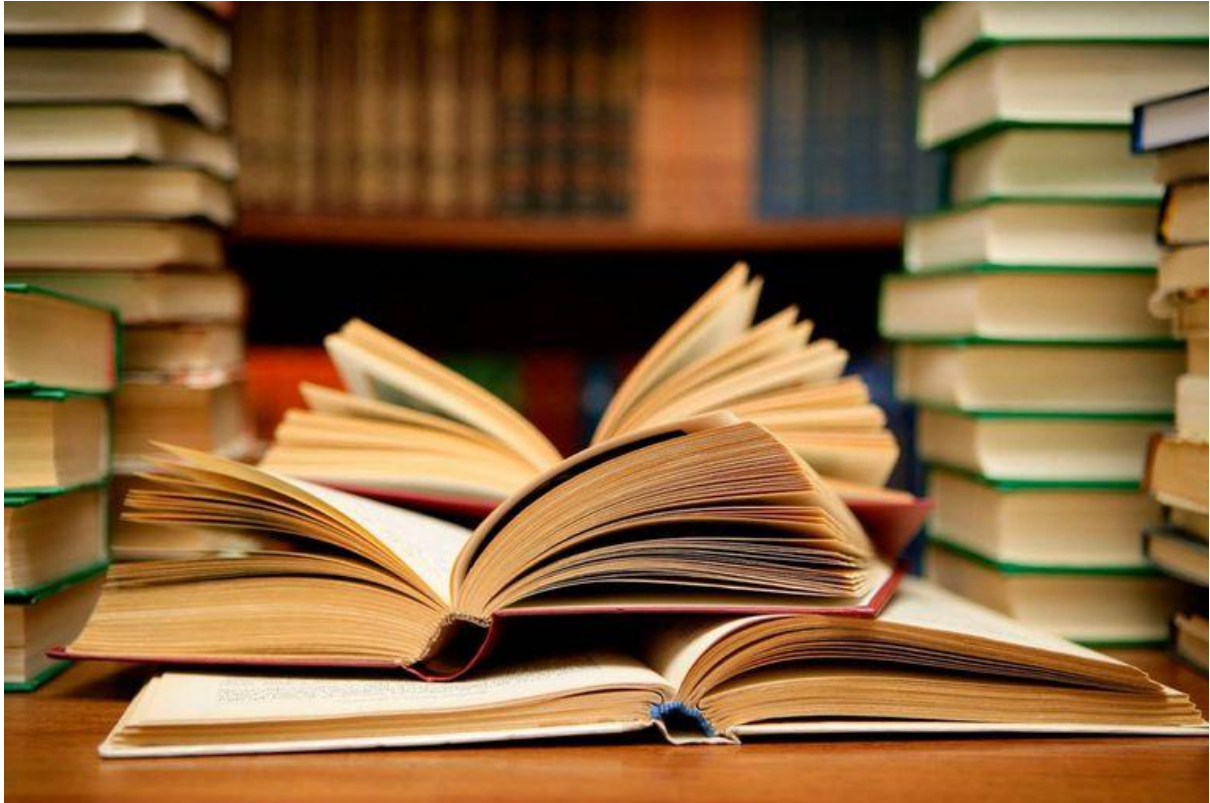


# **M03: Programación orientada a objetos**

## **Biblioteca Tierra Alta**



Emma Rueda Hinojosa  
Adrián Menéndez Fernández  
2021-2022

## Requerimientos para la app

La Biblioteca Tierra Alta ha contactado con nuestra empresa porque quiere crear una aplicación de escritorio basada en Java, que permita mejorar la gestión de las operaciones más comunes que se realizan en esta pequeña biblioteca que da servicio a los pueblos de la comarca de la Tierra Alta.

