

**UNIVERSIDAD TECNOLOGICA NACIONAL FACULTAD REGIONAL GENERAL PACHECO**

**TÉCNICO SUPERIOR EN PROGRAMACIÓN**

**LABORATORIO DE COMPUTACIÓN V**

**TP 1**

**Ejercicio de consola**

**PROFESOR/RA:**

* **Tamara Herrera**

**GRUPO NRO. 12 – INTEGRANTES:**

* **Sergio Adrián Palacio**
* **Juan Manuel Oreguy**
* **Pablo Ernesto Giménez**
* **Marcos José Zone**

Índice

[**a) Constructores:** 3](#_Toc99675416)

[**b) Encapsulamiento:** 4](#_Toc99675417)

[**c) Método toString():** 5](#_Toc99675418)

[**d) Variables Constantes:** 7](#_Toc99675419)

[**e) Variables Estáticas:** 8](#_Toc99675420)

[**f) Herencia:** 9](#_Toc99675421)

[**g) Clase abstracta con un método abstracto:** 10](#_Toc99675422)

[**h) Interfaz:** 11](#_Toc99675423)

[**i) Método Main:** 12](#_Toc99675424)

[**j) Polimorfismo:** 13](#_Toc99675425)

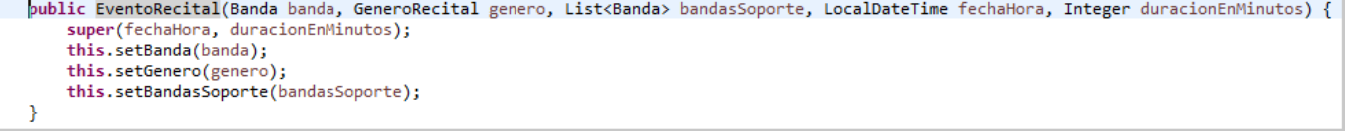
# **a) Constructores:**

Para mostrar los constructores decidimos poner dos ejemplos de cuando se instancian los eventos de recital y teatro.

En este caso es el constructor por parámetros de la clase “**EventoRecital**”.

Constructor por parámetros

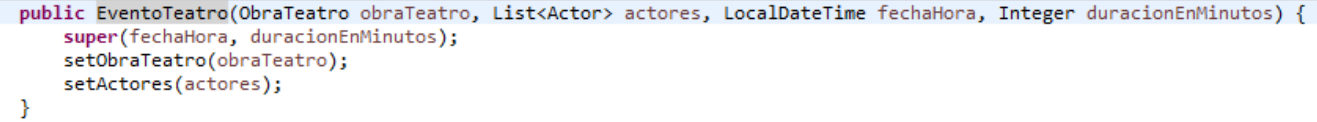




Constructor por parámetros de la clase “**EventoTeatro**”.

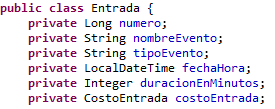
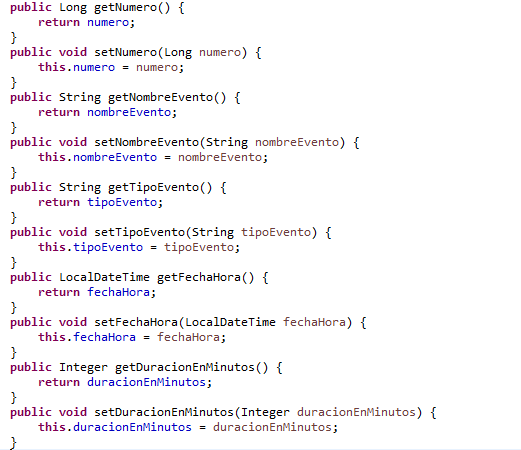
Constructor por parámetros





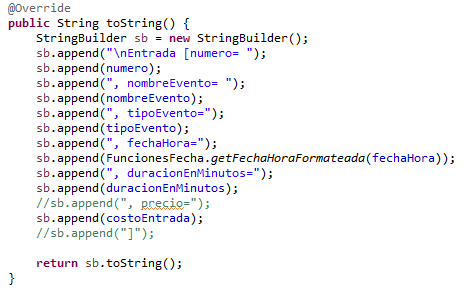
# **b) Encapsulamiento:**

Para el mostrar el Encapsulamiento se ve como ejemplo en la clase “**Entrada**” sus atributos definidos como “**private**” para proteger la información y a continuación se verá los “**get**” y “**set**” de estos atributos.

