ODAW 1B | MÓDULO: PROGRAMACIÓN | CURSO: 2025-2026

MEMORIA DE PROYECTO *ODAW1.3.UD4*

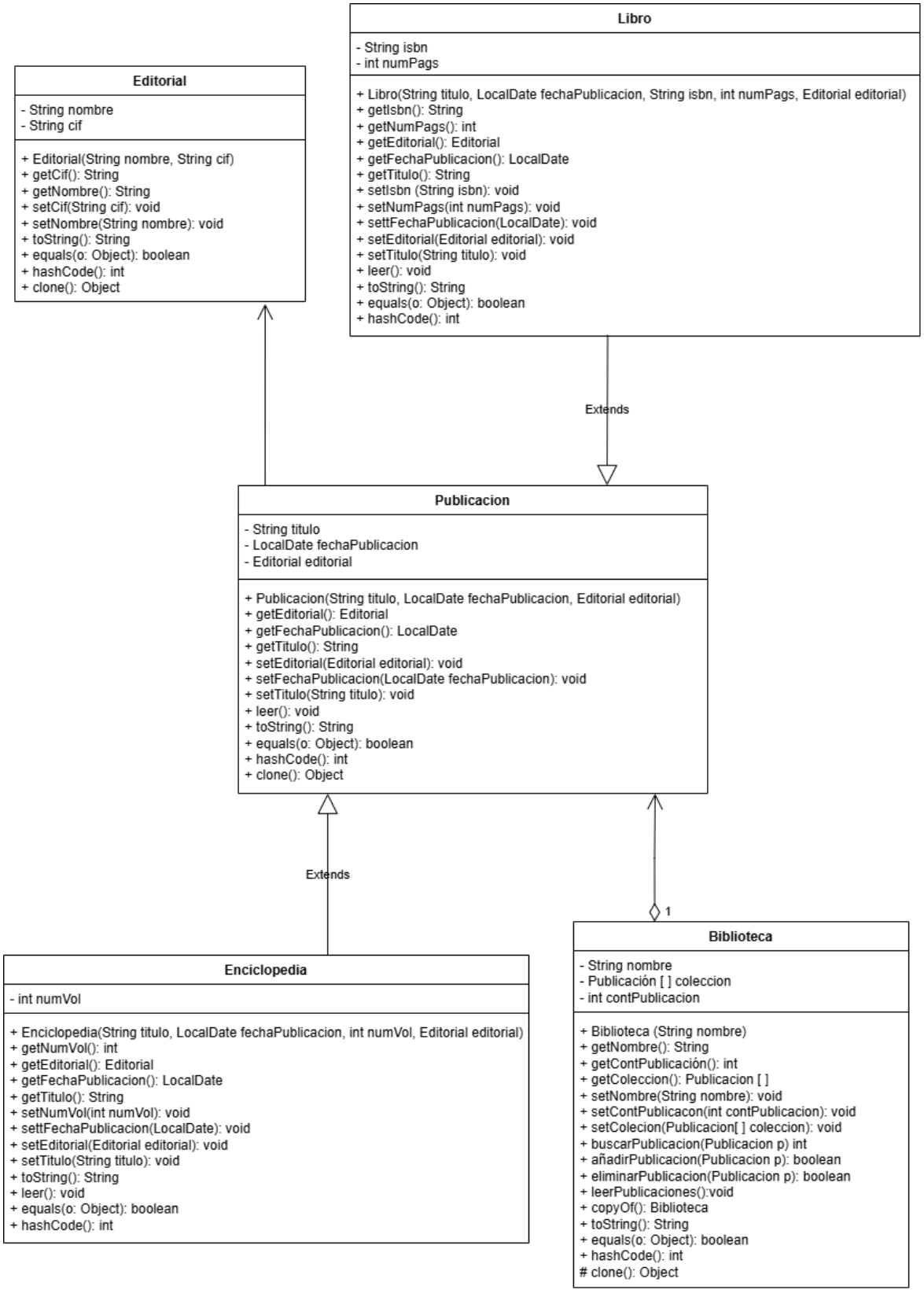
GRUPO 3: MILLARES TALLÓN, AROA MARÍA & VÁZQUEZ ZAS, ISMAEL

ÍNDICE

MEMORIA DE PROYECTO

ODAW1.3.UD4

A) DIAGRAMA DE CLASES UML



B) JUSTIFICACIÓN DE RELACIONES

Asociación entre Publicacion y Editorial:

- Se ha definido como una relación estructural mediante un atributo.
- Una Publicacion conoce a su Editorial, pero no hay una relación de propiedad o control del ciclo de vida.
- La flecha indica navegabilidad: desde el objeto publicación podemos acceder a la información de la empresa editorial que la emite.

