The proxy pattern

START

Team:

Phan Huỳnh Anh Thư - 17095
Võ Công Minh - 10421040

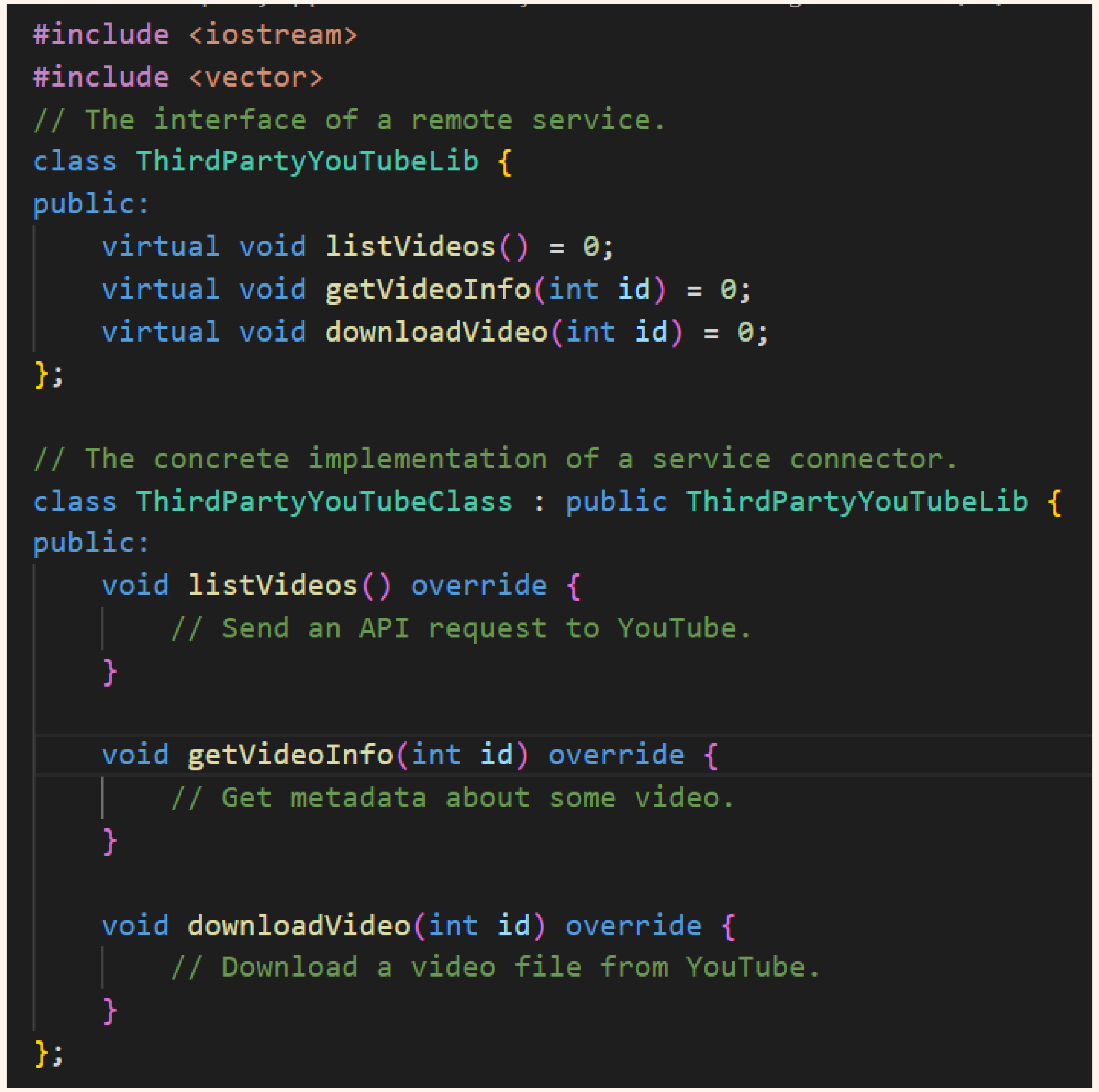
Real-world Analogy: Cash & Credit Card

Types of proxy pattern

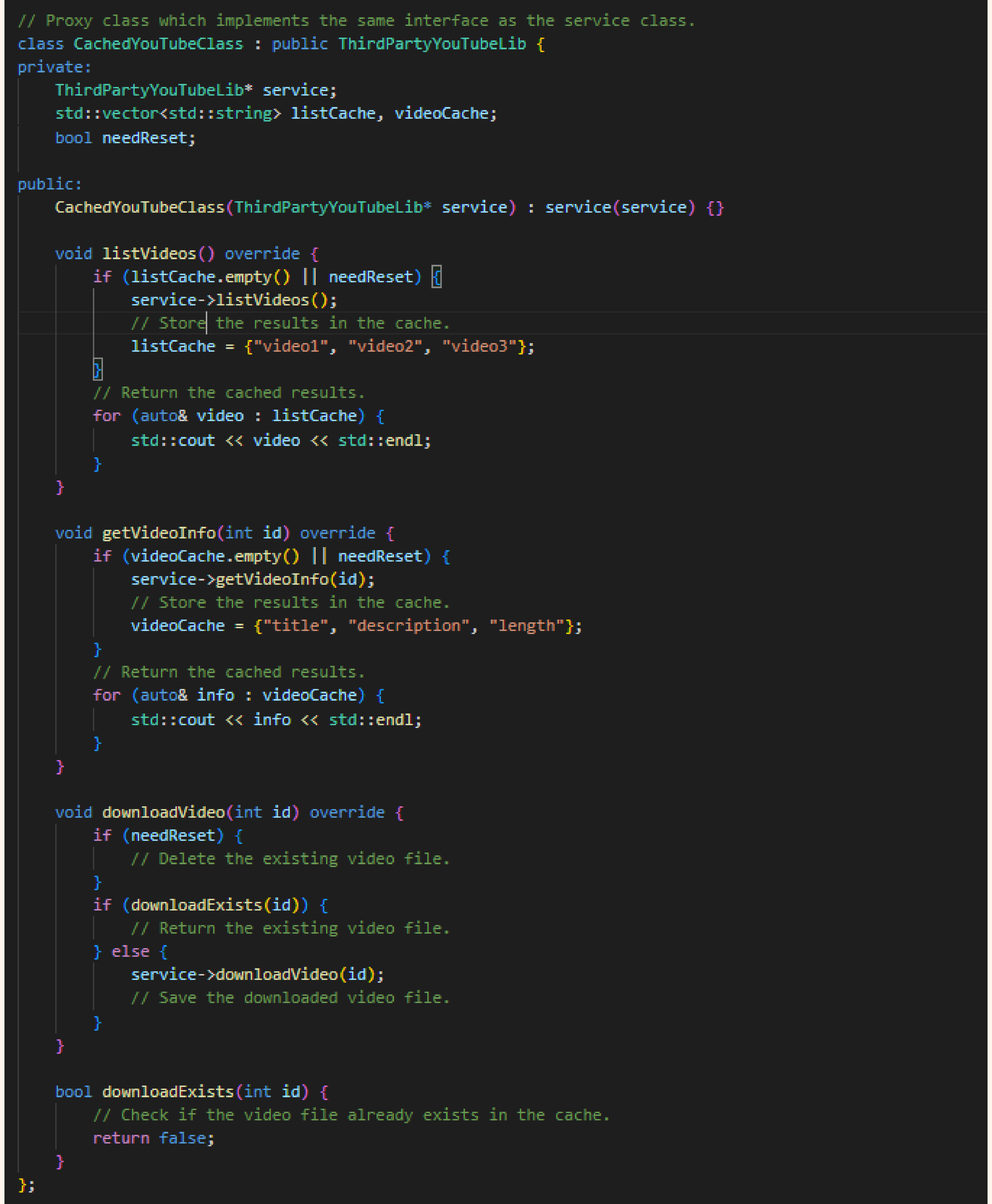
A protection proxy controls access to a subject based on access rights. It only allows access to authorized users, and blocks access for unauthorized users. For example, a proxy may authenticate users before allowing them to edit a document.

