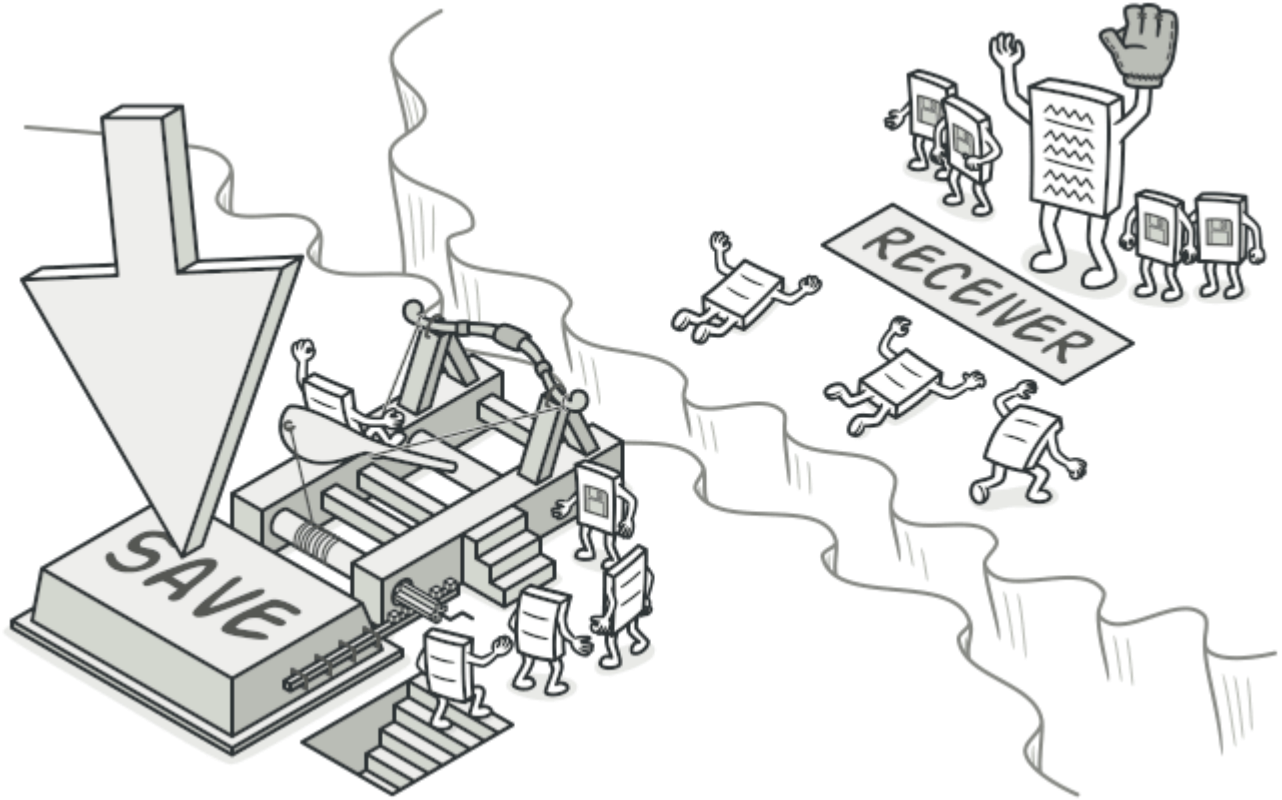
COMMAND

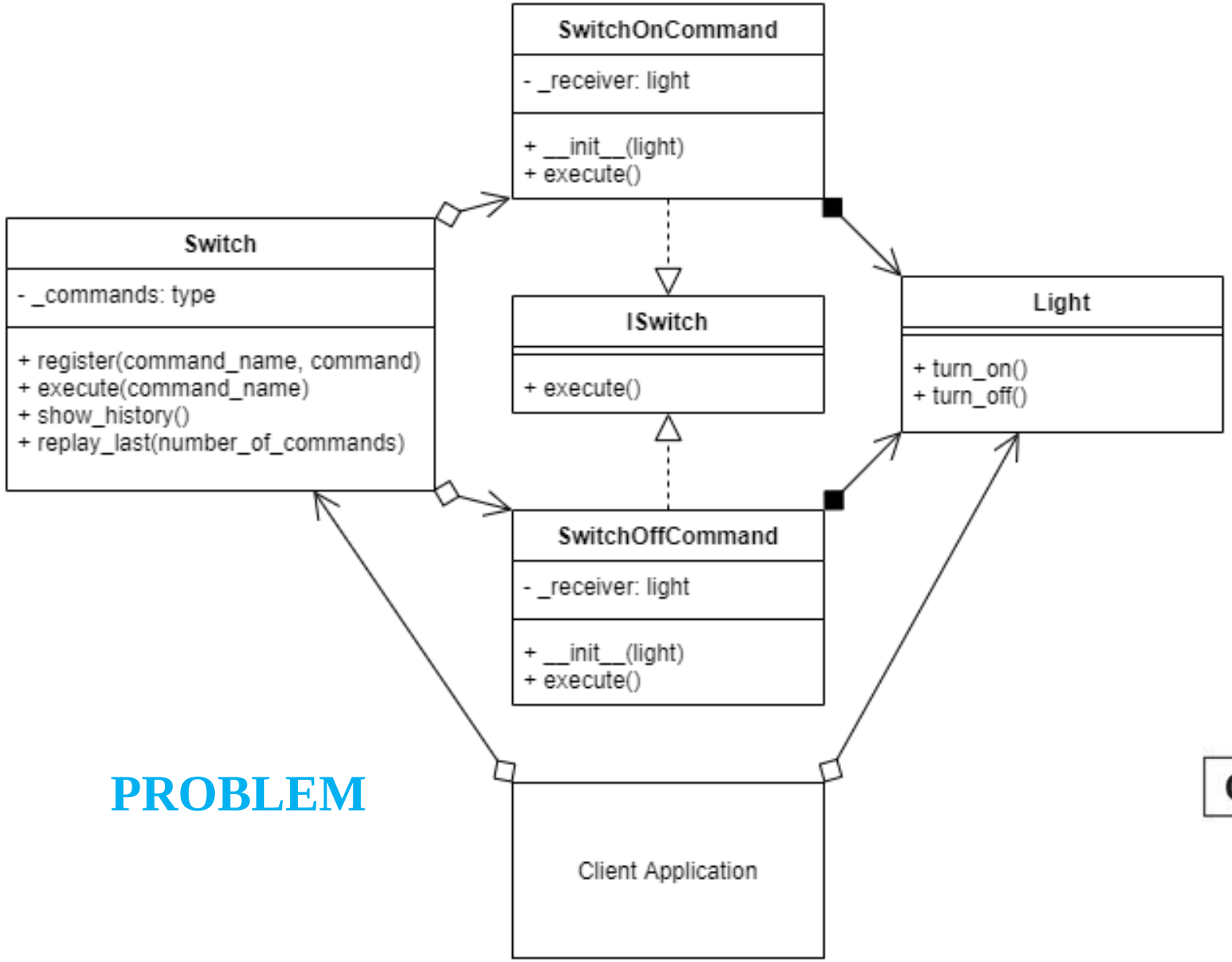
- This Design pattern helps the program to react based on **Command** from the user**.**
- The Command Pattern uses an object to represent a command, which can be call upon later by an invoker when needed.