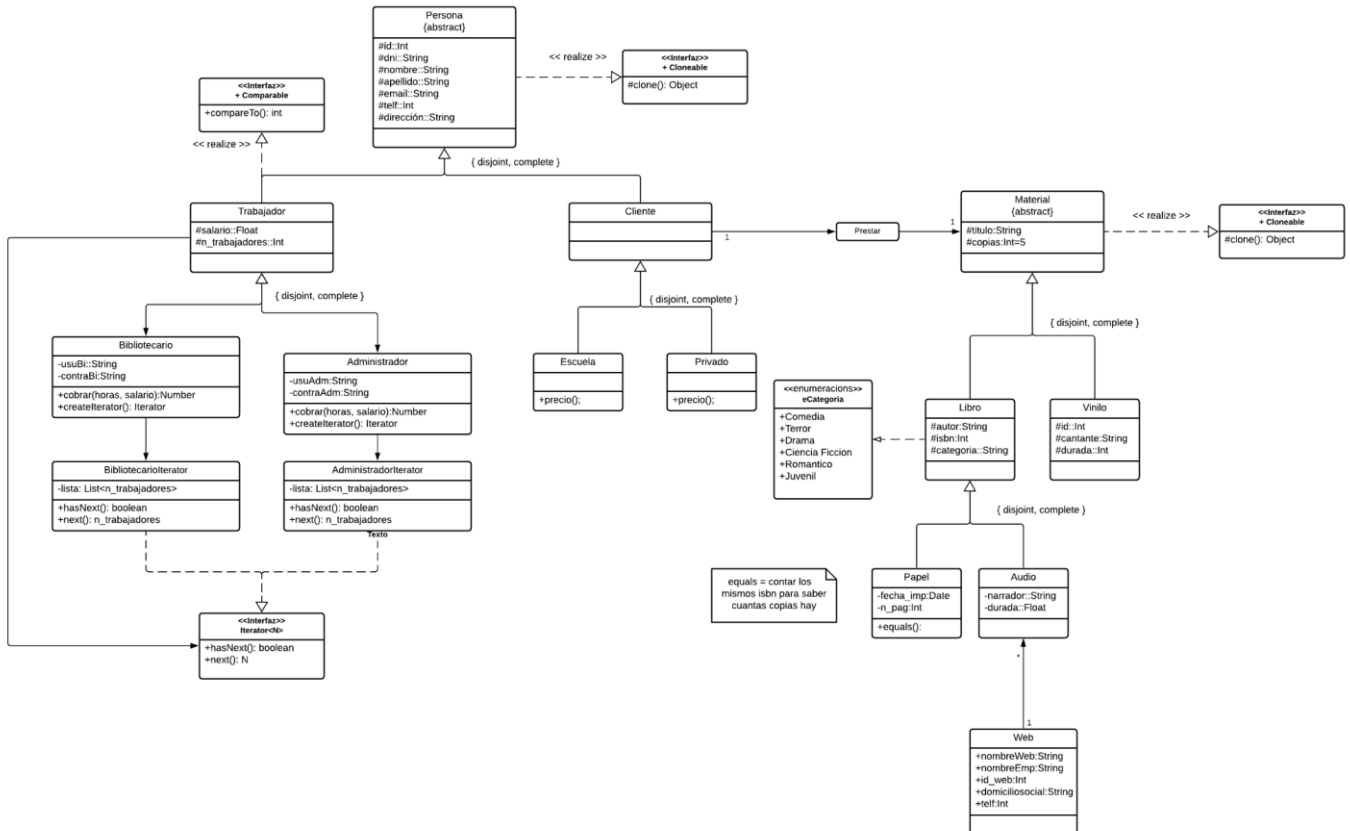
### Condiciones de la app:

- Existen dos tipos de productos: libros (ya sean en papel o en audiolibro) y discos de vinilos.
- Existen dos tipos de clientes: cliente privado y cliente escuela de música.
- Los clientes privados sólo pueden solicitar libros, tanto en papel como en audiolibro, mientras que las escuelas de música sólo pueden pedir discos de vinilo.
- Sólo se puede pedir un producto, es decir, hasta que los clientes no hagan la devolución, este no podrá solicitar otro producto.
- Los audiolibros son suministrados por una web, de la cual queremos almacenar los siguientes datos: nombre de la web, nombre de la empresa propietaria, identificador de la web en el registro mercantil, el domicilio social y el teléfono de contacto. Una web puede suministrar más de un audiolibro, y uno audiolibro es suministrado sólo por una web.
- Por otro lado, la biblioteca necesita también almacenar y gestionar información de sus trabajadores. La aplicación debe tener dos tipos de usuarios:
  - Administrador: lleva a cabo cualquier tarea dentro de la aplicación (usuario admin y contraseña admin).
  - Bibliotecario: encargado de gestionar los préstamos de la biblioteca (usuario bibliotecario y contraseña bibliotecario).
- Nuestra empresa deberá determinar qué información debe tener en cuenta para desarrollar la aplicación. La biblioteca ofrece un listado de ejemplos:
  - Clientes: dni, nombre, apellido, email, etc.
  - Libros: título, isbn, autores, la cantidad de libros del mismo tipo, etc.
  - Libros en papel: fecha de impresión, cantidad de hojas, etc.

