

# ***Programación Avanzada***

## **IIC2233 2025-2**

Cristian Ruz - Pablo Araneda - Francisca Ibarra - Tamara Vidal - Daniela Concha



# Experiencia 1

OOP

---

# DCCPALOOZA

## Requisitos

- Menú de inicio
- Clase administradora: DCCpalooza
- Clase artista: Artista

# Datos

artista.csv

nombre;genero;dia\_presentacion;hit\_del\_momento

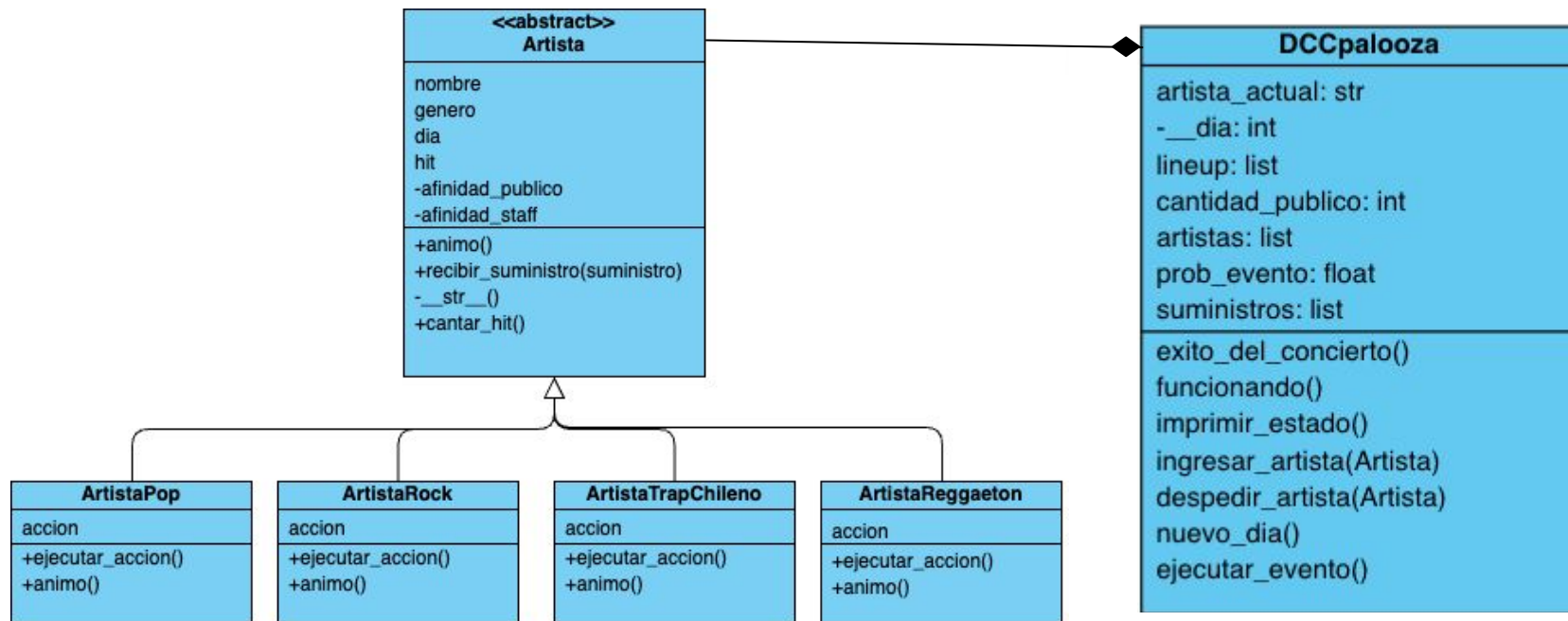
nombre	Nombre del artista
genero	Género del artista (Pop, Rock, Trap Chileno y Reggaeton)
dia_presentacion	Día que le corresponde presentarse al artista
hit_del_momento	Nombre de la canción más escuchada del artista

suministros.csv

nombre;valor\_de\_satisfaccion

nombre	Nombre del suministro
valor_de_satisfaccion	Valor de satisfacción que entrega el suministro

