AD340 Mobile Application Development

Week 03

Outline

Make an app interactive

Building an Android App

Accessibility

Layouts in general

ConstraintLayout

Last Week

- Android Studio Tour
- Create your first Android Studio Project
- Device Manager & Virtual Device
- Layouts
- App Resources
- Activity

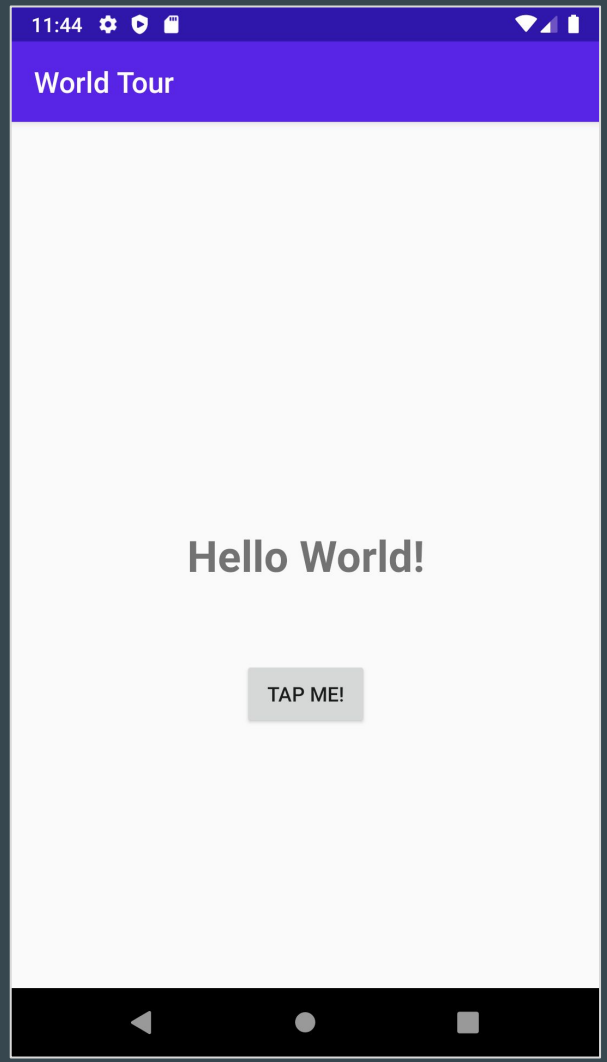
Make an app interactive

Define app behavior in Activity

You can define how your app will behave in the Activity file of your app.

Example

- modify the MainActivity so that the "Hello World" is displayed when the button is tapped.



Modify a View dynamically

You can also dynamically modify the views in your layout

Example:

- Get a reference to the View in the view hierarchy using findViewById() method
- Pass in the resource ID for that TextView
- Change properties or call methods on the TextView

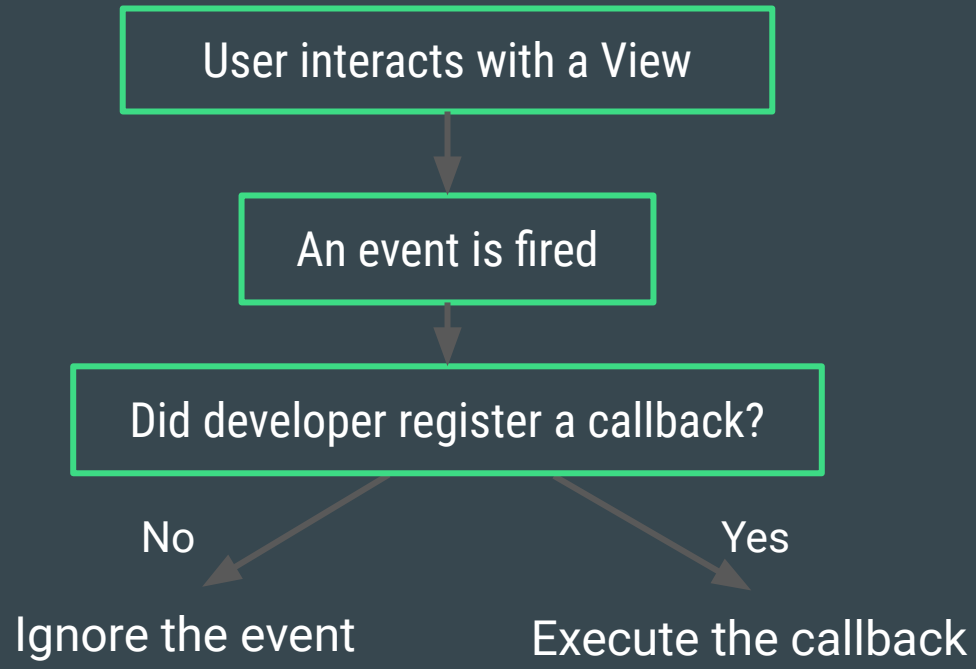
```
val resultTextView: TextView = findViewById(R.id.textView)

resultTextView.text = "Goodbye!"
```

