



Universidad de las Fuerzas Armadas

Informe N° 2:

Creación de Objetos y UML

Nombre y Apellido:

Sheyla Bernal

Programación Orientada a Objet

Docente:

LUIS ENRIQUE JARAMILLO MONTAÑO

Fecha:

8 de diciembre del 2024

CONTENIDO

Contenido

INTRODUCCION	3
DESARROLLO.....	4
• 5 objetos:.....	4
• Relación:.....	4
• Código:	5
• Pelicula:	5
• Cliente:	5
• Entrada:	6
• Sala:	6
• Cine:	7
• General	7
• Uml:.....	8
CONCLUSIONES	9
RECOMENDACIONES	9
BIOGRAFIA	10

INTRODUCCION

Un diagrama UML es una forma de visualizar sistemas y software utilizando el Lenguaje Unificado de Modelado (UML). Los ingenieros de software crean diagramas UML online para comprender los diseños, la arquitectura del código y la implementación propuesta de sistemas de software complejos.

Los diagramas UML ayudan a realizar un seguimiento de las relaciones y jerarquías entre líneas de código importantes.

Un diagrama de objetos se enfoca en los atributos de un conjunto de objetos y cómo esos objetos se relacionan entre sí.

Por ejemplo, en el siguiente diagrama de objetos, las tres cuentas bancarias están ligadas al banco mismo. Los títulos de clase muestran el tipo de cuentas (ahorros, corriente y tarjeta de crédito) que un cliente dado podría tener con este banco en particular. Los atributos de clase son diferentes para cada tipo de cuenta.

Por ejemplo, el objeto de tarjeta de crédito tiene un límite de crédito, mientras que las cuentas de ahorros y corriente tienen tasas de interés.

DESARROLLO

5 objetos:

- **Cine:** Administrara todas las películas que tienen
- **Película:** Representará todas las películas disponibles en el cine, contendrá título, autor, año publicación
- **Entrada:** Registran las entradas compradas
- **Sala:** Lugar donde se proyectarán las películas
- **Cliente:** Se les presentara una lista de las películas y ellos podrán comprar entradas

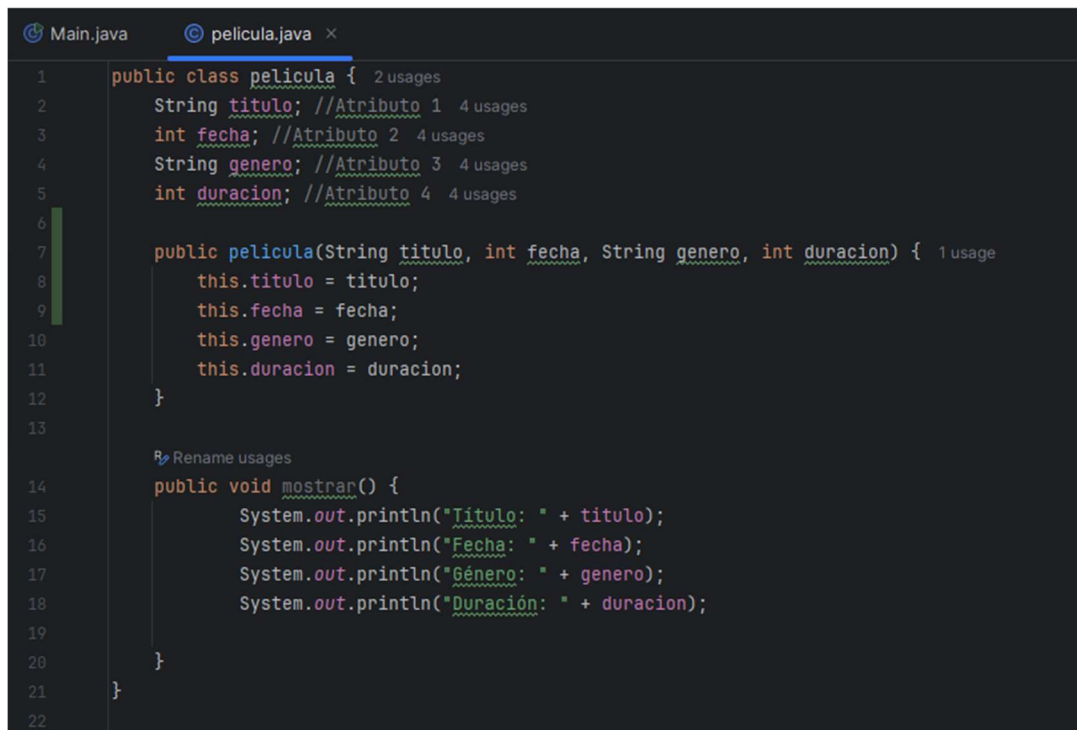
Relación:

