



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorios de docencia

# Laboratorio de Computación Salas A y B

*Profesor(a):* René Adrián Dávila Pérez

*Asignatura:* Programación Orientada a Objetos

*Grupo:* 7

*No de Práctica(s):* Proyecto 1

*Integrante(s):* 322044339

322094152

322114461

425093384

322150984

*No. de brigada:* 6

*Semestre:* 2026-1

*Fecha de entrega:* 24 de septiembre de 2025

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Marco Teórico</b>	<b>2</b>
2.1. Array List . . . . .	2
2.2. Clases . . . . .	3
2.3. Objetos . . . . .	3
2.4. Constructores . . . . .	3
<b>3. Desarrollo</b>	<b>3</b>
3.1. Proyecto.java . . . . .	4
3.2. Libro.java . . . . .	4
3.3. Libreria.java . . . . .	4
<b>4. Resultados</b>	<b>5</b>
4.1. Ingresar libros . . . . .	5
4.2. Imprimir libros . . . . .	6
4.3. Eliminar libros . . . . .	6
4.4. Segunda impresión . . . . .	7
4.5. Opción no válida . . . . .	7
4.6. Salida . . . . .	8
<b>5. Conclusiones</b>	<b>8</b>
<b>6. Referencias bibliográficas</b>	<b>9</b>

# 1. Introducción

Imaginemos que somos dueños de una librería: ahí los libros se acumulan, se venden y algunos se pierden en el desorden. El manejo de todos a la larga se vuelve inviable por la cantidad que son, ya que si no se lleva un registro claro, podemos olvidar o perder información sobre los libros, como sus nombres o sus autores. Este problema, que parece trivial a pequeña escala, se vuelve importante cuando la colección aumenta, pues más libros implican la necesidad de más tiempo buscando información y menos tiempo atendiendo a los clientes. Para darle solución, nos podemos apoyar en la tecnología, aunque hay que tener presente que si no contamos con una forma rápida y confiable de agregar, eliminar o consultar los libros, esto nos va a limitar demasiado; por ello, para gestionarlos podemos emplear colecciones que, con las herramientas adecuadas, nos permitirán ofrecer un servicio de calidad.

La motivación de este proyecto la encontramos en que, al emular digitalmente el manejo de una librería, nos desligamos de los errores humanos y ganamos orden, velocidad y claridad; aunque lo que más importa es que este tipo de sistemas prepara las bases para aplicaciones más complejas, como, por ejemplo, las búsquedas avanzadas.

El propósito de este proyecto es crear una aplicación en Java que posibilite gestionar de forma intuitiva y eficaz un conjunto de libros. Para conseguirlo, vamos a generar clases como Librería y Libro, que se dedicarán a la creación de objetos y al manejo de información por medio de estructuras de datos como ArrayList para guardar los libros, así como acciones como añadir, quitar o mostrar libros. Con esto, se pretende evidenciar cómo un fragmento de código puede ser útil para algo cotidiano y aplicar los principios de la Programación Orientada a Objetos.

## 2. Marco Teórico

### 2.1. Array List

La clase Array List permite el uso de todas las operaciones con listas conocidas y permite introducir todo tipo de elementos, incluido null, además, esta clase permite trabajar con el tamaño de la matriz en la que se almacenan los datos internamente mediante el uso de métodos.

Cada ArrayList tiene una capacidad relacionada con el tamaño de la matriz, esto quiere decir que la matriz es "dinámica" pues mientras se agregan elementos al ArrayList, aumenta su capacidad y por ende el tamaño de la matriz.

