Mobile Application Development Overview

Presented by:

Himshikhar Gayan

Where do we start from?

Where do we start from?

Native Android

Where do we start from?

Native Android

Native iOS

Where do we start from?

Native Android

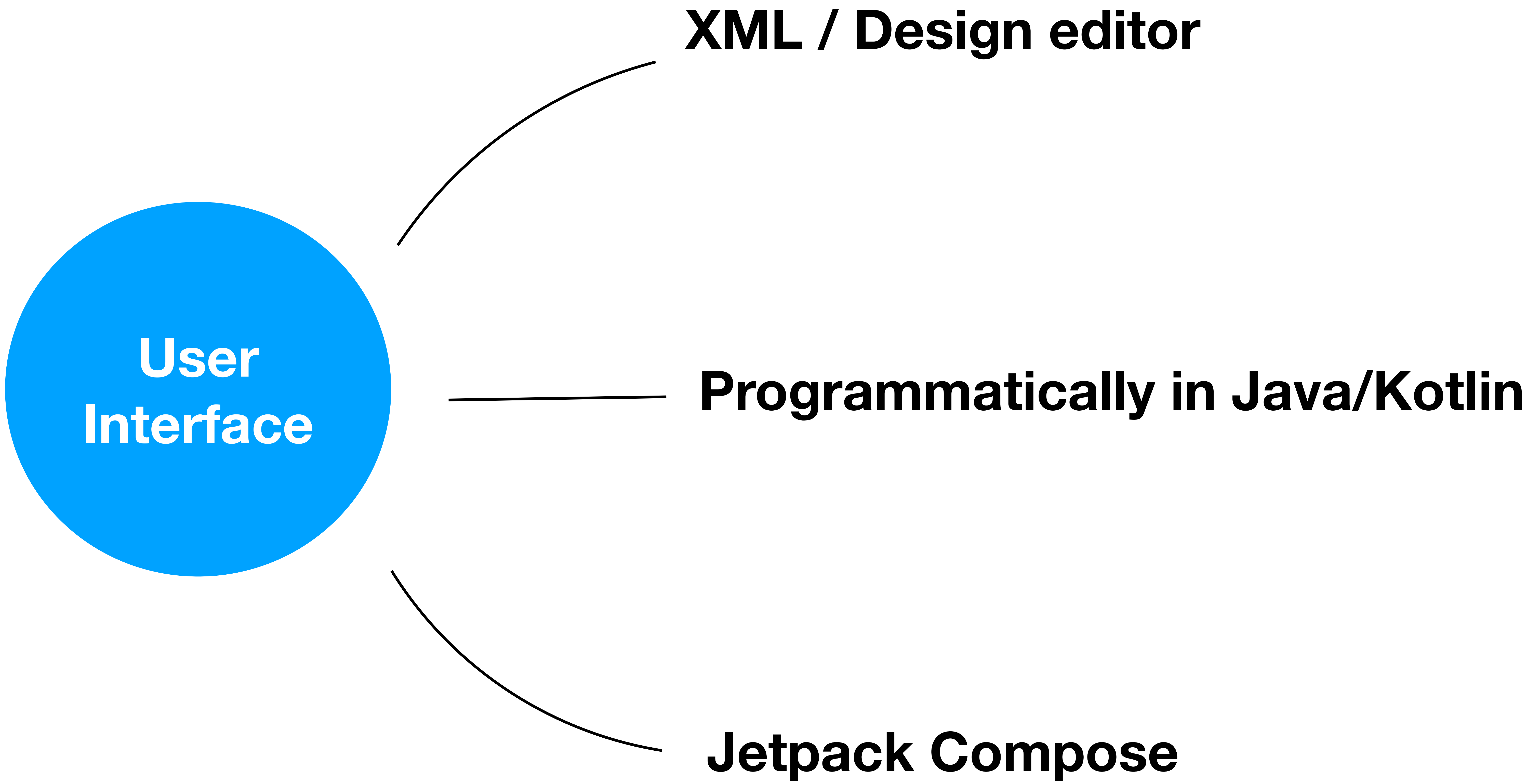
Native iOS

Hybrid

Native Android

UI

Native Android



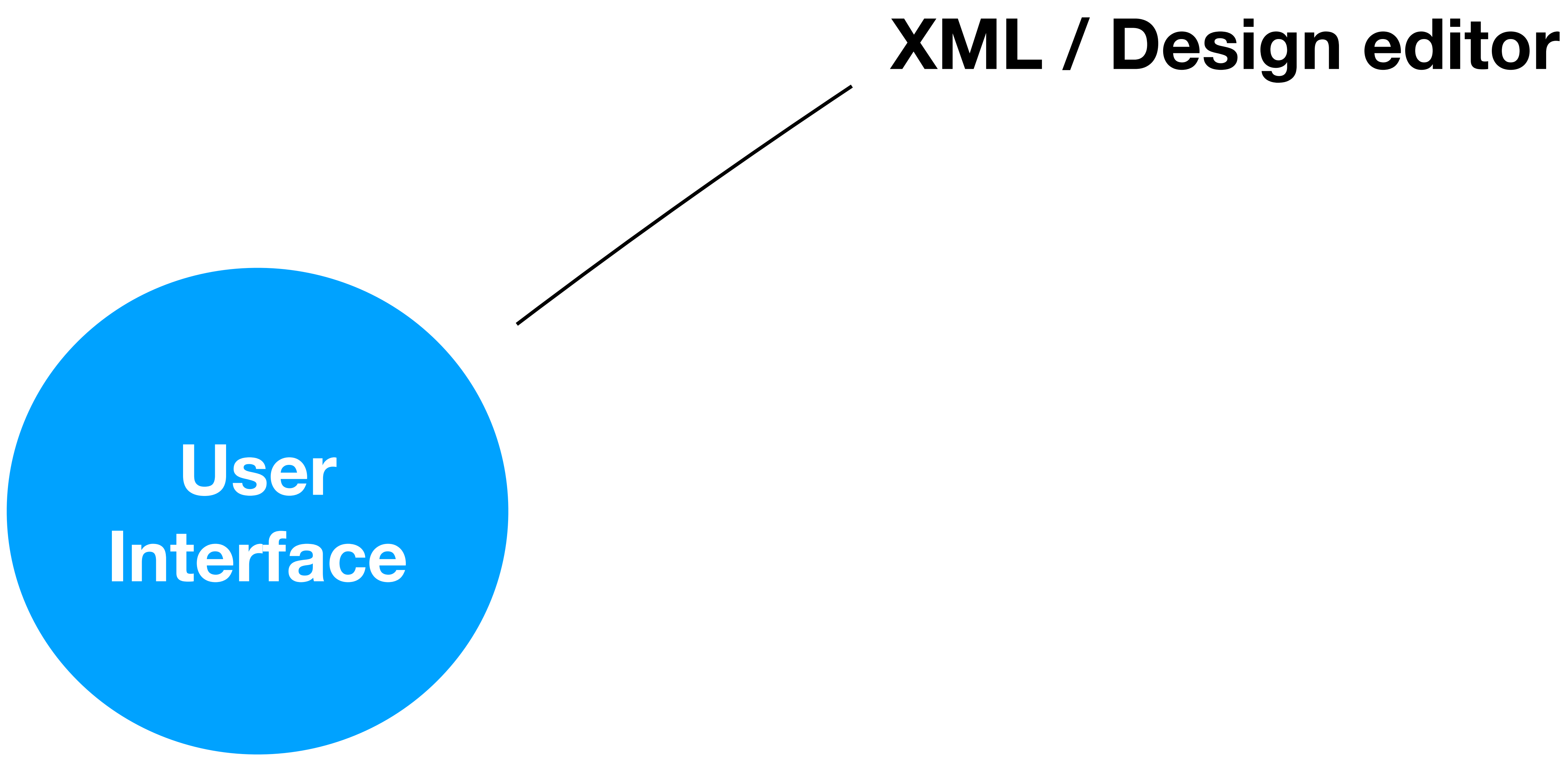
User Interface

XML / Design editor

