

# Android for .NET Developers Series

## Building an Android App

### Views and Layouts

Jim Wilson

[jimw@jwhh.com](mailto:jimw@jwhh.com)

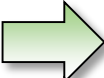


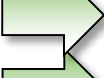

@hedgehogjim

<http://facebook.com/hedgehogjim>

<http://hedgehogjim.wordpress.com>

**pluralsight**  
hardcore developer training

# Outline

-  The role of Views and Layouts
-  Using Layouts
-  LinearLayout
-  RelativeLayout
-  Handling View event callbacks

# The role of Views and Layouts

➔ **Android UI is composed of Layouts and Views**

➔ Views are UI components

- ViewGroups are Views that hold other Views

➔ Layouts are specialized ViewGroups

- Describe View positioning
- A Layout is commonly the root of the UI
- Layouts are often layered within one another

➔ Constructing the UI

- Layout resource
  - Resources are preferred and most adaptive
- Code

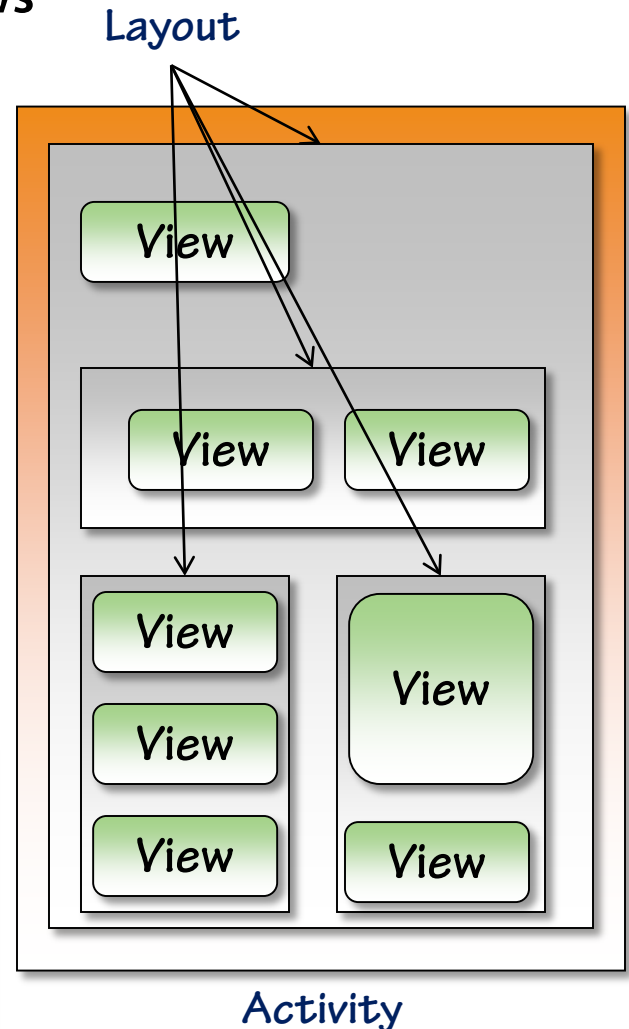


**activity\_main.xml**

```
<LinearLayout ...>
  <TextView ... />
  ...
  ...
</LinearLayout>
```

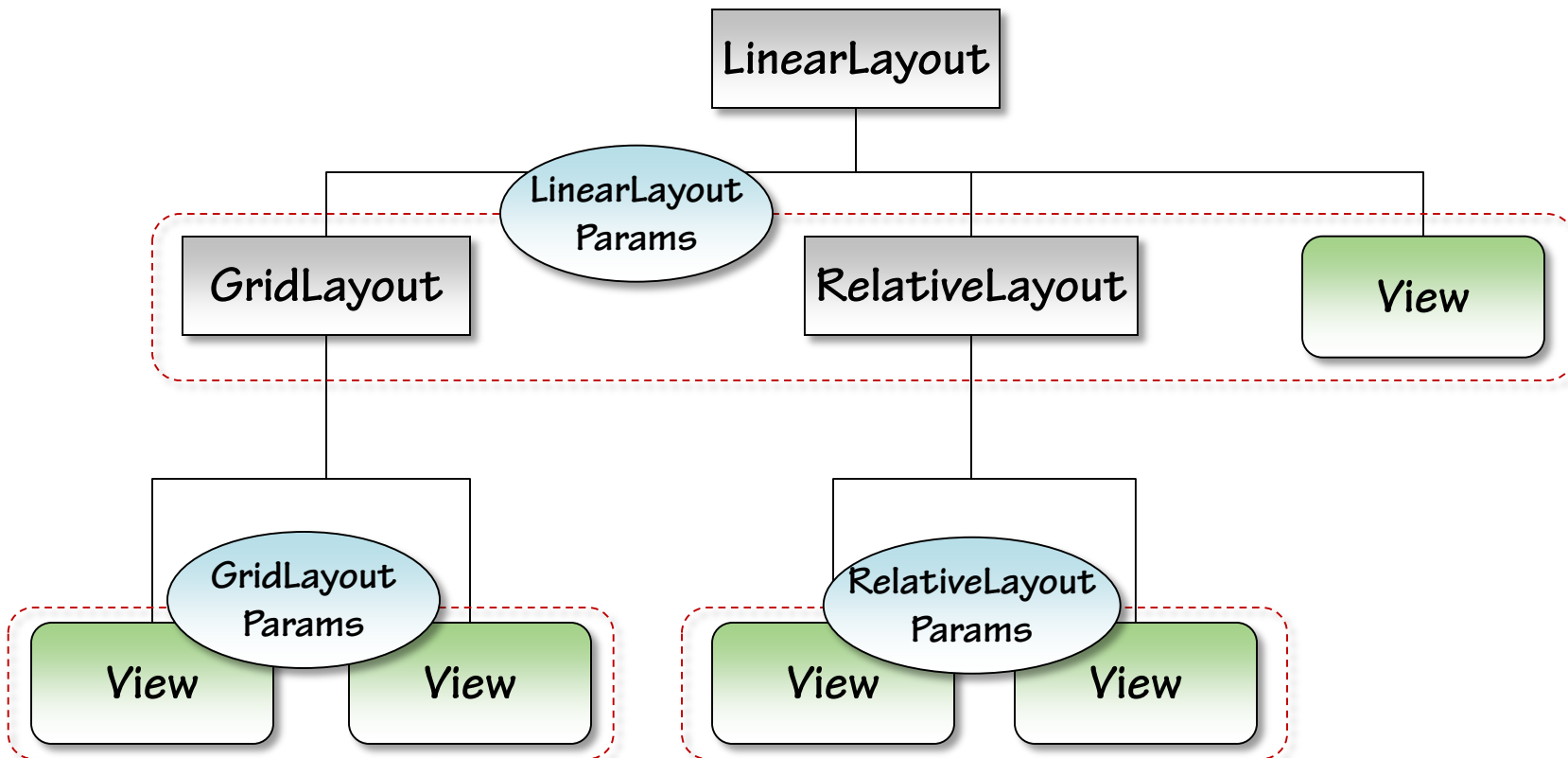
**MainActivity.js**

```
LinearLayout layout =
  new LinearLayout(...);
...
...
setContentView(layout);
```



# Understanding Layout

- ➡ **Layouts influence the positioning of Views**
- ➡ View layout is designed around hints
- ➡ Each layout provides a specific positioning behavior for contained views

