1. **Synchronized vs Asynchronized**
2. **Synchronized Methods**

* Định nghĩa bằng cách them keyword “synchronized” trước kiểu trả về của hàm.
* Khi có 2 hay nhiều Threads được shared cùng một Object để thao tác, các threads sẽ đồng thời gọi phương thức của Object đó để thực thi
* Các methods không được định nghĩa “synchronized” thì các thread sẽ cùng truy cập vào them/sửa/xóa tại cùng 1 thời điểm, gây ra lỗi. Thêm từ khóa giúp đạt được sự giao tiếp giữa các thread, chỉ một thread được truy cập vào tài nguyên tại một thời điểm. Nó thực hiện việc giống như khóa tài nguyên chỉ định lại.

1. **Synchronized Blocks**

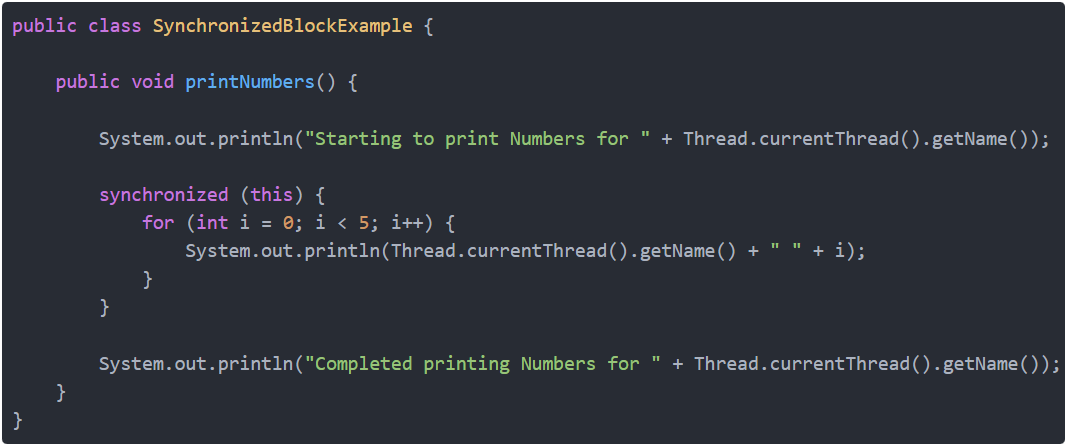
* Khi phương thức được đồng bộ hóa, các luồng sẽ bị giảm khả năng thực thi song song, điều đó ảnh hưởng đến hiệu năng của chương trình. Tuy nhiên, chúng ta có thể giảm số lượng mã được thực thi theo kiểu đồng bộ hóa bằng cách tạo một synchronized block. Thay vì đồng bộ hóa toàn bộ phương thức, chỉ cần đồng bộ hóa một vài dòng mã trong phương thức là được.

1. **Keyword “Synchronized”**

* Chỉ dùng trong method, static method hoặc block trong method.
* Synchronized(this, other params,…) {//…}

1. **Asynchronized**

* Nếu các luồng đều dùng chung một tài nguyên và phương thức dùng chung tài nguyên đó là mặc định thì sẽ dẫn đến Asynchronized.

****

1. **Asynchromous and Synchronous Callback**
2. Observer design pattern

A picture containing text, font, screenshot, algebra

Description automatically generated

* Cho phép Object được đăng kí và nhận các thông báo về các sự kiện hoặc trạng thái thay đổi của một Object khác.
* Các thành phần chính của Observer Design Pattern:

+ Subject: Đây là đối tượng đang được quan sát. Nó có một danh sách những observers quan tâm đến trạng thái hoặc sự kiện của nó.

+ Observer: Đây là đối tượng quan tâm đến trạng thái hoặc sự kiện của Subject. Nó đăng ký với Subject để nhận thông báo khi có thay đổi.

+ Concrete Subject: Đây là một triển khai của Subject interface. Nó duy trì một danh sách những Observer và thông báo cho họ khi có sự thay đổi trạng thái hoặc một sự kiện xảy ra.

+ Concrete Observer: Đây là một triển khai của Subject interface. Nó đăng ký với Subject để nhận thông báo và thực hiện một số hành động khi nhận được thông báo.

* Trong ví dụ:

+ Tạo một Subject interface và một Observer interface. Subject sẽ là đối tượng để một danh sách các Observer quan sát, các Observer sẽ nhận thông báo mỗi khi đối tượng Subject đó thay đổi thông tin (Trong VD này là thay đổi nhiệt độ).

+ Tạo một thể hiện WeatherStation triển khai từ Subject để các đối tượng Observer đăng kí với nó để quan sát.

+ Các Observer được khởi tạo sẽ đăng kí nó với Subject (Weather Station).

+ Các Observer sẽ lắng nghe cùng một thể hiện được khởi tạo bên trên.

+ Khi Subject thay đổi nhiệt độ, nó sẽ thông báo cho tất cả các Observer đã đăng kí ở trong danh sách của nó qua hàm setTemperature().

+ Muốn tạo task Async: tạo một thread mới trong hàm update dữ liệu hoặc thay đổi state của subject đang theo dõi trong mỗi observer. Một async task luôn được thực thi trong một thread mới. Tạo Thread giúp cho các code không bị block với nhau nên việc thực thi không phải đợi.\

+ Synchronous: Các đoạn code sẽ block nhau hoặc chờ cho sự kiện xảy ra xong rồi mới tiếp tục.

1. **Callback Function: Là** một function (Callee) được truyền vào một hàm khác (Caller) như một Argument và được kì vọng sẽ thực thi sau một số event. **Mục đích của Callback** là để thông báo cho một chức năng Sync/Async khi một công việc của một Class (Object hay Instance của Class) khác đã xong.

* Điều này rất hữu ích khi làm việc với các tác vụ Asynchronous. Giả sử chúng ta muốn thực hiện một số tác vụ thông thường như thực hiện một số thao tác hoặc hiển thị nội dung sau khi nhấp vào nút hoặc tìm nạp dữ liệu từ internet. Điều này cũng được sử dụng trong xử lý sự kiện, vì chúng tôi nhận được thông báo khi nhấp vào nút thông qua chức năng gọi lại.

1. Thread

* New 🡪 Runnable 🡪 Block 🡪 Waiting 🡪 Terminate. (Life Cycle)