Programmatically in Java/Kotlin

Jetpack Compose

Native Android

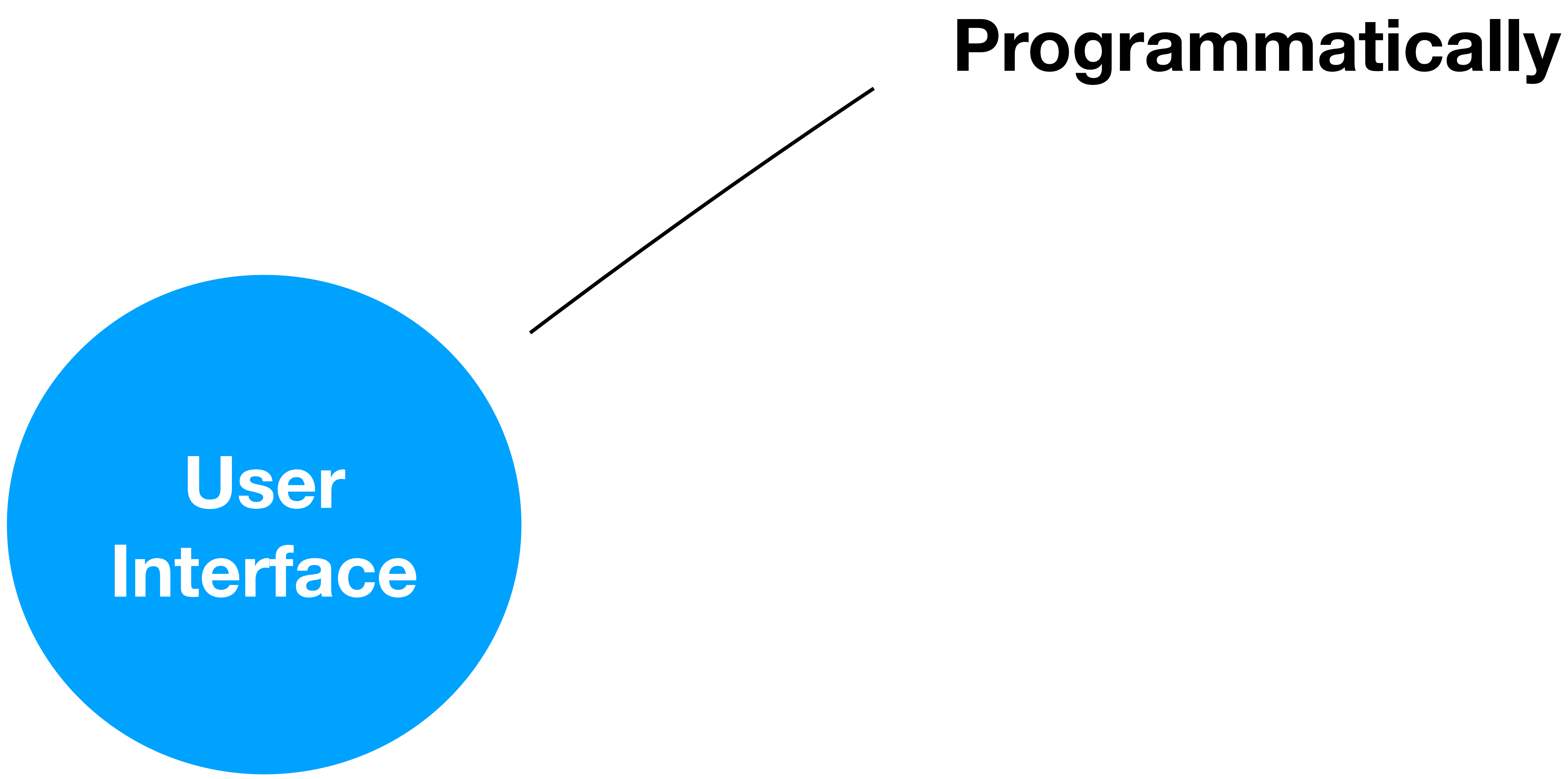


User Interface

XML / Design editor

```
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text=“Click to activate”
 />
```

Native Android

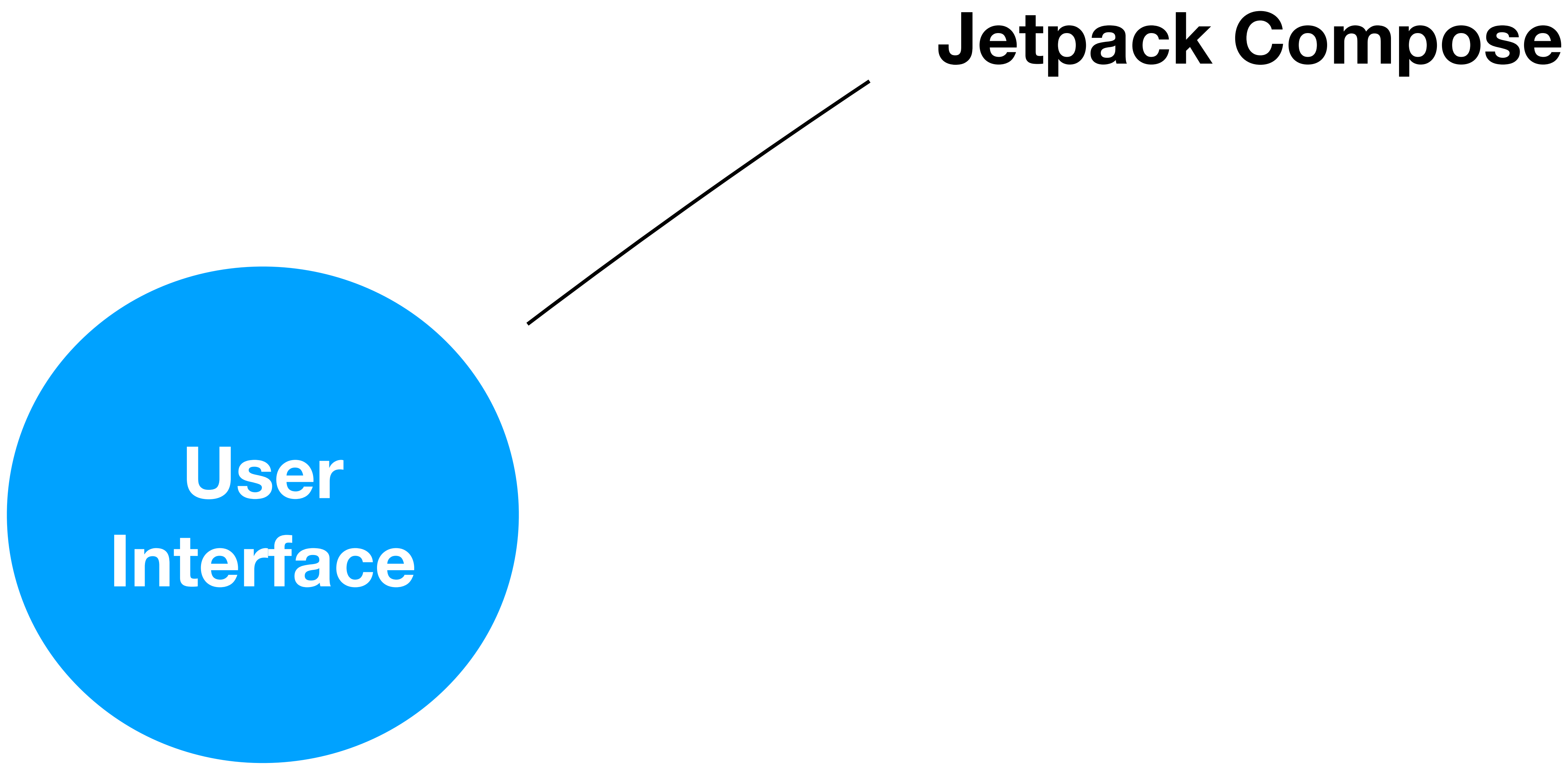


User Interface

Programmatically

```
Button button = new Button(this);
button.setText(“Click to activate”);
```

