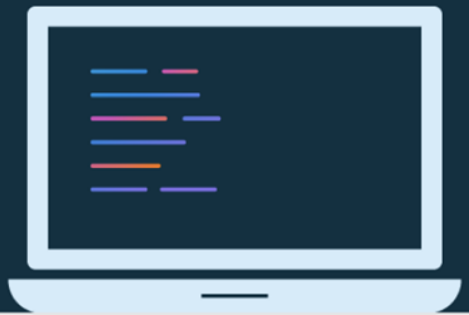
Bài học 6: Di chuyển trong ứng dụng



Giới thiệu về bài học này

Bài học 6: Di chuyển trong ứng dụng

- Nhiều hoạt động và ý định
- Thanh ứng dụng, ngăn điều hướng và các trình đơn
- Mảnh
- Di chuyển trong một ứng dụng
- Hành vi di chuyển tùy chỉnh khác
- Giao diện người dùng điều hướng
- Tóm tắt

Nhiều hoạt động và ý định

Nhiều màn hình trong một ứng dụng

Đôi khi, chức năng của ứng dụng có thể được tách thành nhiều màn hình.

Ví dụ:

- Xem thông tin chi tiết về một mục (ví dụ: sản phẩm trong ứng dụng mua sắm)
- Tạo một mục mới (ví dụ: email mới)
- Hiển thị các chế độ cài đặt của một ứng dụng
- Truy cập vào các dịch vụ trong những ứng dụng khác (ví dụ: thư viện ảnh hoặc duyệt xem tài liệu)

Ý định

Yêu cầu một thao tác từ thành phần ứng dụng khác, chẳng hạn như một Hoạt động khác

- `Ý định` thường có 2 đoạn thông tin chính:
  - Thao tác cần thực hiện (ví dụ: `ACTION_VIEW`, `ACTION_EDIT`, `ACTION_MAIN`)
  - Dữ liệu được thao tác (ví dụ: hồ sơ của một người trong cơ sở dữ liệu danh bạ)
- Thường dùng để chỉ định yêu cầu chuyển đổi sang một Hoạt động khác

Ý định tường minh

- Thực hiện yêu cầu **bằng một thành phần cụ thể**
- Chuyển nội bộ đến một Hoạt động trong ứng dụng của bạn
- Chuyển đến một ứng dụng cụ thể của bên thứ ba hoặc một ứng dụng khác mà bạn đã viết

Ví dụ về ý định tường minh

Chuyển đổi giữa các hoạt động trong ứng dụng của bạn:

```
fun viewNoteDetail() {
    val intent = Intent(this, NoteDetailActivity::class.java)
    intent.putExtra(NOTE_ID, note.id)
    startActivity(intent)
}
```

Chuyển đến một ứng dụng bên ngoài cụ thể:

```
fun openExternalApp() {
    val intent = Intent("com.example.workapp.FILE_OPEN")
    if (intent.resolveActivity(packageManager) != null) {
        startActivity(intent)
    }
}
```

Ý định ngầm ẩn

- Cung cấp thao tác chung mà ứng dụng có thể thực hiện
- Được phân giải bằng cách liên kết loại dữ liệu và thao tác với các thành phần đã biết
- Cho phép bất kỳ ứng dụng nào đáp ứng tiêu chí xử lý yêu cầu