Herencia entre Publicacion y Libro:

- Se ha empleado esta relación porque existe una jerarquía de tipo "es-un".
- El Libro es un tipo concreto de Publicacion que comparte atributos comunes como el título y la fecha.
- Esto permite la redefinición de métodos (como el método leer()) y el uso de polimorfismo.

Herencia entre Publicacion y Enciclopedia:

- Se ha empleado esta relación porque existe una jerarquía de tipo "es-un".
- La Enciclopedia es un tipo concreto de Publicacion que comparte atributos comunes como el título y la fecha.
- Esto permite la redefinición de métodos (como el método leer()) y el uso de polimorfismo.

Agregación entre Biblioteca y Publicacion:

- Se ha elegido la agregación porque representa una relación de "todo-parte" con independencia.
- La Biblioteca actúa como el "todo" que gestiona una colección de publicaciones, pero las publicaciones (las partes) pueden existir sin necesidad de la biblioteca.
- A nivel estructural, la clase Biblioteca contiene un atributo de tipo array (Publicacion[]) para almacenar estas partes.

C) ELECCIÓN DE VISIBILIDAD, TIPO DE DATO Y PARÁMETROS EN CONSTRUCTORES Y ATRIBUTOS EN LAS CLASES PUBLICACIÓN Y LIBRO

Se ha establecido una visibilidad privada (private) para todos los atributos tanto en la clase Publicacion como en su subclase Libro. Esta decisión responde al principio de encapsulamiento, protegiendo el estado interno de los objetos y obligando a utilizar los métodos públicos o los constructores para cualquier acceso o modificación.

En cuanto a los tipos de datos, destaca la elección de java.time.LocalDate para el atributo fechaPublicacion (en lugar de Date o String), lo que permite realizar operaciones cronológicas precisas y validar fechas reales. Para el resto de atributos se han utilizado tipos estándar adecuados a su naturaleza: String para cadenas de texto (titulo, isbn) y el tipo primitivo int para valores numéricos contables (numPags).

```java
public class Publicacion implements Cloneable {

    private String titulo;
    private LocalDate fechaPublicacion;
    private Editorial editorial;

    @Contract(pure = true)
    public Publicacion(String titulo, LocalDate fechaPublicacion, Editorial editorial){
        this.titulo = titulo;
        this.fechaPublicacion = fechaPublicacion;
        this.editorial = editorial;
    }
```

Respecto a los constructores, la estrategia se basa en la delegación. La clase Publicacion inicializa sus atributos básicos mediante asignación directa. Por su parte, el constructor de Libro recibe todos los parámetros necesarios (propios y heredados) e invoca inmediatamente a super(...) para garantizar que la parte genérica de la publicación se construya correctamente antes de asignar el ISBN y el número de páginas.

```java
public class Libro extends Publicacion {

    private String isbn;
    private int numPags;

    public Libro(String titulo, LocalDate fechaPublicacion, String isbn, int numPags, Editorial editorial){
        super(titulo, fechaPublicacion, editorial);
        this.isbn = isbn;
        this.numPags = numPags;
    }
```

D) ESTRATEGIAS PARA REDEFINIR MÉTODOS HEREDADOS DE OBJECT. PUBLICACION: COMPORTAMIENTO DE LA IMPLEMENTACIÓN DE UNA COPIA SUPERFICIAL

1. Estrategia en la clase Editorial

En la clase Editorial, la redefinición de los métodos se centra en la identificación unívoca de la empresa.

Equals y HashCode: Se ha determinado que dos editoriales son iguales si comparten el mismo CIF, independientemente del nombre. Esto evita duplicidades si una empresa cambia de denominación comercial pero mantiene su identidad fiscal.