Native Android

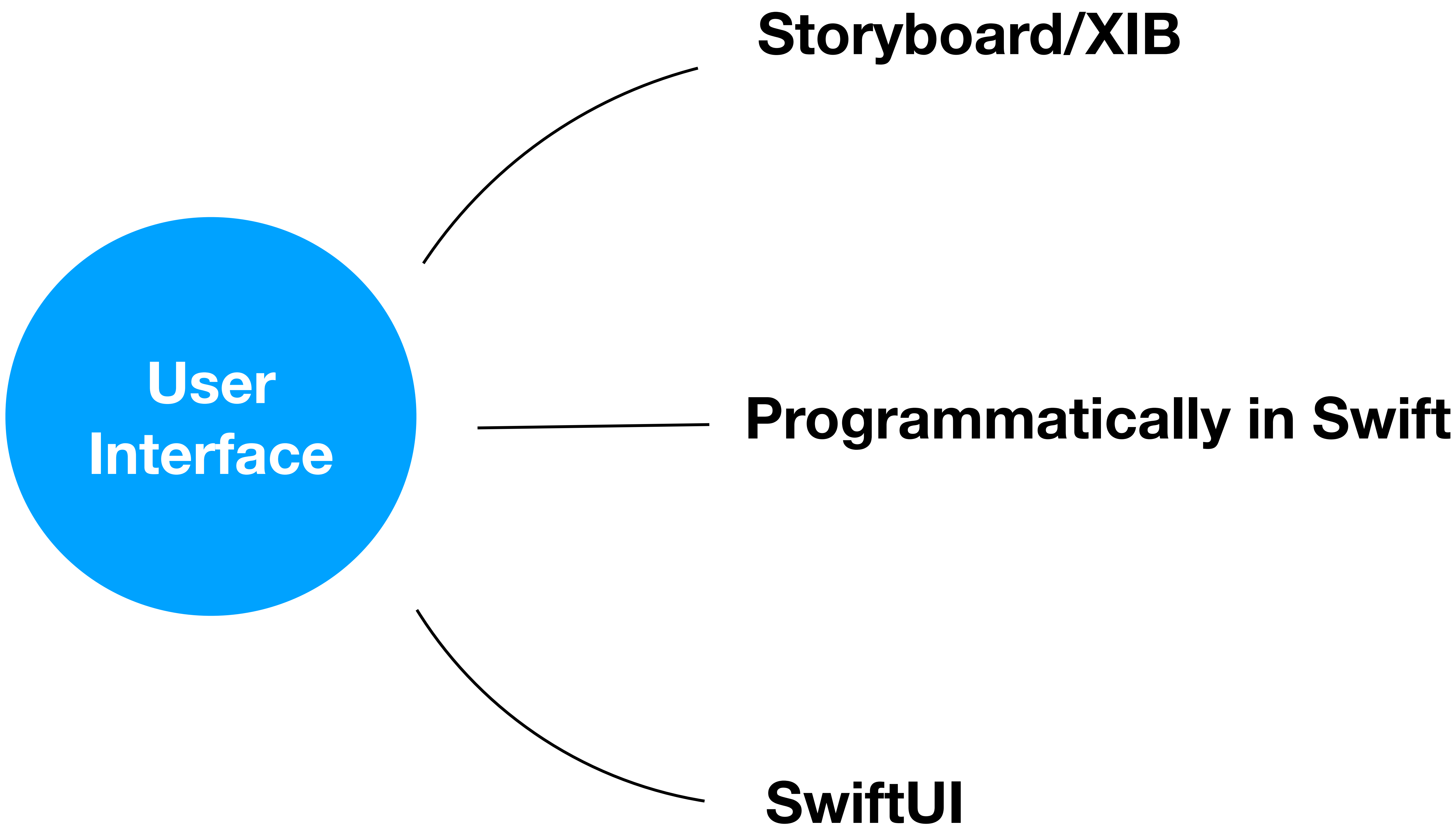


User Interface

Jetpack Compose

```
Text("Hacktoberfest")
```

Native iOS



User Interface

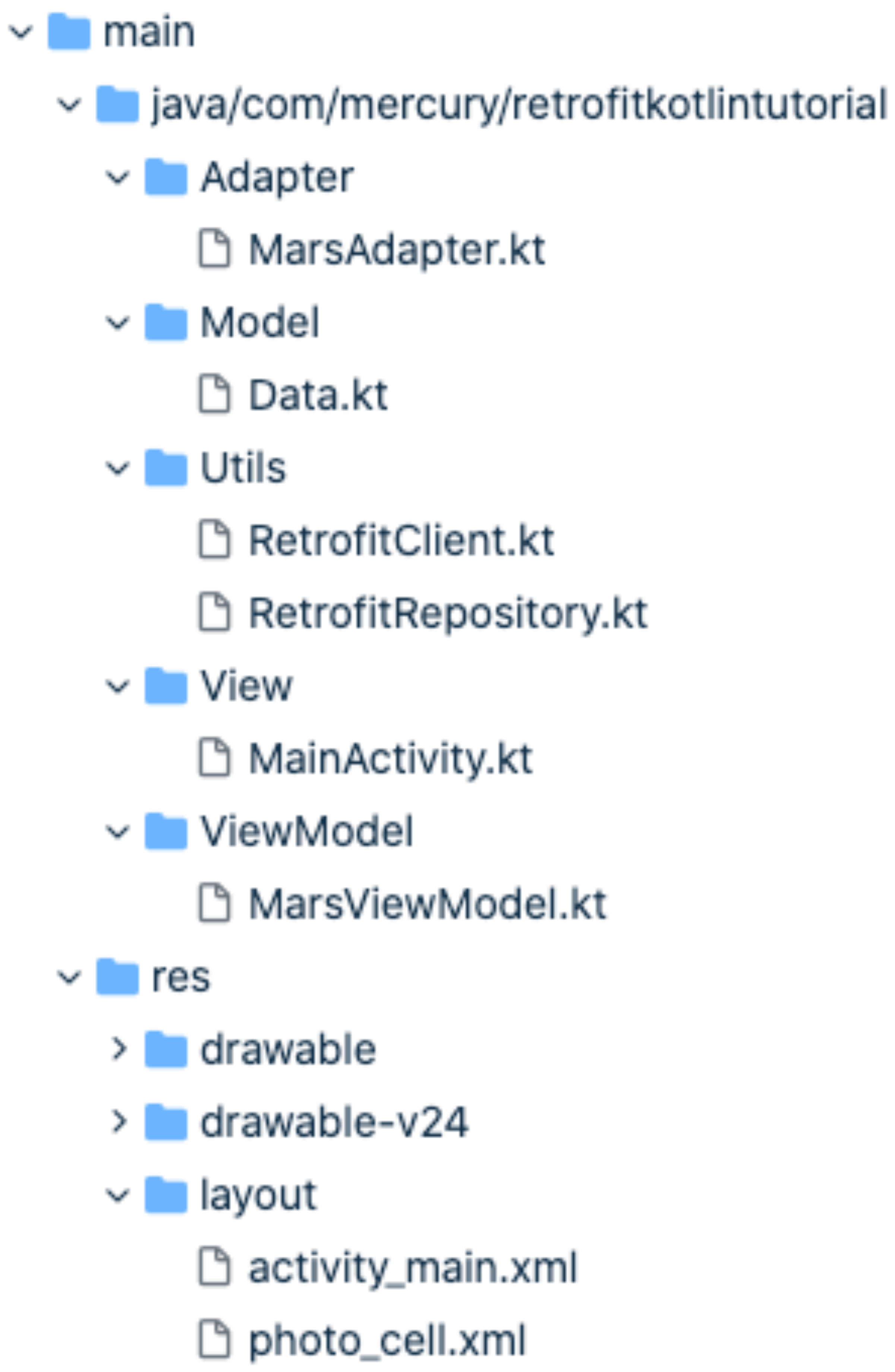
Storyboard/XIB

Programmatically in Swift

SwiftUI

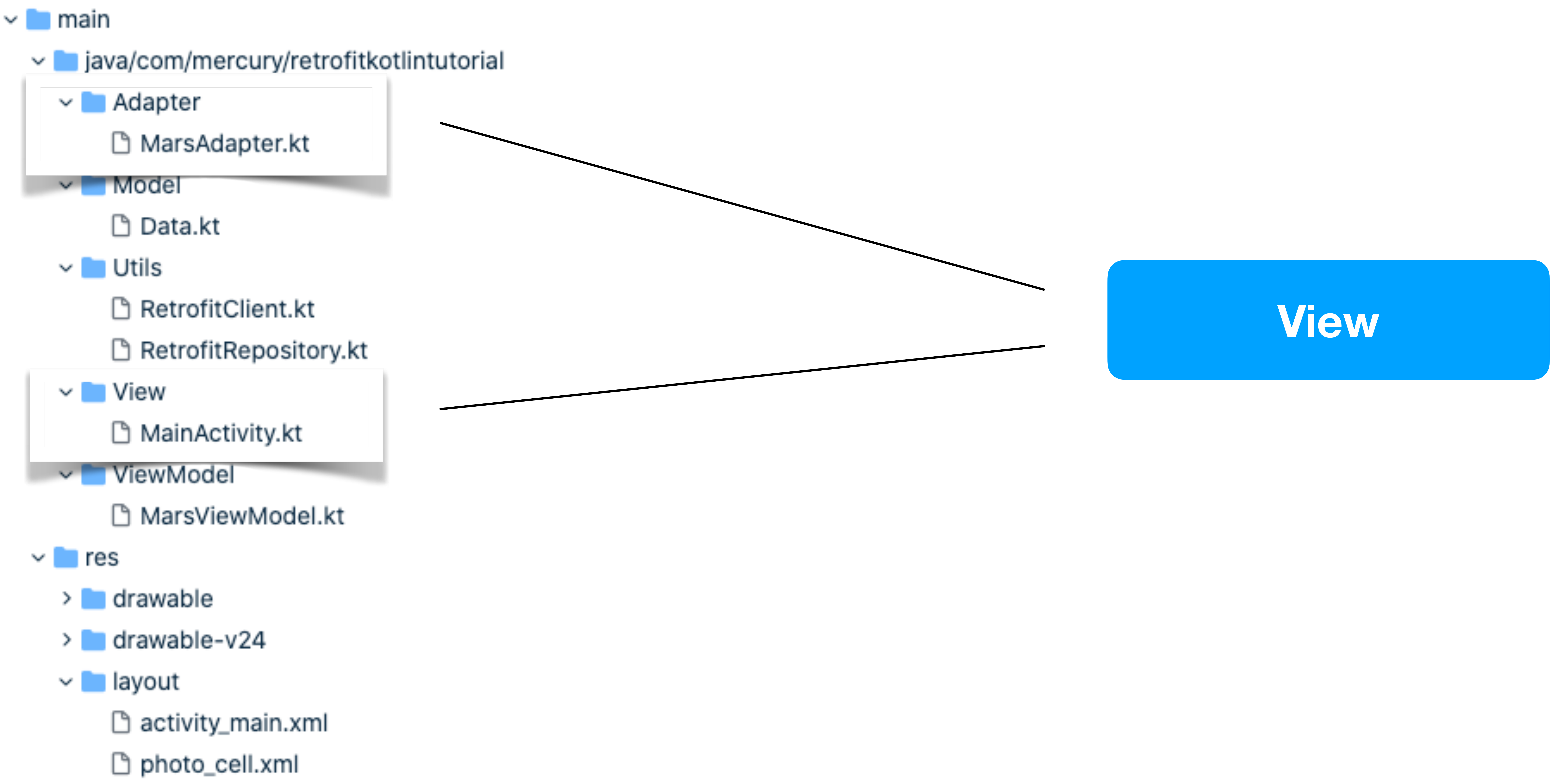
Structuring the code

Structuring the code



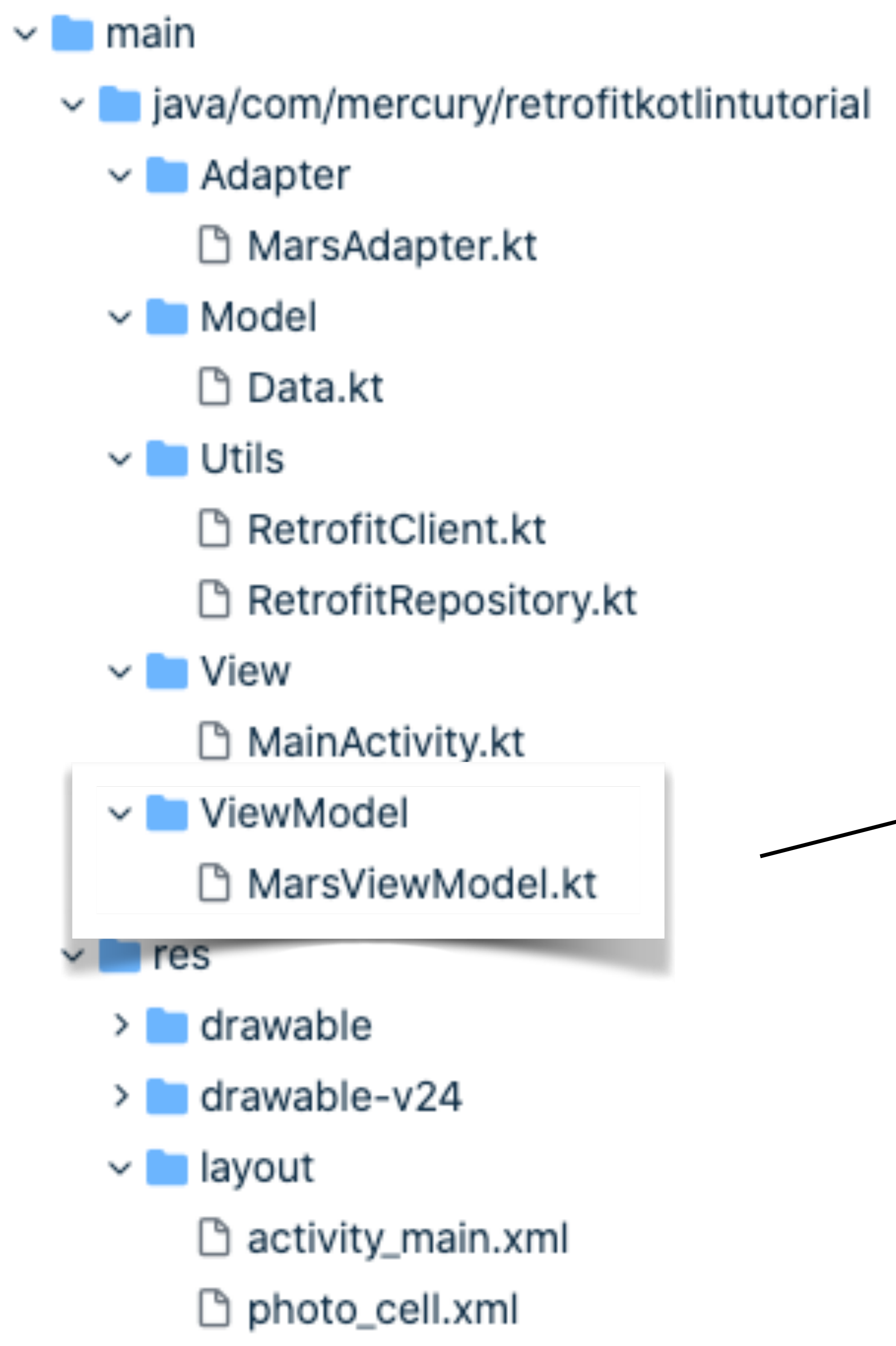
```
main
  java/com/mercury/retrofitkotlintutorial
    Adapter
      MarsAdapter.kt
    Model
      Data.kt
    Utils
      RetrofitClient.kt
      RetrofitRepository.kt
    View
      MainActivity.kt
    ViewModel
      MarsViewModel.kt
  res
    drawable
    drawable-v24
    layout
      activity_main.xml
      photo_cell.xml
```