A remote proxy acts as an interface to an object in a different address space. The proxy communicates with the remote object to carry out operations. For example, proxies are often used to encapsulate remote procedure calls or web service calls.

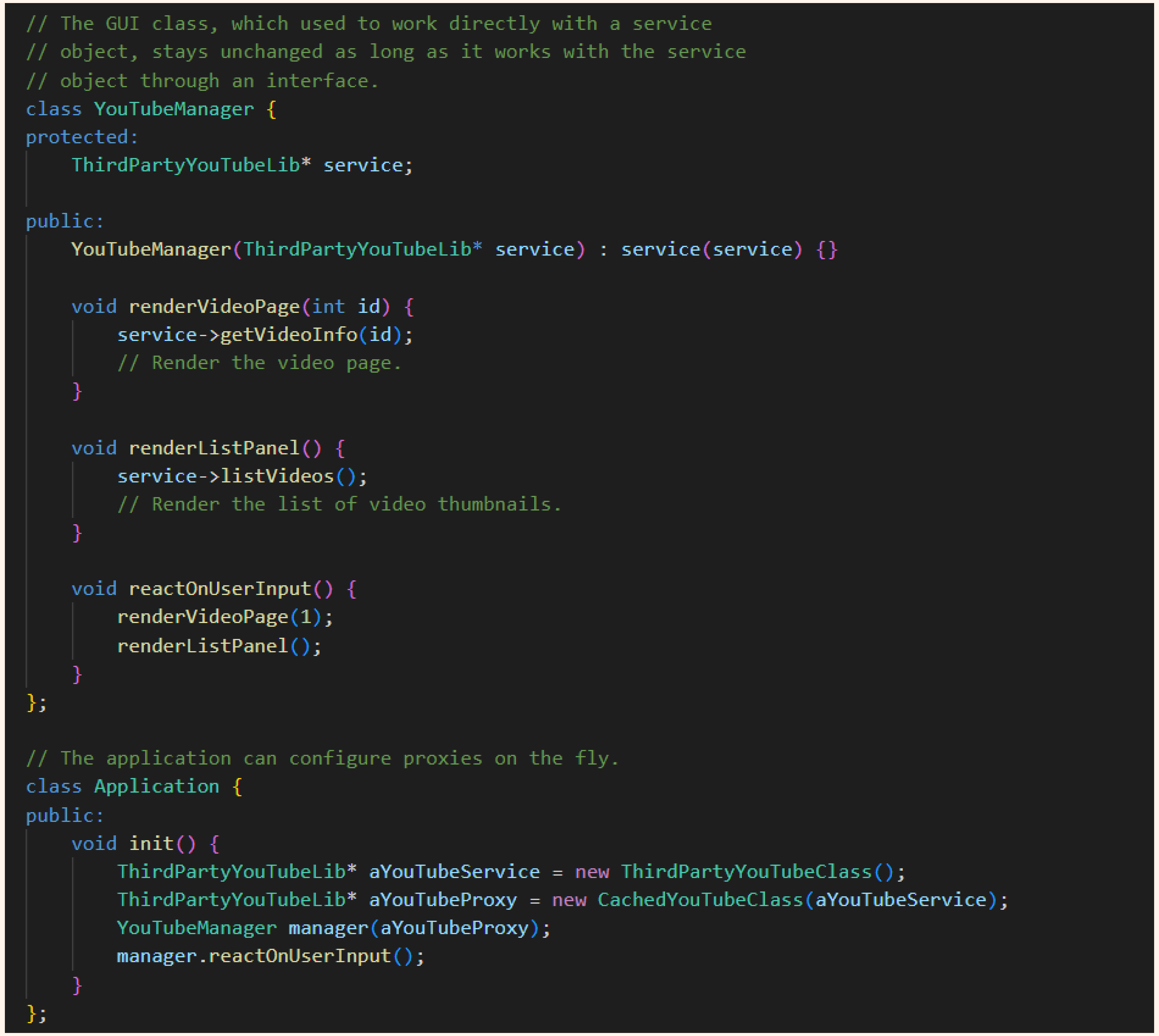
Implemetation



```cpp
#include <iostream>
#include <vector>
// The interface of a remote service.
class ThirdPartyYouTubeLib {
public:
    virtual void listVideos() = 0;
    virtual void getVideoInfo(int id) = 0;
    virtual void downloadVideo(int id) = 0;
};

// The concrete implementation of a service connector.
class ThirdPartyYouTubeClass : public ThirdPartyYouTubeLib {
public:
    void listVideos() override {
        // Send an API request to YouTube.
    }

    void getVideoInfo(int id) override {
        // Get metadata about some video.
    }

    void downloadVideo(int id) override {
        // Download a video file from YouTube.
    }
};
```



```cpp
// Proxy class which implements the same interface as the service class.
class CachedYouTubeClass : public ThirdPartyYouTubeLib {
private:
    ThirdPartyYouTubeLib* service;
    std::vector<std::string> listCache, videoCache;
    bool needReset;

public:
    CachedYouTubeClass(ThirdPartyYouTubeLib* service) : service(service) {}

    void listVideos() override {
        if (listCache.empty() || needReset) {
            service->listVideos();
            // Store the results in the cache.
            listCache = {"video1", "video2", "video3"};
        }
        // Return the cached results.
        for (auto& video : listCache) {
            std::cout << video << std::endl;
        }
    }

    void getVideoInfo(int id) override {
        if (videoCache.empty() || needReset) {
            service->getVideoInfo(id);
            // Store the results in the cache.
            videoCache = {"title", "description", "length"};
        }