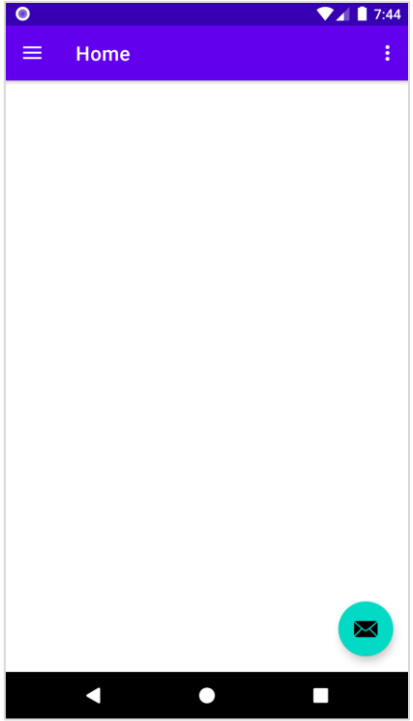
Ví dụ về ý định ngầm ẩn

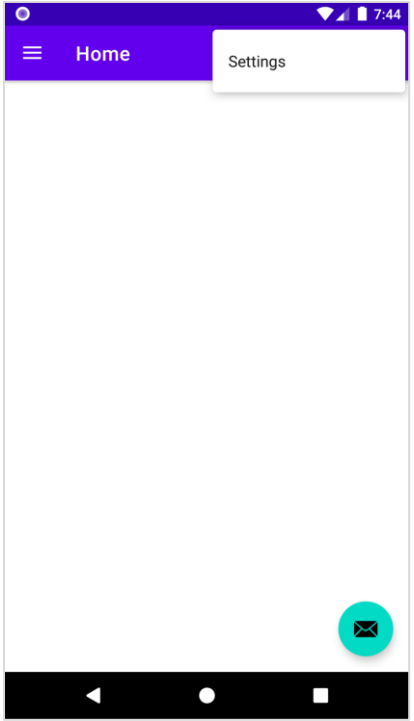
```
fun sendEmail() {
    val intent = Intent(Intent.ACTION_SEND)
    intent.type = "text/plain"
    intent.putExtra(Intent.EXTRA_EMAIL, emailAddresses)
    intent.putExtra(Intent.EXTRA_TEXT, "How are you?")

    if (intent.resolveActivity(packageManager) != null) {
        startActivity(intent)
    }
}
```

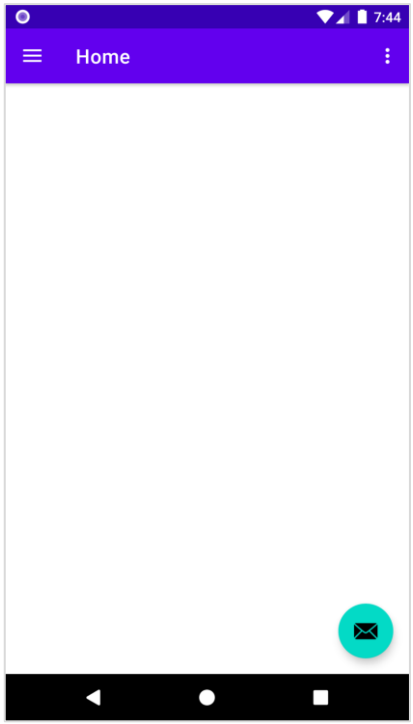
Thanh ứng dụng, ngăn điều hướng và các trình đơn

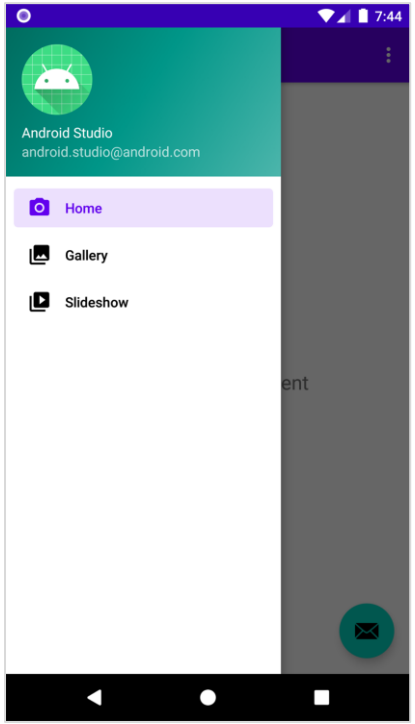
Thanh ứng dụng





Ngăn điều hướng





Trình đơn

Xác định các mục trong trình đơn trên tài nguyên trình đơn XML (có trong thư mục `res/menu`)

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never" />
</menu>
```

Các tùy chọn trình đơn khác

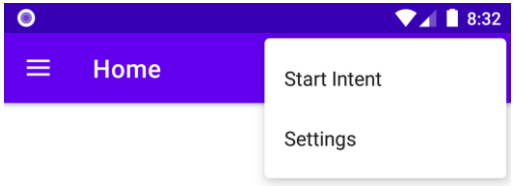
```
<menu>
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_home"
            android:icon="@drawable/ic_menu_camera"
            android:title="@string/menu_home" />
        <item
            android:id="@+id/nav_gallery"
            android:icon="@drawable/ic_menu_gallery"
            android:title="@string/menu_gallery" />
        <item
            android:id="@+id/nav_slideshow"
            android:icon="@drawable/ic_menu_slideshow"
            android:title="@string/menu_slideshow" />
    </group>
</menu>
```

Ví dụ về trình đơn tùy chọn

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/action_intent"
        android:title="@string/action_intent" />

    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never" />
</menu>
```