Hablando de gestión del tiempo, las operaciones "size", "isEmpty", "get", "set", "iterator" y "listIterator" son operaciones de tiempo constante, mientras que "add" es de orden  $n$ . [4]

## 2.2. Clases

Una clase es una abstracción de un modelo de la realidad que representa en un único elemento un conjunto de objetos, definen el comportamiento y la estructura de los objetos que se van a instanciar a partir de la clase. El ejemplo clásico al momento de tratar de explicar lo que es una clase, es verlo como un molde de galletas y las galletas que se crean a partir del molde serían los objetos. [1]

Entre algunos de los conceptos fundamentales de las clases están:

- Atributos: También conocidos como variables de instancia representan las características o propiedades de un objeto.
- Métodos: Estos representan el comportamiento de los objetos, son funciones que pueden realizar acciones o manipular los atributos de los objetos.

## 2.3. Objetos

Son una abstracción de una parte del mundo real, así como conceptos abstractos con características y comportamientos, cuentan con una estructura interna que combina variables, funciones y estructuras de datos. [3]

## 2.4. Constructores

Cada objeto de una clase debe ser creado de forma explícita por medio de un método especial que se denomina constructor, por convención, dicho método tiene el mismo nombre que la clase en la que se encuentra y no se le asocia un modo de acceso (es público).

En lenguajes como Java, se provee al programador de un constructor por defecto en cada clase además de permitir la creación de otros métodos constructores definidos por el usuario.

Aunque una clase pueda tener un número indefinido de métodos constructores, debe existir una forma de identificarlos entre uno y otro, esto se logra por medio de la lista de parámetros que recibe el constructor, con este cambio el lenguaje entiende que estamos haciendo referencia a diferentes tipos de métodos constructores[2].

## 3. Desarrollo

Para el proyecto retomamos conceptos vistos con anterioridad como lo son la creación de objetos, la comunicación entre los mismos, el recoger datos ingresados desde la línea de comandos y las estructuras HashTable, HashMap, y ArrayList, eligiendo un desarrollo directo y efectivo.

Se crearon tres documentos .java para la realización del proyecto, siendo estos Libro.java, Libreria.java y Proyecto.java.

### 3.1. Proyecto.java

Esta es la clase principal debido a que contiene al método main, pero antes de ello, como usaremos datos proporcionados por el usuario importamos la biblioteca `java.util.Scanner`. Ya dentro de la clase inicializamos a las variables de tipo `int` clave, opcion e indice en cero, a las cadenas autor y titulo, y a dos objetos, el primero de tipo `Scanner` y de nombre `e`, el segundo de tipo `Libreria` y de nombre `Libreria`.

Para un entorno más ameno creamos un menú con `do-while` y `switch`, el cual funcionaría siempre y cuando la opción no fuera 4, dado que la opción 1 se relaciona con la operación de ingresar un libro nuevo, la opción 2 se relacionaba a eliminar un libro existente y la opción tres imprimía los libros almacenados.

### 3.2. Libro.java

La parte más simple del programa y en cual no tuvimos muchas especulaciones más el camino predeterminado de hacer dos constructores. En este apartado nuestro objetivo es crear con los datos que llegan un objeto libro. Primero declaramos dentro de la clase a un `String` autor y un `String` titulo, además de una variable clave inicializada en cero. Después se elaboran dos constructores, uno por defecto sin parámetros por si se llegara a dar el caso de un libro vacío, aunque en este caso no, y un constructor que recibe como parámetros la clave, el título y el autor, usando la palabra reservada `.this` que distingue al atributo y al parámetro

### 3.3. Libreria.java

Esta fue la parte más complicada del programa y en la que más propusimos cambios para su implementación, porque primero teníamos la idea de usar un `HashMap` accediendo mediante clave o valor, sin embargo esto nos acarreó confusiones en un principio, por lo que decidimos usar la opción más simple, siendo este un `ArrayList` en el cual ordenaríamos nuestros objetos libro y los almacenaríamos en un `ArrayList` al cual accederíamos por índice.

Para ello, importamos la biblioteca de `java.util.ArrayList` para hacer uso de esta estructura, por lo que dentro de la clase el primer paso es crear un objeto libreria de tipo `ArrayList` que contendría objetos libro. Dentro de esta clase también se crean tres métodos vacíos, uno por cada operación solicitada para el proyecto.