Structuring the code



```
main
  java/com/mercury/retrofitkotlintutorial
    Adapter
      MarsAdapter.kt
    Model
      Data.kt
    Utils
      RetrofitClient.kt
      RetrofitRepository.kt
    View
      MainActivity.kt
    ViewModel
      MarsViewModel.kt
  res
    drawable
    drawable-v24
    layout
      activity_main.xml
      photo_cell.xml
```

Structuring the code

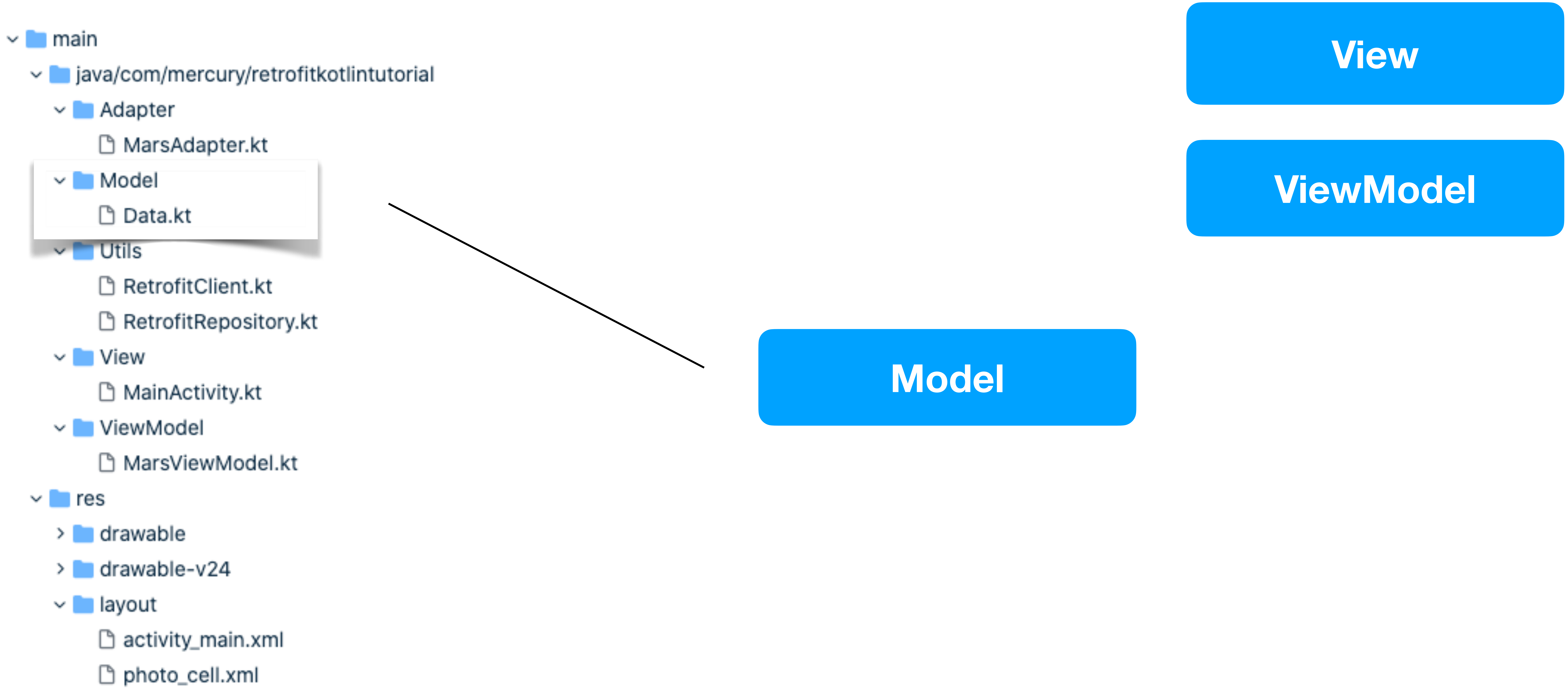


```
main
  java/com/mercury/retrofitkotlintutorial
    Adapter
      MarsAdapter.kt
    Model
      Data.kt
    Utils
      RetrofitClient.kt
      RetrofitRepository.kt
    View
      MainActivity.kt
    ViewModel
      MarsViewModel.kt
  res
    drawable
    drawable-v24
    layout
      activity_main.xml
      photo_cell.xml
```

View

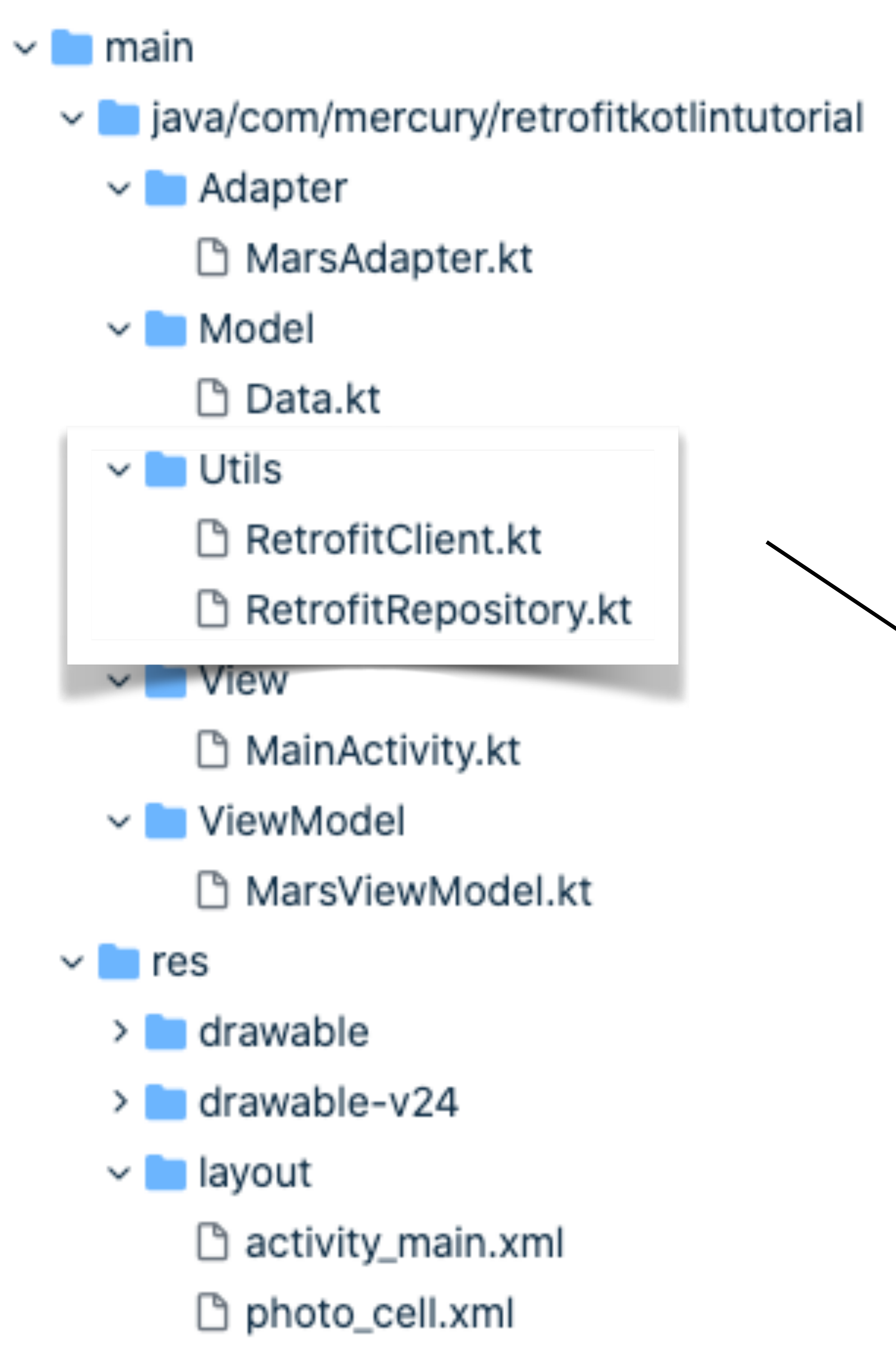
ViewModel

Structuring the code



```
main
  java/com/mercury/retrofitkotlintutorial
    Adapter
      MarsAdapter.kt
    Model
      Data.kt
    Utils
      RetrofitClient.kt
      RetrofitRepository.kt
    View
      MainActivity.kt
    ViewModel
      MarsViewModel.kt
  res
    drawable
    drawable-v24
    layout
      activity_main.xml
      photo_cell.xml
```

