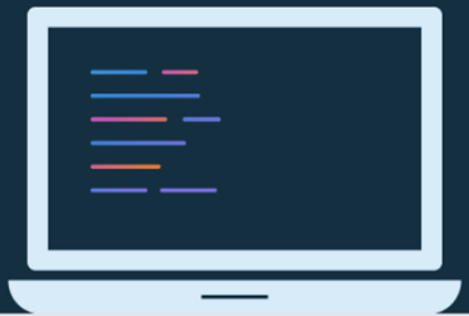
Bài học 7: Vòng đời của hoạt động và mảnh



Giới thiệu về bài học này

Bài học 7: Vòng đời của hoạt động và mảnh

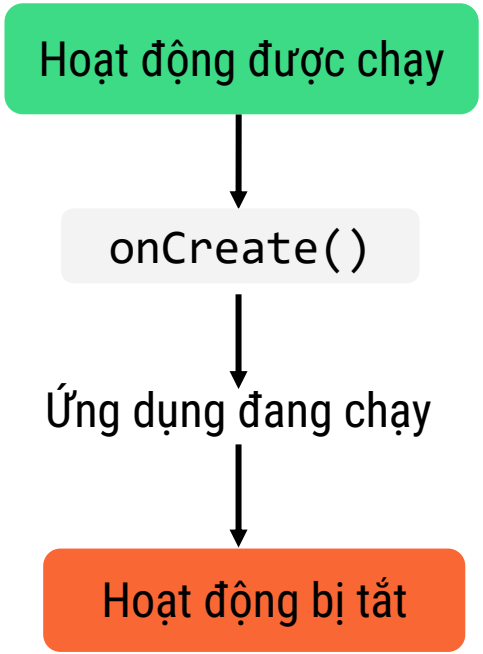
- Vòng đời của hoạt động
- Ghi nhật ký
- Vòng đời của mảnh
- Các thành phần nhận biết vòng đời
- Tác vụ và ngăn xếp lùi
- Tóm tắt