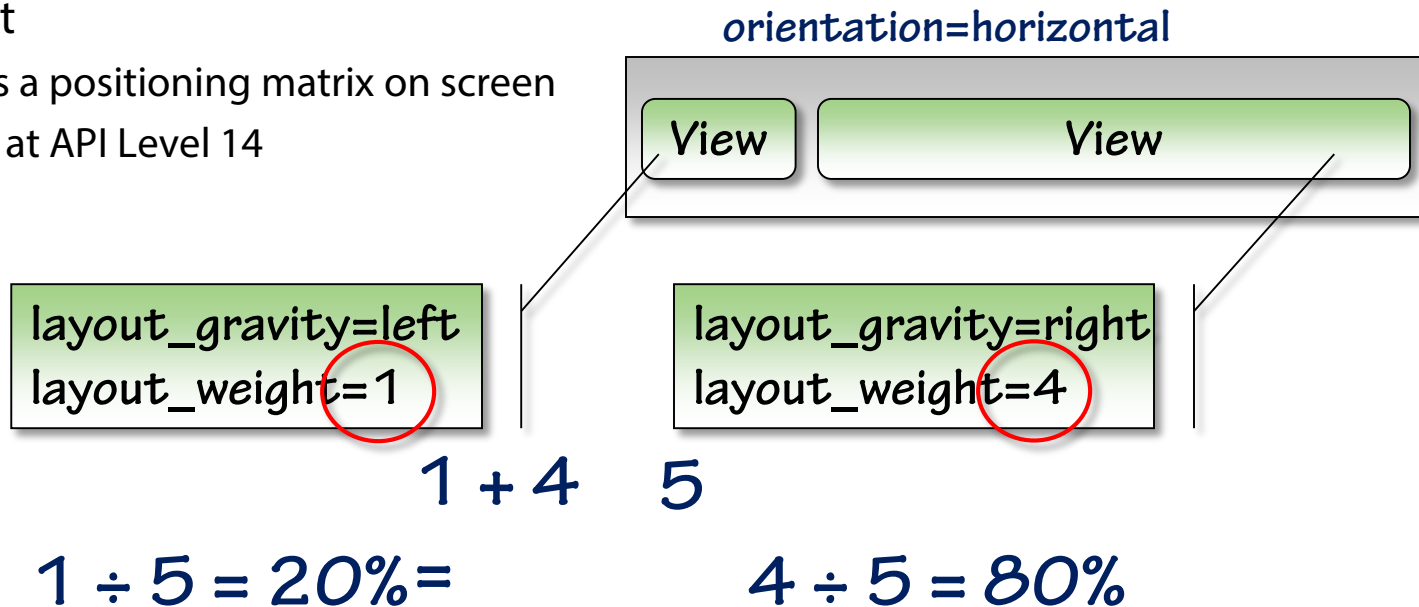


# Using Layout

➡ Each Layout provides positioning behavior

➡ LinearLayout

- Distributes Views in a horizontal or vertical line
- RelativeLayout
  - Positions Views relative to other Views
- GridLayout
  - Creates a positioning matrix on screen
  - Added at API Level 14

