Guía Taller Herencia y Polimorfismo en DART - 02

Para el desarrollo del siguiente taller, el aprendiz deberá tener claros los siguientes conceptos:

Herencia: es una propiedad que permite que los objetos sean creados a partir de otros ya existentes, obteniendo características (métodos y atributos) similares a los ya existentes. Es la relación entre una clase general y otra clase más específica. Es un mecanismo que nos permite crear clases derivadas a partir de clase base. Nos permite compartir automáticamente métodos y datos entre clases subclases y objetos. Por ejemplo: Si declaramos una clase párrafo derivada de una clase texto todos los métodos y variables asociadas con la clase texto son automáticamente heredados por la subclase párrafo

Polimorfismo: en programación orientada a objetos, el polimorfismo se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos. El único requisito que deben cumplir los objetos que se utilizan de manera polimórfica es saber responder al mensaje que se les envía.

Clases Abstractas: estas clases se caracterizan por tener algunos de sus métodos declarados como abstractos. Un método abstracto es aquel que está definido, pero no tiene cuerpo, es decir, le declaramos el nombre, los parámetros y el tipo de devolución pero no le declaramos lo que va entre llaves "{}", es más, no le ponemos llaves.

Estas clases se utilizan para permitir que otras clases hereden de ella y proporcionar un modelo a seguir, pero no se puede crear como tal un objeto de la misma (no pueden ser instanciadas). La declaración de una clase y método abstracto es el siguiente:

```
abstract class Vehiculo{
        // Método abstracto
        void conducir();
}
```

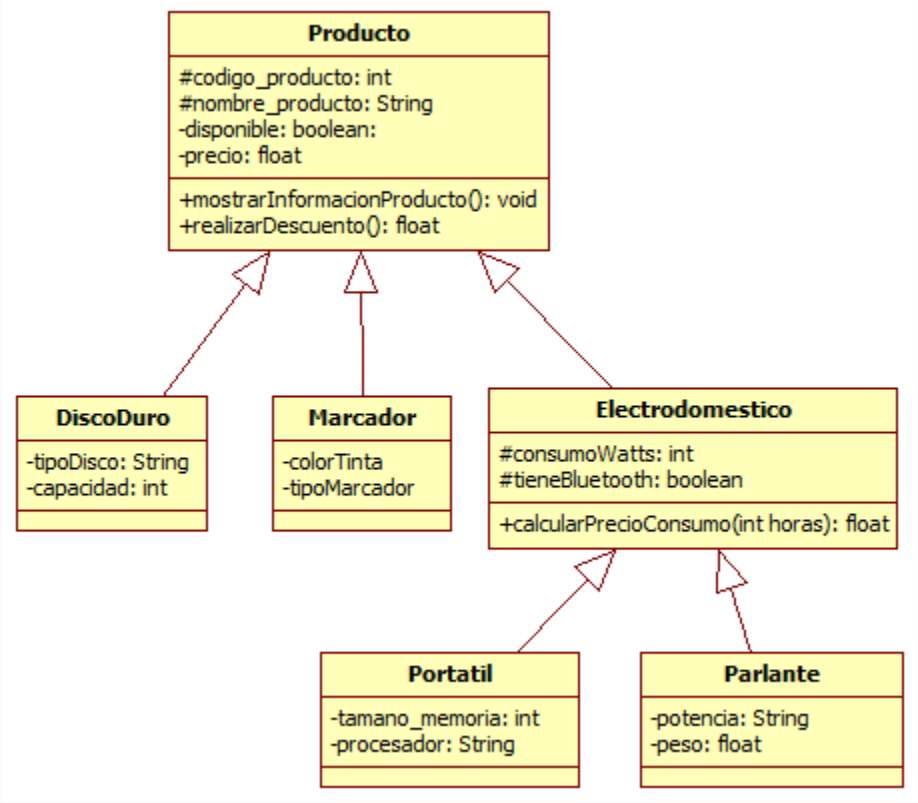
Reglas básicas de este tipo de clases:

- Una clase se declara abstracta si tiene algún método abstracto.
- Una clase abstracta no puede crear objetos, es decir, no podemos utilizar "new" para una clase abstracta, ya que tiene métodos que no están definidos.
- Una clase abstracta no puede ser a la vez "final" (En lenguajes como Java).
- Si una clase hereda de una clase abstracta, deberá de sobrescribir todos los métodos abstractos, si no es así, la clase que hereda deberá ser pasada a abstracta.
    - abstractos, pero solo con que exista un método abstracto, la clase deberá ser pasada a abstracta.
- Como los métodos estáticos no pueden ser redefinidos, un método abstracto no puede ser estático.
- Una clase abstracta si puede tener constructores. Como sabemos, no es posible crear objetos de una clase abstracta, pero de una clase, la cual herede de una clase abstracta, si se pueden crear objetos, y como sabemos cuándo creamos un objeto de una clase que

hereda, la primera llamada de su constructor es una llamada al constructor de la clase "padre", lo que permite que aunque no cree un objeto, si se pueda utilizar su constructor.

A partir de este momento y con base a lo anterior, el aprendiz deberá desarrollar los siguientes puntos:

1. Crear un proyecto en Dart y seguir las siguientes instrucciones, tomando como base el siguiente diagrama de clases:



   a. Construir las 6 clases en Dart (utilizando Visual Studio Code) tomando como base el diagrama de clases anterior y teniendo en cuenta la secuencia de herencia manejada. Deben crearse todos los métodos SET y GET de cada clase
   b. La clase **Producto** deberá crearse como clase abstracta, al igual que el método **realizarDescuento()**
   c. Para llamar el constructor de la clase **Producto** desde la clase **DiscoDuro** y **Marcador**, se deberá hacer utilizando el método **super()**
   d. Para llamar el constructor de la clase **Producto** desde la clase **Electrodoméstico**, se deberán settear directamente los atributos: código_producto y nombre_producto de la clase padre, y hacer el llamado de un constructor de la clase padre que reciba solamente los valores de los campos privados: disponible y precio
   e. Sobreescribir el método **realizarDescuento()** para que quede de la siguiente manera:
      i. El descuento de los discos duros será del 20%

ii. El descuento de los Marcadores será del 10%
iii. El descuento de los Electrodomésticos será del 30%

f. El método **mostrarInformacionProducto()** de la clase **Producto**, mostrará todos los atributos de dicha clase
g. El método **calcularPrecioConsumo()** de la clase **Electrodoméstico**, calculará el precio a pagar según el consumo de watts y la cantidad de horas
h. ¿Intentar crear una instancia (objeto) de la clase Producto y decir si se puede o no y por qué?
i. Al ejecutar el programa, se deberá mostrar un menú como el siguiente:
   Elige una opción:
   1) Crear Disco Duro
   2) Crear Marcador
   3) Crear Portátil
   4) Crear Parlante
   5) Vender Disco Duro
   6) Vender Marcador
   7) Vender Portátil
   8) Vender Parlante
   9) Calcular precio consumo Portátil
   10) Calcular precio consumo Parlante
   11) Salir
j. Si se elige entre la opción 1 y 4, se deberán crear las intancias (objetos) según la clase que corresponda
k. Si se elige entre la opción 5 y 8, se deberá llamar al método **realizarDescuento()** según el producto escogido y mostrar toda la información del producto vendido
l. Si se elige la opción 9 o 10, se deberá mostrar el valor que se pagaría por el consumo de Watts del electrodoméstico escogido según la cantidad de horas ingresadas
m. Si se elige la opción 11, se deberá terminar con la ejecución del programa