Vòng đời của hoạt động

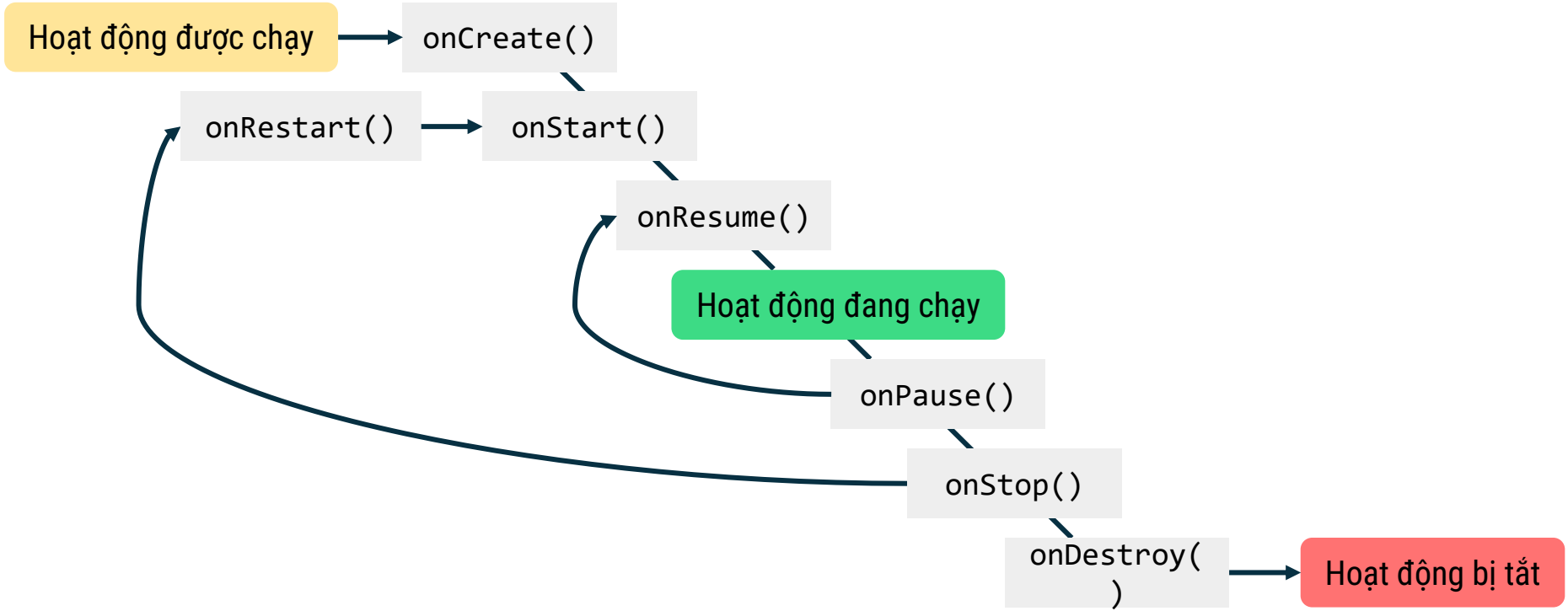
Tại sao vòng đời lại quan trọng

- Duy trì dữ liệu và trạng thái của người dùng nếu:
  - Người dùng tạm thời rời khỏi ứng dụng rồi quay lại
  - Người dùng bị gián đoạn (ví dụ: một cuộc gọi điện thoại)
  - Người dùng xoay thiết bị
- Tránh rò rỉ bộ nhớ và sự cố với ứng dụng.

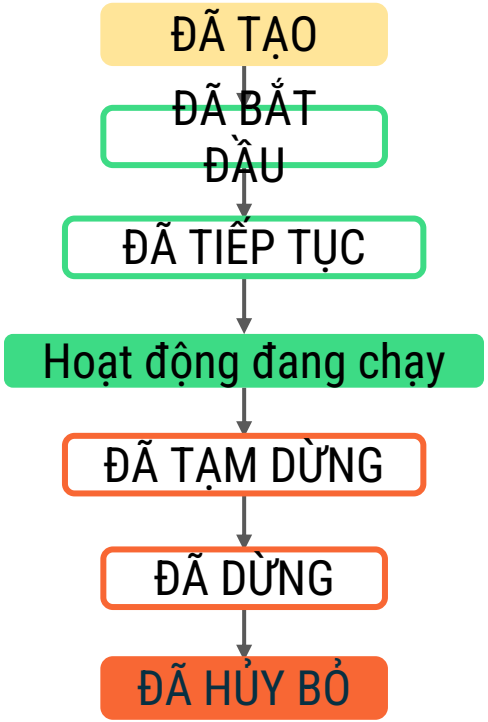
Vòng đời hoạt động được đơn giản hóa



Vòng đời của hoạt động



Trạng thái của hoạt động



onCreate()

- Hoạt động được tạo và tác vụ khởi tạo khác sẽ diễn ra
- Bạn phải triển khai lệnh gọi lại này
- Tăng cường giao diện người dùng của hoạt động và thực hiện logic khởi động ứng dụng khác

onStart()

- Hoạt động sẽ hiển thị với người dùng
- Được gọi sau hoạt động:
  - `onCreate()`

    hoặc
  - `onRestart()` nếu hoạt động bị dừng trước đó

onResume()

- Hoạt động lấy lại tiêu điểm nhập:
  - Người dùng có thể tương tác với hoạt động
- Hoạt động vẫn ở trạng thái đã tiếp tục cho đến khi hệ thống kích hoạt lệnh tạm dừng hoạt động

onPause()

- Hoạt động không còn là tiêu điểm (không chạy ở nền trước)
- Hoạt động vẫn hiển thị nhưng người dùng hiện không tích cực tương tác với hoạt động đó
- Trái ngược với `onResume()`

onStop()

- Hoạt động không còn hiển thị với người dùng
- Hủy bỏ các tài nguyên không cần thiết nữa
- Lưu mọi trạng thái cố định mà người dùng đang trong quá trình chỉnh sửa để họ không mất dữ liệu

onDestroy()

- Hoạt động sắp bị hủy bỏ, nguyên nhân có thể là do:
  - Hoạt động đã kết thúc hoặc bị người dùng đóng
  - Sự thay đổi về cấu hình
- Thực hiện thao tác dọn dẹp tài nguyên cuối cùng.
- Đừng dùng phương thức này để tiết kiệm dữ liệu người dùng (hãy làm điều đó sớm hơn)

Tóm tắt các trạng thái hoạt động

Trạng thái	Lệnh gọi lại	Nội dung mô tả
Đã tạo	`onCreate()`	Hoạt động đang được khởi tạo.
Đã bắt đầu	`onStart()`	Hoạt động hiển thị với người dùng.
Đã tiếp tục	`onResume()`	Hoạt động có tiêu điểm nhập.
Đã tạm dừng	`onPause()`	Hoạt động không có tiêu điểm nhập.
Đã dừng	`onStop()`	Hoạt động không còn hiển thị.
Đã hủy bỏ	`onDestroy()`	Hoạt động bị hủy bỏ.