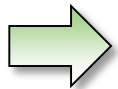


# Using Layout

- Each Layout provides positioning behavior

- LinearLayout

- Distributes Views in a horizontal or vertical line

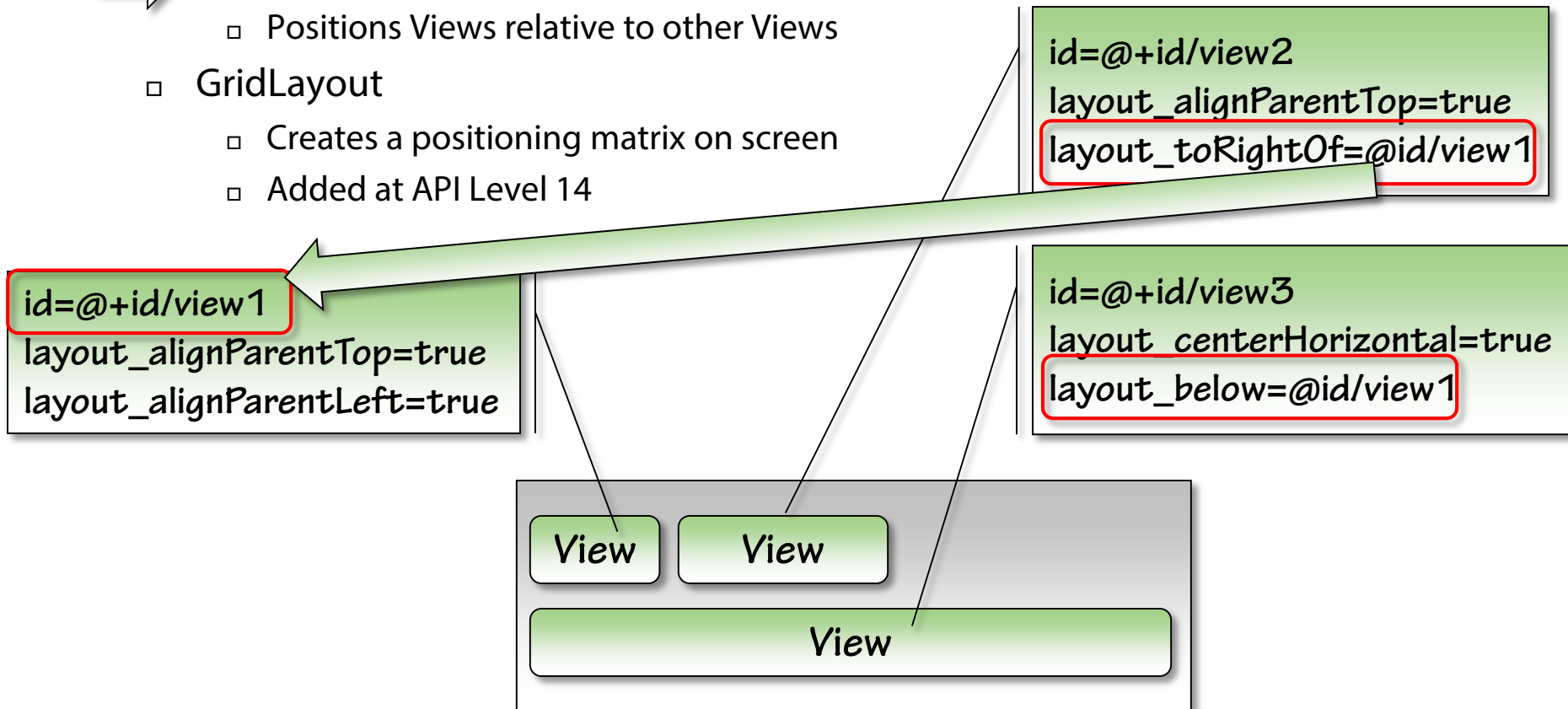


- RelativeLayout

- Positions Views relative to other Views

- GridLayout

- Creates a positioning matrix on screen
    - Added at API Level 14



# Using Layout

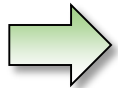
- Each Layout provides positioning behavior