Clone: Al estar formada por atributos inmutables (String), la implementación de clone devuelve una copia estándar (super.clone()), lo cual es suficiente y eficiente.

```java
public class Editorial implements Cloneable {
    @Override
    public String toString() {
        return "Editorial{nombre='" + nombre + "', cif='" + cif + "'}";
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Editorial editorial = (Editorial) o;
        return Objects.equals(cif, editorial.cif);
    }
    @Override
    public int hashCode() {
        return Objects.hash(cif);
    }

    @Override
    public Object clone() {
        try {
            return super.clone();
        } catch (CloneNotSupportedException e) {
            return null;
        }
    }
```

2. Estrategia en la clase Publicacion y el problema de la copia

Para la clase Publicacion, la estrategia de igualdad combina el título de la obra y la editorial asociada. Sin embargo, el punto crítico reside en la redefinición del método clone.

Análisis de copia superficial vs. profunda: El enunciado plantea qué ocurriría si implementásemos una copia superficial (simplemente retornando super.clone()). En ese escenario, el objeto clonado tendría sus propios atributos primitivos, pero compartiría la misma referencia en memoria al objeto Editorial que la publicación original.

Consecuencia: Si modificásemos el nombre de la editorial en la copia, el cambio se propagaría automáticamente a la publicación original, rompiendo el principio de independencia de los objetos.

Solución implementada: Para evitar este efecto colateral, se ha implementado una copia profunda. Dentro del método clone, forzamos explícitamente la clonación del objeto editorial (clon.editorial = (Editorial) this.editorial.clone()). De esta forma, la nueva publicación apunta a una nueva instancia de editorial, garantizando una independencia total.

```java
public class Publicacion implements Cloneable {

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Publicacion that = (Publicacion) o;
        return Objects.equals(titulo, that.titulo) &&
                Objects.equals(editorial, that.editorial);
    }
```

```java
@Override
public int hashCode() { return Objects.hash(titulo, editorial); }

@Override
public Object clone() {
    try {
        Publicacion clon = (Publicacion) super.clone();
        if (this.editorial != null) {
            clon.editorial = (Editorial) this.editorial.clone();
        }
        return clon;
    } catch (CloneNotSupportedException e) {
        return null;
    }
}
```

```java
public class Biblioteca implements Cloneable {

    public Biblioteca copyOf() {
        try {
            Biblioteca copia = (Biblioteca) super.clone();

            copia.coleccion = new Publicacion[20];

            for (int i = 0; i < this.contPublicacion; i++) {
                copia.coleccion[i] = (Publicacion) this.coleccion[i].clone();
            }

            copia.contPublicacion = this.contPublicacion;

            return copia;
        } catch (CloneNotSupportedException e) {
            return null;
        }
    }
```

```java
public class App {
    public static void main(String[] args) {
        Biblioteca copiaSeguridad = miBiblio.copyOf();
```

3. Estrategia en la clase Libro

La clase Libro utiliza una estrategia de extensión del comportamiento heredado para evitar la duplicidad de código:

ToString: Invoca a super.toString() para obtener la cadena base ("Publicación...") y le concatena únicamente la información específica del libro (ISBN y número de páginas).

Equals: Primero valida la igualdad base llamando a super.equals(o), que comprueba título y editorial, y, solo si esta se cumple, verifica la coincidencia del ISBN. Esto asegura que dos libros con el mismo título pero diferente edición (ISBN) sean tratados como objetos distintos.

```java
public class Libro extends Publicacion {
    @Override
    public String toString() {
        return super.toString() + " -> Libro{isbn= '" + isbn +
                "', paginas =" + numPags + "}";
    }

    @Override
    public boolean equals(Object o) {
        if (!super.equals(o)) return false;
        Libro libro = (Libro) o;
        return Objects.equals(isbn, libro.isbn);
    }
```

E) EXPLICACIÓN DE LA CLASE BIBLIOTECA: OPERACIONES DE LECTURA, COMPARACIÓN Y COPIA

1. Operación de Lectura: Polimorfismo

La operación de lectura, implementada en el método leerPublicaciones, es un ejemplo práctico de polimorfismo. El método recorre el array coleccion e invoca el método .leer() de cada objeto almacenado.