Lưu trạng thái

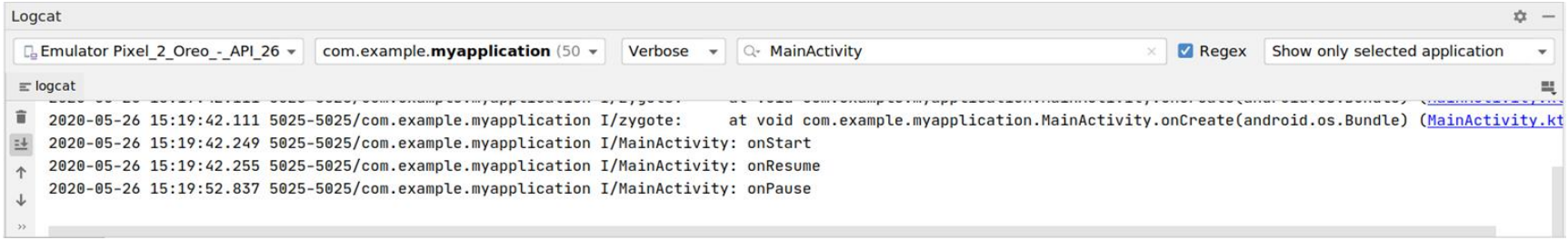
Người dùng muốn trạng thái giao diện người dùng vẫn giữ nguyên sau khi có sự thay đổi về cấu hình hoặc nếu ứng dụng bị chấm dứt khi chạy ở chế độ nền.

- Hoạt động bị hủy bỏ và khởi động lại hoặc ứng dụng bị chấm dứt và hoạt động được bắt đầu.
- Lưu trữ dữ liệu người dùng cần thiết để tạo lại những thay đổi về Vòng đời của ứng dụng và hoạt động:
  - Dùng `Gói` do `onSaveInstanceState()` cung cấp.
  - `onCreate()` nhận `Gói` làm đối số khi hoạt động được tạo lại.

Ghi nhật ký

Ghi nhật ký trong Android

- Giám sát luồng sự kiện hoặc trạng thái ứng dụng của bạn.
- Dùng lớp `Log` tích hợp sẵn hoặc thư viện của bên thứ ba.
- Ví dụ về lệnh gọi phương thức `Log`: `Log.d(TAG, "Message")`