- **Cine:** Se relaciona con películas ya que maneja todo el catálogo de las películas
- **Película:** Se relaciona con **Sala** ya que hay se proyectan todas las películas
- **Cliente:** Se relaciona con **Entrada** y **Sala** ya que el cliente si no tiene la entrada no pasa a la sala y el cliente tiene que llevar consigo la entrada
- **Entrada:** Se relaciona con **Película** y **Sala** depende de la entrada para poder pasar a la sala y ver la película
- **Sala:** Se relaciona con **Película** para que los clientes puedan visualizar la película

Código:

Vamos a realizar el código en java

- Pelicula:

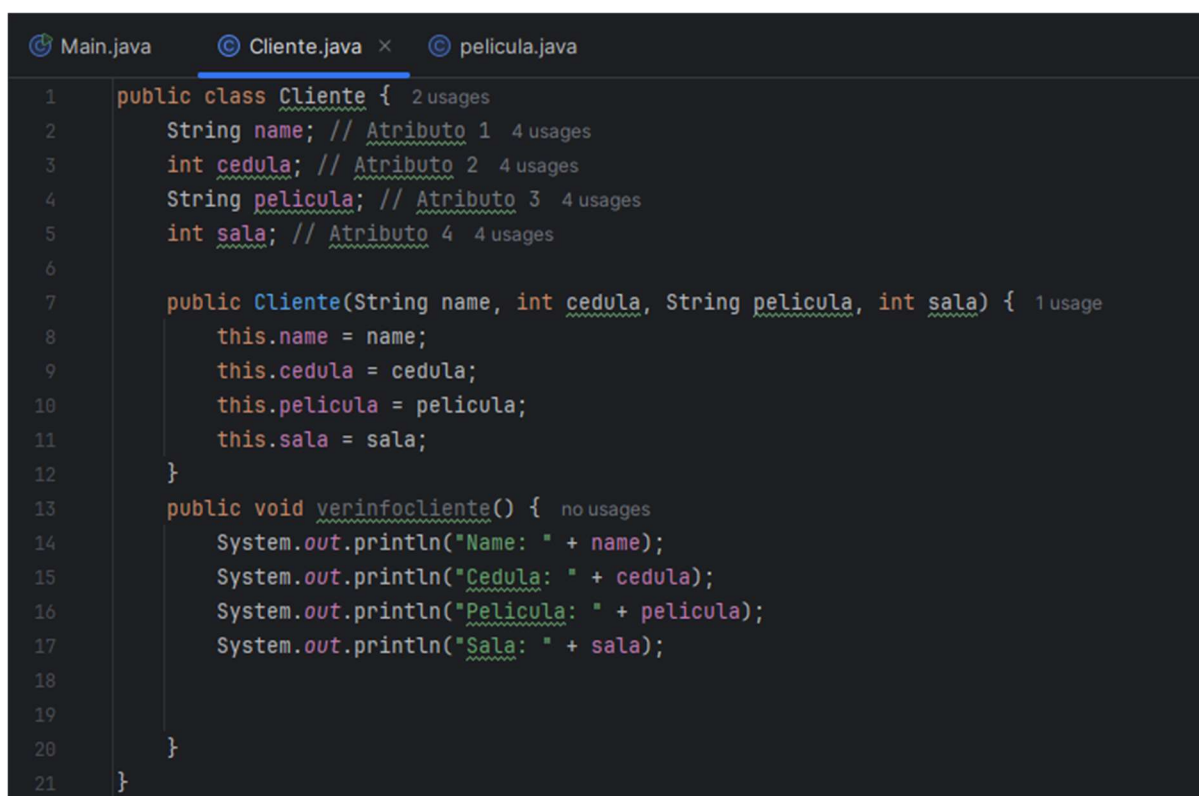


```

1  public class pelicula { 2 usages
2      String titulo; //Atributo 1 4 usages
3      int fecha; //Atributo 2 4 usages
4      String genero; //Atributo 3 4 usages
5      int duracion; //Atributo 4 4 usages
6
7      public pelicula(String titulo, int fecha, String genero, int duracion) { 1 usage
8          this.titulo = titulo;
9          this.fecha = fecha;
10         this.genero = genero;
11         this.duracion = duracion;
12     }
13
14     public void mostrar() {
15         System.out.println("Titulo: " + titulo);
16         System.out.println("Fecha: " + fecha);
17         System.out.println("Género: " + genero);
18         System.out.println("Duración: " + duracion);
19     }
20 }
21
22

