        System.out.println("Empleado " + nombre + " está gestionando el libro '" +
libro.getTitulo() + "'");
    }

    // Getter
    public String getNombre() {
        return nombre;
    }
}
```

Clase Transaccion

Registra las transacciones de préstamo y devolución. El método registrarTransaccion() asocia un usuario y un libro con una transacción específica.

```
Main.java : Libro.java : Usuario.java : Biblioteca.java : Empleado.java : Transaccion.java :
1- import java.util.Date;
2
3- public class Transaccion {
4-     private Date fecha;
5-     private String tipo; // "prestamo" o "devolucion"
6
7-     // Constructor
8-     public Transaccion(Date fecha, String tipo) {
9-         this.fecha = fecha;
10-        this.tipo = tipo;
11-    }
12
13-    // Método para registrar la transacción
14-    public void registrarTransaccion(Usuario usuario, Libro libro) {
15-        System.out.println("Transacción registrada: " + tipo + " - Usuario: " + usuario.getNombre() + ", Libro: " + libro.getTitulo());
16-    }
17
18-    // Getters
19-    public Date getFecha() {
20-        return fecha;
21-    }
22
23-    public String getTipo() {
24-        return tipo;
25-    }
26
27- }
```

```
import java.util.Date;

public class Transaccion {
    private Date fecha;
    private String tipo; // "prestamo" o "devolucion"

    // Constructor
    public Transaccion(Date fecha, String tipo) {
        this.fecha = fecha;
        this.tipo = tipo;
    }

    // Método para registrar la transacción
    public void registrarTransaccion(Usuario usuario, Libro libro) {
        System.out.println("Transacción registrada: " + tipo + " - Usuario: " +
usuario.getNombre() + ", Libro: " + libro.getTitulo());
    }

    // Getters
    public Date getFecha() {
        return fecha;
    }

    public String getTipo() {
        return tipo;
    }
}
```

Clase Main

Se crean instancias de las clases y se simula el préstamo, devolución y gestión de libros en la biblioteca, mostrando cómo interactúan entre sí las clases.

```
Main.java : Libro.java : Usuario.java : Biblioteca.java : Empleado.java : Transaccion.java :
1 import java.util.ArrayList;
2 import java.util.Date;
3 import java.util.List;
4
5 public class Main {
6     public static void main(String[] args) {
7         // Crear instancias de libros
8         Libro libro1 = new Libro("Java para Principiantes", "John Doe", "12345");
9         Libro libro2 = new Libro("Programación Avanzada", "Jane Smith", "67890");
10        Libro libro3 = new Libro("Fundamentos de Python", "Alice Johnson", "11121");
11        Libro libro4 = new Libro("Introducción a C#", "Bob Martin", "11223");
12        Libro libro5 = new Libro("Bases de Datos SQL", "Carlos Pérez", "33445");
13
14        // Crear una instancia de la biblioteca y agregar libros al catálogo
15        Biblioteca biblioteca = new Biblioteca();
16        biblioteca.agregarLibro(libro1);
17        biblioteca.agregarLibro(libro2);
18        biblioteca.agregarLibro(libro3);
19        biblioteca.agregarLibro(libro4);
20        biblioteca.agregarLibro(libro5);
21
22        // Crear 5 instancias de usuarios
23        Usuario usuario1 = new Usuario("Carlos Perez");
24        Usuario usuario2 = new Usuario("Ana Ruiz");
25        Usuario usuario3 = new Usuario("Luis Gómez");
26        Usuario usuario4 = new Usuario("Marta Sánchez");
27        Usuario usuario5 = new Usuario("José Martín");
28
29        // El usuario1 pide prestado el libro1
30        usuario1.pedirPrestado(libro1);
31
32        // El usuario2 pide prestado el libro2
33        usuario2.pedirPrestado(libro2);
34
35        // El usuario3 pide prestado el libro3
36        usuario3.pedirPrestado(libro3);
37    }
```

Main.java	Libro.java	Usuario.java	Biblioteca.java	Empleado.java	Transaccion.java
44				<i>// Crear 5 instancias de empleados</i>	
45				Empleado empleado1 = new Empleado("Ana Ruiz");	
46				Empleado empleado2 = new Empleado("David Torres");	
47				Empleado empleado3 = new Empleado("Clara López");	
48				Empleado empleado4 = new Empleado("Raúl Gómez");	
49				Empleado empleado5 = new Empleado("Elena Fernández");	
50					
51				<i>// Los empleados gestionan a los usuarios</i>	
52				empleado1.gestionarUsuario(usuario1);	
53				empleado2.gestionarUsuario(usuario2);	
54				empleado3.gestionarUsuario(usuario3);	
55				empleado4.gestionarUsuario(usuario4);	
56				empleado5.gestionarUsuario(usuario5);	
57					
58				<i>// Los empleados gestionan los libros</i>	
59				empleado1.gestionarLibro(libro1);	
60				empleado2.gestionarLibro(libro2);	
61				empleado3.gestionarLibro(libro3);	
62				empleado4.gestionarLibro(libro4);	
63				empleado5.gestionarLibro(libro5);	
64					
65				<i>// Crear 5 instancias de transacciones</i>	
66				Transaccion transaccion1 = new Transaccion(new Date(), "prestamo");	
67				Transaccion transaccion2 = new Transaccion(new Date(), "prestamo");	
68				Transaccion transaccion3 = new Transaccion(new Date(), "prestamo");	
69				Transaccion transaccion4 = new Transaccion(new Date(), "prestamo");	
70				Transaccion transaccion5 = new Transaccion(new Date(), "prestamo");	
71					
72				<i>// Registrar las transacciones</i>	
73				transaccion1.registrarTransaccion(usuario1, libro1);	
74				transaccion2.registrarTransaccion(usuario2, libro2);	
75				transaccion3.registrarTransaccion(usuario3, libro3);	
76				transaccion4.registrarTransaccion(usuario4, libro4);	
77				transaccion5.registrarTransaccion(usuario5, libro5);	
78					
79				<i>// Los usuarios devuelven los libros</i>	
80				usuario1.devolverLibro(libro1);	
72				<i>// Registrar las transacciones</i>	
73				transaccion1.registrarTransaccion(usuario1, libro1);	
74				transaccion2.registrarTransaccion(usuario2, libro2);	
75				transaccion3.registrarTransaccion(usuario3, libro3);	
76				transaccion4.registrarTransaccion(usuario4, libro4);	
77				transaccion5.registrarTransaccion(usuario5, libro5);	
78					
79				<i>// Los usuarios devuelven los libros</i>	
80				usuario1.devolverLibro(libro1);	
81				usuario2.devolverLibro(libro2);	
82				usuario3.devolverLibro(libro3);	
83				usuario4.devolverLibro(libro4);	
84				usuario5.devolverLibro(libro5);	
85					
86				<i>// Los empleados gestionan los libros después de la devolución</i>	
87				empleado1.gestionarLibro(libro1);	
88				empleado2.gestionarLibro(libro2);	
89				empleado3.gestionarLibro(libro3);	
90				empleado4.gestionarLibro(libro4);	
91				empleado5.gestionarLibro(libro5);	
92				}	
93				}	
94					