Tăng cường trình đơn tùy chọn

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {
    menuInflater.inflate(R.menu.main, menu)
    return true
}
```

Xử lý các tùy chọn trình đơn được chọn

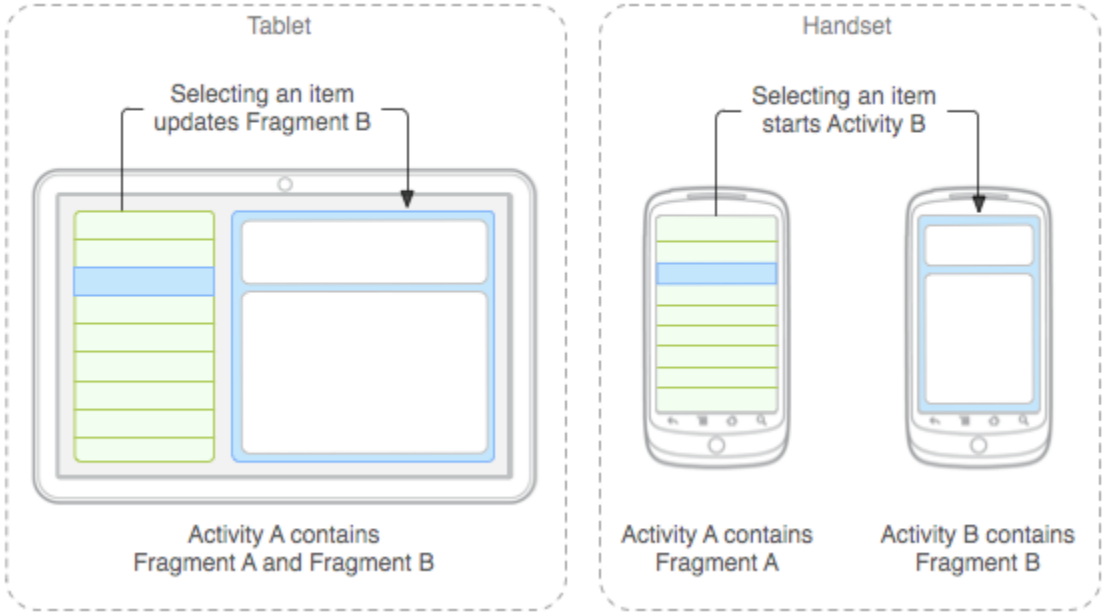
```
override fun onOptionsItemSelected(item: MenuItem): Boolean {

    when (item.itemId) {
        R.id.action_intent -> {
            val intent = Intent(Intent.ACTION_WEB_SEARCH)
            intent.putExtra(SearchManager.QUERY, "pizza")
            if (intent.resolveActivity(packageManager) != null) {
                startActivity(intent)
            }
        }

        else -> Toast.makeText(this, item.title, Toast.LENGTH_LONG).show()
    ...
```

Mảnh

Mảnh cho bố cục trên máy tính bảng



Mảnh

- Biểu thị một hành vi hoặc một phần giao diện người dùng trong một hoạt động ("hoạt động vi mô")
- Phải được lưu trữ trong một hoạt động
- Vòng đời gắn liền với vòng đời của hoạt động lưu trữ
- Có thể được thêm vào hoặc bị xóa đi trong thời gian chạy

Lưu ý về mảnh

Dùng phiên bản AndroidX của lớp `Fragment`.
(`androidx.fragment.app.Fragment`).

Không dùng phiên bản nền tảng của lớp `Fragment`
(`android.app.Fragment`) vì phiên bản này không còn dùng nữa.

Di chuyển trong một ứng dụng

Thành phần điều hướng

- Tập hợp các thư viện và công cụ, bao gồm cả một trình chỉnh sửa tích hợp, để tạo các đường dẫn điều hướng thông qua một ứng dụng
- Giả định một `Hoạt động` trên mỗi sơ đồ với nhiều đích của `Mảnh`
- Bao gồm 3 phần chính:
  - Sơ đồ điều hướng
  - Máy chủ điều hướng (`NavHost`)
  - Bộ điều khiển điều hướng (`NavController`)

Thêm phần phụ thuộc

Trong tệp `build.gradle`, bên dưới `phần phụ thuộc`:

```
implementation "androidx.navigation:navigation-fragment-ktx:$nav_version"

implementation "androidx.navigation:navigation-ui-ktx:$nav_version"
```

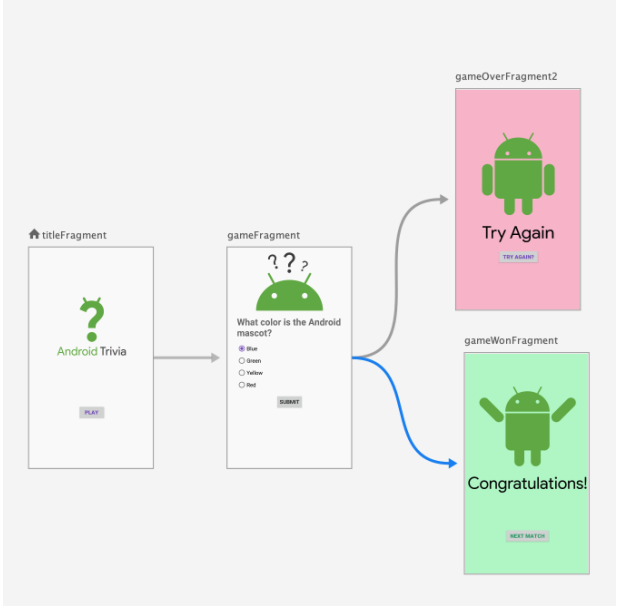
Máy chủ điều hướng (NavHost)

```
<fragment
    android:id="@+id/nav_host"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:defaultNavHost="true"
    app:navGraph="@navigation/nav_graph_name"/>
```

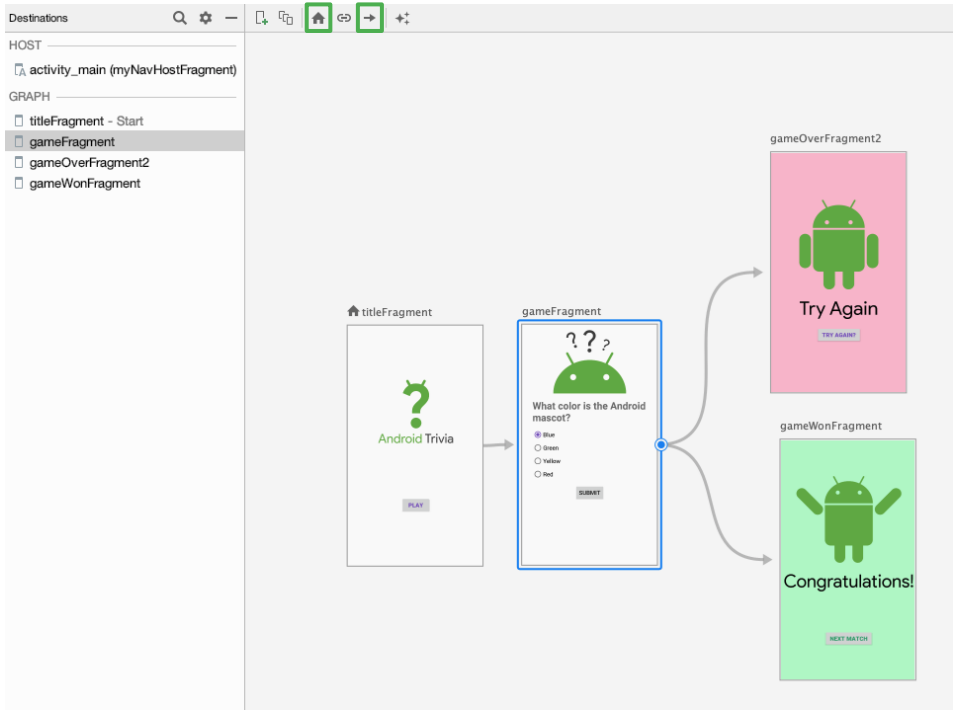
Sơ đồ điều hướng

Loại tài nguyên mới có trong thư mục `res/navigation`

- Tệp XML chứa tất cả các đích và thao tác di chuyển của bạn
- Liệt kê tất cả các đích (Mảnh/Hoạt động) có thể được chuyển đến
- Liệt kê các thao tác liên quan để chuyển giữa các thao tác đó
- Tùy ý liệt kê các hoạt ảnh để chuyển sang hoặc thoát



Trình chỉnh sửa điều hướng trong Android Studio



Tạo một Mảnh

- Mở rộng lớp `Fragment`
- Ghi đè `onCreateView()`
- Tăng cường bố cục cho Mảnh mà bạn đã xác định trong tệp XML

