**Programación orientada a objetos – Semana 2**

1. **¿Qué representa esta clase? ¿Qué información contiene?**

La clase representa las características de un jugador, tales como los atributos y comportamientos (en el ejemplo aún no se establece ninguno). En este caso las propiedades de la clase se refieren a aquellas que describen y proporcionan relevancia al jugador como su nombre, edad, posición, numero de camiseta y equipo.

1. **¿Por qué es importante inicializar los atributos al crear un objeto? ¿Qué pasaría si no usáramos un constructor?**

Se deben inicializar para que el objeto estés preparado para ser utilizado; en caso de no hacerlo, podría ocasionar algún tipo de error en el programa.

- Declarar un constructor permite atribuir al objeto un estado inicial al momento de instanciarlo, permitiendo consistencia. Si no existe el constructor, el objeto se inicializa de forma implícita con los valores por defecto de los tipos de datos (ejemplo: si es un entero int, el valor por defecto es 0. Si es un valor de tipo string, por defecto será un null). Tener un objeto instanciado de manera implícita puede conllevar a errores de datos no válidos, como es el caso del null, que puede provocar NullPointerException.

Por tanto, la data no estaría validada correctamente.

1. **¿Por qué no accedemos directamente a los atributos desde otras clases? ¿Qué ventajas tiene usar getters y setters?**

Es recomendable que los atributos de una clase sean privados, ya que esto puede evitar que se manipulen de manera incorrecta y así asegurar que el usuario siga la ruta especificada debido a que los atributos son variables que describen las características de un objeto. En otras palabras, tener los atributos privados protege la integridad de los datos ya que no pueden ser accedidos de forma imprudente de manera externa y ajena a la clase. Se trabaja con un acceso controlado a los atributos

- Una de sus ventajas es que nos permiten obtener el valor de una variable de instancia y cambiarlo. Adicional a esto, permite leer y escribir valores de variables privadas desde fuera de la clase donde se crearon. El encapsulamiento a través de getters y setters facilita supervisar las modificaciones u obtención de un valor de cada propiedad (atributo).

1. **¿Qué otros métodos podrían ser útiles para la clase Jugador? ¿Cómo podríamos mejorar este método?**

Se podrían usar también: correr, atacar, mover, hacerDano. Es decir, métodos que provean detalle del comportamiento del jugador y su desempeño.

- Se podría mejorar usando un String en lugar de un sout ya que nos daría más flexibilidad.

1. **¿Qué aprendiste al crear y utilizar objetos? ¿Cómo cambia el estado de un objeto cuando usamos métodos set?**

Aprendí que al crear un objeto estoy asignando memoria para almacenar valores de sus atributos. También comprendí que los objetos tienen un estado que es definido por sus atributos y comportamientos, los cuales son representados por sus métodos.

Algunos conceptos que estudie:

\* La encapsulación nos dice que los objetos agrupan datos y comportamientos en una sola entidad.

\* **El modularidad nos** facilita la reutilización y organización de nuestro código.

\* **La abstracción nos p**ermite ocultar detalles internos y mostrar solo lo necesario.

\* **Con la instancia** creamos objetos a partir de clases que nos sirven como plantillas.

- Su estado cambia al momento de actualizar alguno de sus atributos, ya que con este método podemos modificar sus atributos sin necesidad de cambiar la estructura del objeto.