```

- Cliente:

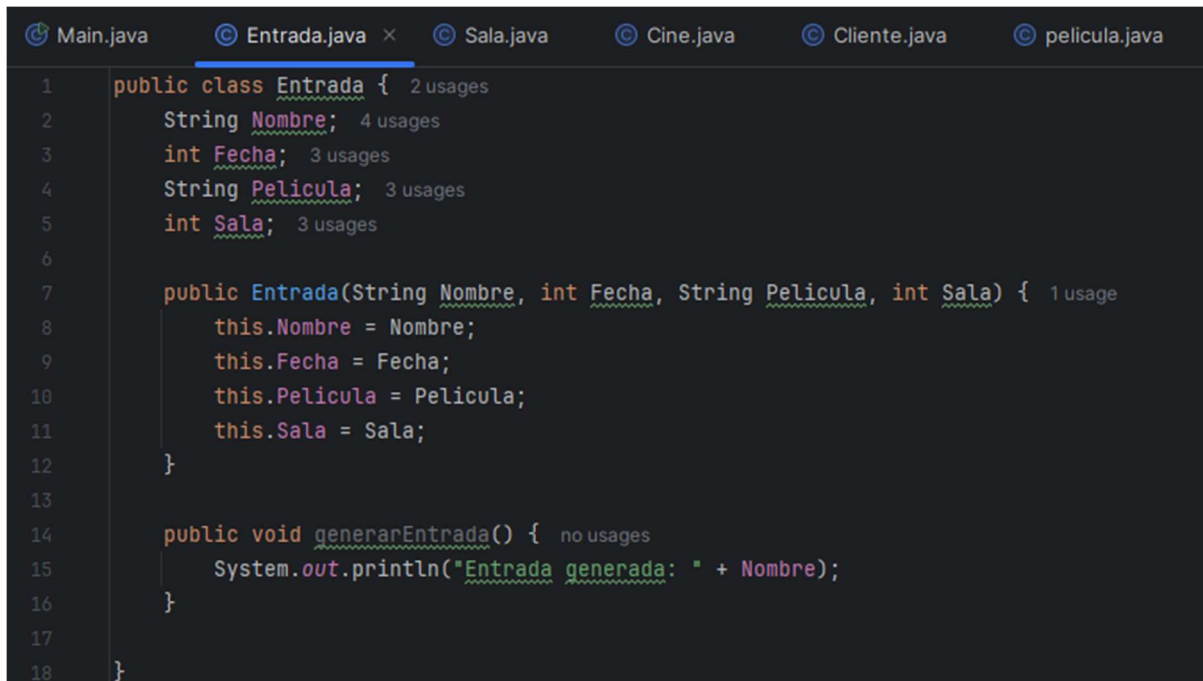


```

1  public class Cliente { 2 usages
2      String name; // Atributo 1 4 usages
3      int cedula; // Atributo 2 4 usages
4      String pelicula; // Atributo 3 4 usages
5      int sala; // Atributo 4 4 usages
6
7      public Cliente(String name, int cedula, String pelicula, int sala) { 1 usage
8          this.name = name;
9          this.cedula = cedula;
10         this.pelicula = pelicula;
11         this.sala = sala;
12     }
13
14     public void verinfocliente() { no usages
15         System.out.println("Name: " + name);
16         System.out.println("Cedula: " + cedula);
17         System.out.println("Pelicula: " + pelicula);
18         System.out.println("Sala: " + sala);
19     }
20 }
21
22

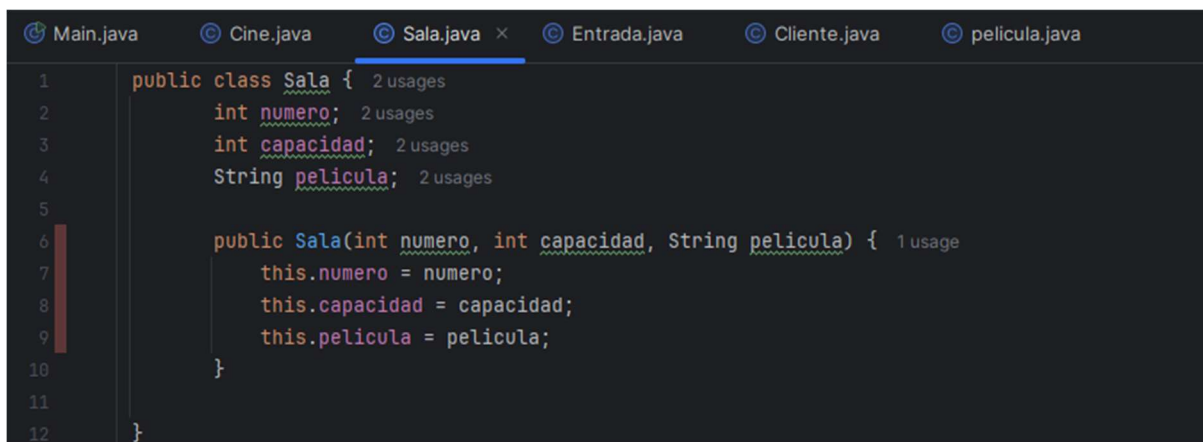
```

