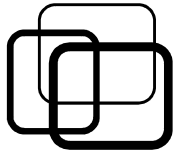
Mẫu Singleton

GV. Nguyễn Minh Huy

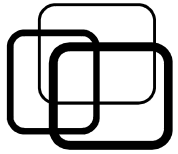
Mẫu Singleton



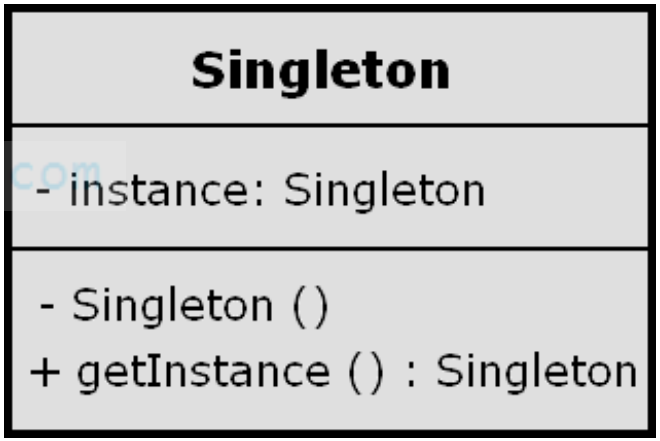
- Ngữ cảnh:
  - Bài toán:
    - Đối tượng Application trong ứng dụng.
    - Đối tượng Cookies trên Web Server.
    - Connection Pool.
  - Mục tiêu:
    - Một lớp chỉ cần có MỘT thể hiện (instance).

      ➔ Mở rộng: một lớp chỉ cần có xác định N thể hiện.
    - Các thể hiện này được chỉa sẻ trên toàn ứng dụng.

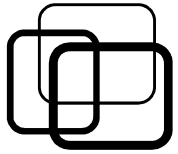
Mẫu Singleton



- Hướng tiếp cận:
  - Chia sẻ thể hiện của lớp trên toàn ứng dụng:
    - Dùng thuộc tính static để lưu một thể hiện của lớp.
  - Đảm bảo lớp chỉ có MỘT thể hiện:
    - Đưa constructor vào tầm vực private.
    - Cung cấp hàm static để trả về thể hiện đã lưu.
    - ➔ Điều khiển được quá trình khởi tạo đối tượng.



Mẫu Singleton



- Cài đặt:

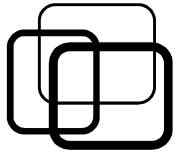
```
class Cookies
{
private:
        static Cookies* m_cookies;
        Cookies();
public:
        static Cookies* getCookies()
        {
              return m_cookies;
        }
};

// Khởi tạo thể hiện dùng chung.
Cookies* Cookies::m_cookies = new Cookies();
```

```
void main()
{
        Cookies *c1 = Cookies.getCookies();
        // Thêm cookies…
        // Bớt cookies…

        Cookies *c2 = Cookies.getCookies();
        // Thêm bớt cookies…
}
```

Mẫu Singleton



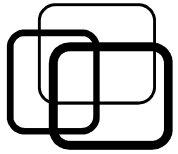
- Các vấn đề xung quanh:
  - Khởi tạo trễ (lazy loading):
    - Thuộc tính static được khởi tạo khi truy xuất lớp lần đầu.

      ➔ Sử dụng bộ nhớ không hiệu quả.
    - Cách thức khởi tạo trễ:
      - B1: Nếu chưa khởi tạo → Khởi thể hiện dùng chung.
      - B2: Trả về thể hiện dùng chung.


```
// Khởi tạo NULL.
Cookies* Cookies::m_cookies = NULL;

Cookies* Cookies::getCookies()
{
        if (m_cookies == NULL)
                m_cookies = new Cookies();
        return m_cookies;
}
```

Mẫu Singleton



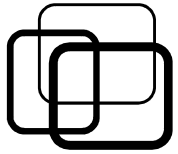
- Các vấn đề xung quanh:
  - Chia sẻ đa luồng (multi-threading):
    - Vấn đề truy xuất đồng thời trên Cookies::getCookies().

```
Cookies* Cookies::getCookies()
{
        if (m_cookies == NULL)
                m_cookies = new Cookies();
        return m_cookies;
}
```

Vùng không an toàn
Critical Section!!

    - Cách giải quyết:
      - Dùng Mutex.
      - Dùng Semaphore.

Mẫu Singleton



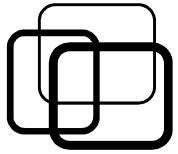
- Các vấn đề xung quanh:
  - Chia sẻ đa luồng (multi-threading):
    - Dùng Mutex trong Visual C++:

```
class Cookies
{
private:
        static Mutex * m_mutex;
        static Cookies* m_cookies;
        Cookies();
public:
        static Cookies* getCookies();
};
```

```
Mutex* Cookies::m_mutex = new Mutex();
Cookies* Cookies::m_cookies = NULL;

Cookies* Cookies::getCookies()
{
        mutex->WaitOne();
        if (m_cookies == NULL)
                m_cookies = new Cookies();
        mutex->ReleaseMutex();
        return m_cookies;
}
```

Mẫu Singleton

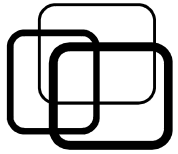


- Các vấn đề xung quanh:
  - Chia sẻ đa luồng (multi-threading):
    - Dùng synchronized trong Java:

```
class Cookies
{
        private static final Cookies m_cookies;
        private Cookies();

        public static synchronized Cookies getCookies()
        {
                if (m_cookies == null)
                        m_cookies = new Cookies();
                return m_cookies;
        }
};
```

Mẫu Singleton



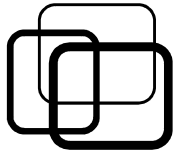
- Các vấn đề xung quanh:
  - Chia sẻ đa luồng (multi-threading):
    - Dùng inner class (Bill Pugh, University of Maryland):

```
class Cookies
{
        private static class CookiesHolder {
                public static final Cookies m_cookies = new Cookies();
        }

        private Cookies();

        public static Cookies getCookies()
        {
                return CookiesHolder.m_cookies;
        }
};
```

Mẫu Singleton



- Các vấn đề xung quanh:
  - Kiểm thử đơn vị (unit testing):
    - Cô lập các đơn vị mã nguồn để kiểm tra.
    - Singleton tạo ra Global State trong ứng dụng.
    - ➔ Gây khó khăn cho việc kiểm thử đơn vị.
  - Mẫu nên tránh (anti-pattern):
    - Tạo liên kết chồng chéo trong mã nguồn.
    - Liên kết chồng chéo bị che dấu.
    - Không an toàn trong ứng dụng đa luồng.
    - Tạo Global State làm kiểm thử khó khăn.
    - ➔ Dùng Singleton một cách cẩn trọng!!