Set up listeners for specific events

set up listeners for certain things to happen based on user input

base View class has a set of events that you can register callbacks to respond to

View.OnClickListener is the one you'll interact with the most.



how to set an OnClickListener on a View

need to create the OnClickListener

the MainActivity implement the OnClickListener interface

set the click listener (this) on the Button when the Activity is created

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val button1: Button = findViewById(R.id.button)
        button1.setOnClickListener { it: View!
            // do something
        }
    }
```

Late initialization

Kotlin has a strong preference for null safety

needing to create properties or fields that may depend on function calls such as onCreate

Option 1: make the type nullable and set the initial value to null.

- Downside: need to have a null check

Option 2: use lateinit.(tells the Kotlin compiler that the programmer is responsible for the initialization)

```
class MainActivity : AppCompatActivity() {

    lateinit var imageResult: ImageView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        imageResult = findViewById(R.id.imageView)
```

Assignments

[Assignment] Interactive Button - Display a Toast

[Assignment] Click Counter

[Assignment] Calm Shiba Inu, Angry Shiba Inu

Building an Android app

What is Gradle?

Builds automation system

Manages the build cycle via a series of tasks (for example, compiles Kotlin sources, runs tests, installs app to device)

Determines the proper order of tasks to run

Manages dependencies between projects and third-party libraries

Gradle build file

Declare plugins

Define Android properties

Handle dependencies

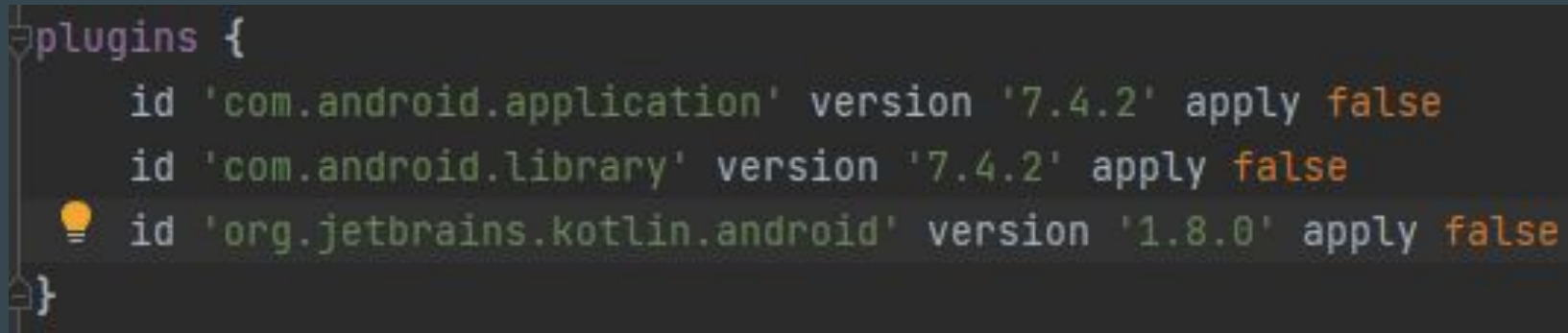
Connect to repositories

Plugins

build.gradle file will usually include a plugin section

Plugins provide additional functionality as supplementary tasks and dependencies

- No need to declare all the dependencies



Android configuration

set the specifics of what makes your app run on Android

Example

- SDK versions
- test runners
- build targets
- etc

```
android {
    namespace 'com.example.dicerollerapp'
    compileSdk 33

    defaultConfig {
        applicationId "com.example.dicerollerapp"
        minSdk 33
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
```

Dependencies

lets you refer to libraries you want to use

- Both locally or remote
- sub-dependencies

```
dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

Repositories

You can find in settings.gradle

there are collections of libraries organized in repositories

Use the repositories block to specify where to look for them

Gradle will start at the first listed repository and fall through to each subsequent

```
pluginManagement {
    repositories {
        google()
        mavenCentral()
        gradlePluginPortal()
    }
}
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode
    repositories {
        google()
        mavenCentral()
    }
}
rootProject.name = "DiceRollerApp"
include ':app'
```

Common Gradle Tasks

Clean

- The clean task deletes all compiled files from the build directory.

