



UNIVERSIDAD
DE GRANADA

Desarrollo de Software

Práctica 1.2

Granada, 24 de marzo 2021

Índice:

Parte Obligatoria	1
Diagrama de clases	2
Parte Opcional	2
Diagrama de clases	3
Capturas de la aplicación	4
Parte obligatoria	4
Parte opcional	5

Parte Obligatoria

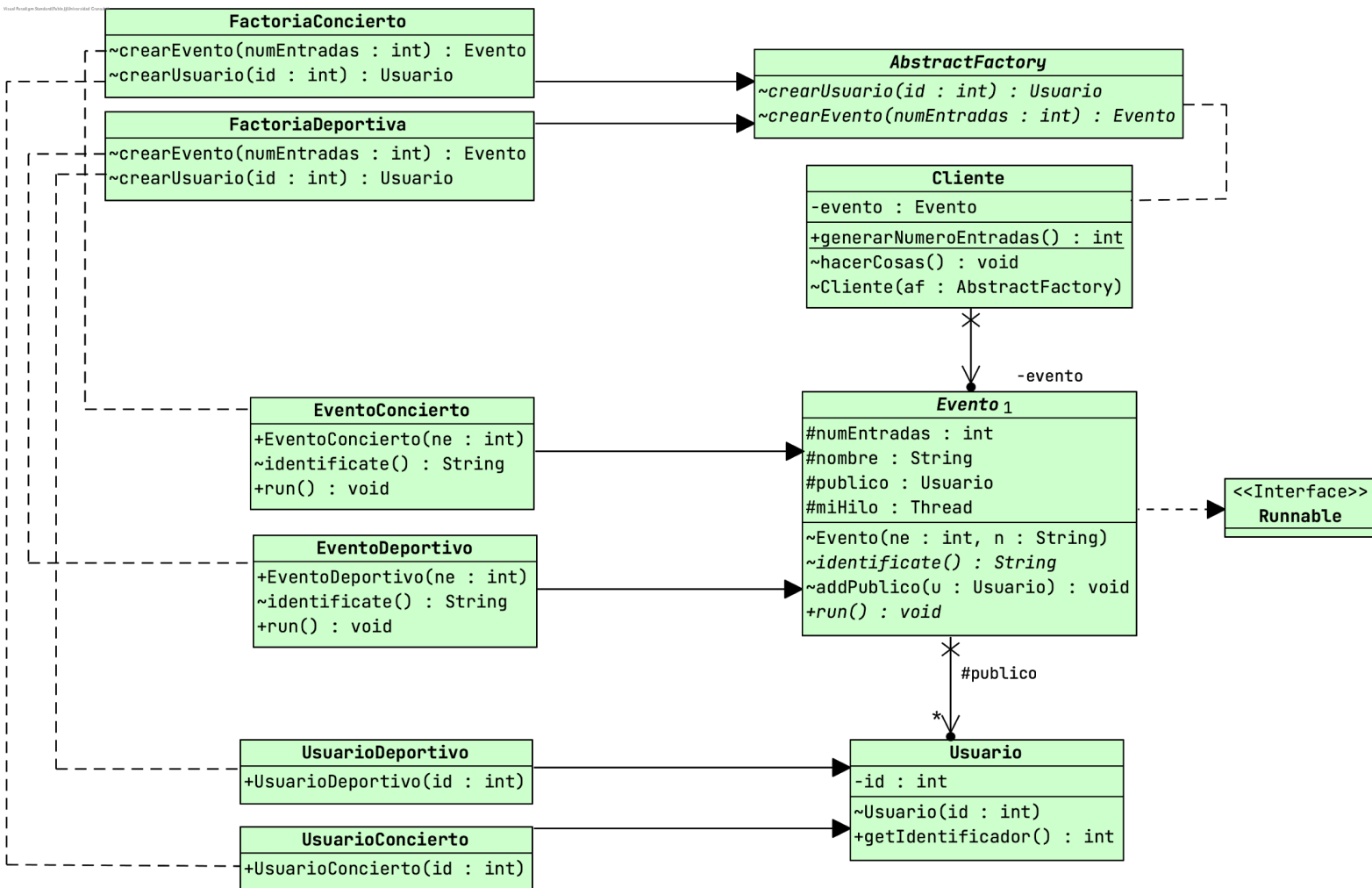
Para abordar este ejercicio hemos considerado dos familias, *Evento* y *Usuario* y estas familias tienen dos tipos de productos, productos de tipo *concierto* o productos de tipo *deportivo*.

El uso conjunto de los patrones de diseño *Factoría Abstracta* y *Método Factoría* aplicado en nuestro caso quedaría como se ve en el diagrama de clases asociado. De forma que son las factorías concretas *FactoriaDeportiva* y *FactoriaConcierto* las encargadas de crear los eventos o usuarios de los diferentes tipos cuando el cliente lo requiera. Al heredar de *AbstractFactory*, siendo esta una clase abstracta, se obliga a las diferentes factorías concretas a implementar sus métodos de forma que en el caso de necesitar más tipos de factorías concretas, por ejemplo porque el sistema crezca en funcionalidades, permite una alta adaptabilidad además de hacer el diseño mucho más intuitivo y una codificación mucho más llana.