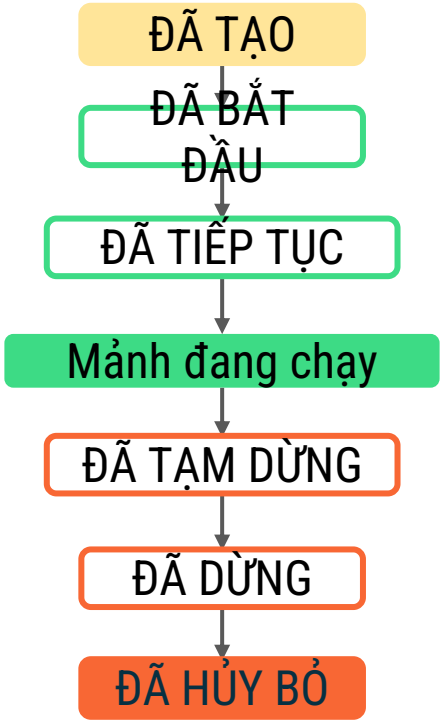


Viết nhật ký

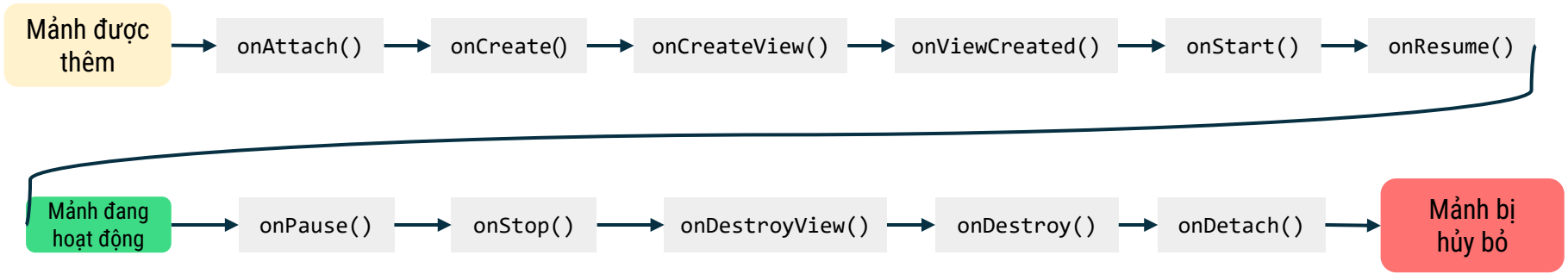
Mức độ ưu tiên	Phương thức ghi nhật ký
Chi tiết	`Log.v(String, String)`
Gỡ lỗi	`Log.d(String, String)`
Thông tin	`Log.i(String, String)`
Cảnh báo	`Log.w(String, String)`
Lỗi	`Log.e(String, String)`

Vòng đời của mảnh

Trạng thái của mảnh



Sơ đồ vòng đời của mảnh



onAttach()

- Được gọi khi một mảnh được đính kèm vào bối cảnh
- Đứng ngay trước `onCreate()`

onCreateView()

- Được gọi để tạo hệ phân cấp chế độ xem liên kết với mảnh
- Tăng cường bố cục mảnh ở đây và quay lại chế độ xem gốc

onViewCreated()

- Được gọi khi hệ phân cấp chế độ xem đã được tạo
- Thực hiện mọi quá trình khởi tạo còn lại ở đây (ví dụ: khôi phục trạng thái từ `Gói`)

onDestroyView() và onDetach()

- `onDestroyView()` được gọi khi hệ phân cấp chế độ xem của mảnh bị xóa.
- `onDetach()` được gọi khi mảnh không còn được đính kèm vào hoạt động lưu trữ.

Tóm tắt các trạng thái của mảnh

Trạng thái	Lệnh gọi lại	Nội dung mô tả
Đã khởi tạo	`onAttach()`	Mảnh được đính kèm vào hoạt động lưu trữ.
Đã tạo	`onCreate()`, `onCreateView()`, `onViewCreated()`	Mảnh được tạo và bố cục đang được khởi tạo.
Đã bắt đầu	`onStart()`	Mảnh được bắt đầu và hiển thị.
Đã tiếp tục	`onResume()`	Mảnh có tiêu điểm nhập.
Đã tạm dừng	`onPause()`	Mảnh không có tiêu điểm nhập.
Đã dừng	`onStop()`	Mảnh không hiển thị.
Đã hủy bỏ	`onDestroyView()`, `onDestroy()`, `onDetach()`	Mảnh bị xóa khỏi hoạt động lưu trữ.

Lưu trạng thái của mảnh trong các thay đổi về cấu hình

Duy trì trạng thái giao diện người dùng trong các mảnh bằng cách lưu trữ trạng thái trong `Gói`:

- `onSaveInstanceState(outState: Bundle)`

Truy xuất dữ liệu đó bằng cách nhận Gói trong các lệnh gọi lại mảnh sau đây:

- `onCreate()`
- `onCreateView()`
- `onViewCreated()`

Các thành phần nhận biết vòng đời

Các thành phần nhận biết vòng đời

Điều chỉnh hành vi của thành phần dựa trên vòng đời của hoạt động hoặc mảnh

- Dùng thư viện `androidx.lifecycle`
- `Vòng đời` theo dõi trạng thái vòng đời của một hoạt động hoặc mảnh
  - Lưu giữ trạng thái vòng đời hiện tại
  - Gửi các sự kiện trong vòng đời (khi có các thay đổi về trạng thái)

LifecycleOwner

- Giao diện cho biết lớp này có vòng đời
- Trình triển khai phải triển khai phương thức `getLifecycle()`

  Ví dụ: `Fragment` và `AppCompatActivity` là các phương thức triển khai `LifecycleOwner`

LifecycleObserver

Triển khai giao diện `LifecycleObserver`:

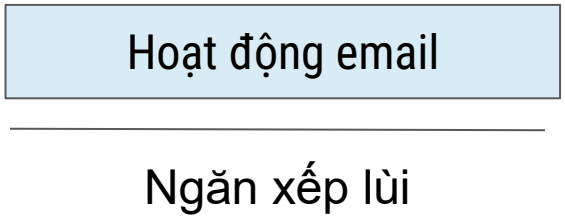
```
class MyObserver : LifecycleObserver {
    @OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
    fun connectListener() {
        ...
    }
```

Thêm đối tượng tiếp nhận dữ liệu vào vòng đời:

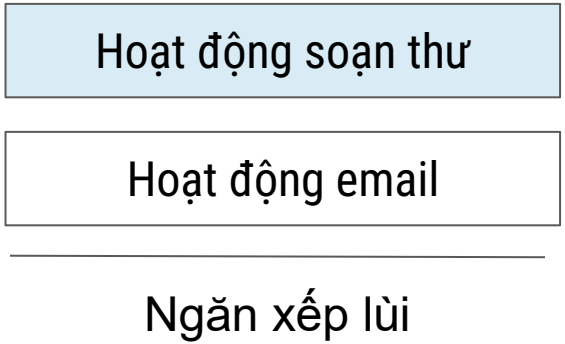
```
myLifecycleOwner.getLifecycle().addObserver(MyObserver())
```