Tasks

- Tasks outputs a list of tasks for your project and any installed plugins.

IntallDebug

- InstallDebug compiles the app if necessary, builds a debug APK, and installs it on a connected physical or emulated device.

Accessibility

Accessibility

Often abbreviated as “a11y”

Refers to improving the design and functionality of your app to make it easier for more people, including those with disabilities, to use

Making your app more accessible leads to an overall better user experience and benefits all your users

Tips: Make apps more accessible

Increase text visibility with foreground and background color contrast ratio

Use large, simple controls

- Touch target size should be at least 48dp x 48dp

Describe each UI element

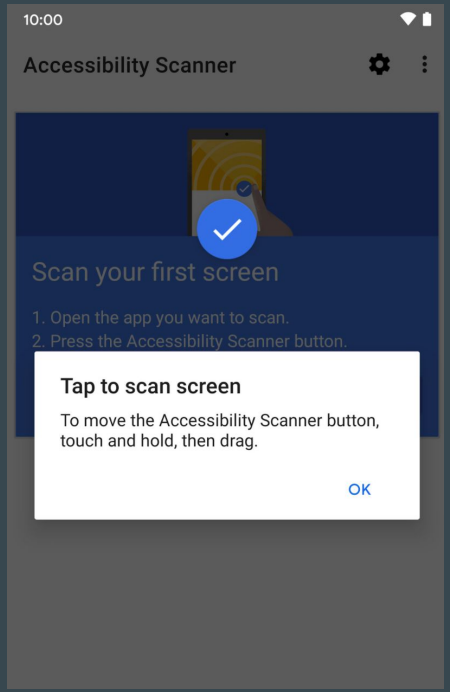
- Set content description on images and controls

https://developer.android.com/guide/topics/ui/accessibility

Accessibility Scanner

Tool that scans your screen and suggests improvements to make your app more accessible, based on:

- Content labels
- Touch target sizes
- Clickable views
- Text and image contrast



https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.auditor&pli=1

TalkBack

- Google screen reader included on Android devices
- Provides spoken feedback so you don't have to look at the screen to use your device
- Lets you navigate the device using gestures
- Includes braille keyboard for Unified English Braille

It is good to test your app with TalkBack turned on

https://developer.android.com/guide/topics/ui/accessibility/testing#talkback

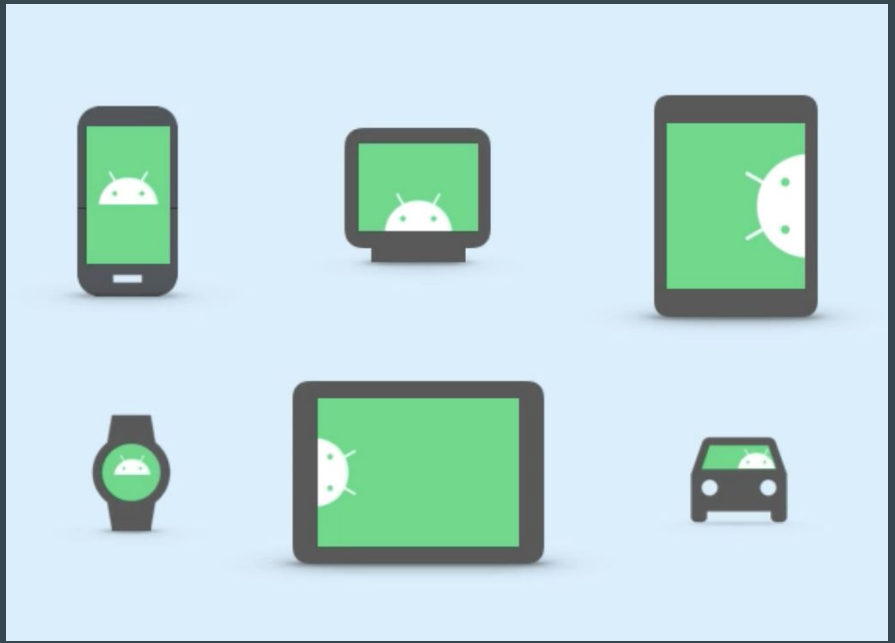
Layouts in General

Android devices

Android devices come in many different form factors.

