

04/06/2025

# Proyecto SISTEMA DE GESTIÓN DE LA FERIA DE LOJA

Programación Orientada a Objetos

David González  
Luis Quito  
Carlos Correa

## Problemática:

Desarrollar un sistema de gestión de entradas para la Feria internacional de Loja, el cual permitirá facturar boletos de entrada a la feria, considerando dos categorías: entradas normales y entradas para funciones especiales, estas últimas relacionadas con presentaciones de artistas nacionales e internacionales, que se realizarán los jueves, viernes y sábados en horario de 5 pm a 2 am.



## Objetivo Claro:

Desarrollar un sistema que permita gestionar eficientemente el acceso, control y análisis de información de una feria, incluyendo visitantes, entradas y eventos especiales.



## Solución Propuesta:

Implementar una solución en Java que:

- Registre visitantes y sus características.
- Permita asociar eventos a cada visitante.
- Calcule estadísticas de forma automática y organizada.

## Entrada

```
+categoria: String  
+precio: double = 2.5  
+fechaHora: LocalDateTime  
+visitante: Visitante  
  
+Entrada()  
+Entrada(categoría:String, precio:double,  
          fechaHora:localDateTime)  
+gestionVentas(acumulacionEntradas:ArrayList<Entrada>): void  
+verificarHora(): boolean
```

**Propósito:** Representa las entradas vendidas en la feria.

**Métodos:**

**gestionVentas():** Maneja el proceso de venta de entradas

**verificarHora():** Valida si el tipo de entrada es válida según la hora para verificar si la entrada se encuentra en el rango obligatorio de los eventos

## Visitante

```
*+discapacidad: boolean  
*+cantEntradasEventos: int  
*+cantEntradasNormales: int  
*+cantidadTotalEntradas: int  
*+totalPagar: double  
*+descuentoAplicado: double  
*+eventoAsistido: String  
  
*+Visitantes()  
*+Visitantes(discapacidad:boolean, cantEntradasNormales:int,  
            cantEntradasEventos:int, cantidadTotalEntradas:int,  
            eventoAsistido:String)  
*+insertarVisitante(acumulacionVisitantes:ArrayList<Visitante>): void
```

**Propósito:** Acumula datos sobre los visitantes y su compra de entradas para poder calcular el total de entradas vendidas en la clase Estadística.

### Métodos:

**insertarVisitante():** Agrega los datos del visitante por teclado y los almacena automáticamente en un arreglo dinámico para la facturación.

## Evento

```
+nombreArtista: String  
+diaPresentacion: String  
+cantidadVisitantes: int  
  
+Evento()  
+Evento(nombreArtista:String, diaPresentacion:int)  
+insertarEvento(eventosPresentables:ArrayList<Evento>,  
               nombreArtista:String, diaPresentacion:String): void  
+calcularVisitantes(acumulacionVisitantes:ArrayList<Visitante>,  
                     eventosPresentables:ArrayList<Evento>): void
```

**Propósito:** Principalmente es para calcular el total del entradas de eventos ingresadas por los visitantes y para ingresar los eventos que se van a celebrar en la feria.

**Métodos:**

**insertarEvento():** Agrega los eventos que se van a celebrar al sistema.

**calcularVisitantes():** Calcula el total de entradas de evento vendidas.

## **Estadistica**

```
*+totalVisitantes: int  
*+TotalPersonasDiscapacitadas: int  
*+ingresosTotales: double  
*+descuentosTotales: double  
*+asistenciaPorDia: int[]  
*+eventoMayorAsistencia: Evento  
*+eventoMenorAsistencia: Evento  
  
*+Estadistica()  
*+Estadistica(totalVisitantes:int, totalPersonasDiscapacitadas:int,  
    ingresoTotales:double, descuentoTotales:double,  
    asistenciaPorDia:int[], eventoMayorAsistencia:Evento,  
    eventoMenorAsistencia:Evento)  
*+calcularEstadistica(visitantes:ArrayList<Visitante>,  
    eventos:ArrayList<Evento>): void  
*+imprimirEstadistica(acumulacionVisitantes:ArrayList<Visitante>,  
    eventosPresentables:ArrayList<Evento>): void  
*+guardarEstadistica(nombreArchivo:String,  
    eventosPresentables:ArrayList<Evento>,  
    asistenciaPorDia:int[]): void
```

**Propósito:** Calcula el total de todo de las entradas normales, de las de eventos, que evento fue el mas vendido, etc. Principalmente presenta un informe completo sobre el funcionamiento de la feria.

**Métodos:**

**calcularEstadistica():** Realiza los cálculos estadísticos sobre la feria.

**imprimirEstadistica():** Muestra los resultados.

**guardarEstadistica():** Almacena las estadísticas en un archivo.

## EjecutorSistemaFeria

```
*+acumulacionEntradas: ArrayList<Entrada>
*+acumulacionVisitantes: ArrayList<Visitante>
*+eventos: ArrayList<Evento>
*+main(String[] arg): void
```

## **CONCLUSIONES**

- El diagrama UML es muy esencial para la creación organizada de nuestro código ya que este nos ayuda a tener una mejor organización al momento de codificar.
- los arreglos dinámicos son muy importantes para el correcto manejo de estructuras de almacenamiento cuando el tamaño no esta definido.

# Dificultades

- Al momento de definir los atributos de la clase en el modelado UML se denota la falla en la definición de estos a causa de conocer en profundidad las necesidades del modelo.
- En el momento de la inserción de la información en un archivo es importante tener en cuenta el orden y la secuencia del ingreso de la información para tener un correcto resultado legible para persona no aptas en la materia.

# Aprendizaje

- Llegar a tener un mejor análisis del problema para la creación del modelado UML.
- Tener un mejor orden y lógica al momento de crear las diferentes clases para la codificación y no sobrecargar las clases de datos incensarios.

# Experiencias

- El trabajo en equipo es esencial para definir y realizar la estructura del modelado en la arquitectura del UML y que la codificación sea mas eficiente.
- Llevar un correcto orden y lógica permite que el trabajo sea mas eficaz y rápido.

**¡GRACIAS!**