## **Clases, objetos y atributos:**

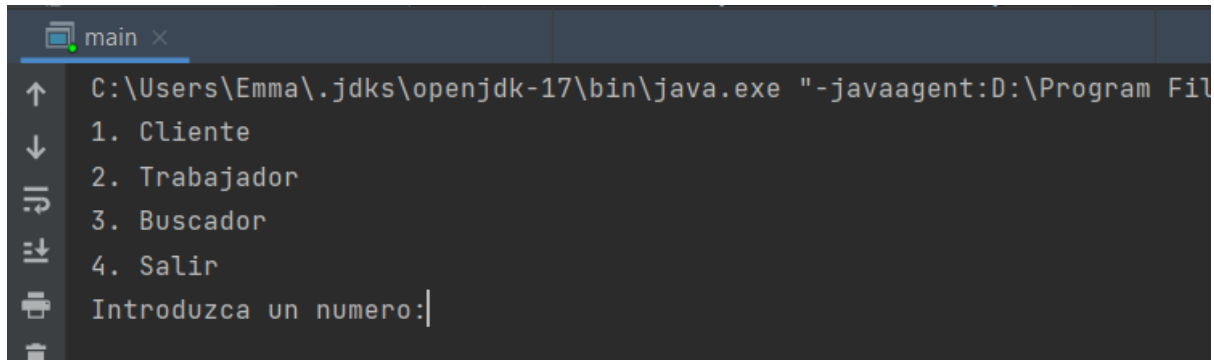
- Persona: id, dni, nombre, apellido, email, telf, dirección.
- Trabajador: id, dni, nombre, apellido, email, telf, dirección, salario.
- Bibliotecario: id, dni, nombre, apellido, email, telf, dirección, salario, usuBi, contraBi.
- Administrador: id, dni, nombre, apellido, email, telf, dirección, salario, usuAdm, contraAdm.
- Cliente: id, dni, nombre, apellido, email, telf, dirección.
- Escuela: id, dni, nombre, apellido, email, telf, dirección.
- Privado: id, dni, nombre, apellido, email, telf, dirección.
- Material: Titulo, copias.
- Libro: título, copias, autor, isbn, categoría.
- Vinilo: título, copias, id, cantante, durada.
- Papel: título, copias, autor, isbn, categoría, fecha\_imp, n\_pag.
- Audio: título, copias, autor, isbn, categoría, narrador, durada.
- Web: nombreWeb, nombreEmp, id\_web, domiciliosocial, telf.