More and more pixels per inch are being packed into device screens.

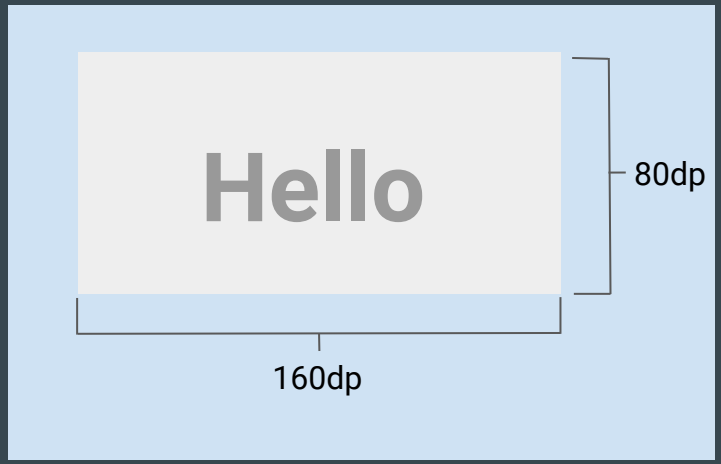
Developers need the ability to specify layout dimensions that are consistent across devices.



Density-independent pixels (dp)

Use dp when specifying sizes in your layout, such as the width or height of views.

- Density-independent pixels (dp) take screen density into account.
- Android views are measured in density-independent pixels.
- Using dp and adaptive layouts like ConstraintLayout ensure that your design and layout will work well across different devices
- dp = (width in pixels * 160)/screen density



Screen-density buckets

Screens are organized in several density buckets (Dots Per Inch)

- Low density (~120 dpi)
- Medium density (~160dpi)
- High density (~240dpi)
- Extra-high density (~320dpi)
- Extra-extra-high density (~480dpi)
- Extra-extra-extra-high density (~640dpi)

Most consumer devices currently fall between hdpi and xxxhdpi.

Device name? DPI?

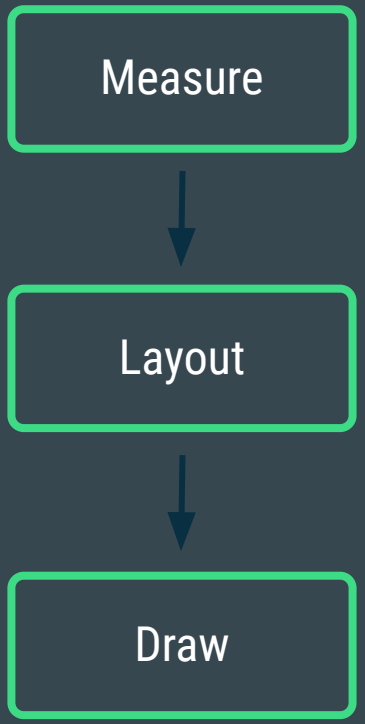
Android View rendering cycle

When Android draws information on the screen, three distinct passes happen in quick succession:

Measure pass - calculate the precise dimensions of each View, including expensive operations like wrap_contents you used.

Layout pass - align views based on the rules of the layout manager.

Draw - render the results based on the previous two steps.



Drawing region

when it comes to drawing it on-screen

the draw calls are bounded by rectangles

an invisible border to the View

What we see:

How it's drawn:

ConstraintLayout

Deeply nested layouts are costly

Placing views in ViewGroups, like LinearLayout, can help you organize your layout

nesting too many layouts within each other, can make your UI unresponsive

- Why? every element on the screen has to be measured precisely before it can be drawn
- Possible solution: ConstraintLayout

What is ConstraintLayout?