Structuring the code



```
main
  java/com/mercury/retrofitkotlintutorial
    Adapter
      MarsAdapter.kt
    Model
      Data.kt
    Utils
      RetrofitClient.kt
      RetrofitRepository.kt
    View
      MainActivity.kt
    ViewModel
      MarsViewModel.kt
  res
    drawable
    drawable-v24
    layout
      activity_main.xml
      photo_cell.xml
```

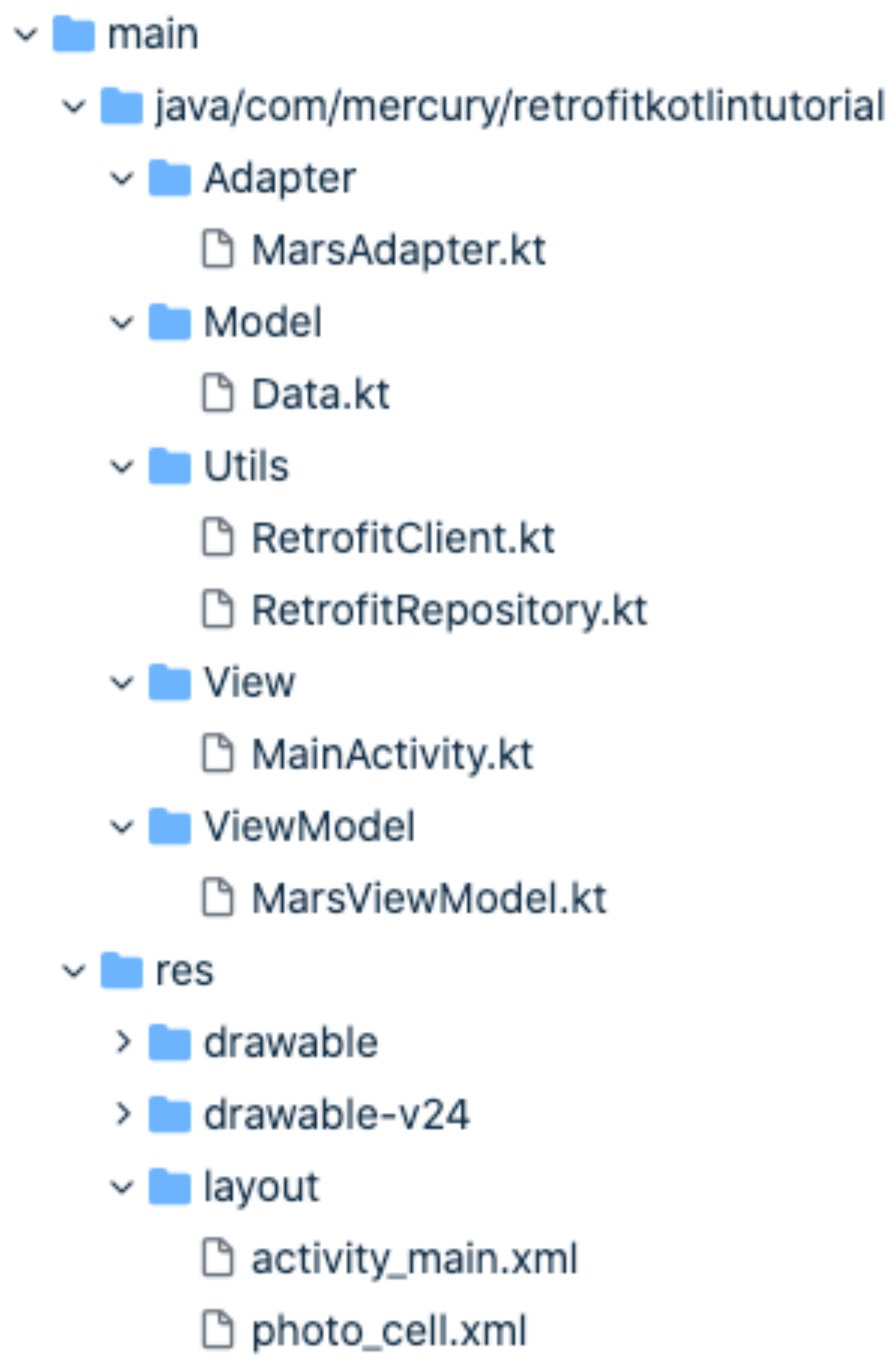
View

ViewModel

Model

Repository

Structuring the code



```
main
  java/com/mercury/retrofitkotlintutorial
    Adapter
      MarsAdapter.kt
    Model
      Data.kt
    Utils
      RetrofitClient.kt
      RetrofitRepository.kt
    View
      MainActivity.kt
    ViewModel
      MarsViewModel.kt
  res
    drawable
    drawable-v24
    layout
      activity_main.xml
      photo_cell.xml
```

