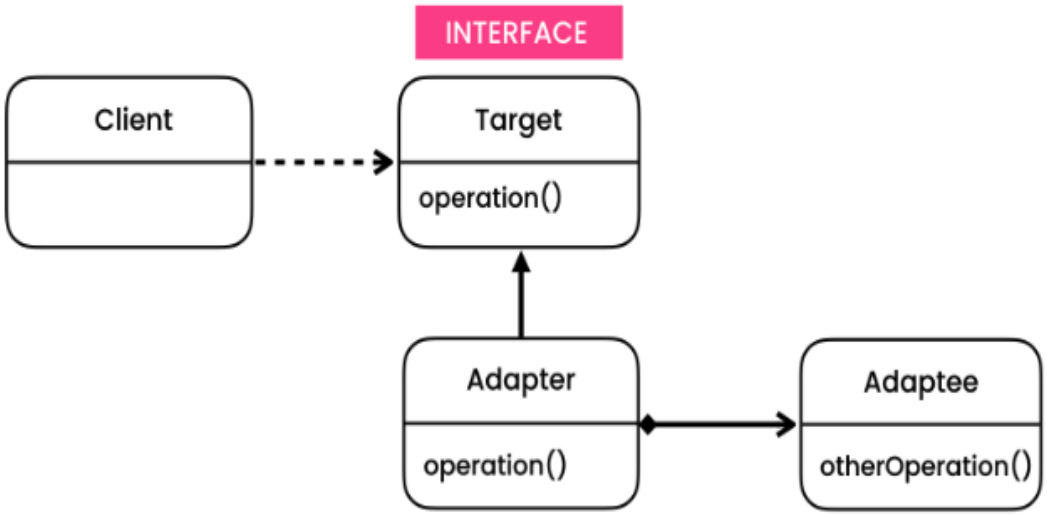
Adapter Pattern:

Definition: Allows converting the interface of a class into another interface that clients expect.



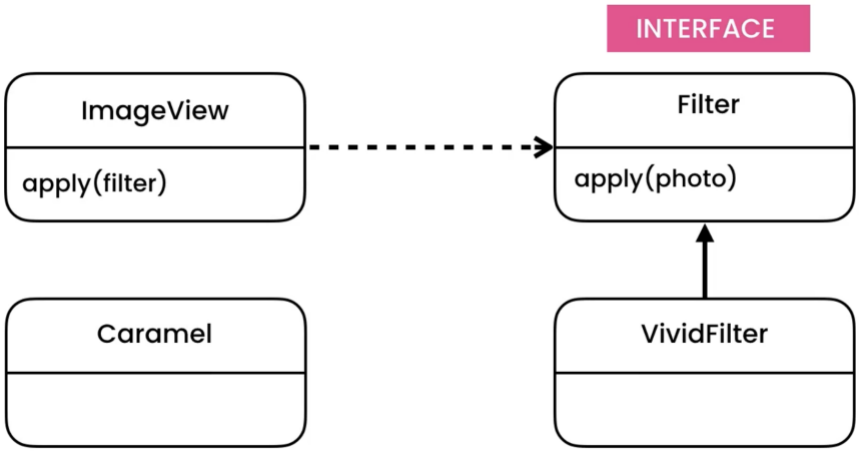
Scenario:

We want to implement apply filters functionality on images. Our all new filters should implement the interface Filter, but we don't want to re-implement already made filters e.g. Caramel which already implements render image. We want to use its implementation in our solution.

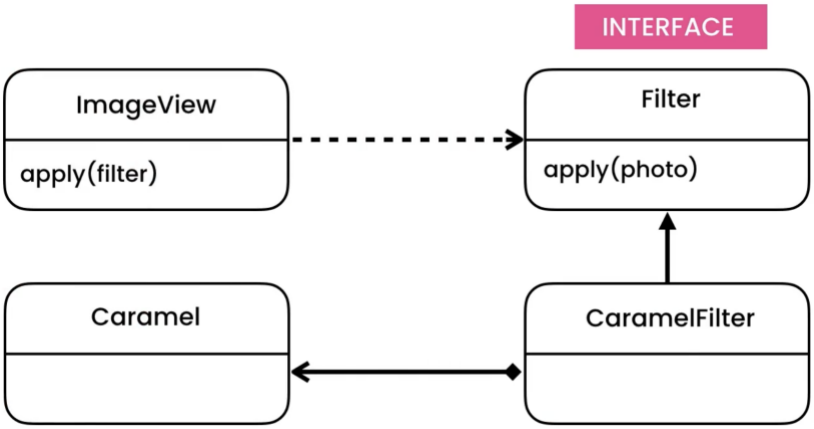
Implementation:

```
public interface Filter {
    void apply (Image image);
}

public class Caramel {
    public void init(){

    }

    public void render(Image image){
        System.out.println("Applying caramel filter");
    }
}
```

Problem: We need to use some existing thing but that class is not implementing our interface.



Solution: We adapt to it by creating our adapter and use its functionality. We will create a class **CaramelAdapter** which implements our interface. In our adapter, we will construct Caramel object (with composition) and use its functionality.




```
public class CaramelAdapter implements Filter {
    private Caramel caramel;
    @Override
    public void apply(Image image) {
        caramel.init();
        caramel.render(image);
    }
}
```