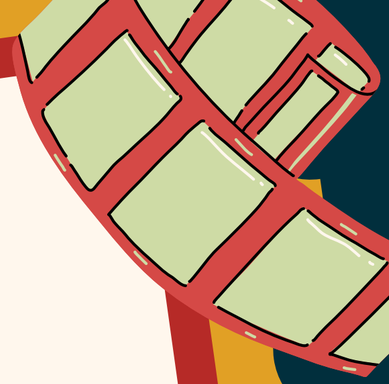
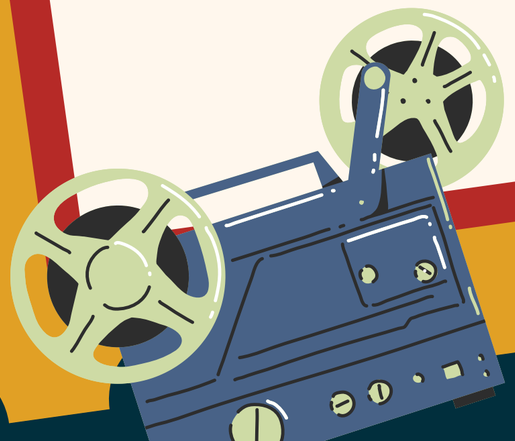
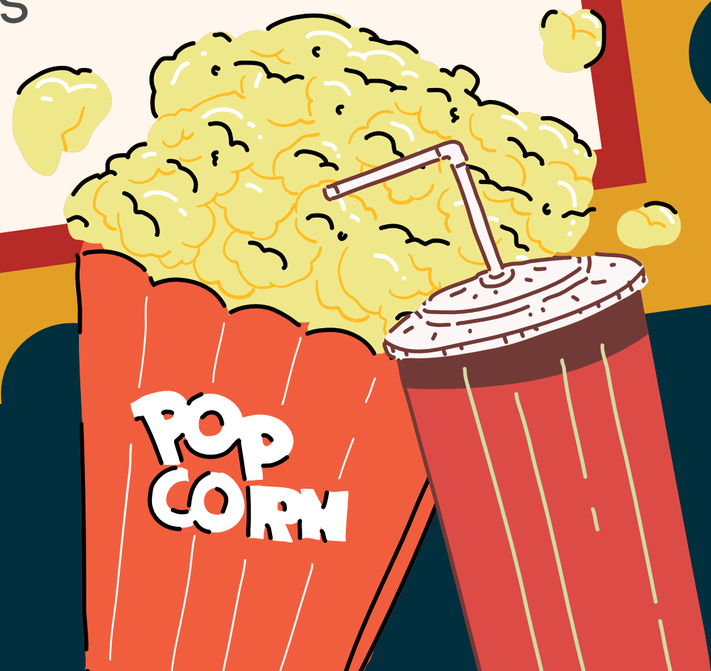
# PRIMER PORTAFOLIO

Amanda Padilla Leiva

Programación Orientada a Objetos







# OBJETIVO DEL PORTAFOLIO

Biblioteca de películas

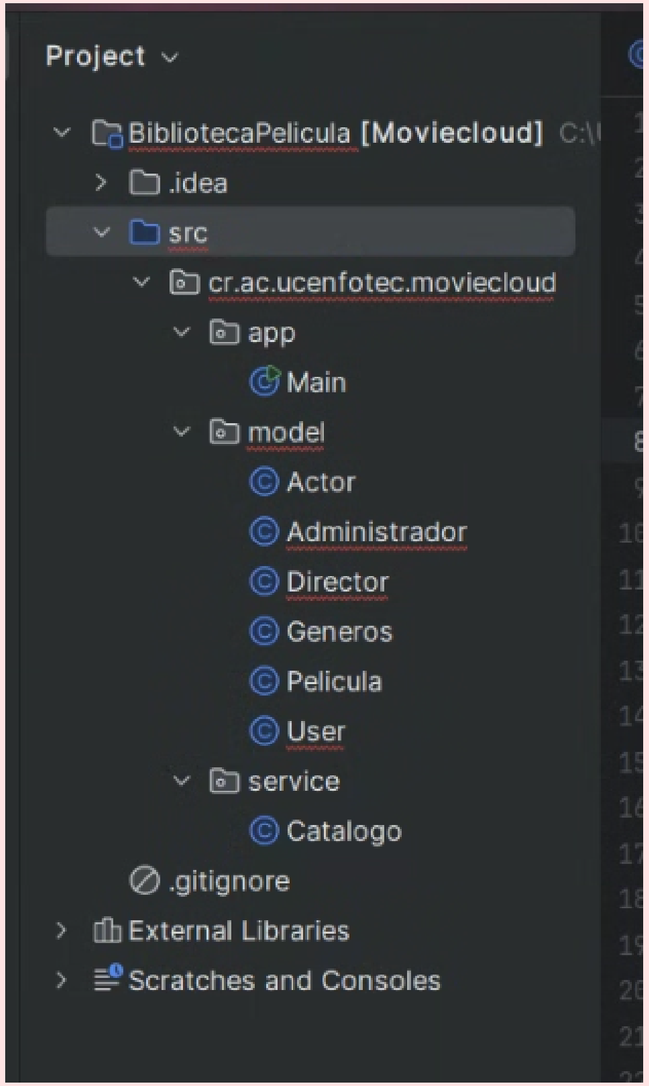
Una biblioteca donde hay un administrador que guarda películas, las clasifica por nombre, elenco, director, duración en minutos y género.
Y hay un usuario básico que puede ver las películas y agregarlas a favoritas.