# Diagrama de clases



# Class Artista



<code>self.nombre</code>	Un <code>str</code> que representa el nombre del artista.
<code>self.genero</code>	Un <code>str</code> que representa el género musical del artista. Puede ser pop, rock, reggaeton o trap chileno.
<code>self.dia_presentacion</code>	Un <code>int</code> que representa el día en el que el artista tendrá su concierto. Tiene un valor de 1, 2 o 3.
<code>self.hit_del_momento</code>	Un <code>str</code> que representa la canción favorita del público de ese artista.
<code>self._afinidad_con_publico</code>	Un <code>int</code> que representa que tanta afinidad está teniendo el artista con el público. Es importante que este valor no puede bajar de 0 y tampoco puede superar 100. <b>Debes implementarlo como una <i>property</i>.</b>
<code>self._afinidad_con_staff</code>	Un <code>int</code> que representa que tanta afinidad está teniendo el artista con el staff del concierto. Es importante que este valor no puede bajar de 0 y tampoco puede superar 100. <b>Debes implementarlo como una <i>property</i>.</b>

# Class Artista



```
def animo(self) -> None:
```

*Property* que calcula el ánimo del artista como una ponderación de sus atributos `self._afinidad_con_staff` y `self._afinidad_con_publico`.

Se calcula de la siguiente manera:

$$animo = \lfloor afinidad\_con\_publico * 0.5 \rfloor + \lfloor afinidad\_con\_staff * 0.5 \rfloor$$



# Class Artista

```
def recibir_suministro(self,  
                        suministro: Suministro) -> None:
```

Esta función se llama cada vez que quieras atender al Artista.

Recibe una instancia de `Suministro` y modifica el atributo

`self.afinidad_con_staff` sumando el valor de satisfacción que tenga el suministro.

Este valor puede aumentar la afinidad o disminuirla. En cada caso, debes imprimir un mensaje indicando lo que pasó.

```
"{self.nombre} recibió {suministro.nombre} en malas condiciones"  
"{self.nombre} recibió un {suministro.nombre} a tiempo!"
```



# Class Artista



```
def cantar_hit(self) -> None:
```

Esta función se llama cada vez que quieras que el Artista cante su `hit_del_momento`. Esto aumenta la `afinidad_con_publico` en `AFINIDAD_HIT`, y deberás imprimir el siguiente mensaje:

```
"{self.nombre} está tocando su hit: {self.hit_del_momento}!"
```

Este valor puede aumentar la afinidad o disminuirla. En cada caso, debes imprimir un mensaje indicando lo que pasó.

# Class ArtistaPop



```
def __init__(self) -> None:
```

Llama el constructor de la clase padre. Además define los siguientes atributos:

- `self.accion = "Cambio de vestuario"`
- `self._afinidad_con_publico = AFINIDAD_PUBLICO_POP`
- `self._afinidad_con_staff = AFINIDAD_STAFF_POP`

```
def ejecutar_accion(self) -> None:
```

Este método aumenta la `afinidad_con_publico` en `AFINIDAD_ACCION_POP`.  
Además, imprime un mensaje específico para su género:

```
"{self.nombre} hará un {self.accion}"
```

# Class ArtistaPop

```
def animo(self) -> None:
```

Deberás sobrescribir la función `animo` de la clase.

Retornar el valor de la *property* `animo` de la clase padre. Si el valor obtenido es menor a 10, imprime el siguiente mensaje:

```
"ArtistaPop peligrando en el concierto. Animo: {valor_animo}"
```



# Class ArtistaRock



```
def __init__(self) -> None:
```

Llama el constructor de la clase padre. Además define los siguientes atributos:

- `self.accion = "Solo de guitarra"`
- `self._afinidad_con_publico = AFINIDAD_PUBLICO_ROCK`
- `self._afinidad_con_staff = AFINIDAD_STAFF_ROCK`

```
def ejecutar_accion(self) -> None:
```

Este método aumenta la `afinidad_con_publico` en `AFINIDAD_ACCION_ROCK`.  
Además, imprime un mensaje específico para su género:

```
"{self.nombre} hará un {self.accion}"
```

# Class ArtistaRock

```
def animo(self) -> None:
```

Deberás sobrescribir la función `animo` de la clase.

Retornar el valor de la *property* `animo` de la clase padre. Si el valor obtenido es menor a 5, imprime el siguiente mensaje:

```
"ArtistaRock peligrando en el concierto. Animo: {valor_animo}"
```



# Class ArtistaTrapChileno

```
def __init__(self) -> None:
```

Llama el constructor de la clase padre. Además define los siguientes atributos:

- `self.accion = "Malianteo"`
- `self._afinidad_con_publico = AFINIDAD_PUBLICO_TRAP_CHILENO`
- `self._afinidad_con_staff = AFINIDAD_STAFF_TRAP_CHILENO`

```
def ejecutar_accion(self) -> None:
```

Aumenta la `afinidad_con_publico` en `AFINIDAD_ACCION_TRAP_CHILENO`. Además, imprime un mensaje específico para su género:

```
"{self.nombre} hará un {self.accion}"
```



# Class ArtistaTrapChileno

```
def animo(self) -> None:
```

Deberás sobrescribir la función `animo` de la clase.