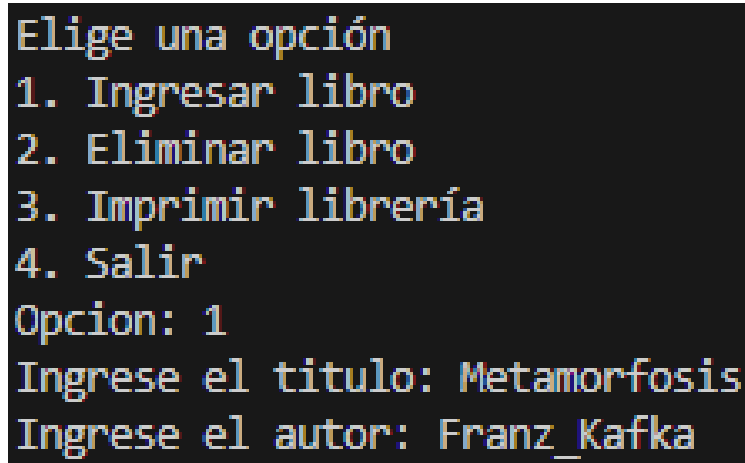
El primer método es agregarLibro, recibe como parámetro un objeto Libro, en este método nuestro objeto libreria mediante .add agrega un nuevo elemento Libro a su ArrayList, y se manda un mensaje de "Agregado." Para confirmar que la operación fue un éxito.

El segundo método es eliminarLibro que recibe como parámetro el índice del libro a eliminar, de manera similar, pero contraria a agregar libro, tomamos a nuestra librería y operamos sobre ella .remove con el parámetro del índice. Para decirle al usuario que la operación se efectuó, entonces le imprimimos un mensaje "Eliminado"

El tercer método es imprimirLibreria que no toma ningún parámetro. Primero imprimimos la línea "Clave | Título | Autor" para darle formato a nuestra impresión, después con la ayuda de un for each de cada libro en la libreria imprimimos con el siguiente formato (libro.clave) + " | " + libro.titulo + " | " + libro.autor. Dando un resultado simple pero entendible.

## 4. Resultados

### 4.1. Ingresar libros



```
Elige una opción
1. Ingresar libro
2. Eliminar libro
3. Imprimir librería
4. Salir
Opcion: 1
Ingresa el titulo: Metamorfosis
Ingresa el autor: Franz_Kafka
```

Figura 1: Ingreso de un libro

Se ejecuta nuestro proyecto y se selecciona la opción de añadir un libro, posteriormente se ingresan los datos del mismo; su título y su autor, mismas que se almacenan mediante el primer método de nuestro programa.

## 4.2. Imprimir libros

```
Elige una opción
1. Ingresar libro
2. Eliminar libro
3. Imprimir librería
4. Salir
Opcion: 3
Clave | Titulo | Autor
0 | La_dama_de_las_camelias | Alejandro_Dumas
1 | La_Iliada | Homero
2 | Metamorfosis | Franz_Kafka
```

Figura 2: Impresión de la lista de todos los libros

Una vez que almacenamos 3 libros diferentes, se selecciona la opción de imprimir los libros, mismos que se muestran mediante el tercer método, clasificados por clave, título y autor.

## 4.3. Eliminar libros

```
Elige una opción
1. Ingresar libro
2. Eliminar libro
3. Imprimir librería
4. Salir
Opcion: 2
Ingresa la clave a eliminar: 1
Eliminado.
```

Figura 3: Eliminación de un libro

Ahora se selecciona la opción de eliminar libros, se ingresa la clave del libro que se quiere eliminar y se elimina mediante el segundo método de nuestro programa.