# ESTRUCTURA

**Paquete model →** guarda las clases que representan cosas reales, como Película, Usuario, Actor.

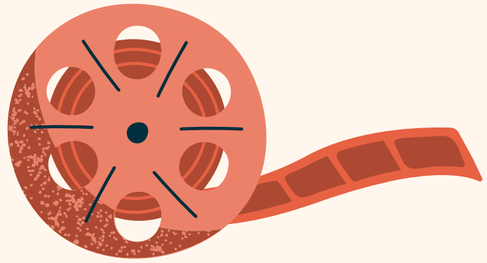
**Paquete service →** contiene la lógica o funciones del sistema, como buscar, filtrar o administrar datos. Y aqui es donde ponemos los arreglos que manejan los datos.

**Paquete app →** es donde está el punto de inicio del programa (tu clase Main), donde se ejecuta todo.



# CLASES

## CLASE PELÍCULA (MOVIE)



- id: String
- título: String
- año: int
- duraciónMinutos: int
- clasificación: ContentRating (enum: G, PG, PG13, R, NC17)
- géneros: Set<Género>
- elenco: Set<Actor>
- director: Director
- calificaciónPromedio: double

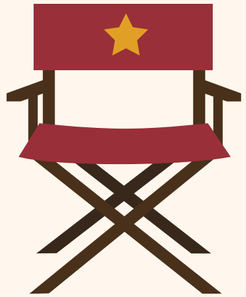
## CLASE GÉNERO (GENRE)

- nombre: String (ejemplo: Acción, Drama, Comedia, Terror, etc.)

# CLASES

## CLASE DIRECTOR

- id: String
- nombre: String
- películasDirigidas: Set<Película>



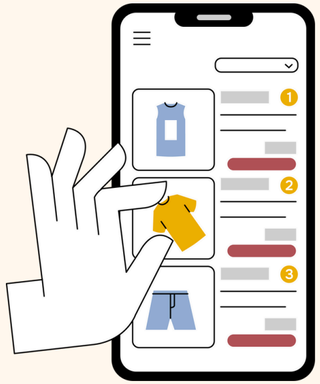
## CLASE ADMIN (HEREDA DE USER)

- username: String
- email: String
- privilegios: List<String> (ejemplo: agregar, eliminar, editar películas)

# CLASES

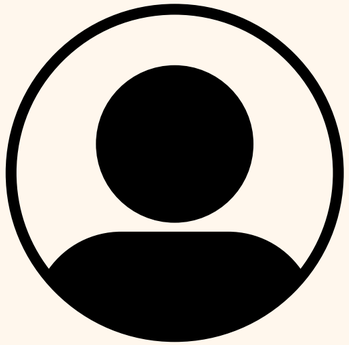
## CLASE CATÁLOGO

- películas: Map<String, Película>
- actores: Map<String, Actor>
- directores: Map<String, Director>

