Adapter

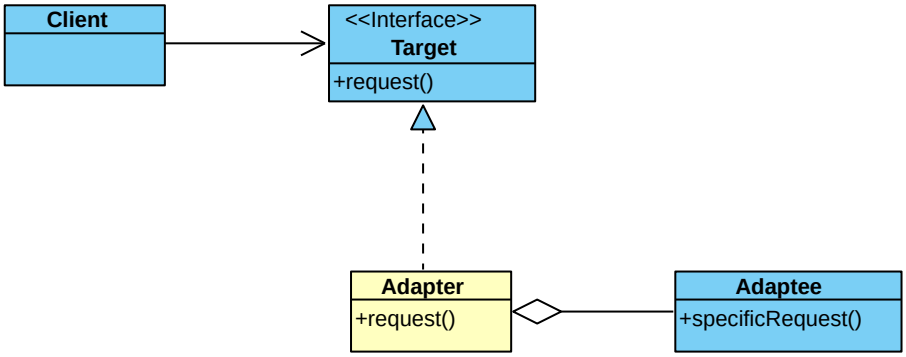
Category : structural

It allows to convert an interface of a class to another interface expected by the client.

Adapter makes it work together classes that could not work because of incompatibility between interfaces.

The adapter design pattern solves problems like :

- How can a class be reused that does not have an interface that a client requires ?
- How can classes that have incompatible interfaces work together ?
- How can an alternative interface be provided for a class ?



Example : adapt a square to a rectangle

Adaptee

```
public class Square {

    private Integer side;

    public Square(Integer side) {
        this.side = side;
    }

    public Integer getSide() {
        return side;
    }

    public Integer calculateArea() {
        return side * side;
    }
```

```
}
```

Target

```
public interface Rectangle {

    Integer getWidth();
    Integer getHeight();
}
```

Adapter

```
public class Adapter implements Rectangle {

    private Square square;

    public Adapter(Square square) {
        this.square = square;
    }

    @Override
    public Integer getWidth() {
        return square.getSide();
    }

    @Override
    public Integer getHeight() {
        return square.getSide();
    }
}
```

Client

```
public class Client {

    private static Rectangle rectangle;

    public static void main(String[] args) {

        Square square = new Square(5);
        rectangle = new Adapter(square);
        System.out.println("Rectangle width = " + rectangle.getWidth());
        System.out.println("Rectangle height = " + rectangle.getHeight());
    }
```

```
}
```