- LinearLayout

- Distributes Views in a horizontal or vertical line

- RelativeLayout

- Positions Views relative to other Views



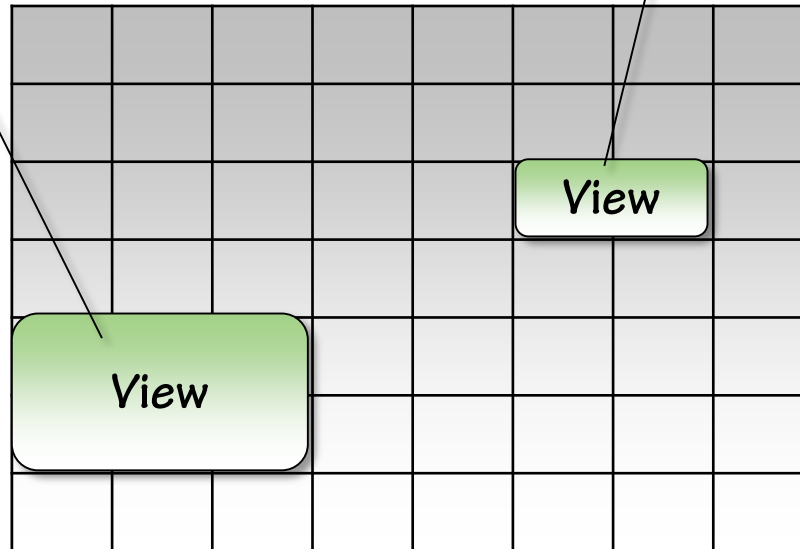
- GridLayout

- Creates a positioning matrix on screen

- Added at API Level 14

`layout_row=2`  
`layout_column=5`  
`layout_rowSpan=1`  
`layout_columnSpan=2`

`layout_row=4`  
`layout_column=0`  
`layout_rowSpan=2`  
`layout_columnSpan=3`



`columnCount=8`  
`rowCount=7`

# Mimicking delegates for event handling

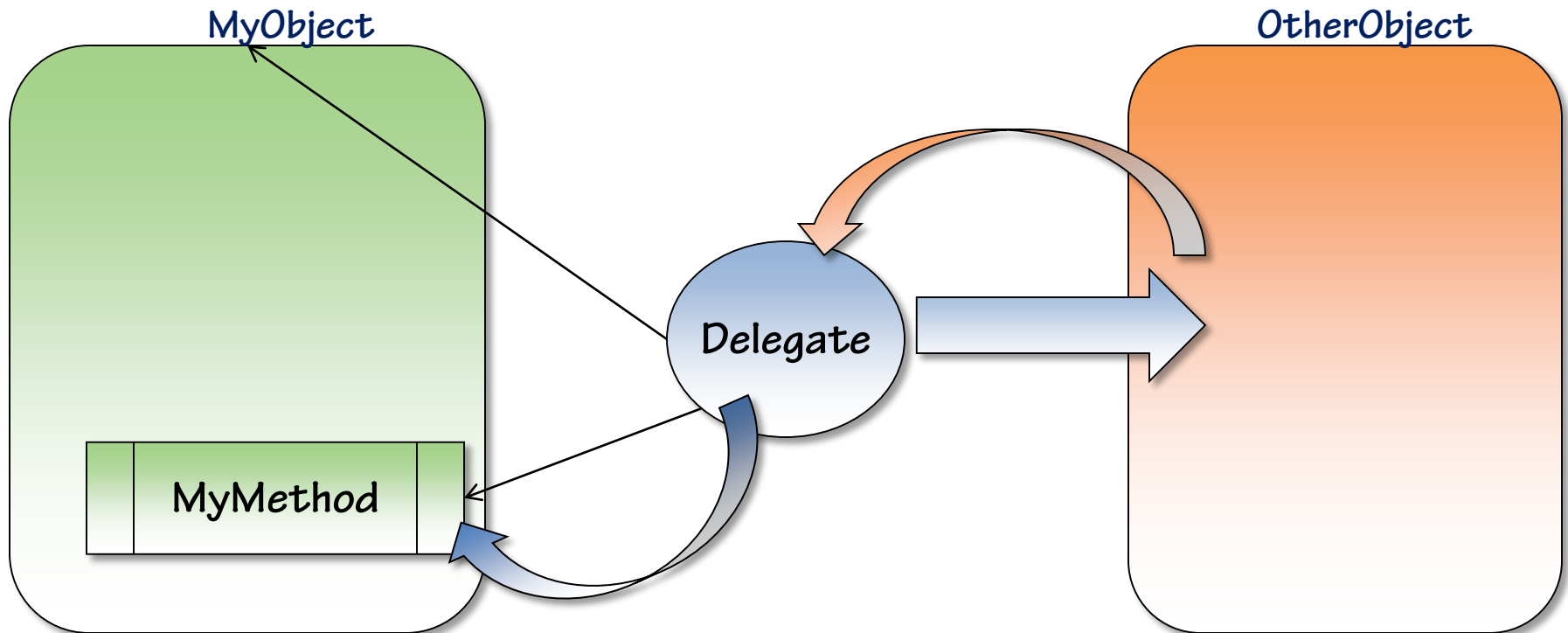
➡ **Java doesn't support delegates but we can fake it**

➡ .NET delegates allow callbacks into object methods

- Has a reference to an object instance
- Has a reference to a method within that object