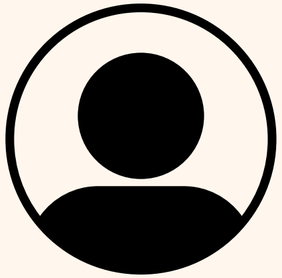


## CLASE USER (USUARIO)

- id: String
- username: String
- email: String
- favoritos: Set<Película>

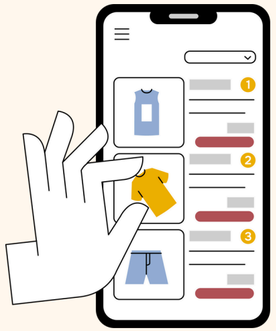


# MINI-DIAGRAMA

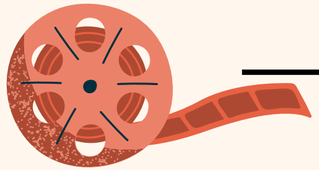


Usuario

Administrador



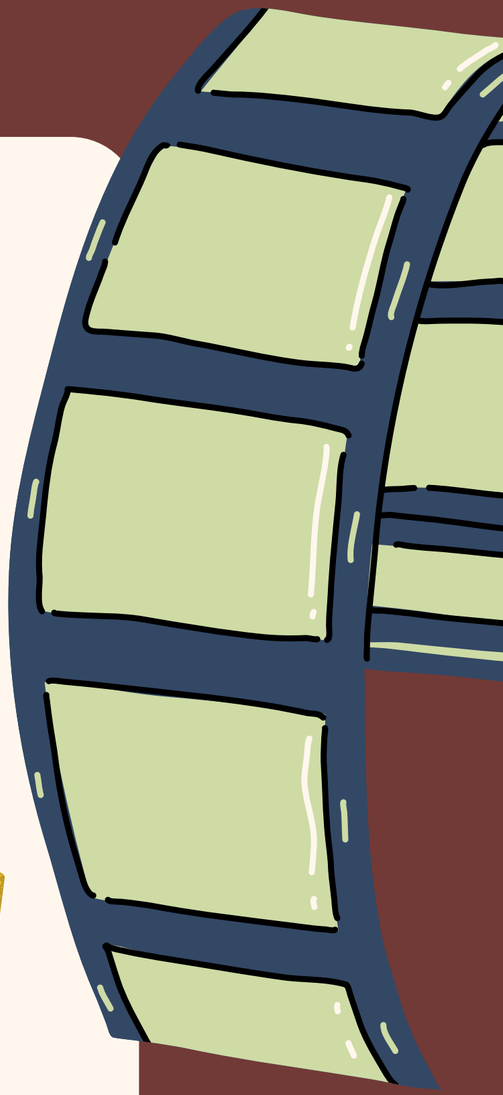
Catálogo



Película

Director

Género



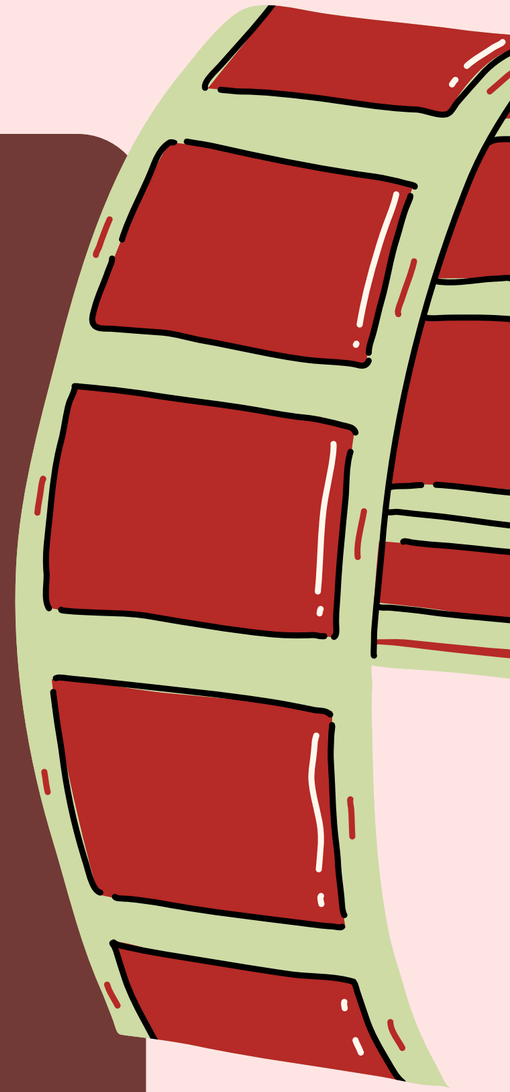
# PROCESO Y ESTRUCTURA

**C** CREATE (crear): Agregar nuevos elementos.

**R** READ (leer): Consultar o ver información.

**U** UPDATE (actualizar): Modificar datos existentes.

**D** DELETE (borrar): Borrar registros.



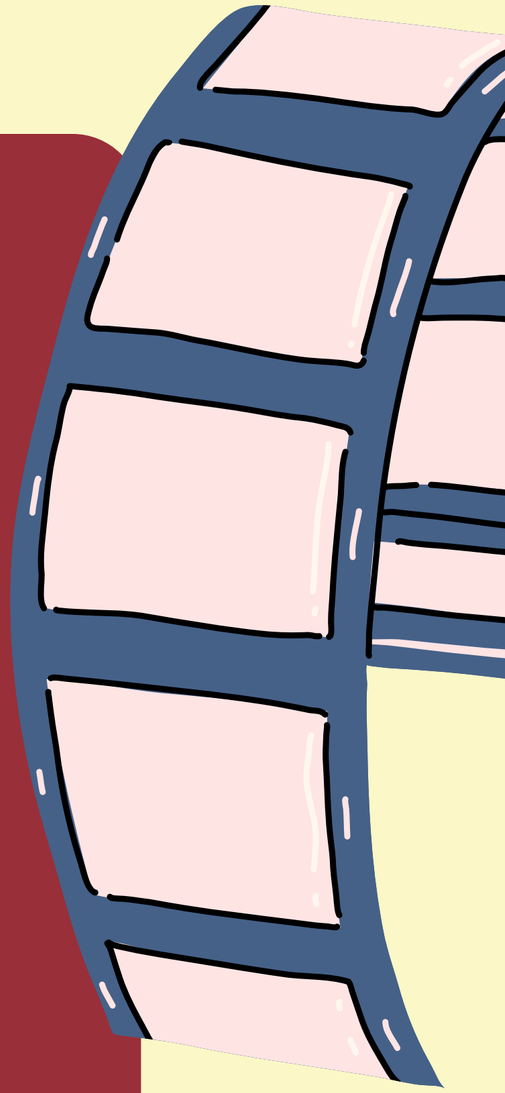
# CONCEPTOS

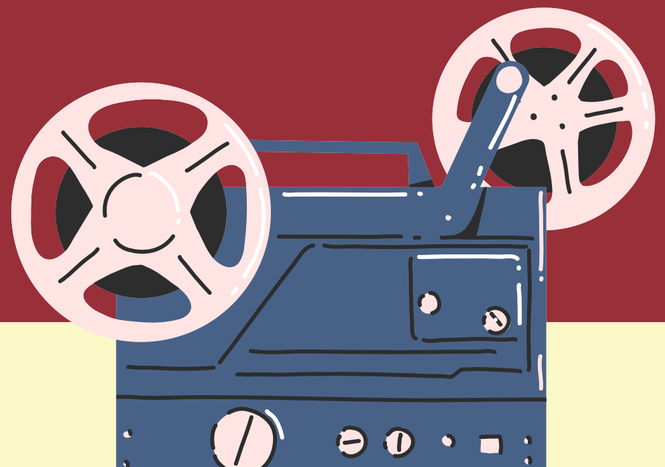
## Abstracción

```
public class Pelicula {  40 usages
    private String id;  4 usages
    private String titulo;  6 usages
    private int anio;  6 usages
    private int duracionMinutos;
    private String clasificacion;
    private Set<Genero> generos;
    private Set<Actor> elenco;  4 u
    private Director director;  4 u
```

## Encapsulamiento

```
private String id;  4 us