Recommended default layout for Android

ConstraintLayout duplicates all the functionality of LinearLayout and RelativeLayout

Layout flatter with less hierarchy and nesting, while allowing complex behavior

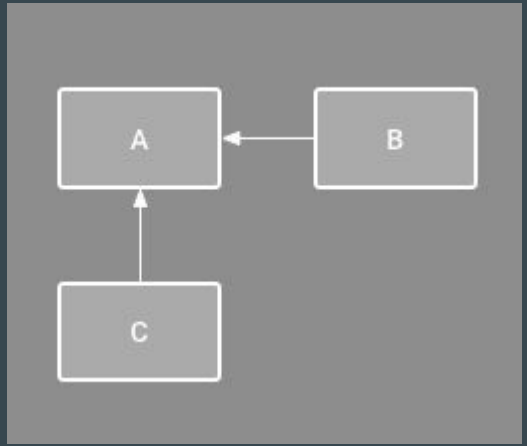
Position and size views within it using a set of constraints

What is a constraint?

A restriction or limitation on the properties of a View that the layout attempts to respect

Example

- B is constrained to always be to the right of A
- C is constrained below A



Relative positioning constraints

Can set up a constraint relative to the parent container

- the source is the current TextView
- target is the Parent container

Constraint 1:

- align the top of this TextView to the top of the parent layout.

Constraint 2:

- align the left of this TextView to the left of the parent layou

Relative positioning constraints : Vertical

Starting in the vertical direction

We can add a constraint to the

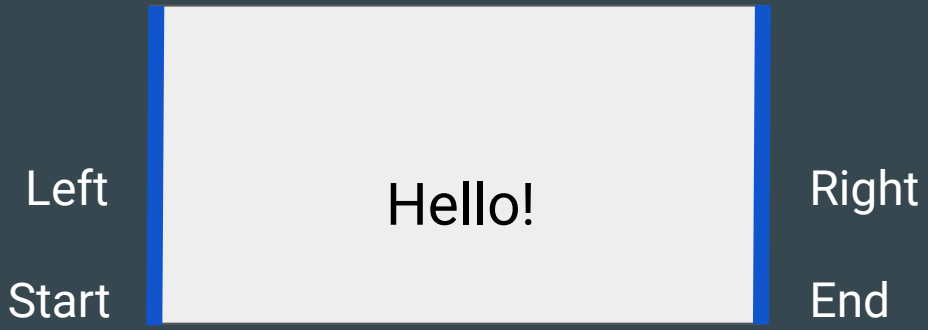
- top or bottom of an element
- text's baseline if the View contains text

Relative positioning constraints : Horizontal

Horizontally, we can constrain the start and end edges of a View.

best practice: default to start and end instead of left and right

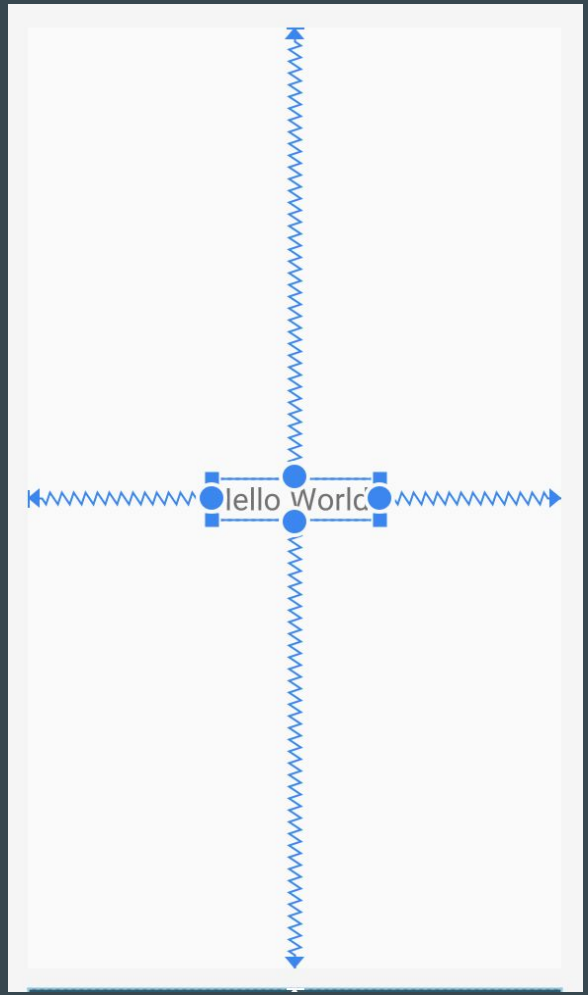
- **Why?** Some languages use Right-to-Left scripts



Simple ConstraintLayout example

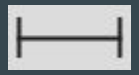
ConstraintLayout containing one child TextView