```
class DetailFragment : Fragment() {

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
            savedInstanceState: Bundle?): View? {
        return inflater.inflate(R.layout.detail_fragment, container, false)
    }
}
```

Chỉ định các đích của Mảnh

- Các đích của mảnh được biểu thị bằng thẻ `action` trong sơ đồ điều hướng.
- Các thao tác có thể được xác định ngay trong tệp XML hoặc trong Trình chỉnh sửa điều hướng bằng cách kéo từ nguồn tới đích.
- Mã thao tác được tạo tự động có dạng `action_<sourceFragment>_to_<destinationFragment>`.

Ví dụ về đích của mảnh

```
<fragment
    android:id="@+id/welcomeFragment"
    android:name="com.example.android.navigation.WelcomeFragment"
    android:label="fragment_welcome"
    tools:layout="@layout/fragment_welcome" >

    <action
        android:id="@+id/action_welcomeFragment_to_detailFragment"
        app:destination="@id/detailFragment" />

</fragment>
```

Bộ điều khiển điều hướng (NavController)

`NavController` quản lý việc di chuyển trên giao diện người dùng trong một máy chủ điều hướng.

- Việc chỉ định đường dẫn đích sẽ chỉ đặt tên cho thao tác, chứ không thực thi thao tác đó.
- Để truy cập vào một đường dẫn, hãy dùng `NavController`.

Ví dụ về NavController

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        ...
        val navController = findNavController(R.id.myNavHostFragment)
    }

    fun navigateToDetail() {
        navController.navigate(R.id.action_welcomeFragment_to_detailFragment)
    }
}
```

Hành vi di chuyển tùy chỉnh khác

Chuyển dữ liệu giữa các đích

Sử dụng Safe Args:

- Đảm bảo các đối số có loại hợp lệ
- Cho phép bạn cung cấp các giá trị mặc định
- Tạo một lớp `<SourceDestination>Directions` với các phương thức cho mọi thao tác trong đích đó
- Tạo một lớp để đặt đối số cho mọi thao tác được đặt tên
- Tạo một lớp `<TargetDestination>Args` cung cấp quyền truy cập vào các đối số của đích

Thiết lập Safe Args

Trong tệp `build.gradle` của dự án:

```
buildscript {
    repositories {
        google()
    }
    dependencies {
        classpath "androidx.navigation:navigation-safe-args-gradle-plugin:$nav_version"
    }
}
```

Trong tệp `build.gradle` của ứng dụng hoặc mô-đun:

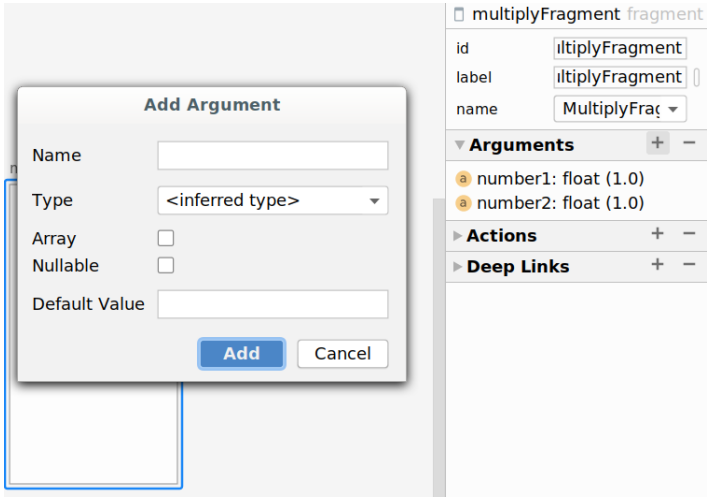
```
apply plugin: "androidx.navigation.safeargs.kotlin"
```

