Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA: 3.18)

Department of Information Technology

COURSE CODE: DJ19ITC802 **DATE:**21-02-2024

COURSE NAME: Design Patterns Laboratory **CLASS:** BE – IT I1-2

NAME: Rishikesh Sharma **SAP ID:** 60003200102

EXPERIMENT NO. 4

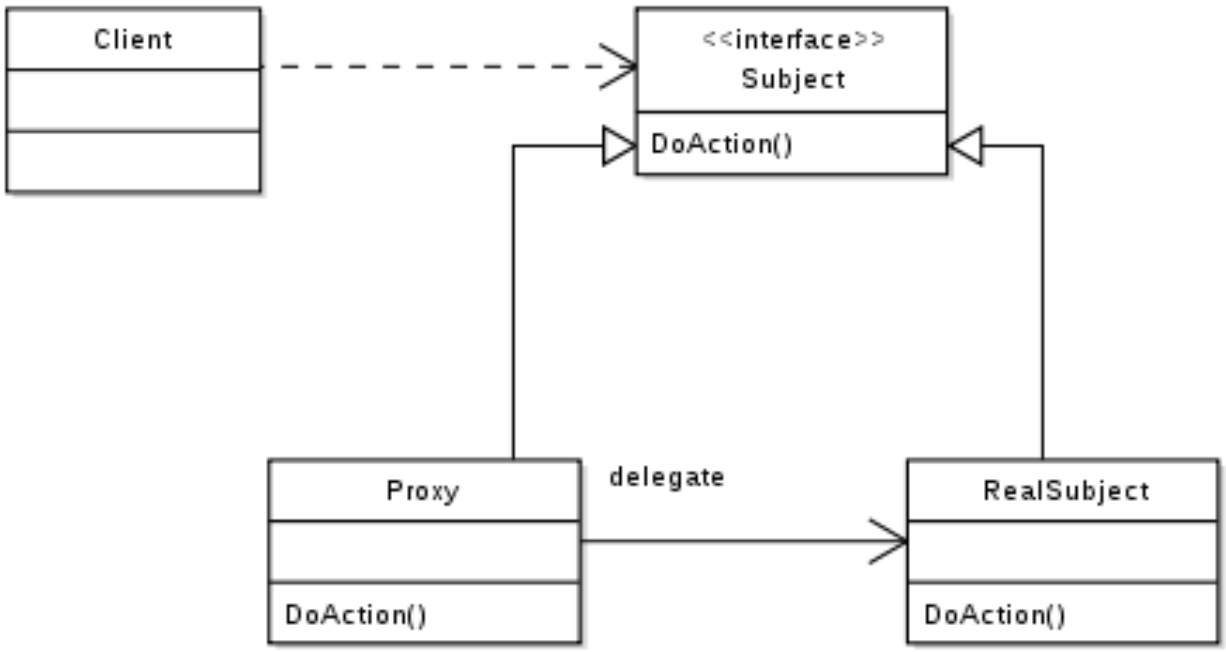
CO/LO: Identify and apply the most suitable design pattern to address a given application design problem.

AIM: Design a UML class diagram for Proxy Pattern and implement the same for any real-life scenario.

DESCRIPTION:

In proxy pattern, a class represents functionality of another class. This type of design pattern comes under structural pattern. In proxy pattern, we create object having original object to interface its functionality to outer world.

SOURCE CODE:



UML Diagram for Proxy Pattern

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA: 3.18)

Department of Information Technology

CODE:

```
from abc import ABC, abstractmethod
class FileLoader(ABC):
    @abstractmethod
    def load_file(self, file_path):
        pass
class RealFileLoader(FileLoader):
    def load_file(self, file_path):
        print(f"Loading file from {file_path}...")
        # Load file from disk and return its contents
        with open(file_path, 'r') as f:
            return f.read()
class VirtualFileLoader(FileLoader):
    def __init__(self):
        self._real_loader = None
    def load_file(self, file_path):
        if not self._real_loader:
            self._real_loader = RealFileLoader()
        print(f"Loading file (virtual) from {file_path}...")
        return self._real_loader.load_file(file_path)
# Client code
file_path = "example.txt"
real_loader = RealFileLoader()
real_file_content = real_loader.load_file(file_path)
print(real_file_content)
virtual_loader = VirtualFileLoader()
virtual_file_content = virtual_loader.load_file(file_path)
print(virtual_file_content)
```

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA: 3.18)

Department of Information Technology

OUTPUT

```
Loading data from database...
Data loaded successfully.
Accessing data through proxy...
Accessing data through proxy...
Accessing data through proxy...
Caching data...
Retrieving data from cache...
Data retrieved: {'id': 1, 'name': 'John Doe', 'age': 30}
```

CONCLUSION:

We have a FileLoader abstract class that defines the load_file() method. We also have a RealFileLoader class that implements the FileLoader interface and loads files from disk. Finally, we have a VirtualFileLoader class that also implements the FileLoader interface, but it creates a RealFileLoader object on demand rather than immediately. In the client code, we create a RealFileLoader object and use it to load the contents of a file. We then create a VirtualFileLoader object and use it to load the same file. The VirtualFileLoader object creates a RealFileLoader object on the first call to load_file(), but subsequent calls reuse the same object. We print the contents of both files to show that they are the same.

REFERENCES:

[1] https://www.tutorialspoint.com/design_pattern/proxy_pattern.htm