INTENT

- Turns a request into a stand-alone object that contains all information about the request thereby letting you parametrize clients with different requests, queue or log requests, and support undoable operations.
- An object-oriented callback

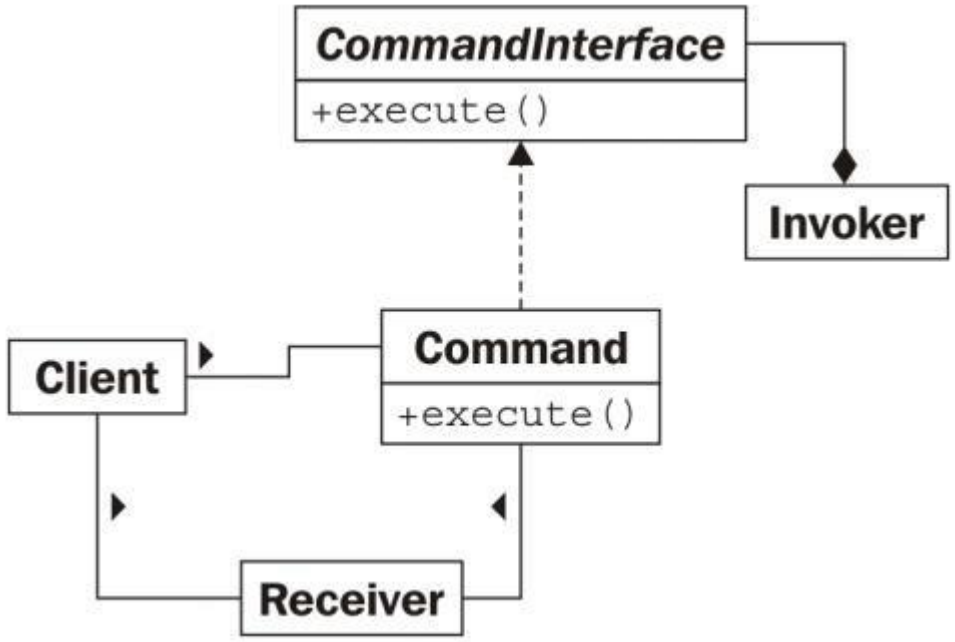


Courtesy : https://refactoring.guru/design-patterns



PROBLEM

CLASS DIAGRAM



```
"""Command Pattern"""
from abc import ABCMeta, abstractmethod


# Receiver
class Light:
    def switchon(self):
        print("Light is on")

    def switchoff(self):
        print("Light is off")
```

```
# Command
class ICommand(metaclass=ABCMeta):

    @abstractmethod
    def execute(self):
        """static interface"""


# Switch ON Command
class oncommand(ICommand):
    def __init__(self, light):
        self._light = light

    def execute(self):
        self._light.switchon()


# Switch OFF Command
class offcommand(ICommand):
    def __init__(self, light):
        self._light = light

    def execute(self):
        self._light.switchoff()
```

```
#Invoker
class Switch:
    def __init__(self):
        self._input = {}

    def register(self, name, command):
        self._input[name]=command

    def execute(self, name):
        if name in self._input:
            self._input[name].execute()
        else:
            print("Enter valid command")
```

```
#Client Receiver
Led = Light()          #class Light

# Create Commands
SWITCH_ON = oncommand(Led)
SWITCH_OFF = offcommand(Led)

# Register the commands with the invoker
SWITCH = Switch()        #class Switch
SWITCH.register("ON", SWITCH_ON)
SWITCH.register("OFF", SWITCH_OFF)

# Execute the commands that are registered on
the Invoker
a = input("Do you want to turn on lights? - Y or
N")
a = a.strip()
if a == "Y" or a == "y":
   SWITCH.execute("ON")
else:
   SWITCH.execute("OFF")
```

OUTPUT

```
Do you want to turn on lights? - Y or N y
Light is on
```

```
Do you want to turn on lights? - Y or N n
Light is off
```