# UML: Unified Modeling Language

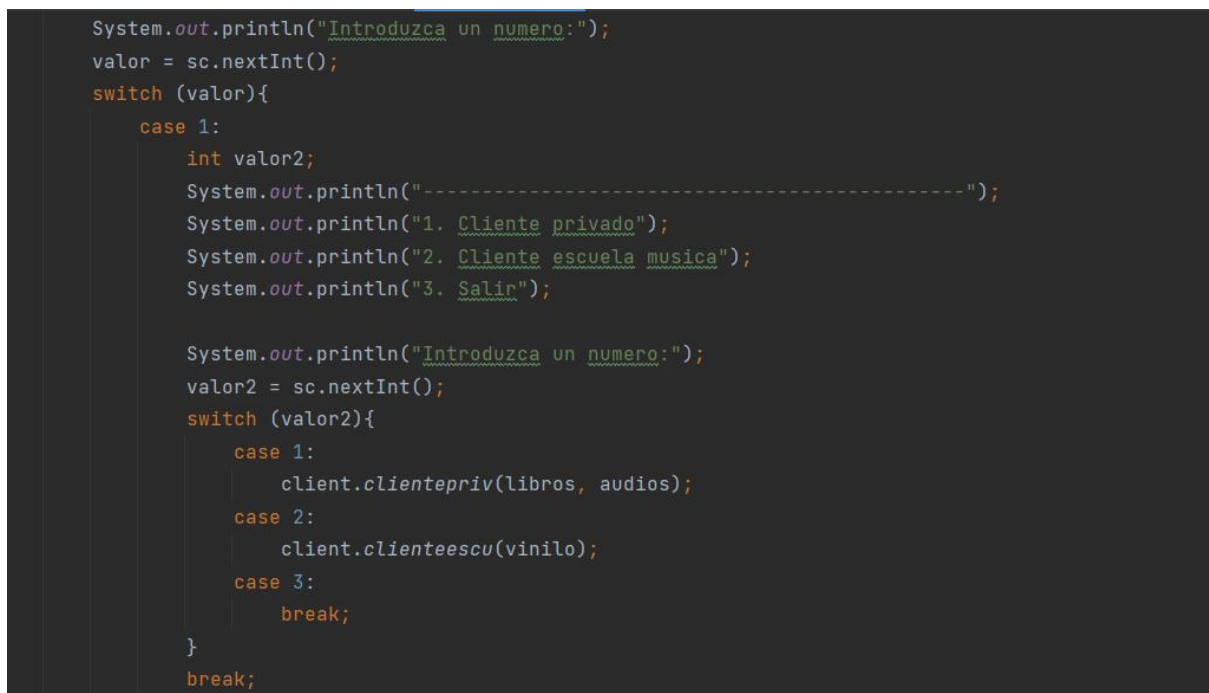


## Demostraciones de código:

La creación de nuestro menú principal:



```
main x
C:\Users\Emma\.jdk\openjdk-17\bin\java.exe "-javaagent:D:\Program Fil
1. Cliente
2. Trabajador
3. Buscador
4. Salir
Introduzca un numero:|
```



```
System.out.println("Introduzca un numero:");
valor = sc.nextInt();
switch (valor){
    case 1:
        int valor2;
        System.out.println("-----");
        System.out.println("1. Cliente privado");
        System.out.println("2. Cliente escuela musica");
        System.out.println("3. Salir");

        System.out.println("Introduzca un numero:");
        valor2 = sc.nextInt();
        switch (valor2){
            case 1:
                client.clientepriv(libros, audios);
            case 2:
                client.clienteescu(vinilo);
            case 3:
                break;
        }
        break;
}
```

Mostrar listado de libros, audiolibros, vinilos siguiendo la misma lógica:

```
1. Listado libros
2. Listado audio libros
3. Prestar
4. Sortir

titol='Harry Potter 1', autor='jk', genere='Ficcio', codigo='1a', disponible='true', paginas='700', ISBN='25636985'
titol='Harry Potter 2', autor='jk', genere='Ficcio', codigo='2b', disponible='true', paginas='550', ISBN='74696885'
-----
```

```
@Override
public String toString() {
    return "nom='" + nom + '\'' +
        ", Cognom='" + cognom + '\'' +
        ", DNI='" + DNI + '\'' +
        ", email='" + email + '\'';
}
```

```
@Override
public String toString() {
    return super.toString() + '\'' +
        ", paginas='" + numPagines + '\'' +
        ", ISBN='" + ISBN + '\'';
}
```

Añadir libros, audiolibros, vinilos siguiendo la misma lógica:

```
public static void AñadirLibroPapel() throws IOException {

    Scanner sc = new Scanner(System.in);
    String nombreP;
    System.out.println("-----");
    System.out.println("Titol: ");
    String titol = sc.nextLine();
    System.out.println("Autor: ");
    String autor = sc.nextLine();
    System.out.println("Genere: ");
    String genere = sc.nextLine();
    System.out.println("Codigo: ");
    String codigo = sc.nextLine();
    System.out.println("Disponible (Si/No)");
    String Disponible = sc.nextLine();
    System.out.println("Numero de pagines: ");
    int numPagines = sc.nextInt();
    System.out.println("ISBN: ");
    int ISBN = sc.nextInt();

    nombreP = String.valueOf(new llibrePaper(titol, autor, genere, codigo, Disponible, numPagines, ISBN));
    System.out.println("Titol: " + titol + " Autor: " + autor + " Genere: " + genere + " Codi: " + codigo + " Disponibil");
    llibrePaper.add(nombreP);

}
```

