Programación Avanzada
IIC2233 2025-2

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

Experiencia 1

OOP

Requisitos

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

Datos

artista.csv

```
nombre;genero;dia_presentacion;hit_del_momento
```

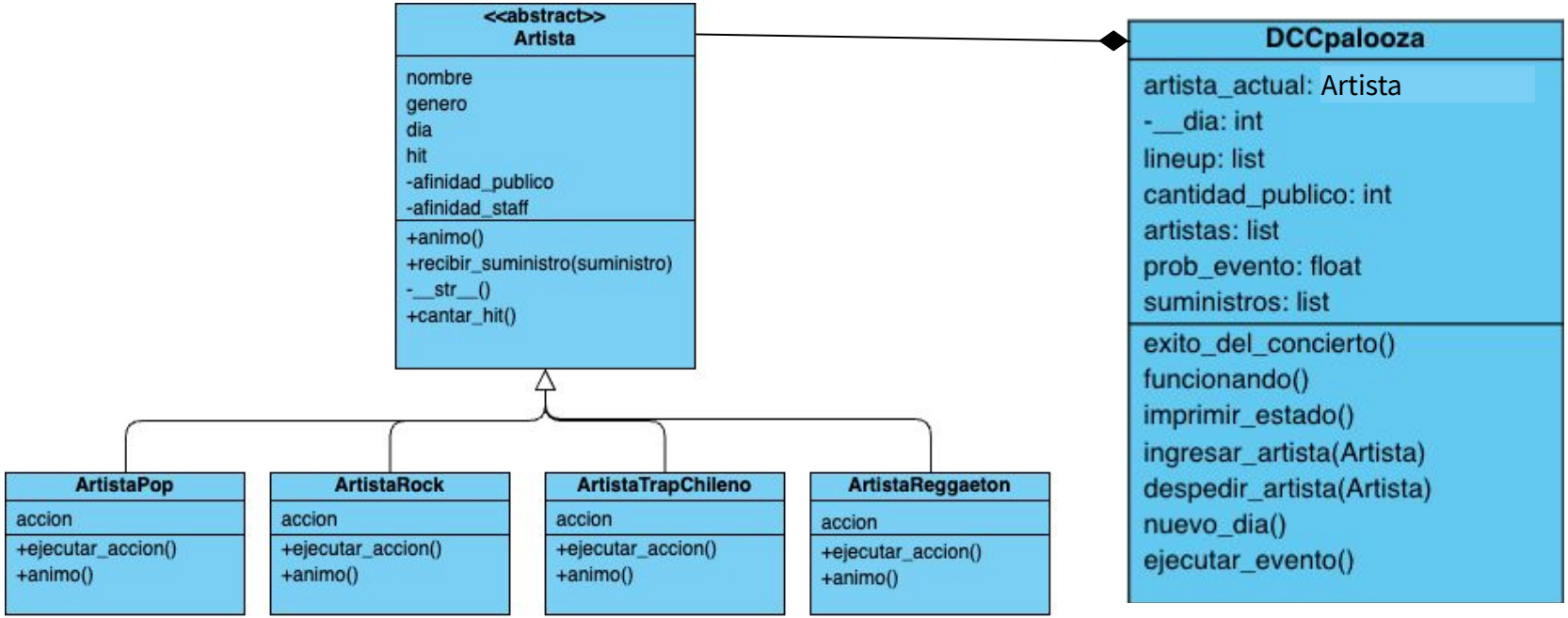
`nombre`	Nombre del artista
`genero`	Género del artista (Pop, Rock, Trap Chileno y Reggaeton)
`dia_presentacion`	Día que le corresponde presentarse al artista
`hit_del_momento`	Nombre de la canción más escuchada del artista

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`

`self.nombre`	Un `str` que representa el nombre del artista.
`self.genero`	Un `str` que representa el género musical del artista. Puede ser pop, rock, reggaeton o trap chileno.
`self.dia_presentacion`	Un `int` que representa el día en el que el artista tendrá su concierto. Tiene un valor de 1, 2 o 3.
`self.hit_del_momento`	Un `str` que representa la canción favorita del público de ese artista.
`self._afinidad_con_publico`	Un `int` 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. **Debes implementarlo como una *property*.**
`self._afinidad_con_staff`	Un `int` 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. **Debes implementarlo como una *property*.**

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:

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

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` canta 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`


```python
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`

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

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 `DCCPalooza`

```
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 `DCCPalooza`

```
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 `DCCPalooza` en una cantidad `PUBLICO_TERREMOTO` 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 `DCCPalooza` 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