#### 4.4. Segunda impresión

```
Elige una opción
1. Ingresar libro
2. Eliminar libro
3. Imprimir librería
4. Salir
Opcion: 3
Clave | Titulo | Autor
0 | La_dama_de_las_camelias | Alejandro_Dumas
2 | Metamorfosis | Franz_Kafka
```

Figura 4: Lista actualizada

Se vuelve a seleccionar la opción de imprimir la lista de los libros, solo que en esta ocasión ya no se muestra el libro que se eliminó anteriormente.

#### 4.5. Opción no válida

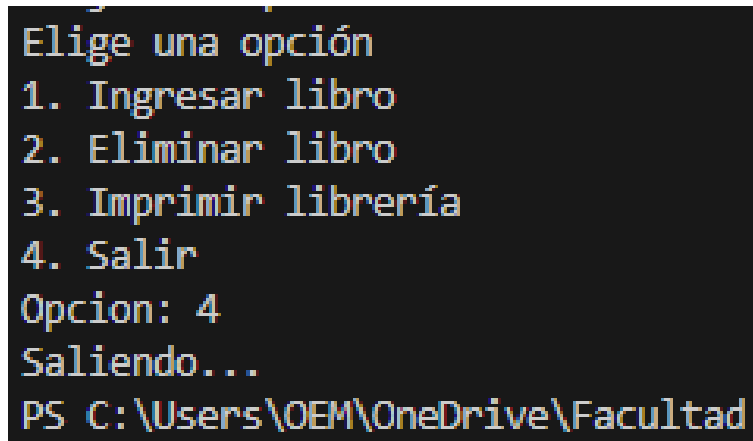
```
Elige una opción
1. Ingresar libro
2. Eliminar libro
3. Imprimir librería
4. Salir
Opcion: 7
Elija una opcion correcta
```

Figura 5: Mensaje de opción inválida

Se muestra un mensaje de invalidación cuando se quiere ingresar una opción que no está dentro de las que definimos en nuestro programa.



## 4.6. Salida



```
Elige una opción
1. Ingresar libro
2. Eliminar libro
3. Imprimir librería
4. Salir
Opcion: 4
Saliendo...
PS C:\Users\OEM\OneDrive\Facultad
```

Figura 6: Salida del programa

El programa muestra un mensaje de salida una vez que se selecciona la opción correspondiente y se termina la ejecución.

## 5. Conclusiones

El desarrollo de este proyecto permitió aplicar los conceptos fundamentales de la programación orientada a objetos vistos en este primer parcial: clases, objetos, constructores, métodos y distribución de responsabilidades. A través de la implementación de una librería digital sencilla, se dio uso a estructuras de datos/colecciones como **ArrayList** que facilitó la gestión de los libros y eficiencia para agregar, quitar e imprimir dichos datos.

La interacción entre las clases nos ayudó a reforzar la idea de modularidad, que cada componente tenga un propósito definido y que coopere en conjunto para solucionar un determinado problema. De este modo, el proyecto ilustra la aplicación práctica de los principios teóricos de este nuevo paradigma para construir nuevos códigos y paquetes propios para el desarrollo de software organizado y robusto en el futuro.

## 6. Referencias bibliográficas

### Referencias

- [1] Jorge López Blasco. *Introducción a POO en Java: Objetos y clases*. Accedido: 14 de septiembre de 2025. Oct. de 2023. URL: <https://openwebinars.net/blog/introduccion-a-poo-en-java-objetos-y-clases/>.
- [2] Yusneky Carballo. *Programación Orientada a Objetos (POO)*. Universidad Central de Venezuela, s.f.
- [3] EBAC. *Objeto en programación: qué es y para qué sirve, características, ejemplos y tipos*. Accedido: 14 de septiembre de 2025. Sep. de 2023. URL: <https://ebac.mx/blog/objeto-en-programacion>.
- [4] Oracle. *Class ArrayList<E>, Java Platform, Standard Edition 8 API Specification*. URL: <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html> (visitado 09-09-2025).