Professorship of Embedded Systems and Internet of Things
Department of Electrical and Computer Engineering
Technical University of Munich

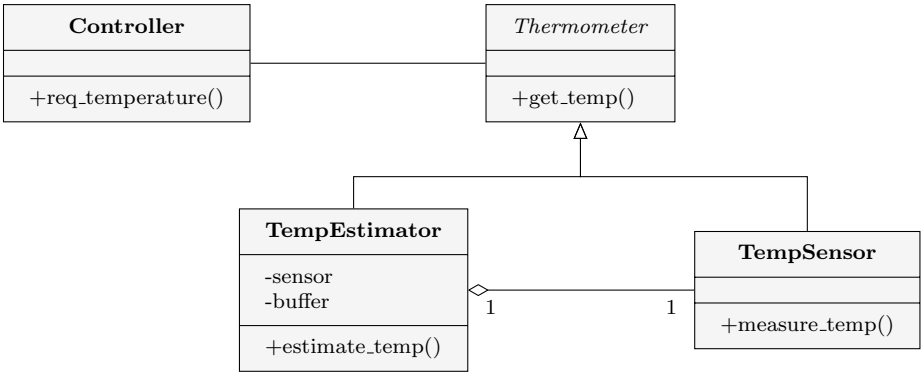
Exercise Sheet 3

Software Architecture for Distributed Embedded Systems, WS 2021

Prof. Steinhorst, M. Sc. Regnath, M.Sc. Ernstberger

? Exercise 3.1: Patterns, Patterns, Patterns

1. Which categories of software patterns exists and what are the differences between them?
2. Name two patterns for each of the three categories.
3. A message buffer is an often used pattern in communication. Which category of pattern would you assign it?
4. Which pattern is shown in the following UML diagram? What is the idea here?



5. When would you use the *Facade* pattern? Which problem does it solve?

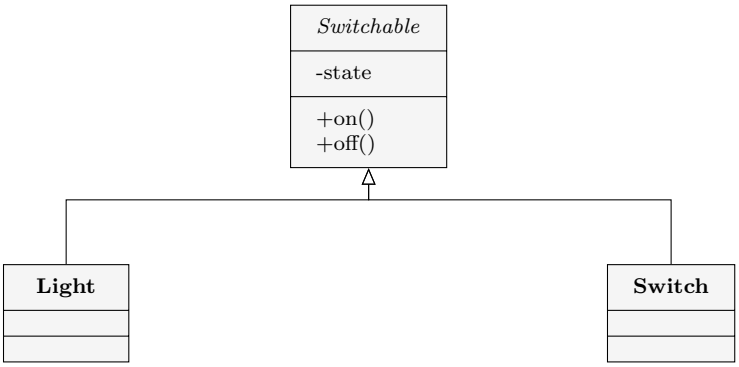
? Exercise 3.2: Patterifying the Code

We want to improve the software of our Startup *AwesomeLights* from exercise 2.

1. Look at the following code. In this example our button wants to connect to a light using sockets. For security reasons, the connect command only accepts an IP address in a specific data format and not as string. Which design pattern would help here? Add your own code using the correct design pattern to make the connection work (see Moodle for full file).

```
class IPv4_Address():
    def __init__(self, byte1, byte2, byte3, byte4):
        self.bytes = (byte1, byte2, byte3, byte4)

class Socket():

    def connect( self, ip_addr ):
        if isinstance(ip_addr, IPv4_Address): # check the correct format
            print("Socket: connecting to {}".format( str(ip_addr) ))
        else:
            print("Socket: Invalid IP format!")

light_ip = "192.168.1.42"

# connecting to the light
sock = Socket()
sock.connect( light_ip )  # THIS WILL FAIL
```

2. From a survey you have figured out that customers already have smart home solution that use Bluetooth or Zigbee instead of WiFi. You decide that it would be a good idea to offer switches and lights for each wireless standard (WiFi, Bluetooth, Zigbee). Extend the following UML diagram of the situation if you would need to support every combination individually, e.g. use classes such as `WiFiSwitch`, `ZigbeeLight`, etc that provide the functions `send()` and `receive()`.



3. Redraw the UML using the bridge pattern. Make use of the class `Switchable` as well as `WirelessSwitchable`, which provide the functions `send()` and `receive()`.

4. Refactor the following code and implement the bridge pattern according to the UML diagram in the previous question (see Moodle for full file).

```python
# Defining the Wireless Protocols
class WiFi():
    def sendTCP(self, msg):
    def recvTCP(self):

class Bluetooth():
    def request_service(self, srv):
    def handle_service(self):

class ZigBee():
    def sendIEEE(self, msg):
    def recvIEEE(self):


# Base Classes
class Switchable():
    def on(self):  raise NotImplementedError
    def off(self): raise NotImplementedError

# Derived Classes for all combinations
class WiFiSwitch(Switchable):
    def __init__(self, name):
        self.name = name
        self._wifi = WiFi()

    def on(self):
        self._wifi.sendTCP("ON")

    def off(self):
        self._wifi.sendTCP("OFF")

# ...
class WiFiLight(Switchable)
class BluetoothSwitch(Switchable)
class BluetoothLight(Switchable)
class ZigbeeSwitch(Switchable)
class ZigbeeLight(Switchable)
```

5. What is the difference between Adapter and Bridge pattern?