Gửi dữ liệu tới một Mảnh

1. Tạo các đối số mà mảnh đích sẽ nhận được.
2. Tạo thao tác để liên kết từ nguồn đến đích.
3. Đặt các đối số trong phương thức thao tác trên `<Source>FragmentDirections`.
4. Di chuyển theo thao tác đó bằng Bộ điều khiển điều hướng.
5. Truy xuất các đối số trong mảnh đích.

Đối số đích

```
<fragment
    android:id="@+id/multiplyFragment"
    android:name="com.example.arithmetic.MultiplyFragment"
    android:label="MultiplyFragment" >
    <argument
        android:name="number1"
        app:argType="float"
        android:defaultValue="1.0" />
    <argument
        android:name="number2"
        app:argType="float"
        android:defaultValue="1.0" />
</fragment>
```



Các loại đối số được hỗ trợ

Loại	**Cú pháp loại** `app:argType=<type>`	**Hỗ trợ các giá trị mặc định**	**Hỗ trợ các giá trị null**
Integer	`"integer"`	Có	Không
Float	`"float"`	Có	Không
Long	`"long"`	Có	Không
Boolean	`"boolean"`	Có (`"true"` hoặc `"false"`)	Không
String	`"string"`	Có	Có
Array	loại ở trên + `"[]"` (ví dụ: `"string[]"` `"long[]"`)	Có (chỉ `"@null"`)	Có
Enum	Tên đủ điều kiện thuộc loại enum	Có	Không
Resource reference	`"reference"`	Có	Không

Các loại đối số được hỗ trợ: Lớp tùy chỉnh

Loại	**Cú pháp loại** `app:argType=<type>`	**Hỗ trợ các giá trị mặc định**	**Hỗ trợ các giá trị null**
Theo tuần tự	Tên lớp đủ điều kiện	Có (chỉ "`@null`")	Có
Theo gói	Tên lớp đủ điều kiện	Có (chỉ "`@null`")	Có

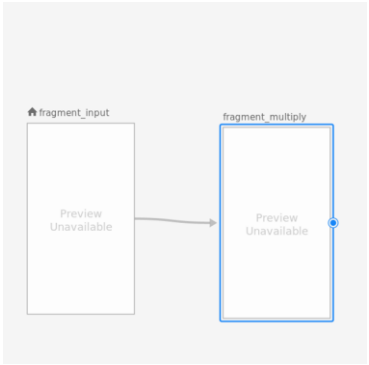
Tạo thao tác từ nguồn đến đích

Trong tệp `nav_graph.xml`:

```
<fragment
    android:id="@+id/fragment_input"
    android:name="com.example.arithmetic.InputFragment">

    <action
        android:id="@+id/action_to_multiplyFragment"
        app:destination="@id/multiplyFragment" />

</fragment>
```



Di chuyển bằng các thao tác

Trong tệp `InputFragment.kt`:

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    binding.button.setOnClickListener {
        val n1 = binding.number1.text.toString().toFloatOrNull() ?: 0.0
        val n2 = binding.number2.text.toString().toFloatOrNull() ?: 0.0

        val action = InputFragmentDirections.actionToMultiplyFragment(n1, n2)
        view.findNavController().navigate(action)
    }
}
```

Truy xuất các đối số của Mảnh

```
class MultiplyFragment : Fragment() {
    val args: MultiplyFragmentArgs by navArgs()
    lateinit var binding: FragmentMultiplyBinding
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val number1 = args.number1
        val number2 = args.number2
        val result = number1 * number2
        binding.output.text = "${number1} * ${number2} = ${result}"
    }
}
```

Giao diện người dùng điều hướng

Các trình đơn được xem lại

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    val navController = findNavController(R.id.nav_host_fragment)
    return item.onNavDestinationSelected(navController) ||
            super.onOptionsItemSelected(item)
}
```