- Top constraint
- Bottom constraint
- Start constraint
- End constraint



Constraint Widget in Layout Editor

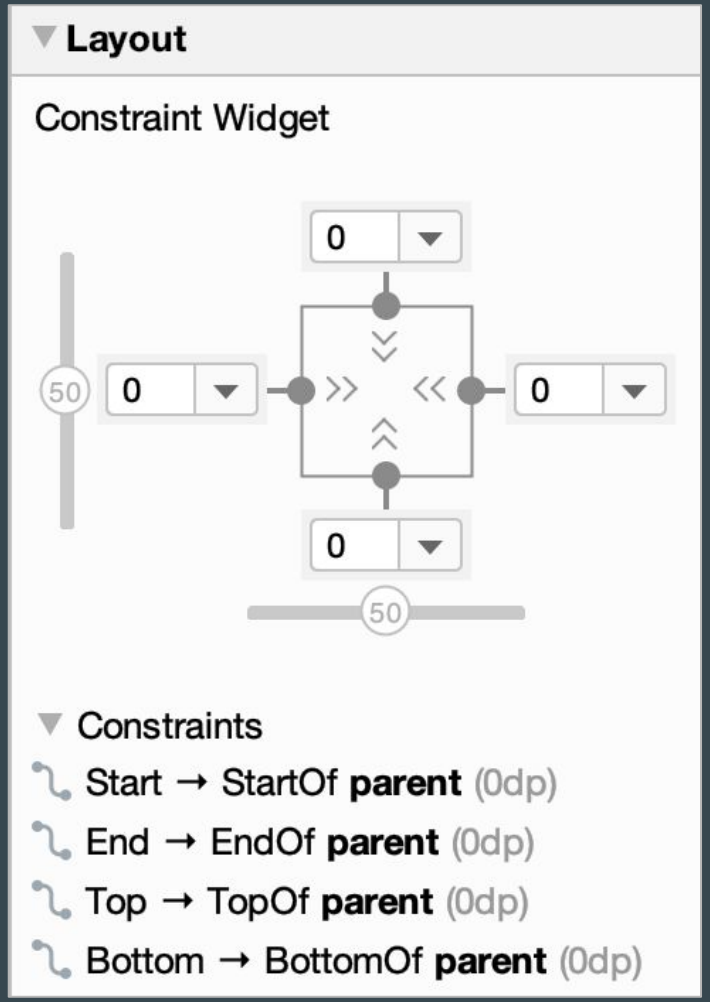
Three types of symbols. Will go over in next slides



Fixed

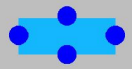
Wrap content

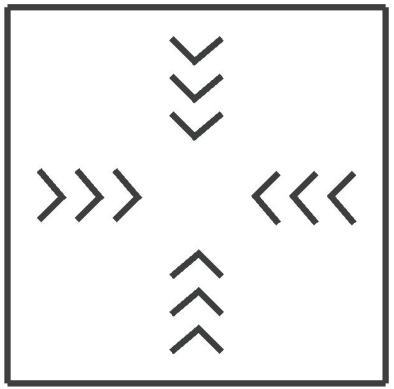
Match constraints



Wrap content for width and height

represents a view with wrap_content on its height and width.

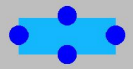


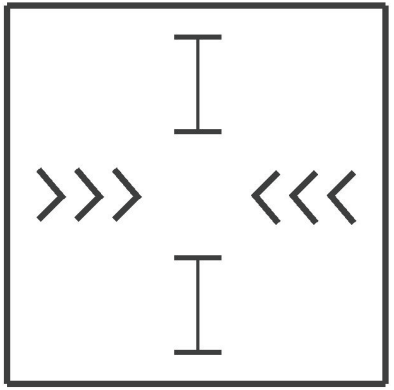


```
layout_width    wrap_content
layout_height   wrap_content
```

Wrap content for width, fixed height

Pipe symbol indicates that dimension has a fixed size in dp (48dp)