private String titulo;
```





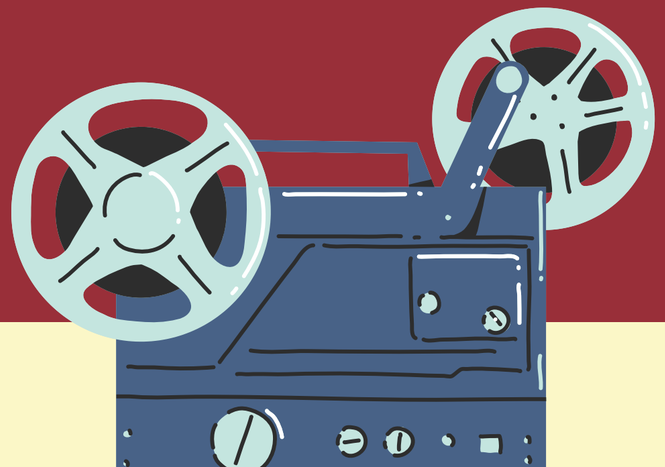
# CONCEPTOS

## Dependencia

```
static void actualizarPelicula() throws IOException {  1 usage
    if (totalPeliculas == 0) {
        System.out.println("No hay peliculas.");
        return;
    }
    String id = leerIdExistentePelicula( label: "ID a actualizar: ");
    int idx = indicePelicula(id);
    Pelicula p = peliculas[idx];

    String nuevoTitulo = leerConDefault( label: "Nuevo titulo (enter = mantener): "
    Integer nuevoAnio = leerEnteroConDefault( label: "Nuevo año (enter = mantener)
    Integer nuevaDur = leerEnteroConDefault( label: "Nueva duración (min, enter =
            p.getDuracionMinutos(),  min: 1,  max: 10000);

    System.out.print("Nueva clasificación (enter = mantener): ");
    String s = null;
    try { s = in.readLine(); } catch (IOException ignored) {}
    String nuevaClas = (s == null || s.trim().isEmpty()) ? p.getClasificacion()

    p.setTitulo(nuevoTitulo);
    p.setAnio(nuevoAnio);
    p.setDuracionMinutos(nuevaDur);
    p.setClasificacion(nuevaClas);
```

