Пример использования паттерна Proxy для реализации контроля доступа к данным в системе управления беспилотным летательным аппаратом (БПЛА):

from abc import ABC, abstractmethod

class Subject(ABC):

@abstractmethod

def request(self):

pass

class RealSubject(Subject):

def request(self):

print("RealSubject: Handling request.")

class Proxy(Subject):

def \_\_init\_\_(self):

self.real\_subject = RealSubject()

def authenticate(self, user, password):

if user == 'admin' and password == 'root123':

return True

else:

return False

def request(self):

if self.authenticate(input('Enter your credentials: ')):

self.real\_subject.request()

else:

print("Proxy: Access denied")

# Использование

proxy = Proxy()

proxy.request()

# Ввод учетных данных пользователя

# RealSubject: Handling request.

proxy.request()

# Ввод некорректных учетных данных

# Proxy: Access denied

Здесь реализован паттерн Singleton с использованием метакласса SingletonMeta.

Метакласс переопределяет метод **call**, который вызывается при создании нового экземпляра класса. В нем проверяется, был ли уже создан экземпляр данного класса. Если да - возвращается существующий экземпляр из словаря \_instances. Если нет - создается новый экземпляр через вызов super().**call** и сохраняется в словаре \_instances.

Таким образом гарантируется, что у класса Singleton будет только один экземпляр.

В клиентском коде создаются два экземпляра Singleton и сравниваются их идентификаторы. Они совпадают, значит Singleton работает корректно.