View

ViewModel

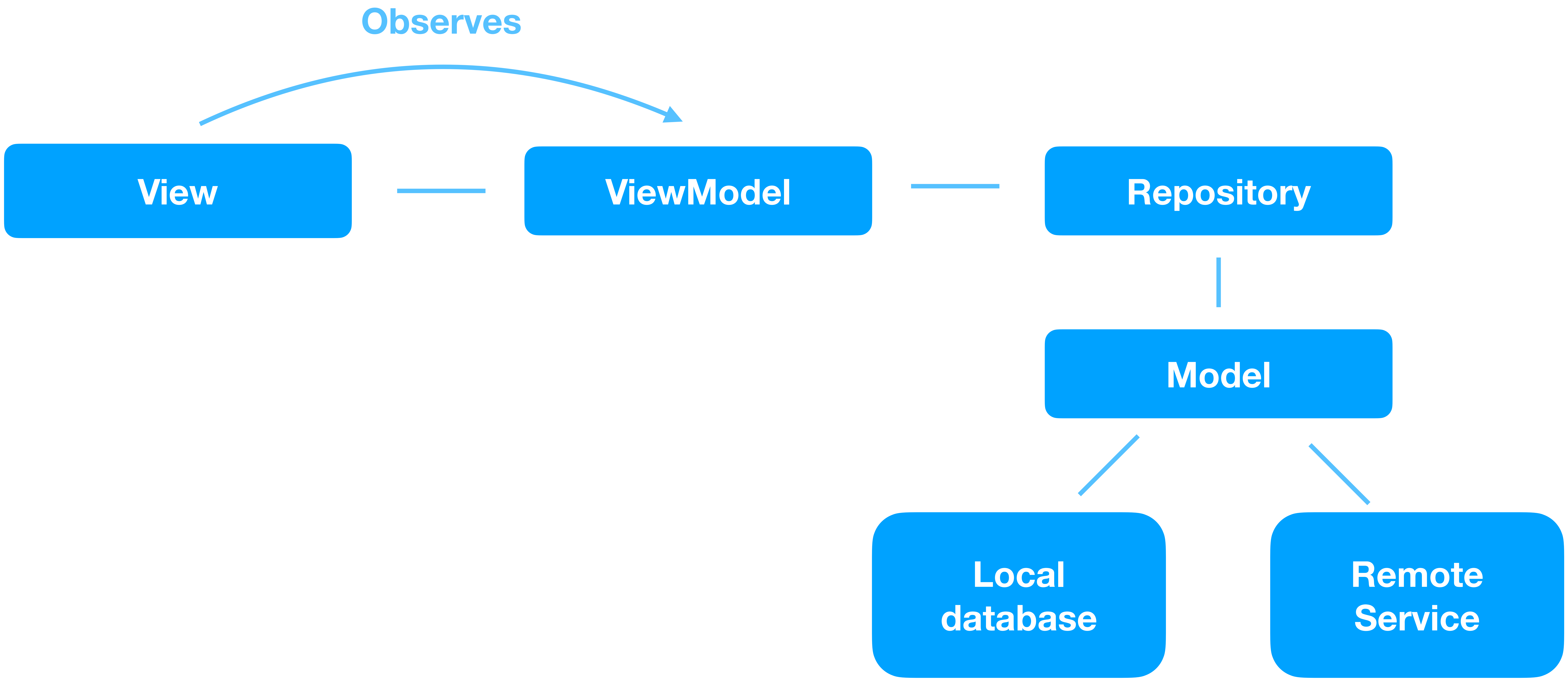
Model

Repository

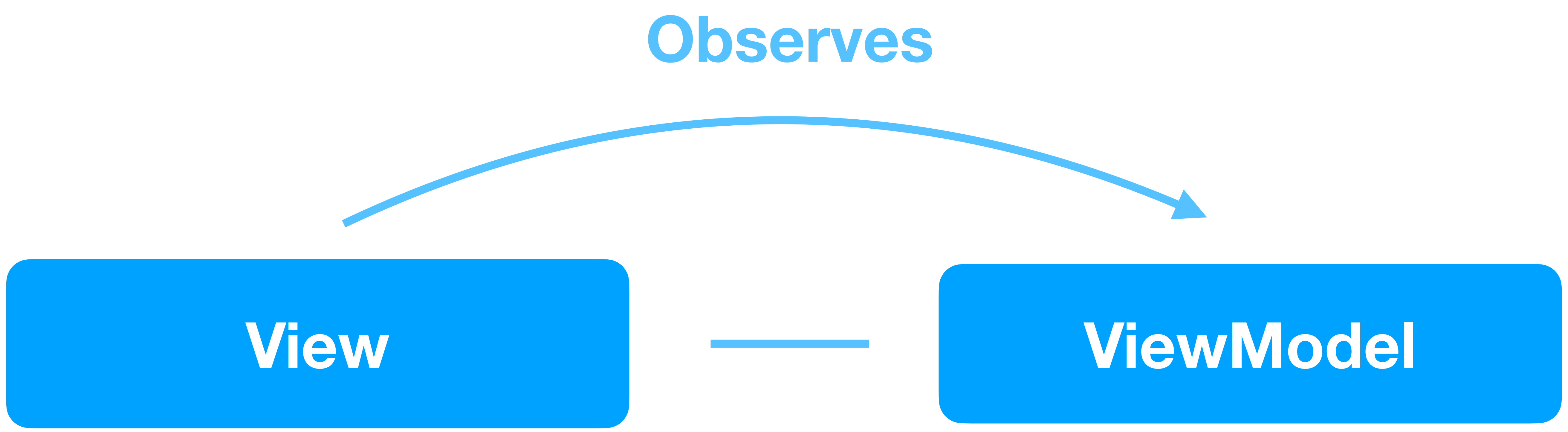
Structuring the code

View	Does UI related work
ViewModel	Exposes data in the required format for views to consume
Model	Holds the data
Repository	Acts as single source of truth

