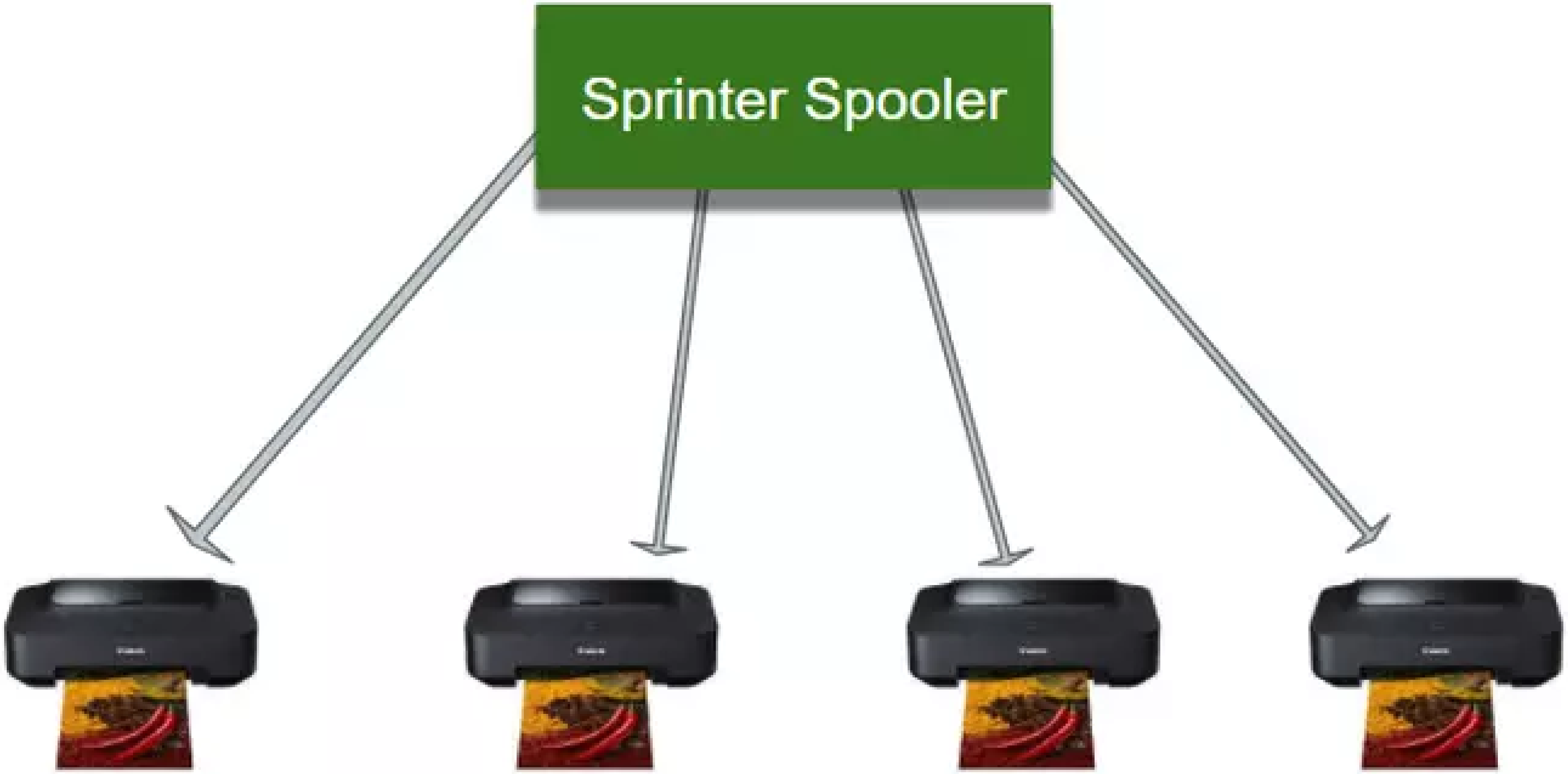
The Singleton Pattern

START

Team:

Phan Huỳnh Anh Thư - 17095
Võ Công Minh - 10421040

Example: System of Printers


Sprinter Spooler

Problems:

The Singleton Pattern ensures that there is only **one instance** of a class while providing a **global point of access** to this instance.

Without the Singleton Pattern, we might end up with multiple instances of a class, which can lead to issues such as inconsistent behavior or wasted resources.

Solution:

The Singleton Pattern is a creational pattern that ensures that a class has only one instance while providing a global point of access to this instance. This is achieved by using a **private constructor** to ensure that the class can only be instantiated once and a static method to provide global access to the instance.

Implemetation

```cpp
private:
    Printer() {}  // private constructor to prevent instantiation
    ~Printer() {}; // private detructor to prevent instantiation fro

    // delete copy constructor and assignment operator to prevent duplica
    Printer(const Printer&) = delete;
    Printer& operator=(const Printer&) = delete;

public:
    //public function to take out the instance
    static Printer& getInstance() {
        static Printer instance;
        return instance;
    }

    void print(const std::string& document){
        cout << "Printing: " << document << '\n';
    }

    static void destroyInstance() {
        Printer& printer = getInstance();
        delete &printer;
    }
```

Avantages

1 Ensuring there is only one instance of the class

2 Providing a global point of access to this instance

3 Allowing for lazy initialization of the instanc e

4 Allowing for thread-safe access to the instance

Disavantages

1 Violates the Single Responsibility Principle as the Singleton Pattern solves two problems at a time

2 The pattern requires special treatment in a multithreaded environment so that multiple threads won’t create a singleton object several times.

3 The Singleton pattern can mask bad design, for instance, when the components of the program know too much about each other.

4 Difficult to unit test the client code.

THANK YOU!