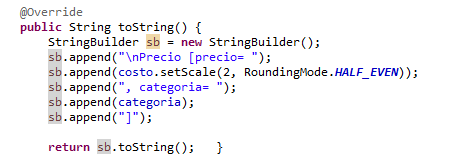


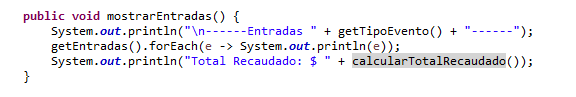
# **c) Método toString():**

Para mostrar el método “**toString()**” hicimos lo siguiente:

Primero tenemos el método “**toString()**” de la clase “**Entrada**”.

Después también tenemos método “**toString()**” en la clase “**CostoEntradaBase**”.



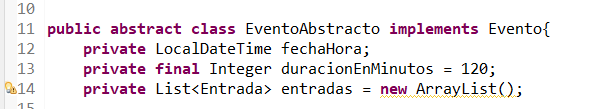
Por último, tenemos la función “**mostrarEntradas()**” en la clase “**EventoAbstracto**” donde con un “**forEach()**” vamos recorriendo los registros de entradas que van a ir mostrando los métodos “**toString()**” mencionados anteriormente y al final se calcula el total recaudado que vendría a ser el total recaudado por cada tipo de evento llamando a la “**funcióncalcularTotalReacaudado**” (Esta función se encuentra en la misma clase).

# **d) Variables Constantes:**

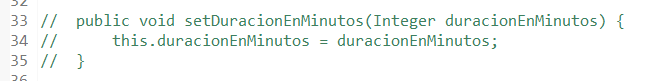
En este caso pusimos como variable constante a la variable que se utiliza para la duración de cada evento.

En este caso damos por hecho que todos los eventos cuentan con la misma duración.

Como aparece en la imagen, esto se encuentra en la clase “**EventoAbstracto**” y la variable constante es “**duracionEnMinutos**”.



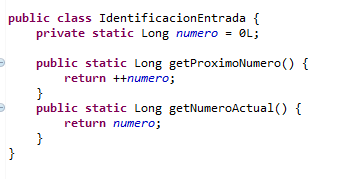
En el caso de querer tener distintas duraciones lo que debemos hacer es descomentar el Setter y quitar el “Final” de la variable



# **e) Variables Estáticas:**

En este caso pusimos como variable estática a la variable que se utiliza para dar el número de identificación único a cada entrada.

Como aparece en la imagen, esto se encuentra en la clase “**IdentificacionEntrada**” y la variable estática se la “**numero**”.



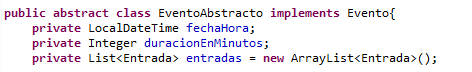
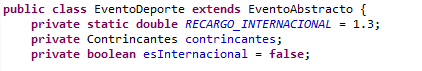
# **f) Herencia:**

En el caso de herencia vamos a mostrar la relación que tenemos entre las clases “**EventoDeporte**” y “**EventoAbstracto**”.

Acá la clase padre es “**EventoAbstracto**” y la clase hija es “**EventoDeporte**”.

A continuación, en las imágenes se ve reflejada esta relación:

Clase padre



Relación de clases

Clase hija

# **g) Clase abstracta con un método abstracto:**

En este caso la clase abstracta es “**EventoAbstracto**” donde tiene dos métodos abstractos (“**getTipoEvento();**” y “**getNombreEvento();**”) los cuales tendrán que si o si ser generados por todas aquellas clases que hereden de la misma.

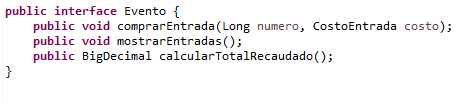
Clase abstracta: “**EventoAbstracto**”.

Métodos abstractos: “**getTipoEvento();**” y “**getNombreEvento();**



# **h) Interfaz:**

Una de las interfaces que utilizamos es la de “**Evento**” que como se vio en las imágenes anteriores esta interfaz la implementamos en la clase “**EventoAbstracto**”.

****

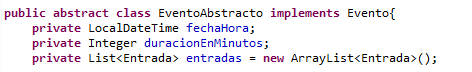
# **i) Método Main:**

El método main de la aplicación es el siguiente:

Esta es una imagen demostrativa del “**main()**”, ya que no se aprecia todo el código completo que se desarrolla dentro del mismo.

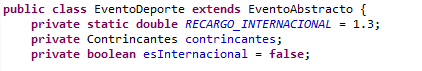
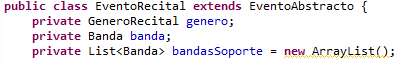
# **j) Polimorfismo:**

Para mostrar el polimorfismo vamos a mostrar el ejemplo de cada una de las clases de los eventos que heredan de la clase padre “**EventoAbstracto**”.



Clase padre

Clase hija



Clase hija

Clase hija

Clase hija