➡ Delegate can be passed to another object

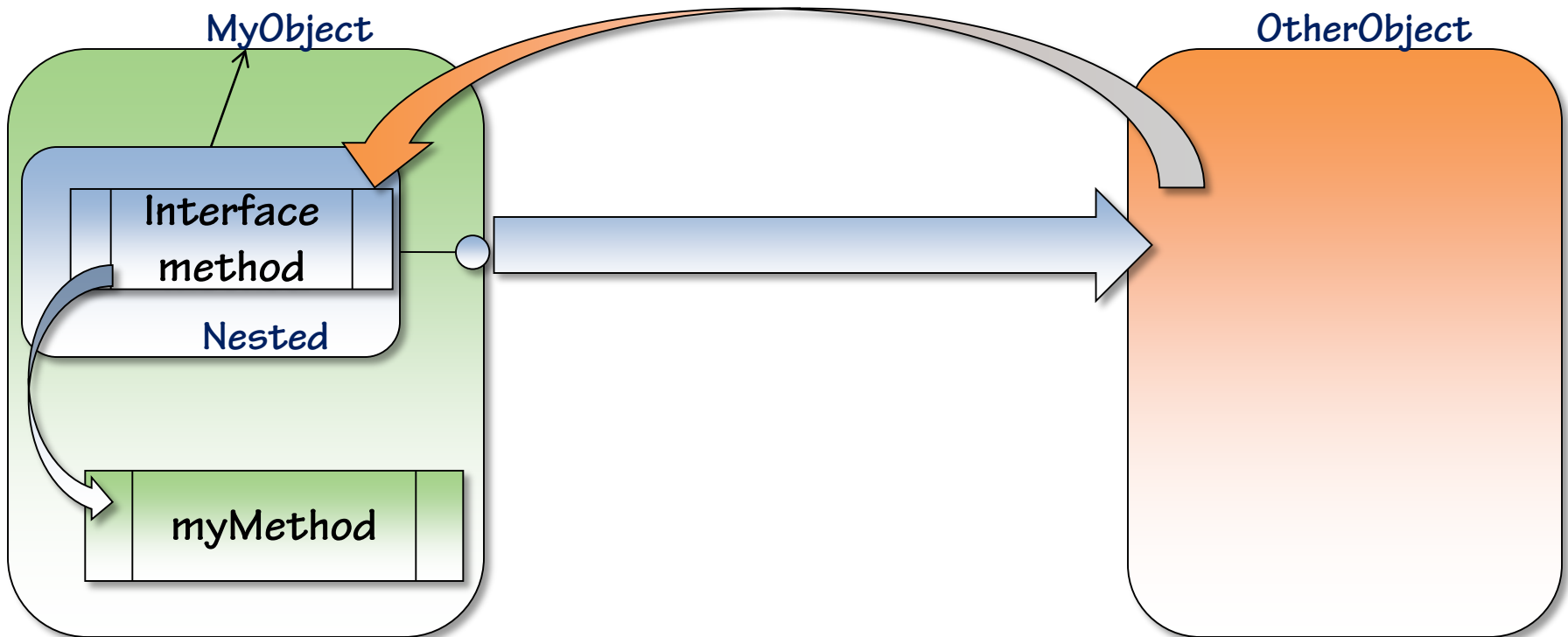
- Receiving object can call back to the specified method





# Mimicking delegates in Java

- ➡ **Nested, anonymous classes give us what we need for many cases**
- ➡ Nested classes give us adequate functionality for common event handling
  - ❑ Can hold a reference to the class in which they are defined
  - ❑ Can call methods on the class on which it's nested
- ➡ Anonymous classes simplify using nested classes for event handling
  - ❑ Provides a simple syntax for creating an interface implementation



# Summary

## **The parts of the UI**

- Views, ViewGroups, and Layouts

## **LinearLayout**

- Provides weighted positioning of Views in a line

## **RelativeLayout**

- Positions Views relative to the Layout or other Views within the Layout

## **Accessing Views within an Activity**

- Use findViewById

## **Handling View events**

- Android relies on interfaces for event handling
- Use Nested, Anonymous classes to simplify event handling