DrawerLayout cho ngăn điều hướng

```
<androidx.drawerlayout.widget.DrawerLayout
    android:id="@+id/drawer_layout" ...>

    <fragment
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:id="@+id/nav_host_fragment" ... />

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/nav_view"
        app:menu="@menu/activity_main_drawer" ... />


</androidx.drawerlayout.widget.DrawerLayout>
```

Hoàn tất việc thiết lập ngăn điều hướng

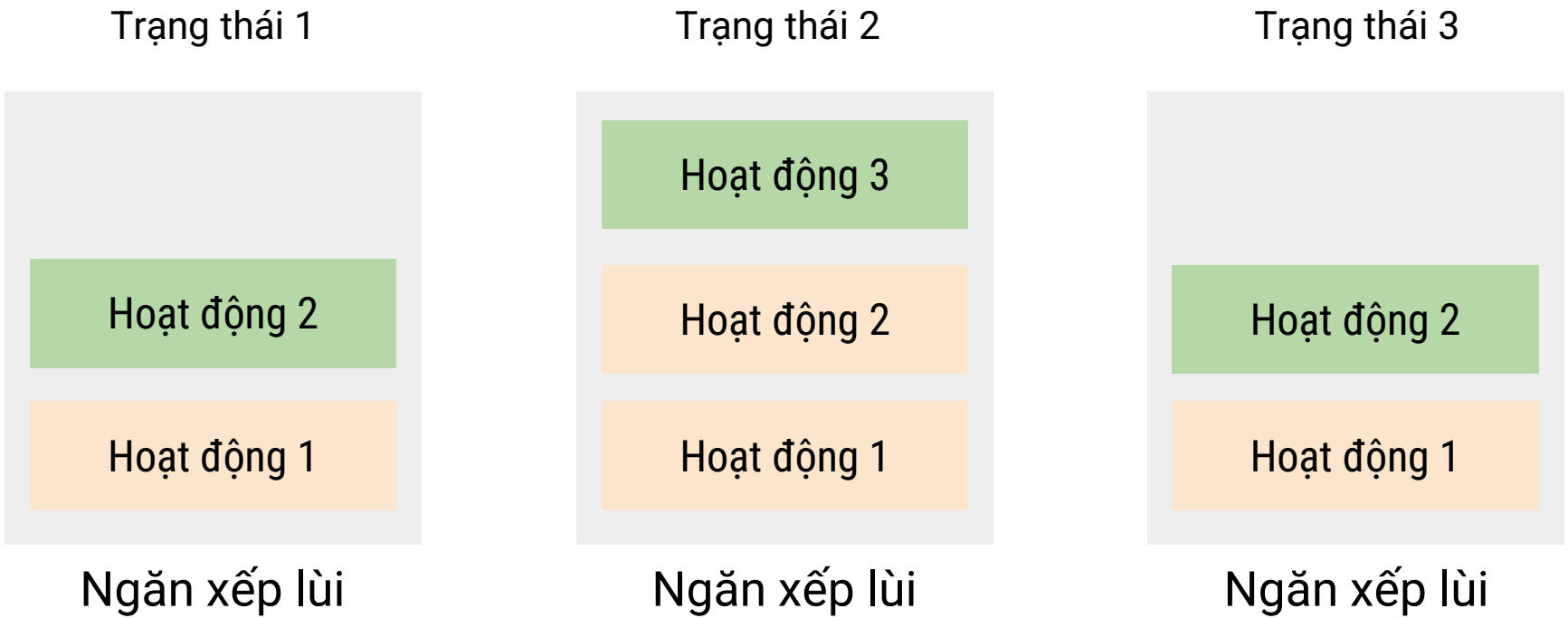
Kết nối `DrawerLayout` với sơ đồ điều hướng:

```
val appBarConfiguration = AppBarConfig(navController.graph, drawer)
```