```
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        // Crear instancias de libros
        Libro libro1 = new Libro("Java para Principiantes", "John Doe", "12345");
        Libro libro2 = new Libro("Programación Avanzada", "Jane Smith", "67890");
        Libro libro3 = new Libro("Fundamentos de Python", "Alice Johnson", "11121");
        Libro libro4 = new Libro("Introducción a C#", "Bob Martin", "11223");
        Libro libro5 = new Libro("Bases de Datos SQL", "Carlos Pérez", "33445");

        // Crear una instancia de la biblioteca y agregar libros al catálogo
        Biblioteca biblioteca = new Biblioteca();
        biblioteca.agregarLibro(libro1);
        biblioteca.agregarLibro(libro2);
        biblioteca.agregarLibro(libro3);
        biblioteca.agregarLibro(libro4);
        biblioteca.agregarLibro(libro5);

        // Crear 5 instancias de usuarios
        Usuario usuario1 = new Usuario("Carlos Perez");
        Usuario usuario2 = new Usuario("Ana Ruiz");
        Usuario usuario3 = new Usuario("Luis Gómez");
        Usuario usuario4 = new Usuario("Marta Sánchez");
        Usuario usuario5 = new Usuario("José Martín");

        // El usuario1 pide prestado el libro1
        usuario1.pedirPrestado(libro1);

        // El usuario2 pide prestado el libro2
        usuario2.pedirPrestado(libro2);

        // El usuario3 pide prestado el libro3
        usuario3.pedirPrestado(libro3);

        // El usuario4 pide prestado el libro4
        usuario4.pedirPrestado(libro4);

        // El usuario5 pide prestado el libro5
        usuario5.pedirPrestado(libro5);

        // Crear 5 instancias de empleados
        Empleado empleado1 = new Empleado("Ana Ruiz");
        Empleado empleado2 = new Empleado("David Torres");
        Empleado empleado3 = new Empleado("Clara López");
        Empleado empleado4 = new Empleado("Raúl Gómez");
        Empleado empleado5 = new Empleado("Elena Fernández");

        // Los empleados gestionan a los usuarios
        empleado1.gestionarUsuario(usuario1);
        empleado2.gestionarUsuario(usuario2);
        empleado3.gestionarUsuario(usuario3);
```

```
    empleado4.gestionarUsuario(usuario4);
    empleado5.gestionarUsuario(usuario5);

    // Los empleados gestionan los libros
    empleado1.gestionarLibro(libro1);
    empleado2.gestionarLibro(libro2);
    empleado3.gestionarLibro(libro3);
    empleado4.gestionarLibro(libro4);
    empleado5.gestionarLibro(libro5);