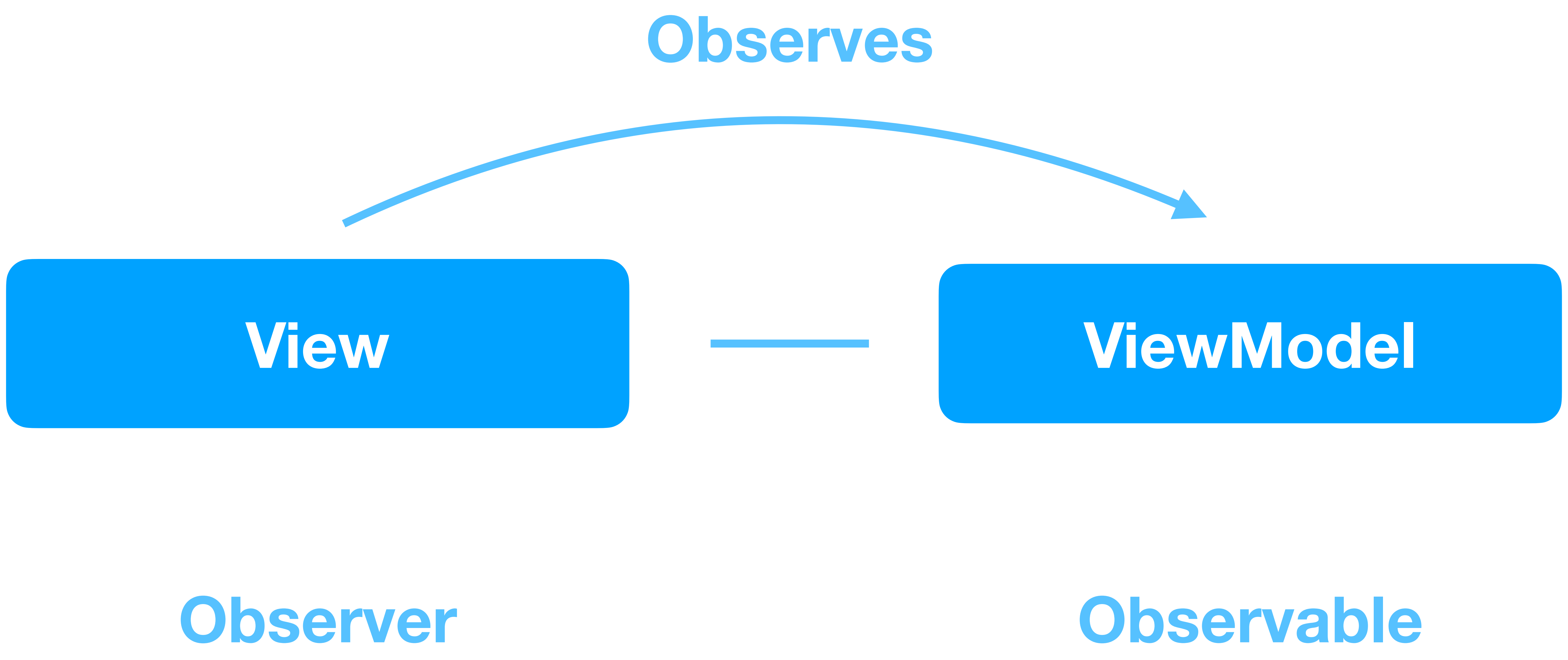
Structuring the code



Structuring the code



Structuring the code



Observes

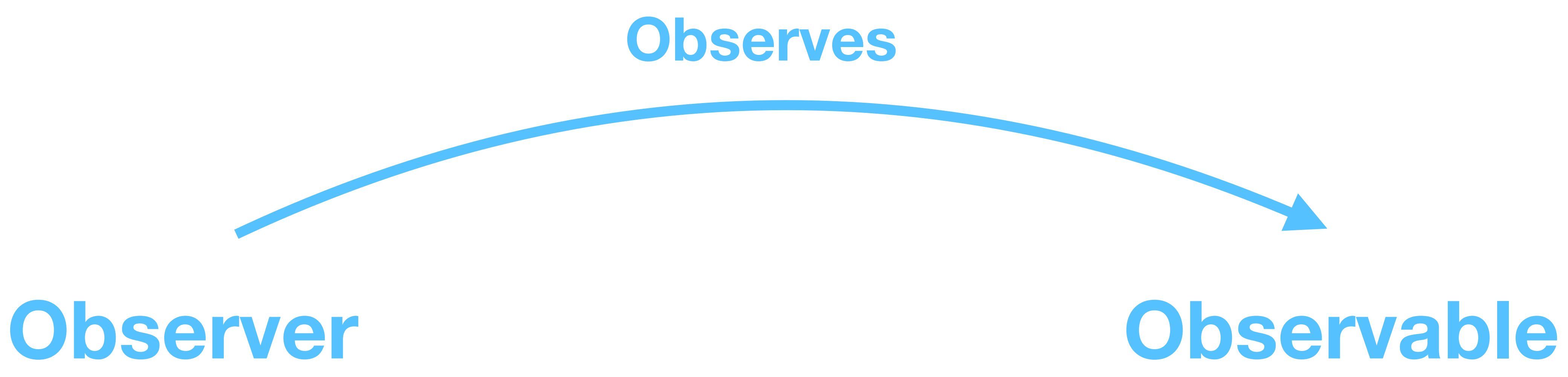
View

ViewModel

Observer

Observable

Structuring the code

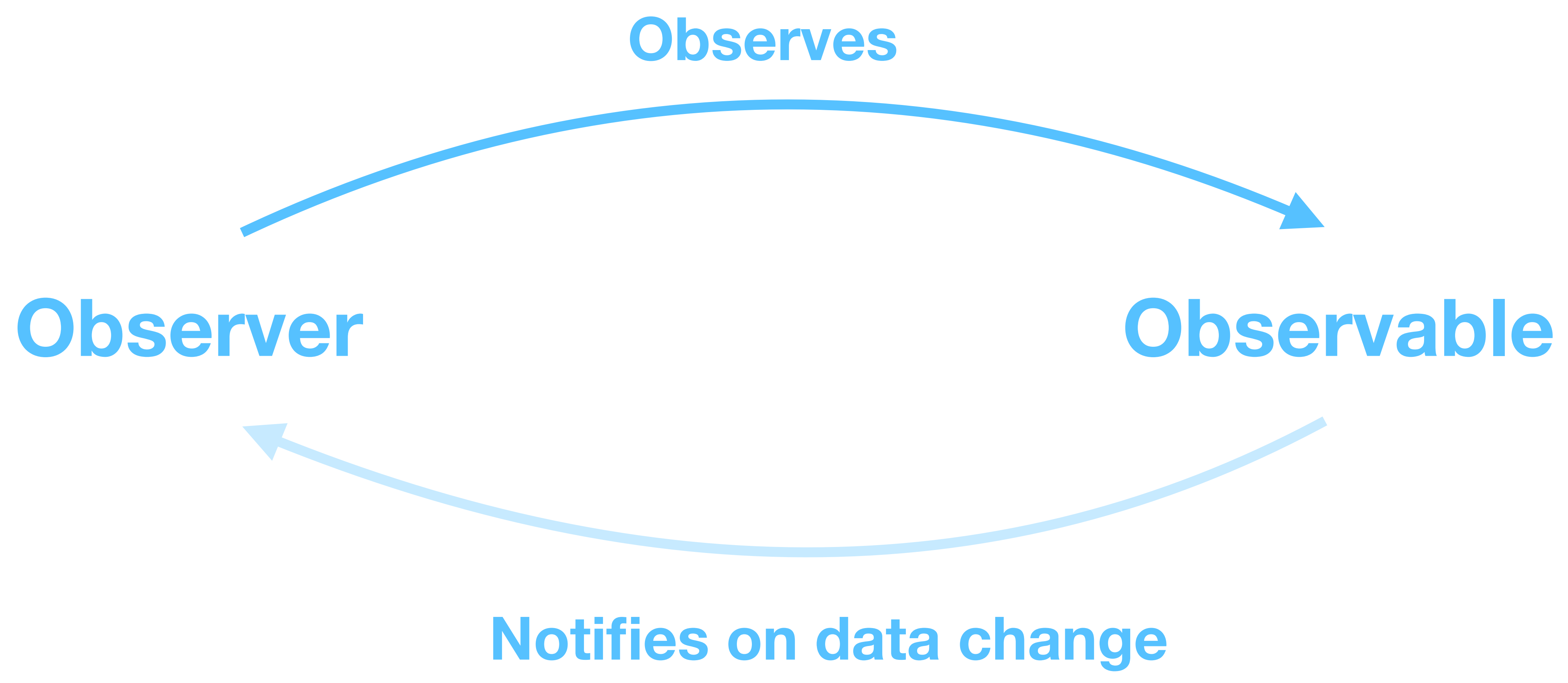


Observes

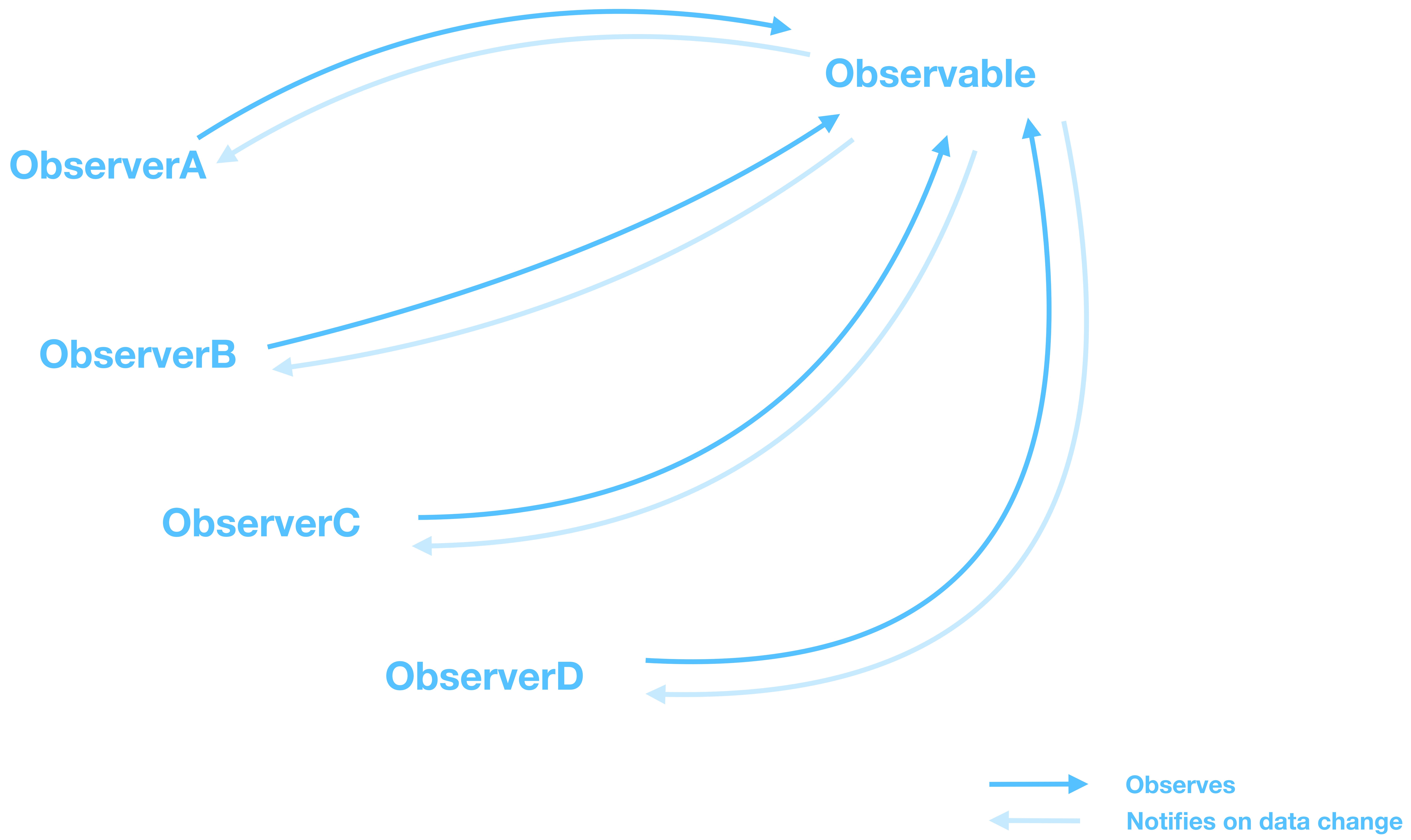
Observer

Observable

Structuring the code



Structuring the code



Structuring the code

What are the benefits?

What are the benefits?

1. Easier to test

    - Less dependencies
    - Easier to write UI test
    - Easier to write unit test

Structuring the code

What are the benefits?

2. Improves readability

Structuring the code

What are the benefits?

3. Single responsibility

Structuring the code

What are the benefits?

4. Reactive UI

Thank you