Retornar el valor de la *property* `animo` de la clase padre. Si el valor obtenido es menor a 20, imprime el siguiente mensaje:

```
"ArtistaTrapChileno peligrando en el concierto. Animo: {valor_animo}"
```



# Class ArtistaReggaeton

```
def __init__(self) -> None:
```

Llama el constructor de la clase padre. Además define los siguientes atributos:

- `self.accion = "Perrear"`
- `self._afinidad_con_publico = AFINIDAD_PUBLICO_REGGAEATON`
- `self._afinidad_con_staff = AFINIDAD_STAFF_REGGAETON`

```
def ejecutar_accion(self) -> None:
```

Este método aumenta la `afinidad_con_publico` en `AFINIDAD_ACCION_REGGAETON`. Además, imprime un mensaje específico para su género:

```
"{self.nombre} hará un {self.accion}"
```





# Class ArtistaReggaeton

```
def animo(self) -> None:
```

Deberás sobrescribir la función `animo` de la clase padre y esta.

Retornar el valor de la *property* `animo` de la clase padre. Si el valor obtenido es menor a 2, imprime el siguiente mensaje:

```
"ArtistaReggaeton peligrando en el concierto. Animo: {valor_animo}"
```



# Class DCCPalooza

<code>self.artista_actual</code>	Un <code>str</code> que representa al artista que está actualmente tocando en el concierto.
<code>self.__dia</code>	Un <code>int</code> que funciona como contador del progreso de la simulación. Debe verificar que se mantenga siempre positivo y con un incremento ascendente.
<code>self.line_up</code>	Un <code>list</code> que contiene instancias de las clases <code>ArtistaPop</code> , <code>ArtistaRock</code> , <code>ArtistaTrapChileno</code> y <code>ArtistaReggaeton</code> que tocarán cada día.
<code>self.cant_publico</code>	Un <code>int</code> que representa la cantidad de público al final de cada día.
<code>self.artistas</code>	Una <code>list</code> que contiene instancias de las clases <code>ArtistaPop</code> , <code>ArtistaRock</code> , <code>ArtistaTrapChileno</code> y <code>ArtistaReggaeton</code> de todo el concierto.
<code>self.prob_evento</code>	Una <code>float</code> que representa la probabilidad de que ocurra algún evento durante el concierto.
<code>self.suministros</code>	Una <code>list</code> que contiene instancias de la clase <code>Suministro</code> .

# Class DCCPalooza

```
def nuevo_dia(self) -> None:
```

Este método simula el paso de un día. Primero verifica la condición de término llamando a `self.exito_del_concierto()`; si está siendo exitoso, aumenta el atributo `self.__dia` en 1, y en caso de que el día sea menor o igual 3 (ya habiendo sumado la unidad) se imprime que comienza un nuevo día.

# Class DCCPa1ooza

```
def ejecutar_evento(self) -> None:
```

Este método se encarga de verificar la ocurrencia de algún evento y ejecutar su efecto en dicho caso.

Si se cumple la probabilidad contenida en `self.prob_evento` deberás escoger alguno de los siguientes eventos de forma **ALEATORIA**:

(Eventos en la siguiente diapositiva)

# Class DCCPa1ooza

```
def ejecutar_evento(self) -> None:
```

- **Lluvia:** Disminuye la afinidad con el público del artista actual en una cantidad AFINIDAD\_LLUVIA e imprime un mensaje que lo indique.
- **Terremoto:** Disminuye la cantidad de público DCCPa1ooza en una cantidad PUBLICO\_EVENTO e imprime un mensaje que lo indique.
- **Ola de calor:** Disminuye la afinidad con el público del artista actual en una cantidad AFINIDAD\_OLA\_CALOR e imprime un mensaje que lo indique. Además, disminuye la cantidad de público de DCCPa1ooza en una cantidad PUBLICO\_OLA\_CALOR e imprimir un mensaje que lo indique.

# ***Programación Avanzada***

## **IIC2233 2025-2**

Cristian Ruz - Pablo Araneda - Francisca Ibarra - Tamara Vidal - Daniela Concha

