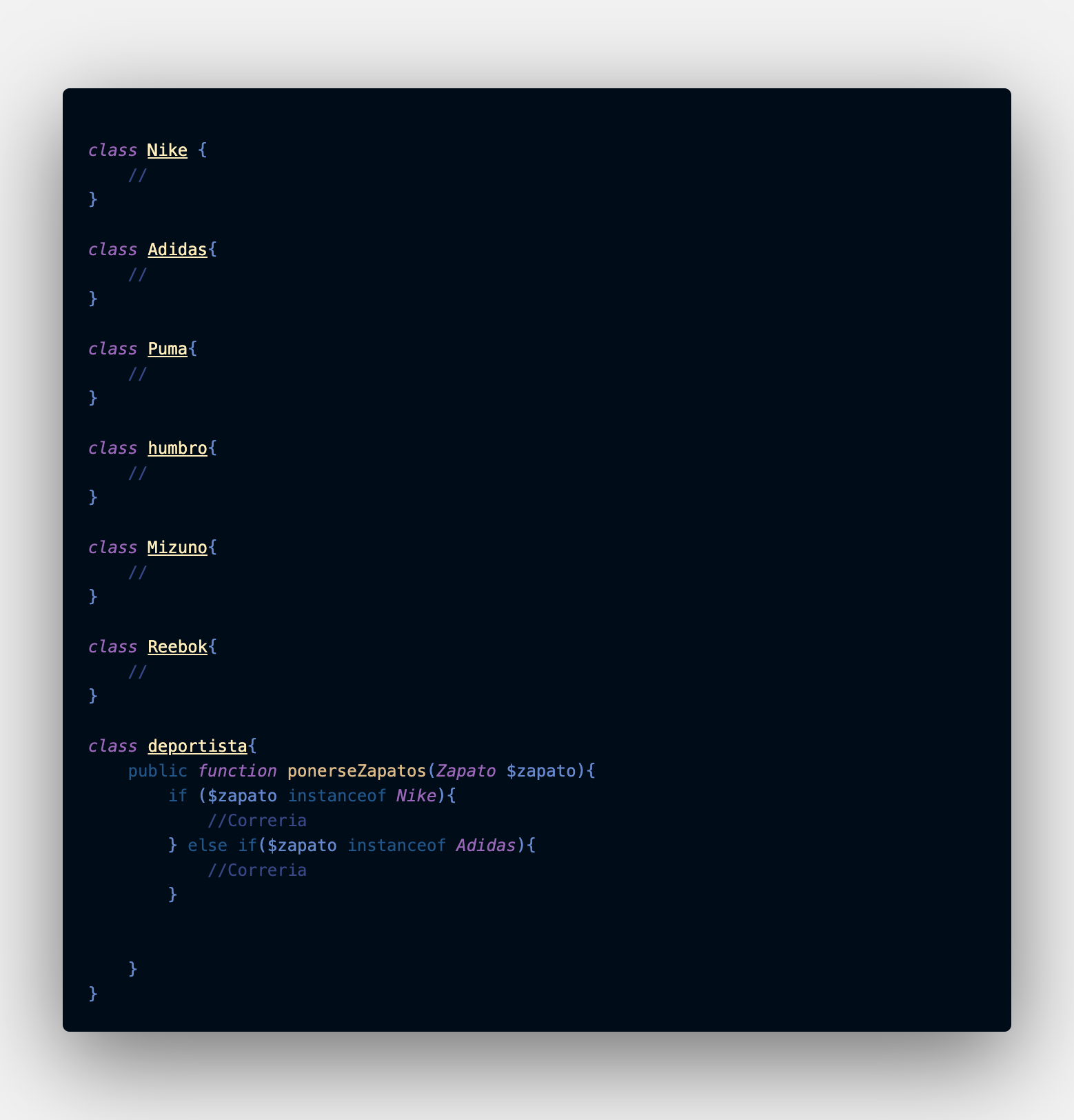
Principio SIngle Responsability (Responsabilidad unica)

En el ejemplo mostrado, se crea una clase llamada FIFA que tenía ciertos métodos, a la hora de solucionarlo se dividió en dos clases, la primera, clase Arquero con las funciones que corresponden a un Arquero y la segunda clase con las funciones que corresponden a un jugador dando como nombre a la segunda clase jugador.



Principio Open-Close (Abierto para extencion, cerrado para modificacion)

En el ejemplo de open - close Tenemos varias clases como Nike Adidas Mizuno entre otras, nos da entender las marcas de zapatos para correr, Y la solución a este ejercicio se hará con una interfase que nos permitirá que nuestro corredor utilice cualquier tipo de zapato para correr y que no tengo ningún problema al hacer lo.



Principio Liskov Substitution (Substitucion de Liskov)

En el ejemplo del principio liskov utilizamos una clase general llamada carro, adentro de ella una función de echar gasolina, de la clase car Se extendió a clase Toyota Corola 2018 y clase tesla model s nos dimos cuenta que el tesla model s no podía recibir la función de recargar gasolina ya que este es un carro eléctrico Así que la función o la solución en este caso fue crear una nueva clase que extendiera al tesla modelo s llamada ElectricCar con la funcion de recargar el carro

+

Principio Interface-Segregation

En este ejemplo creamos una tienda de zapatos, en la interfase habían funciones como talla, color, marca y precio estos se van a extender hacia la clase zapatos pero de repente se puso a vender también accesorios para los zapatos y ropa. Estos ultimos 2 no cumplen con las mismas funciones como de talla, color, marca y precio. Así que la solución fue crear nuevas interfaces con funciones específicas para estas nuevas clases que son accesorios y ropa tengas espicifcamente sus funciones y de esa manera hicimos que nuestro código sea más limpio y que pueda seguir creciendo y mejorando



Principio Dependency-inversion (Inversion de dependencia)

En este ejemplo creamos un supermercado en el cual habían clases como arroz, frijoles, CornFlakes, ChocoCrispi, Carne de Cerdo y Carne Pollo los cuales vamos a utilizar con un constructor, pero nos dimos cuenta que al agregar estas clases a constructor a la hora que quisiéramos agregar más productos en nuestros supermercados dañaria nuestro código y también íbamos a violar una de las normas de SOLID qué es Open Close ya que íbamos a estar modificando nuestro código en cada momento, Así que la solución fue crear nuevas interfaces que pudieran agrupar por clase a los productos ejemplo frijol y arroz van a ir en la interfaz de granos básicos; CornFlakes y ChocoCrispi van en la interfaz de cereales y carne de pollo y carne de cerdo y van en la interfaz de carnes. Asi podremos agregar nuevos productos a nuestro supermercado y no tendremos ningún problema con nuestro código y que no dañara la funcionalidad.

