**Singleton**

* Ý định

Đảm bảo một lớp chỉ có một thể hiện , và thể hiện này có thể truy cập được từ bất cứ đâu

* Động lực

Điều quan trọng đối với một số lớp để có chính xác một thể hiện. Mặc dù có thể có nhiều máy in trong một hệ thống, chỉ nên có một máy in spooler.

Khi hiện thực singleton phải đảm bảo rằng chỉ có một thể hiện (instance) duy nhất được tạo ra và thể hiện tại có thể dùng mọi lúc mọi nơi. Nói cách khác, khi xây dựng lớp cần một kỹ thuật để có thể truy xuất được vào các thành viên public của lớp mà không tạo ra một thể hiện nào (thông qua việc khởi tạo bên ngoài lớp).

* Thể hiện
* Khi hiện thực singleton pattern phải đảm bảo rằng chỉ có một thể hiện (instance) duy nhất được tạo ra và thể hiện tại có thể dùng mọi lúc mọi nơi. Nói cách khác, khi xây dựng lớp cần một kỹ thuật để có thể truy xuất được vào các thành viên public của lớp mà không tạo ra một thể hiện nào (thông qua việc khởi tạo bên ngoài lớp).
* Để chắc chắn rằng, không có bất kỳ thể hiện nào bên ngoài lớp được tạo ra, constructor của lớp đó sẽ có phạm vi truy cập là private. Điều đó cũng có nghĩa là thể hiện duy nhất được tạo ra thông qua chính lớp mà chúng ta xây dựng. Vì vậy, ta sử dụng thành phần static để thực hiện điều này. Ta tạo một thể hiện của lớp kiểu private static và một refractor trả về đối tượng thuộc chính lớp đó.
* Trong C++, singleton định nghĩa một thuộc tính static là một con trỏ trỏ tới thể hiện duy nhất của lớp đó.

Các lớp Singleton được khai báo là

class Singleton { public:

static Singleton\* Instance(); protected:

Singleton(); private:

static Singleton\* \_instance;

};

Việc thực hiện tương ứng là

Singleton\* Singleton::\_instance = 0;

Singleton\* Singleton::Instance () { if (\_instance == 0) {

\_instance = new Singleton;

}

return \_instance;

}