Logbook 16 - Command Pattern

Below is an example of controling a light without using the command pattern

In [3]:
```
# Command interface with a single method execute()
class Command:
def execute(self):
pass
# Receiver class
class Light:
# Method to turn the light on
def turn_on(self):
print("The light is on")
# Method to turn the light off
def turn_off(self):
print("The light is off")
# ConcreteCommand to turn on the light
class TurnOnCommand(Command):
def __init__(self, light):
self._light = light
# The execute method calls the receiver's turn_on method
def execute(self):
self._light.turn_on()
# ConcreteCommand to turn off the light
class TurnOffCommand(Command):
def __init__(self, light):
self._light = light
# The execute method calls the receiver's turn_off method
def execute(self):
self._light.turn_off()
# Invoker class
class RemoteControl:
# Method to execute any command
def submit(self, command):
command.execute()
# Client code
# Receiver
light = Light()
# Command to turn light on
turnOnCommand = TurnOnCommand(light)
# Command to turn light off
turnOffCommand = TurnOffCommand(light)
# Invoker
remote = RemoteControl()
# Turn the light on
remote.submit(turnOnCommand)
# Turn the light off
remote.submit(turnOffCommand)
```

```
The light is on
The light is off
```

Below is an example of controling a light with the command pattern

In [2]:

```python
# Command interface with a single method execute()
class Command:
    def execute(self):
        pass

# Receiver class
class Light:
    # Method to turn the light on
    def turn_on(self):
        print("The light is on")

    # Method to turn the light off
    def turn_off(self):
        print("The light is off")

# ConcreteCommand to turn on the light
class TurnOnCommand(Command):
    def __init__(self, light):
        self._light = light

    # The execute method calls the receiver's turn_on method
    def execute(self):
        self._light.turn_on()

# ConcreteCommand to turn off the light
class TurnOffCommand(Command):
    def __init__(self, light):
        self._light = light

    # The execute method calls the receiver's turn_off method
    def execute(self):
        self._light.turn_off()

# Invoker class
class RemoteControl:
    # Method to execute any command
    def submit(self, command):
        command.execute()



# Client code
# Receiver
light = Light()
# Command to turn light on
turnOnCommand = TurnOnCommand(light)
# Command to turn light off
turnOffCommand = TurnOffCommand(light)

# Invoker
remote = RemoteControl()
# Turn the light on
remote.submit(turnOnCommand)
# Turn the light off
remote.submit(turnOffCommand)
```

```
The light is on
The light is off
```