## Modularidad

```
import cr.ac.ucenfotec.moviecloud.m
import cr.ac.ucenfotec.moviecloud.m
import cr.ac.ucenfotec.moviecloud.m
import cr.ac.ucenfotec.moviecloud.m
import cr.ac.ucenfotec.moviecloud.mo
```



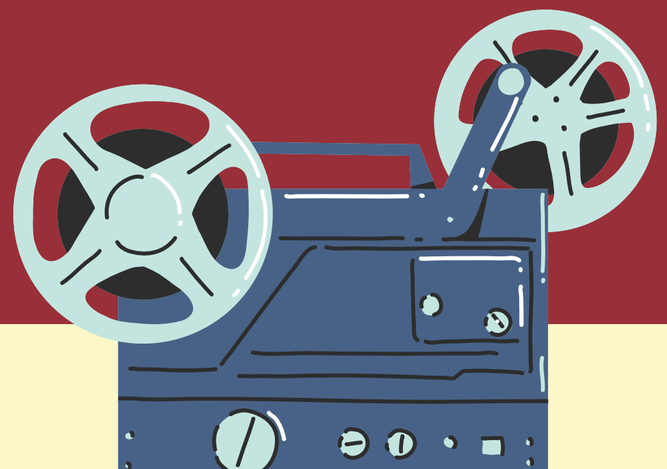
# CONCEPTOS

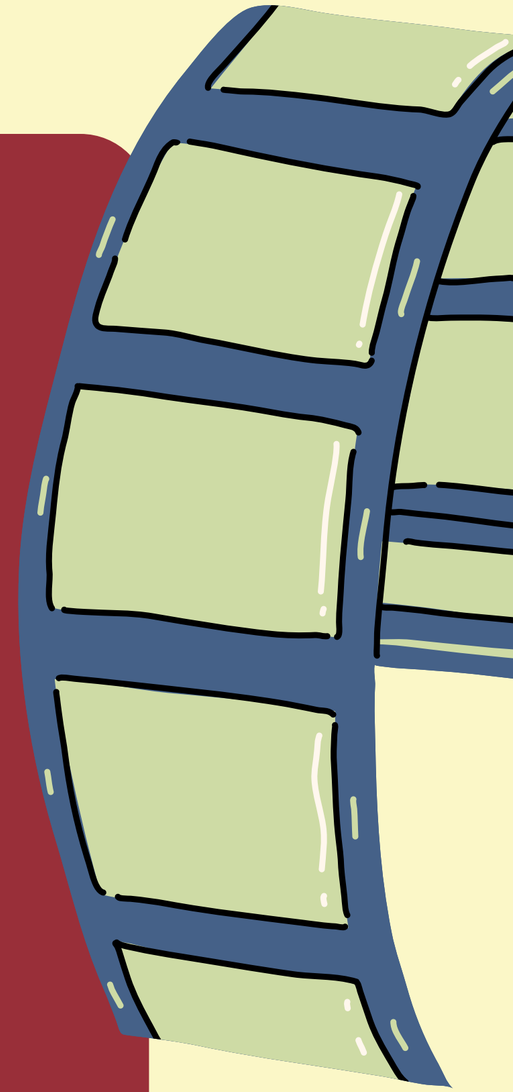
## Composición

```
System.out.print("¿Agregar sinopsis? (si/no): ");
String r = in.readLine();
if ("si".equalsIgnoreCase(r) || "si".equalsIgnoreCase(
    String sinopsis = leerNoVacio( label: "Sinopsis: ");
    p.setFicha(new Pelicula.Ficha(sinopsis));
}
```

## Agregación

```
  private Set<Genero> generos;
```






# APRENDIZAJE PERSONAL

En este portafolio aprendí a organizar el código de manera más eficiente y a aplicar los principios de la programación orientada a objetos junto con el modelo CRUD. Comprendí cómo se relacionan las distintas clases y cómo la modularidad contribuye a que el sistema sea claro, flexible y fácil de ampliar. Además, pude mejorar el manejo de la complejidad, implementando soluciones que facilitan futuras mejoras y permiten que el código sea más limpio y reutilizable.

# GRACIAS!