**Шаг 1: Создание интерфейса класса SomeObject с методами, предоставляющими доступ к данным.**

python

Копировать код

from abc import ABC, abstractmethod

class SomeObject(ABC):

@abstractmethod

def get\_data(self):

pass

@abstractmethod

def set\_data(self, data):

pass

**Шаг 2: Реализация класса Proxy, который реализует интерфейс SomeObject. В его методах добавим проверку прав доступа перед вызовом методов реального объекта.**

python

Копировать код

class RealObject(SomeObject):

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

self.\_data = None

def get\_data(self):

return self.\_data

def set\_data(self, data):

self.\_data = data

class Proxy(SomeObject):

def \_\_init\_\_(self, real\_object, user):

self.\_real\_object = real\_object

self.\_user = user

def get\_data(self):

if self.\_user.has\_permission("get\_data"):

return self.\_real\_object.get\_data()

else:

raise PermissionError("User does not have permission to get data")

def set\_data(self, data):

if self.\_user.has\_permission("set\_data"):

self.\_real\_object.set\_data(data)

else:

raise PermissionError("User does not have permission to set data")

class User:

def \_\_init\_\_(self, name, permissions):

self.name = name

self.permissions = permissions

def has\_permission(self, permission):

return permission in self.permissions

**Шаг 3: Реализация класса SecureProxy, также реализующего интерфейс SomeObject. В его методах добавим дополнительные проверки безопасности.**

python

Копировать код

class SecureProxy(SomeObject):

def \_\_init\_\_(self, real\_object, user):

self.\_real\_object = real\_object

self.\_user = user

def get\_data(self):

if self.\_user.has\_permission("get\_data") and self.\_check\_security():

return self.\_real\_object.get\_data()

else:

raise PermissionError("Security check failed or user does not have permission to get data")

def set\_data(self, data):

if self.\_user.has\_permission("set\_data") and self.\_check\_security():

self.\_real\_object.set\_data(data)

else:

raise PermissionError("Security check failed or user does not have permission to set data")

def \_check\_security(self):

# Дополнительная проверка безопасности, например, проверка двухфакторной аутентификации

# Здесь добавим пример проверки: пользователь должен быть "admin"

return self.\_user.name == "admin"

**Классы для работы с беспилотниками:**

python

Копировать код

class DroneInterface(ABC):

@abstractmethod

def fly(self):

pass

@abstractmethod

def land(self):

pass

class Drone(DroneInterface):

def fly(self):

print("Drone is flying")

def land(self):

print("Drone is landing")

class SecureDroneProxy(DroneInterface):

def \_\_init\_\_(self, drone, user):

self.\_drone = drone

self.\_user = user

def fly(self):

if self.\_check\_access() and self.\_check\_security():

self.\_drone.fly()

else:

raise PermissionError("Access denied or security check failed")

def land(self):

if self.\_check\_access() and self.\_check\_security():

self.\_drone.land()

else:

raise PermissionError("Access denied or security check failed")

def \_check\_access(self):

return self.\_user.has\_permission("fly")

def \_check\_security(self):

return self.\_user.name == "admin"

**Пример использования:**

python

Копировать код

def main():

# Создаем пользователя и проверяем права доступа

user1 = User("admin", ["get\_data", "set\_data", "fly"])

user2 = User("guest", ["get\_data"])

# Создаем реальный объект и прокси

real\_object = RealObject()

proxy = Proxy(real\_object, user1)

secure\_proxy = SecureProxy(real\_object, user1)

# Работа с данными через прокси

proxy.set\_data("some data")

print(proxy.get\_data())

try:

secure\_proxy.set\_data("secure data")

print(secure\_proxy.get\_data())

except PermissionError as e:

print(e)

# Работа с беспилотником через прокси

drone = Drone()

secure\_drone\_proxy = SecureDroneProxy(drone, user1)

secure\_drone\_proxy.fly()

secure\_drone\_proxy.land()

try:

secure\_drone\_proxy = SecureDroneProxy(drone, user2)

secure\_drone\_proxy.fly()

except PermissionError as e:

print(e)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Пояснение:**

1. **SomeObject** - интерфейс, определяющий методы get\_data и set\_data.
2. **RealObject** - класс, который реализует интерфейс SomeObject.
3. **Proxy** - класс, который проверяет права доступа пользователя перед вызовом методов RealObject.
4. **SecureProxy** - класс, который проверяет права доступа и выполняет дополнительные проверки безопасности перед вызовом методов RealObject.
5. **DroneInterface, Drone, SecureDroneProxy** - классы для работы с беспилотниками. SecureDroneProxy проверяет права доступа и выполняет дополнительные проверки безопасности перед вызовом методов Drone.

Этот пример демонстрирует использование различных паттернов проектирования для управления доступом и безопасностью при работе с данными и беспилотниками.