Aunque la referencia en el array es de tipo genérico Publicacion, gracias al enlazamiento dinámico (dynamic binding), la Máquina Virtual de Java determina en tiempo de ejecución el tipo real del objeto (si es un Libro o una Enciclopedia). Esto permite que se ejecute la versión específica del método leer de cada subclase, mostrando datos particulares como el ISBN o los volúmenes, sin necesidad de que la clase Biblioteca conozca esos detalles internos.

```java
public class Biblioteca implements Cloneable {                    6

    public void leerPublicaciones() {
        System.out.println("\n--- Catálogo de la Biblioteca " + this.nombre + " ---");
        for (int i = 0; i < this.contPublicacion; i++) {
            this.coleccion[i].leer();
        }
    }
```

```java
public class App {                                                 5
    public static void main(String[] args) {
        miBiblio.leerPublicaciones();
        miTeca.leerPublicaciones();
```

```
--- Catálogo de la Biblioteca Biblioteca Municipal ---
Estamos leyendo el libro: El Quijote, publicado en 1605-01-01, con ISBN 978-123456 y número de páginas 800
Estamos leyendo el libro: Inventando inventos, publicado en 2026-02-03, con ISBN 978-555555 y número de páginas 9000
Estamos leyendo la enciclopedia: Enciclopedia Britannica, fecha de publicación 2000-05-20 y volumenes: [1,2,3,4,5,6,7,8,9,10,11,12]

--- Catálogo de la Biblioteca Biblioteca estatal ---
Estamos leyendo el libro: El Quijote, publicado en 1605-01-01, con ISBN 978-123456 y número de páginas 800
Estamos leyendo el libro: Pincipito, publicado en 2001-12-13, con ISBN 978-758955 y número de páginas 150
Estamos leyendo la enciclopedia: Enciclopedia Botánica, fecha de publicación 2005-06-10 y volumenes: [1,2,3,4,5,6,7,8,9]
```

2. Operación de Comparación

Para la comparación de bibliotecas, se ha redefinido el método equals basándose en el atributo nombre. Se asume que el nombre actúa como identificador de la entidad. Esto permite verificar si dos objetos biblioteca hacen referencia a la misma institución, independientemente de que el contenido de sus colecciones varíe momentáneamente.

```java
public class Biblioteca implements Cloneable {
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Biblioteca that = (Biblioteca) o;
        return Objects.equals(nombre, that.nombre);
    }
```

```java
public class App {
    public static void main(String[] args) {
        System.out.println("\nUtilizando el método equals con la copia de seguridad y la biblioteca de la que ha sido clonada (Debe dar true): " + copiaSeguridad.equals(miBiblio));
        System.out.println("\nUtilizando el método equals con la copia de seguridad y la biblioteca de la que no ha sido clonada (Debe dar false): " + copiaSeguridad.equals(miTeca));
```

```
Utilizando el método equals con la copia de seguridad y la biblioteca de la que ha sido clonada (Debe dar true): true

Utilizando el método equals con la copia de seguridad y la biblioteca de la que no ha sido clonada (Debe dar false): false
```

3. Operación de Copia: Garantizando la independencia

El requisito más complejo es la copia de la biblioteca. Dado que la biblioteca contiene un array de objetos mutables (Publicacion), una copia superficial sería insuficiente (ambas bibliotecas compartirían los mismos libros y borrar uno en la copia lo borraría en el original).

Para solucionar esto, el método copyOf implementa una estrategia de copia profunda manual:

1. Se clona el objeto Biblioteca base.
2. Se instancia un nuevo array de publicaciones (new Publicacion[20]) para romper el vínculo con el array original.
3. Se recorre la colección original elemento a elemento, invocando el método clone() de cada publicación y asignándola a la nueva posición.

```java
public class Biblioteca implements Cloneable {
    public Biblioteca copyOf() {
        try {
            Biblioteca copia = (Biblioteca) super.clone();
            copia.coleccion = new Publicacion[20];

            for (int i = 0; i < this.contPublicacion; i++) {
                copia.coleccion[i] = (Publicacion) this.coleccion[i].clone();
            }
            copia.contPublicacion = this.contPublicacion;
            return copia;

        } catch (CloneNotSupportedException e) {
            return null;
        }
    }
```

```java
public class App {
    public static void main(String[] args) {

        Biblioteca copiaSeguridad = miBiblio.copyOf();
        System.out.println("\nImprimiendo copia a continuacion");
        copiaSeguridad.leerPublicaciones();
```