De esta manera a la hora de crear un tipo de producto concreto, mediante el método factoría, el cliente solo necesita conocer un objeto del tipo factoría abstracta, por lo que es ajeno a los diferentes tipos de productos concretos que haya, resultando ser altamente escalable pues se pueden añadir más familias a las ya existentes y el cliente no se vería afectado por estos cambios.

En cuanto a la funcionalidad de nuestro programa, el cliente genera un evento al que se asignará un público, formado por usuarios, ambos del mismo tipo, a partir del número de entradas que el evento tenga. En la ejecución, serán hebras las que desempeñen el papel de eventos, de forma que los diferentes tipos de eventos transcurren de distinta manera o lo que es lo mismo cada uno puede llevar asociado un comportamiento diferente, pues estamos hablando de distintos tipos de eventos. En el transcurso de los eventos, las entradas serán vendidas, comenzará el evento y un porcentaje de asistentes irá abandonando los eventos según diferentes causas hasta que el evento finalice.

Diagrama de clases



Parte Opcional

Hemos implementado la casuística anterior haciendo uso ahora del patrón creacional *Abstract Factory* con *Prototype*. En esta ocasión la factoría tiene prototipos que podrán ser de distintos tipos según cómo se cree el objeto de la factoría abstracta. Los objetos requeridos se clonan a partir de un estado base o inicial del objeto, aunque se podría discutir la necesidad de aplicación de este patrón en este ejemplo, es muy utilizado en los casos donde el coste de creación de un objeto es muy elevado (implica realizar consultas a bases de datos, inicializar grandes TDA's) por tanto aunque clonar un objeto no es una operación sencilla, es preferible a hacer una llamada al constructor.

Hemos diseñado una interfaz de forma que todas las futuras clases/modelos que quieran ser prototipos tengan que realizar la interfaz `Cloneable`. Otorgando

Diagrama de clases



Capturas de la aplicación

Parte obligatoria

```
Output - P1_S2 (run) x
run:
    Se han vendido un total de 25 entradas para el evento CONCIERTO
    Se han vendido un total de 39 entradas para el evento DEPORTIVO
    ¡Comienza el evento DEPORTIVO!
    ¡Comienza el evento CONCIERTO!
    El usuario 8 sale del evento deportivo porque tiene sueño
    El usuario 18 sale del concierto porque se marea
    El usuario 15 sale del concierto porque tiene sueño
    El usuario 9 sale del evento deportivo porque se marea
    El usuario 26 sale del evento deportivo porque se marea
    El usuario 22 sale del concierto porque tiene sueño
    El usuario 14 sale del evento deportivo porque se marea
    El usuario 1 sale del evento deportivo porque se marea
    El usuario 17 sale del evento deportivo porque se marea
    El usuario 3 sale del concierto porque no le está gustando
    El usuario 28 sale del evento deportivo porque se marea
    El usuario 10 sale del evento deportivo porque se marea
    El usuario 6 sale del concierto porque no le está gustando
    El usuario 16 sale del evento deportivo porque se marea
    El usuario 38 sale del evento deportivo porque tiene sueño
    El usuario 20 sale del concierto porque se marea
    El usuario 21 sale del concierto porque se marea
    ¡Termina el evento CONCIERTO!
    El usuario 21 sale del evento deportivo porque se marea
    ¡Termina el evento DEPORTIVO!
BUILD SUCCESSFUL (total time: 15 seconds)
```

Parte opcional

```
→ P1_S2_rb git:(main) ruby Principal.rb
Empieza el programa
Se han vendido un total de 32 entradas para el evento Concierto X
Empieza el evento Concierto X
  El usuario 20 sale del concierto porque tiene sueño
  El usuario 29 sale del concierto porque se marea
  El usuario 7 sale del concierto porque no le está gustando
  El usuario 5 sale del concierto porque se marea
  El usuario 13 sale del concierto porque no le está gustando
  El usuario 18 sale del concierto porque se marea
  El usuario 6 sale del concierto porque se marea
  El usuario 31 sale del concierto porque se marea
  El usuario 33 sale del concierto porque se marea
  El usuario 27 sale del concierto porque se marea
Termina el evento Concierto X
Se han vendido un total de 38 entradas para el evento Deportivo Y
Empieza el evento Deportivo Y
  El usuario 48 sale del evento deporivo porque le están llamando por telefono
  El usuario 50 sale del evento deporivo porque se ha cansado
  El usuario 62 sale del evento deporivo porque le están llamando por telefono
  El usuario 58 sale del evento deporivo porque se ha cansado
  El usuario 46 sale del evento deporivo porque le están llamando por telefono
  El usuario 60 sale del evento deporivo porque se ha cansado
  El usuario 71 sale del evento deporivo porque se marea
  El usuario 59 sale del evento deporivo porque se marea
  El usuario 69 sale del evento deporivo porque se marea
  El usuario 52 sale del evento deporivo porque le están llamando por telefono
  El usuario 61 sale del evento deporivo porque se marea
  El usuario 36 sale del evento deporivo porque se ha cansado
Termina el evento Deportivo Y
```