- Entrada:



```
1 public class Entrada { 2 usages
2     String Nombre; 4 usages
3     int Fecha; 3 usages
4     String Pelicula; 3 usages
5     int Sala; 3 usages
6
7     public Entrada(String Nombre, int Fecha, String Pelicula, int Sala) { 1 usage
8         this.Nombre = Nombre;
9         this.Fecha = Fecha;
10        this.Pelicula = Pelicula;
11        this.Sala = Sala;
12    }
13
14    public void generarEntrada() { no usages
15        System.out.println("Entrada generada: " + Nombre);
16    }
17
18 }
```

- Sala:



```
1 public class Sala { 2 usages
2     int numero; 2 usages
3     int capacidad; 2 usages
4     String pelicula; 2 usages
5
6     public Sala(int numero, int capacidad, String pelicula) { 1 usage
7         this.numero = numero;
8         this.capacidad = capacidad;
9         this.pelicula = pelicula;
10    }
11
12 }
```

- Cine:

```
public class Cine { 2 usages
    String nombre; 2 usages
    List<Sala> salas; 3 usages

    public Cine(String nombre) { 1 usage
        this.nombre = nombre;
        this.salas = new ArrayList<>();
    }

    public void agregarSala(Sala sala) { 2 usages
        salas.add(sala);
    }

    // Getters
    public String getNombre() { 1 usage
        return nombre;
    }

    public List<Sala> getSalas() { no usages
        return salas;
    }
}
```

- General

```
Main.java x Sala.java Entrada.java Cliente.java pelicula.java
1 public class Main {
2     public static void main(String[] args) {
3         // Crear peliculas
4         pelicula pelicula1 = new pelicula( titulo: "Culpa Mia", fecha: 148, genero: "Romance", duracion: 120);
5         pelicula pelicula2 = new pelicula( titulo: "Barbie", fecha: 208, genero: "comedia", duracion: 110);
6
7         // Crear una sala
8         Sala sala1 = new Sala( numero: 1, capacidad: 100, pelicula: "Culpa mia");
9         sala1.agregarPelicula(pelicula1);
10        sala1.agregarPelicula (pelicula2);
11
12        // Crear un cliente
13        Cliente cliente1 = new Cliente( daniela: "Daniela", cedula: 1_725_378_999);
14        Entrada entrada1 = new Entrada( e001: "E001", new Fecha(), pelicula1, sala1);
15        cliente1.comprarEntrada(entrada1);
16
17        // Mostrar historial del cliente
18        cliente1.verHistorial();
19    }
20
21    public static class Fecha { 2 usages
22    }
23
24 }
```