Esta estrategia asegura que la 'Copia de Seguridad' sea una entidad totalmente autónoma: modificar, añadir o eliminar libros en ella no tiene ningún efecto sobre la biblioteca original.

```
--- Catálogo de la Biblioteca Biblioteca Municipal ---
Estamos leyendo el libro: El Quijote, publicado en 1605-01-01, con ISBN 978-123456 y número de páginas 800
Estamos leyendo la enciclopedia: Enciclopedia Britannica, fecha de publicación 2000-05-20 y volumenes: [1,2,3,4,5,6,7,8,9,10,11,12]
Estamos leyendo el libro: Inventando inventos, publicado en 2026-02-03, con ISBN 978-555555 y número de páginas 9000

Imprimiendo copia a continuacion

--- Catálogo de la Biblioteca Biblioteca Municipal ---
Estamos leyendo el libro: El Quijote, publicado en 1605-01-01, con ISBN 978-123456 y número de páginas 800
Estamos leyendo la enciclopedia: Enciclopedia Britannica, fecha de publicación 2000-05-20 y volumenes: [1,2,3,4,5,6,7,8,9,10,11,12]
Estamos leyendo el libro: Inventando inventos, publicado en 2026-02-03, con ISBN 978-555555 y número de páginas 9000
```

F) CÓDIGO COMPLETO MAIN Y EJECUCIÓN

1. MAIN

```
import java.time.LocalDate;

public class App {
    public static void main(String[] args) {

        Editorial edAnaya = new Editorial( nombre: "Anaya",  cif: "B-11111111");
        Editorial edSantillana = new Editorial( nombre: "Santillana",  cif: "A-22222222");

        Libro libro1 = new Libro( titulo: "El Quijote", LocalDate.of( year: 1605,  month: 1,  dayOfMonth: 1),  isbn: "978-123456",  numPags: 800, edAnaya);
        Libro libro2 = new Libro( titulo: "Inventando inventos",LocalDate.of( year: 2026,  month: 2,  dayOfMonth: 3), isbn: "978-555555", numPags: 9000, edAnaya);
        Libro libro3 = new Libro( titulo: "Pincipito",LocalDate.of( year: 2001,  month: 12,  dayOfMonth: 13), isbn: "978-758955", numPags: 150, edAnaya);
        Enciclopedia enciclo1 = new Enciclopedia( titulo: "Enciclopedia Britannica", LocalDate.of( year: 2000,  month: 5,  dayOfMonth: 20),  numVol: 12, edSantillana);
        Enciclopedia enciclo2 = new Enciclopedia( titulo: "Enciclopedia Botánica", LocalDate.of( year: 2005,  month: 6,  dayOfMonth: 10),  numVol: 9, edSantillana);

        Biblioteca miBiblio = new Biblioteca( nombre: "Biblioteca Municipal");
        Biblioteca miTeca = new Biblioteca( nombre: "Biblioteca estatal");


        System.out.println("Añadiendo el Quijote: " + miBiblio.anadirPublicacion(libro1));
        System.out.println("Añadiendo el Quijote (a una blibliioteca distinta): " + miTeca.anadirPublicacion(libro1));
        System.out.println("Añadiendo Principito: " + miTeca.anadirPublicacion(libro3));
        System.out.println("Añadiendo Inventando inventos: " + miBiblio.anadirPublicacion(libro2));
        System.out.println("Añadiendo Britannica: " + miBiblio.anadirPublicacion(enciclo1));
        System.out.println("Añadiendo Botánica: " + miTeca.anadirPublicacion(enciclo2));
        System.out.println("Intentando añadir Quijote otra vez (Debe dar false): " + miBiblio.anadirPublicacion(libro1));


        miBiblio.leerPublicaciones();
        miTeca.leerPublicaciones();
```

```
        System.out.println("\nBuscando 'Inventando inventos'. Posición esperada [1]: " + miBiblio.buscarPublicacion(libro2));

        System.out.println("\nEliminando 'Inventando inventos': " + miBiblio.eliminarPublicacion(libro2));

        System.out.println("\nAñadiendo 'Inventando inventos': " + miBiblio.anadirPublicacion(libro2));

        miBiblio.leerPublicaciones();

        Biblioteca copiaSeguridad = miBiblio.copyOf();
        System.out.println("\nImprimiendo copia a continuacion");
        copiaSeguridad.leerPublicaciones();

        System.out.println("\nUtilizando el método equals con la copia de seguridad y la biblioteca de la que ha sido clonada (Debe dar true): " + copiaSeguridad.equals(miBiblio));
        System.out.println("\nUtilizando el método equals con la copia de seguridad y la biblioteca de la que no ha sido clonada (Debe dar false): " + copiaSeguridad.equals(miTeca));

        System.out.println("\nMostrando el hash de 'Principito'" + libro3.hashCode());

        System.out.println("\nUtilizando el método toString de enciclopedia que utiliza a su vez el toString del padre: " + enciclo2);
    }
}
```