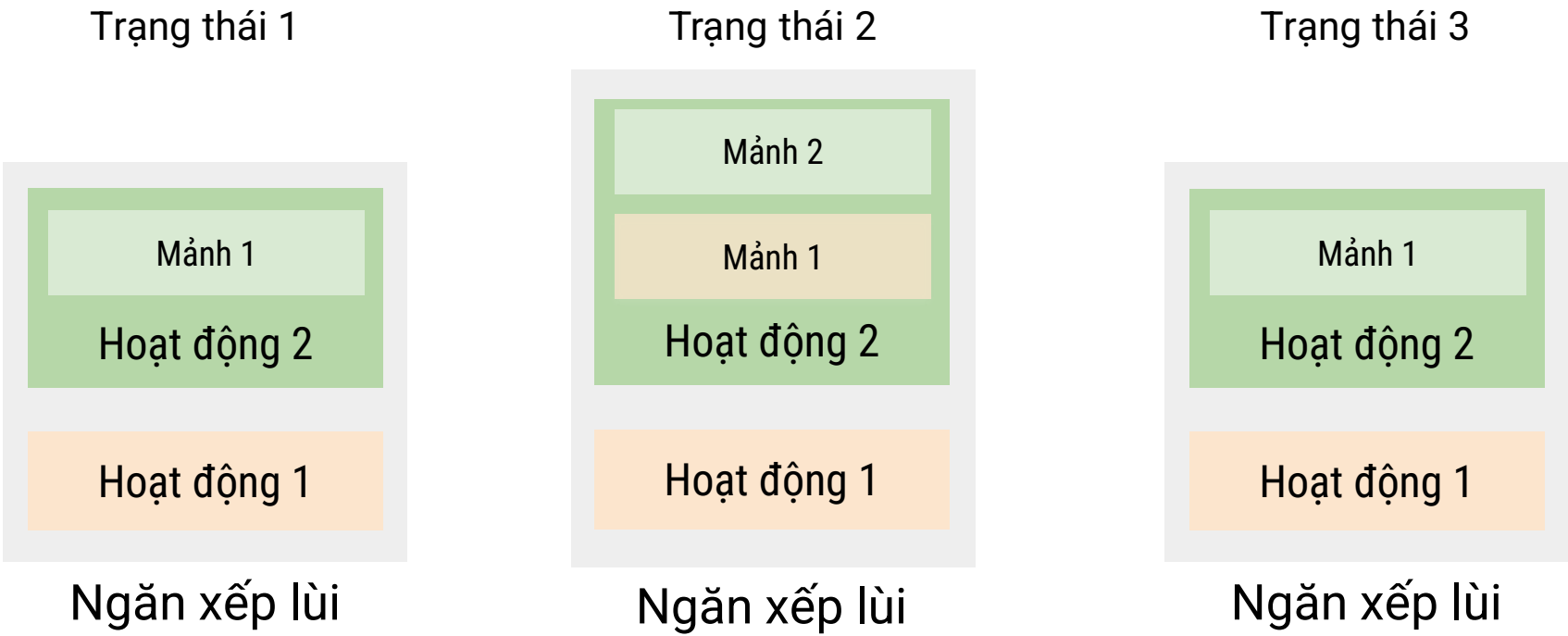
Thiết lập `NavigationView` để dùng với `NavController`:

```
val navView = findViewById<NavigationView>(R.id.nav_view)
navView.setupWithNavController(navController)
```

Tìm hiểu về ngăn xếp lùi



Các mảnh và ngăn xếp lùi



Tóm tắt

Tóm tắt

Trong Bài học 6, bạn đã tìm hiểu cách:

- Sử dụng ý định tường minh và ý định ngầm ẩn để chuyển đổi giữa các Hoạt động
- Định cấu trúc các ứng dụng bằng cách dùng mảnh thay vì đặt toàn bộ mã giao diện người dùng trong Hoạt động
- Xử lý hoạt động di chuyển bằng NavGraph, NavHost và NavController
- Dùng Safe Args để chuyển dữ liệu giữa các đích của mảnh
- Dùng NavigationUI để kết nối thanh ứng dụng trên cùng, ngăn điều hướng và thanh điều hướng dưới cùng
- Android lưu giữ một ngăn xếp lùi gồm tất cả các đích mà bạn đã truy cập, trong đó mỗi đích mới sẽ được đẩy lên đầu ngăn xếp.

Tìm hiểu thêm

- Nguyên tắc di chuyển
- Thành phần điều hướng
- Chuyển dữ liệu giữa các đích
- NavigationUI

Lộ trình

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

Bài học 6: Di chuyển trong ứng dụng