Tác vụ và ngăn xếp lùi

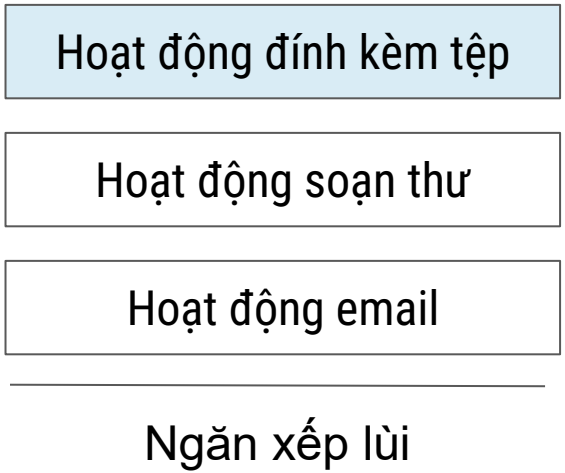
Ngăn xếp lùi các hoạt động



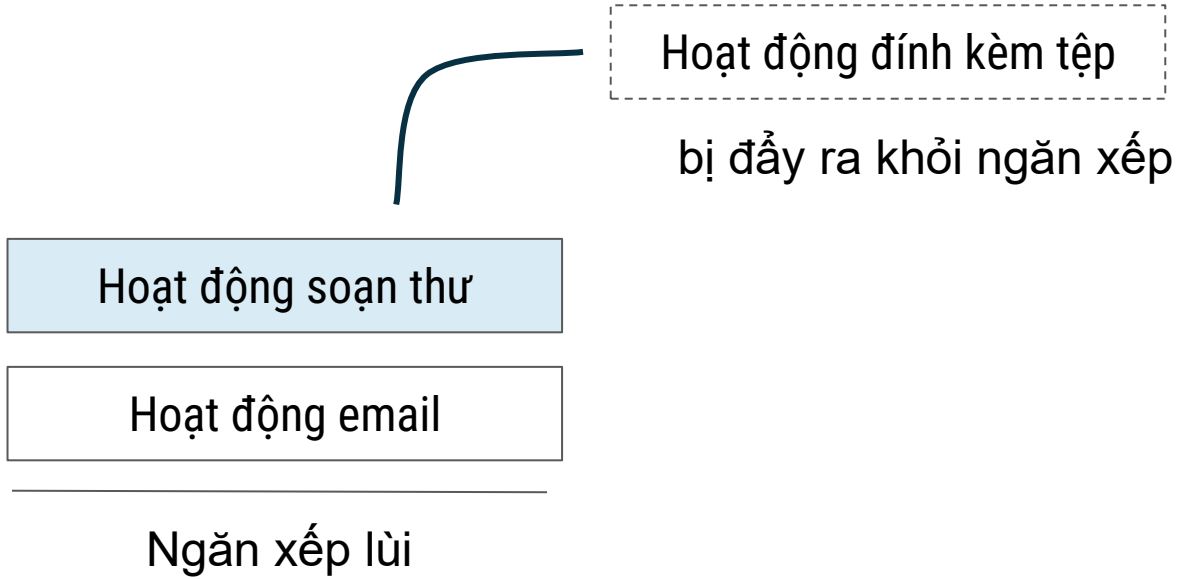
Thêm vào ngăn xếp lùi



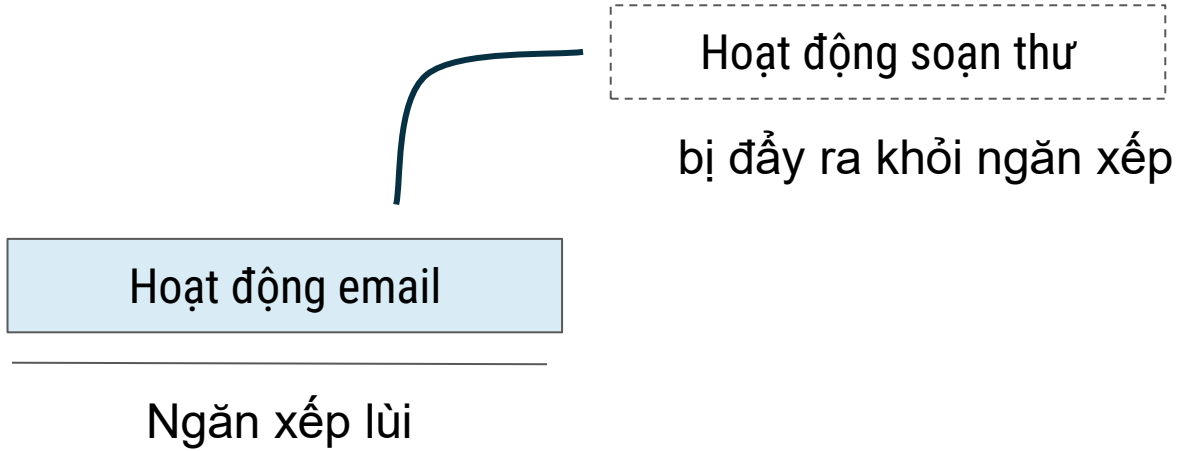
Thêm lại vào ngăn xếp lùi



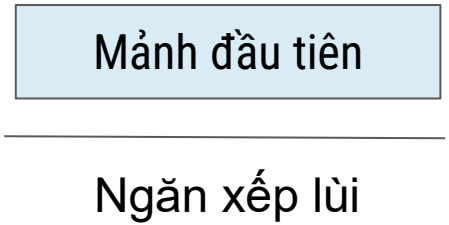
Nhấn vào nút Quay lại



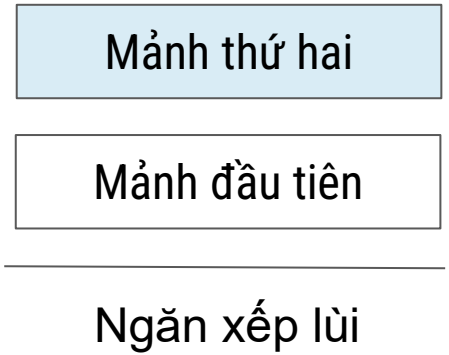
Nhấn vào nút Quay lại lần nữa



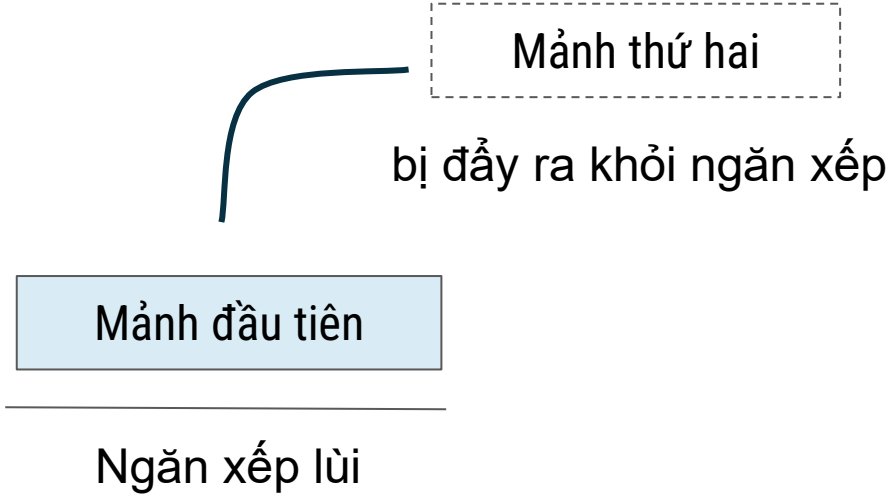
Đích đầu tiên trong ngăn xếp lùi



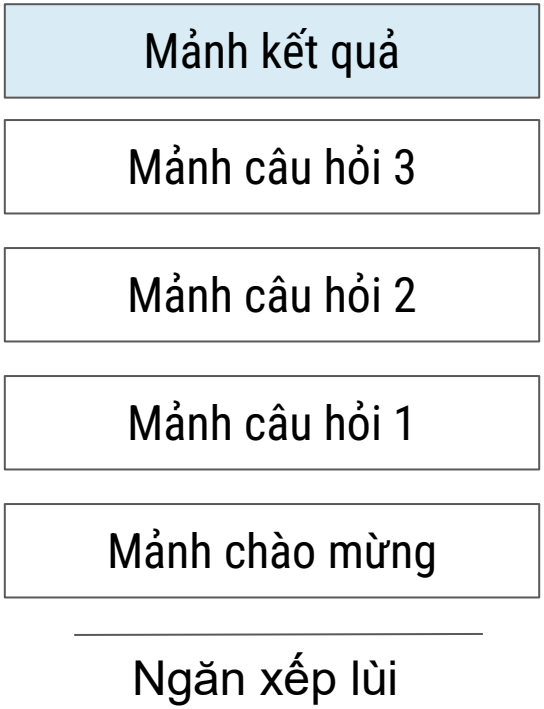
Thêm một đích vào ngăn xếp lùi



Nhấn vào nút Quay lại

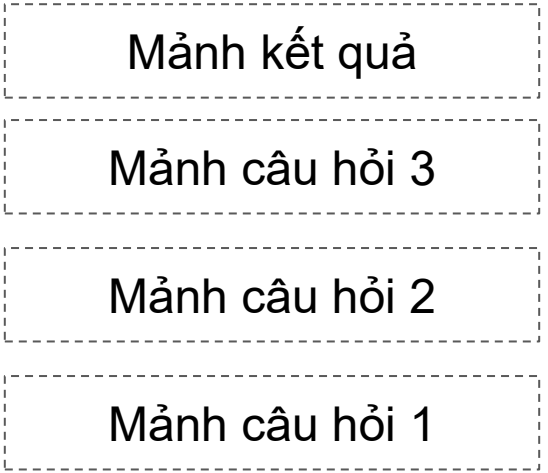


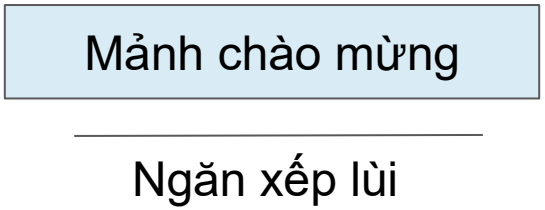
Một ví dụ khác về ngăn xếp lùi



Sửa đổi hành vi của nút Quay lại

đẩy các đích khác ra khỏi ngăn xếp lùi





Tóm tắt

Tóm tắt

Trong Bài học 7, bạn đã tìm hiểu cách:

- Nắm được cách một thực thể của hoạt động chuyển đổi qua nhiều trạng thái vòng đời khi người dùng tương tác hoặc rời khỏi ứng dụng của bạn
- Duy trì trạng thái giao diện người dùng trong các thay đổi về cấu hình thông qua `Gói`
- Các phương thức gọi lại trong vòng đời của mảnh tương tự như hoạt động, nhưng có thêm các phương thức khác
- Dùng các thành phần nhận biết vòng đời để giúp sắp xếp mã ứng dụng của bạn
- Dùng hành vi mặc định hoặc hành vi tùy chỉnh của ngăn xếp lùi
- Dùng tính năng ghi nhật ký để giúp gỡ lỗi và theo dõi trạng thái của ứng dụng

Tìm hiểu thêm

- Tìm hiểu về Vòng đời hoạt động
- Lớp Activity
- Hướng dẫn và vòng đời của mảnh
- Lớp Fragment

Lộ trình

Thực hành những gì bạn đã học được bằng cách hoàn thành lộ trình này:

Bài học 7: Vòng đời của hoạt động và mảnh