2. EJECUCIÓN

```
Añadiendo el Quijote: true
Añadiendo el Quijote (a una bliblioteca distinta): true
Añadiendo Principito: true
Añadiendo Inventando inventos: true
Añadiendo Britannica: true
Añadiendo Botánica: true
Intentando añadir Quijote otra vez (Debe dar false): false

--- Catálogo de la Biblioteca Biblioteca Municipal ---
Estamos leyendo el libro: El Quijote, publicado en 1605-01-01, con ISBN 978-123456 y número de páginas 800
Estamos leyendo el libro: Inventando inventos, publicado en 2026-02-03, con ISBN 978-555555 y número de páginas 9000
Estamos leyendo la enciclopedia: Enciclopedia Britannica, fecha de publicación 2000-05-20 y volumenes: [1,2,3,4,5,6,7,8,9,10,11,12]

--- Catálogo de la Biblioteca Biblioteca estatal ---
Estamos leyendo el libro: El Quijote, publicado en 1605-01-01, con ISBN 978-123456 y número de páginas 800
Estamos leyendo el libro: Pincipito, publicado en 2001-12-13, con ISBN 978-758955 y número de páginas 150
Estamos leyendo la enciclopedia: Enciclopedia Botánica, fecha de publicación 2005-06-10 y volumenes: [1,2,3,4,5,6,7,8,9]

Buscando 'Inventando inventos'. Posición esperada [1]: 1

Eliminando 'Inventando inventos': true

Añadiendo 'Inventando inventos': true

--- Catálogo de la Biblioteca Biblioteca Municipal ---
Estamos leyendo el libro: El Quijote, publicado en 1605-01-01, con ISBN 978-123456 y número de páginas 800
Estamos leyendo la enciclopedia: Enciclopedia Britannica, fecha de publicación 2000-05-20 y volumenes: [1,2,3,4,5,6,7,8,9,10,11,12]
Estamos leyendo el libro: Inventando inventos, publicado en 2026-02-03, con ISBN 978-555555 y número de páginas 9000

Imprimiendo copia a continuacion

--- Catálogo de la Biblioteca Biblioteca Municipal ---
Estamos leyendo el libro: El Quijote, publicado en 1605-01-01, con ISBN 978-123456 y número de páginas 800
Estamos leyendo la enciclopedia: Enciclopedia Britannica, fecha de publicación 2000-05-20 y volumenes: [1,2,3,4,5,6,7,8,9,10,11,12]
Estamos leyendo el libro: Inventando inventos, publicado en 2026-02-03, con ISBN 978-555555 y número de páginas 9000
```

```
Utilizando el método equals con la copia de seguridad y la biblioteca de la que ha sido clonada (Debe dar true): true

Utilizando el método equals con la copia de seguridad y la biblioteca de la que no ha sido clonada (Debe dar false): false

Mostrando el hash de 'Principito'-976160073

Utilizando el método toString de enciclopedia que utiliza a su vez el toString del padre: Publicación {titulo= 'Enciclopedia Botánica', fecha= 2005-06-10} -> Enciclopedia{vols= 9}
```