INFORME PRÁCTICA DS OP2 22-23

Principios de diseño utilizados:

En el código de la práctica, tenemos 3 principios de diseño presentes.

-*Principio de responsabilidad única*: cada objeto debe tener una responsabilidad única enteramente encapsulada en la clase. Todos los servicios que provee el objeto están estrechamente alineados con dicha responsabilidad. La clase **Partido** está repesentando mediante la final, a todos los partidos de ese torneo, actuando como observador de los hijos modificándolos o modificándose cuando es notificado de algún cambio de ellos. Su responsabilidad es esa, observar y modificar cuando sea necesario.

También tienen este principio las clases **GeneraciónAleatoria** y **GeneraciónCabezaSerie**, las cuales crean de distintas maneras las tablas del torneo. En estas dos últimas clases estamos dividiendo las responsabilidades de las dos generaciones en dos clases distintas.

-*Principio abierto-cerrado*: queremos ser capaces de cambiar lo que hace un clase, sin tener que tocar el código de la clase. El código está hecho para que sea sencillo añadir nuevos métodos de generación de tablas u otros estados en un futuro. En el código tenemos la clase **GenerarCuadros**, la cual es una clase sellada, ya que restringe que clases pueden extenderla o implementarla y con ella podríamos añadir fácilmente nuevas creaciones de tablas a raíz de permitir a otras generaciones añadiéndolas al enum generación e incluyéndolas en la clase **GenerarCuadros**, utilizando así estas nuevas clases el método crearPartidos.

-*Principio de inversión de la dependencia*: la inversión de la dependencia es la estrategia de depender de interfaces o clases y funciones abstractas, en vez de depender de clases y funciones concretas. Las clases **Partido**, **Jugador** y **Torneo** son clases abstractas y poseen inversión de la dependencia mediante los getters y setters para obtener y asignar valores, limitando el acceso a las clases.

Patrones de diseño utilizados:

Durante el desarrollo de esta práctica utilicé 2 patrones de diseño. Estes son el patrón estrategia y el patrón observador.

-*Patrón estrategia*: este patrón es utilizado ya que hay diferentes maneras de como generar el cuadro del torneo (mediantes sorteo aleatorio o mediante sorteo por cabezas de serie). La práctica también dice que puede haber más maneras de generar el cuadro del torneo añadidas posteriormente y esto se facilitaría mucho gracias a emplear este patrón. En esta interfaz tenemos la clase **GeneraciónUtilizada**, la cual decide que generador de cuadros utilizamos para crear las tablas. Esta clase tiene dos subclases, **GeneraciónAleatoria** y **GeneraciónCabezaSerie**, que implementan su método según el método asignado en **GeneraciónUtilizada**.

-*Patrón observador*: la clase **Partido** funciona como un nodo de un árbol, donde la final del torneo representa la raíz de ese árbol y por lo tanto el Torneo en cierta parte. Se accede a este árbol mediante la final mencionada anteriormente. Los nodos hojas o partidos de primera ronda notificarán cual es el ganador de su partido y cuando finaliza el partido. Así, los partidos son observadores de sus hijos.

Samuel Varela Gómez