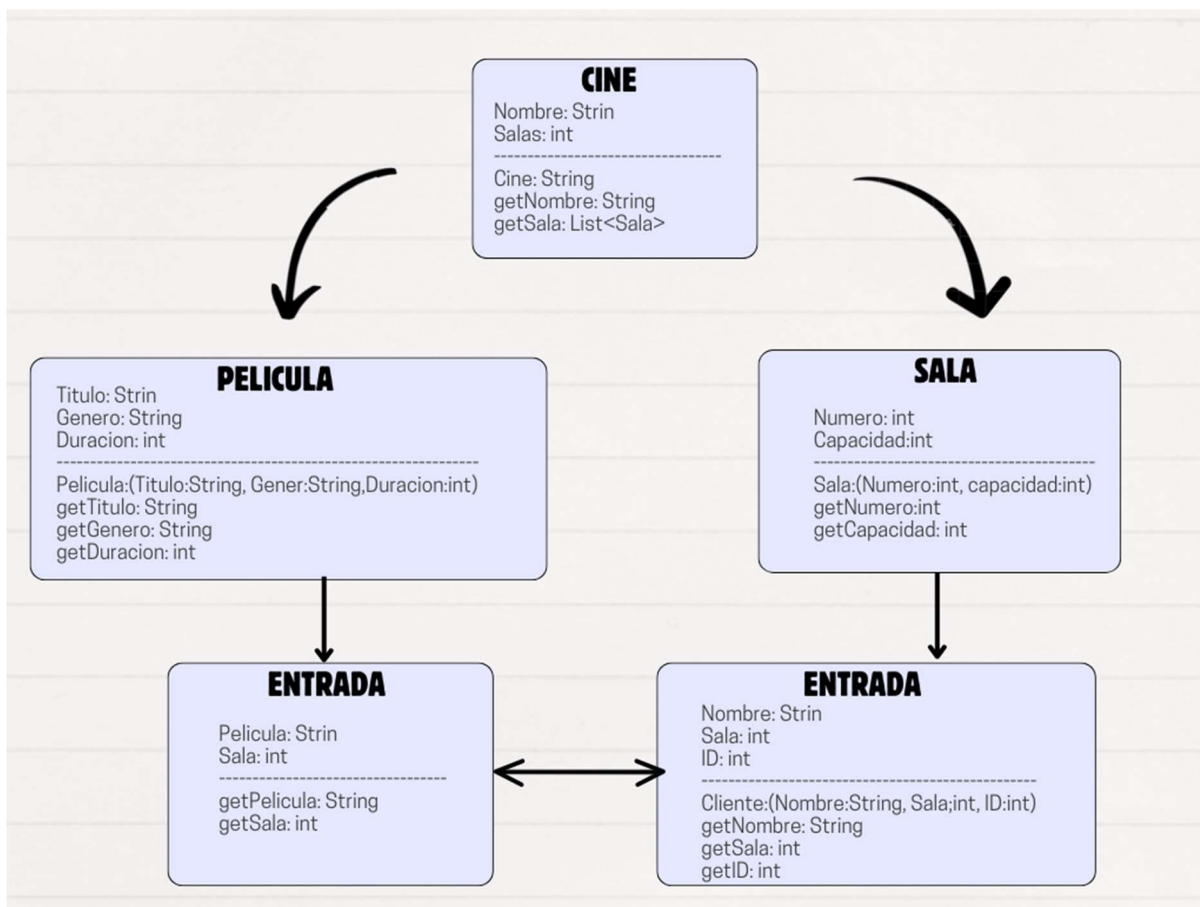
```

Run Main x
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3\lib\idea_rt.jar=52713:C:\
Cine: Cineplex
Cliente: Daniela
Pelicula: null

Process finished with exit code 0

```

Uml:



CONCLUSIONES

- **A lo largo del proceso, se lograron implementar funcionalidades clave como el registro de películas, la gestión de funciones y la venta de entradas, lo que demuestra el potencial del sistema. Sin embargo, la falta de tiempo y la complejidad en la integración de componentes dificultaron el avance y la calidad del producto final.**

RECOMENDACIONES

- **Realizar mas pruebas para poder obtener códigos limpios**
- **Tratar de mantener un orden preciso para no confundirnos**
- **Practicar un poco mas sobre lo que es el diseño de objetos y uml**

BIOGRAFIA

<https://www.lucidchart.com/pages/es/diagrama-de-objetos-uml>

<https://miro.com/es/diagrama/que-es-diagrama-uml/>