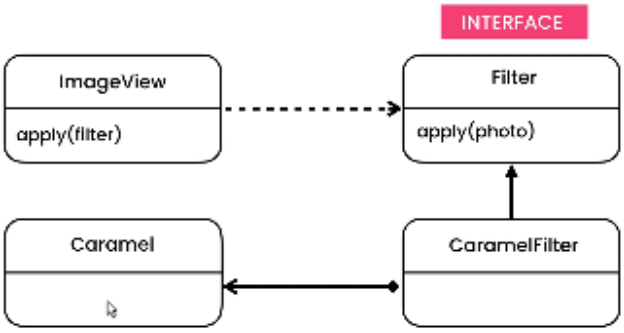
# **Adaptor** Design Pattern

This pattern allows us to convert the interface of an object to a different form.

## Class Diagrams




## Explanation

In this example, we want to create a mobile app for applying image filters and there is a need to use a third-party library. The Image class is a standard class for work with images. The VividFilter class is a real filter that implements the Filter interface. the ImageView class contains an image and allows us to apply a different filter to it. The Caramel class would apply a caramel and it is in the third-party library. Caramel class does not implement the Filter interface and it doesn't have the apply() method. So, there is a need for CaramelFilter class that implements the Filter interface and it forwards the request to a Caramel object. In this implementation, we used composition, because our CaramelFilter is composed of Caramel object.