Modificar libros, audiolibros, vinilos siguiendo la misma lógica:

```
public static void modificarLibroPapel() throws IOException {

    Scanner sc = new Scanner(System.in);
    double valorP, nuevoValorP;
    int indiceP = 0;
    System.out.println("Valor a modificar: ");
    valorP = sc.nextDouble();
    llibrePaper.indexOf(valorP);
    if (indiceP != -1) {
        System.out.println("Nuevo valor: ");
        nuevoValorP = sc.nextDouble();
        llibrePaper.set(indiceP, nuevoValorP);
    } else {
        System.out.println("Dato no encontrado");
    }

}
```

Eliminar libros, audiolibros, vinilos siguiendo la misma lógica:

```
public static void eliminarLibroPapel() throws IOException {

    Scanner sc = new Scanner(System.in);
    double valorP;
    int indiceP = 0;
    System.out.println("Valor a eliminar: ");
    valorP = sc.nextDouble();
    llibrePaper.indexOf(valorP);
    if (indiceP != -1) {
        llibrePaper.remove(indiceP);
        System.out.println("Dato eliminado");
    } else {
        System.out.println("Dato no encontrado");
    }
}
```

Comprobar que existe la escuela y si existe que muestre la cantidad de puntos del carnet:

```
public static void prestarLibro(ArrayList libros, ArrayList clientes, ArrayList reservaLibros, String ISBN, String nif) {
    ArrayList isUsuario = false, isLibro = false, isPrestadoLibro = false, isUsuarioConPrestamo = false, tienePuntos = false;
    for (int i = 0; i < clientes.size(); i++) {
        if (clientes.get(i) instanceof CPrivado) {
            if (((CPrivado) clientes.get(i)).getNif().equals(nif)) {
                isUsuario = true;
                if (((CPrivado) clientes.get(i)).getCarnet() > 0) {
                    tienePuntos = true;
                }
            }
        }
    }
}
```

Comprobamos que existe el libro:

```
for (int i = 0; i < libros.size(); i++) {
    if (((Libro) libros.get(i)).getISBN().equals(ISBN)) {
        isLibro = true;
        if (libros.get(i) instanceof LPapel) isLibroPapel = true;
        if (libros.get(i) instanceof AudioLibro) isAudioLibro = true;
    }
}
```



Creamos la función de devolver el libro:

```
public static void devolverLibro(ArrayList clientes, ArrayList reservaLibros, String nif) {
    for (int i = 0; i < reservaLibros.size(); i++) {
        if (((Reserva) reservaLibros.get(i)).getNif().equals(nif)) {
            Date dataDevolucio = new Date();
            long diferenciaMilesimas = ((Reserva) reservaLibros.get(i)).getDataFi().getTime() + dataDevolucio.getTime();
            int diferenciaDias = (int) TimeUnit.MILLISECONDS.toDays(diferenciaMilesimas);
            if (diferenciaDias < 0) {
                for (int c = 0; c < clientes.size(); c++) {
                    if (clientes.get(c) instanceof CPrivado) {
                        if (((CPrivado) clientes.get(c)).getNif().equals(nif)) {
                            int puntosResta = diferenciaDias * -1;
                            ((CPrivado) clientes.get(c)).restarPuntos(puntosResta);
                        }
                    }
                }
            }
        }
    }
}
```

Comprobamos que el libro que solicitamos esta disponible:

```
if (isUsuario && isLibro && tienePuntos) {
    for (int i = 0; i < reservaLibros.size(); i++) {
        if (((Reserva) reservaLibros.get(i)).getISBN().equals(ISBN)) {
            isPrestadoLibro = true;
        }
    }
}
```

Comprobamos que el usuario no tiene nada en préstamo:

```
for (int i = 0; i < reservaLibros.size(); i++) {
    if (((Reserva) reservaLibros.get(i)).getNif().equals(nif)) {
        isUsuarioConPrestamo = true;
    }
}
```