    // Crear 5 instancias de transacciones
    Transaccion transaccion1 = new Transaccion(new Date(), "prestamo");
    Transaccion transaccion2 = new Transaccion(new Date(), "prestamo");
    Transaccion transaccion3 = new Transaccion(new Date(), "prestamo");
    Transaccion transaccion4 = new Transaccion(new Date(), "prestamo");
    Transaccion transaccion5 = new Transaccion(new Date(), "prestamo");

    // Registrar las transacciones
    transaccion1.registrarTransaccion(usuario1, libro1);
    transaccion2.registrarTransaccion(usuario2, libro2);
    transaccion3.registrarTransaccion(usuario3, libro3);
    transaccion4.registrarTransaccion(usuario4, libro4);
    transaccion5.registrarTransaccion(usuario5, libro5);

    // Los usuarios devuelven los libros
    usuario1.devolverLibro(libro1);
    usuario2.devolverLibro(libro2);
    usuario3.devolverLibro(libro3);
    usuario4.devolverLibro(libro4);
    usuario5.devolverLibro(libro5);

    // Los empleados gestionan los libros después de la devolución
    empleado1.gestionarLibro(libro1);
    empleado2.gestionarLibro(libro2);
    empleado3.gestionarLibro(libro3);
    empleado4.gestionarLibro(libro4);
    empleado5.gestionarLibro(libro5);
}
}
```

3.3. Resultados

```
input
l libro 'Java para Principiantes' ha sido agregado al catálogo.
l libro 'Programación Avanzada' ha sido agregado al catálogo.
l libro 'Fundamentos de Python' ha sido agregado al catálogo.
l libro 'Introducción a C#' ha sido agregado al catálogo.
l libro 'Bases de Datos SQL' ha sido agregado al catálogo.
l libro 'Java para Principiantes' ha sido prestado.
l libro 'Programación Avanzada' ha sido prestado.
l libro 'Fundamentos de Python' ha sido prestado.
l libro 'Introducción a C#' ha sido prestado.
l libro 'Bases de Datos SQL' ha sido prestado.
Empleado Ana Ruiz está gestionando al usuario Carlos Perez
Empleado David Torres está gestionando al usuario Ana Ruiz
Empleado Clara López está gestionando al usuario Luis Gómez
Empleado Raúl Gómez está gestionando al usuario Marta Sánchez
Empleado Elena Fernández está gestionando al usuario José Martín
Empleado Ana Ruiz está gestionando el libro 'Java para Principiantes'
Empleado David Torres está gestionando el libro 'Programación Avanzada'
Empleado Clara López está gestionando el libro 'Fundamentos de Python'
Empleado Raúl Gómez está gestionando el libro 'Introducción a C#'
Empleado Elena Fernández está gestionando el libro 'Bases de Datos SQL'
Transacción registrada: prestamo - Usuario: Carlos Perez, Libro: Java para Principiantes
Transacción registrada: prestamo - Usuario: Ana Ruiz, Libro: Programación Avanzada
Transacción registrada: prestamo - Usuario: Luis Gómez, Libro: Fundamentos de Python
Transacción registrada: prestamo - Usuario: Marta Sánchez, Libro: Introducción a C#
Transacción registrada: prestamo - Usuario: José Martín, Libro: Bases de Datos SQL
l libro 'Java para Principiantes' ha sido devuelto.
l libro 'Programación Avanzada' ha sido devuelto.
l libro 'Fundamentos de Python' ha sido devuelto.
l libro 'Introducción a C#' ha sido devuelto.
l libro 'Bases de Datos SQL' ha sido devuelto.
Empleado Ana Ruiz está gestionando el libro 'Java para Principiantes'
Empleado David Torres está gestionando el libro 'Programación Avanzada'
Empleado Clara López está gestionando el libro 'Fundamentos de Python'
Empleado Raúl Gómez está gestionando el libro 'Introducción a C#'
Empleado Elena Fernández está gestionando el libro 'Bases de Datos SQL'

..Program finished with exit code 0
Press ENTER to exit console.
```

4. Conclusiones

- El uso de diagramas UML en el proceso de diseño de software facilita la visualización de la estructura y las relaciones entre las clases. Este enfoque contribuye significativamente a la comprensión del sistema antes de su implementación, lo que permite detectar problemas en el diseño a tiempo
- El modelado UML proporciona una base buena para la programación orientada a objetos, ayudando a estructurar y organizar el código de manera eficiente. Además, este enfoque permite que los diseñadores y desarrolladores trabajen de forma más colaborativa y comprendan mejor los requisitos del sistema, lo que mejora la calidad del software final.

5. Recomendaciones

- Antes de la implementación, realiza revisiones del diseño UML con el equipo para identificar posibles mejoras y garantizar que todas las relaciones entre clases estén correctamente modeladas.
- Es importante que todos los diagramas UML estén bien documentados para que sean comprensibles para cualquier miembro del equipo de desarrollo. Esto facilita el mantenimiento y las futuras modificaciones del sistema.