        // Return the cached results.
        for (auto& info : videoCache) {
            std::cout << info << std::endl;
        }
    }

    void downloadVideo(int id) override {
        if (needReset) {
            // Delete the existing video file.
        }
        if (downloadExists(id)) {
            // Return the existing video file.
        } else {
            service->downloadVideo(id);
            // Save the downloaded video file.
        }
    }

    bool downloadExists(int id) {
        // Check if the video file already exists in the cache.
        return false;
    }
};
```



```cpp
// The GUI class, which used to work directly with a service
// object, stays unchanged as long as it works with the service
// object through an interface.
class YouTubeManager {
protected:
    ThirdPartyYouTubeLib* service;

public:
    YouTubeManager(ThirdPartyYouTubeLib* service) : service(service) {}

    void renderVideoPage(int id) {
        service->getVideoInfo(id);
        // Render the video page.
    }

    void renderListPanel() {
        service->listVideos();
        // Render the list of video thumbnails.
    }

    void reactOnUserInput() {
        renderVideoPage(1);
        renderListPanel();
    }
};

// The application can configure proxies on the fly.
class Application {
public:
    void init() {
        ThirdPartyYouTubeLib* aYouTubeService = new ThirdPartyYouTubeClass();
        ThirdPartyYouTubeLib* aYouTubeProxy = new CachedYouTubeClass(aYouTubeService);
        YouTubeManager manager(aYouTubeProxy);
        manager.reactOnUserInput();
    }
};
```

Avantages

1 Provides a placeholder for another object and can control access to it

2 Adds an extra level of indirection

3 Delay expensive object creation and provide lazy initialization

4 Restrict access to objects

Disavantages

1 Complicated

2 Proxy itself needs to be maintained

3 Performance

THANK YOU!