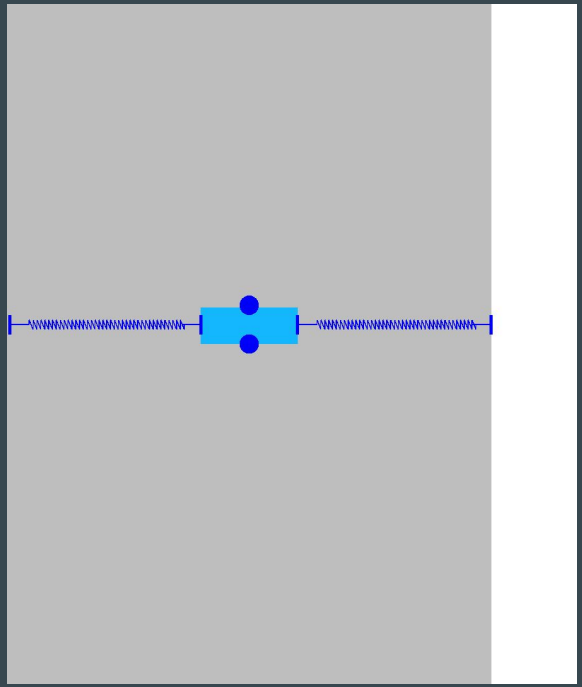


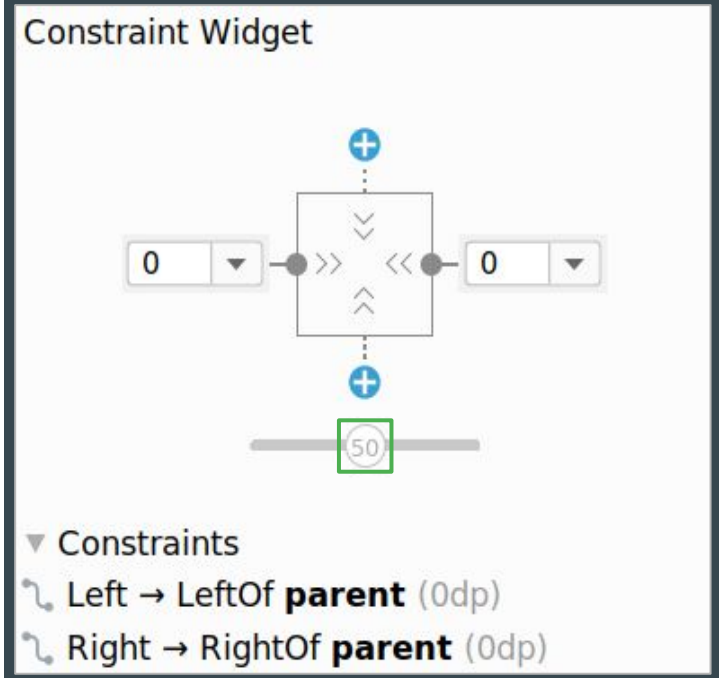
```
layout_width    wrap_content
layout_height   48dp
```

Center a view horizontally

start and end constraints on the View in the horizontal direction.

bias of 50: constraints are equally applied





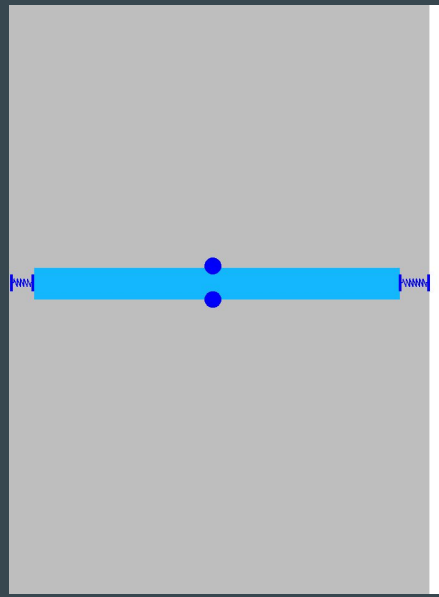
Use match_constraint

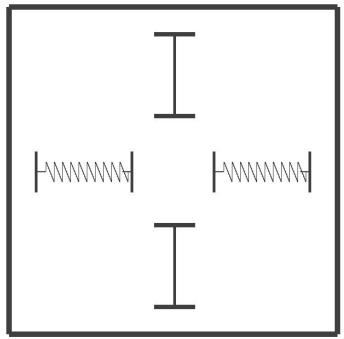
LinearLayout

- Can use match_parent (take up as much space available)

ConstraintLayout

- can't use match_parent on a child view
- Use match_constraint





```
layout_width    0dp(match_constraint)
layout_height   48dp
```

Chains

Let you position views in relation to each other

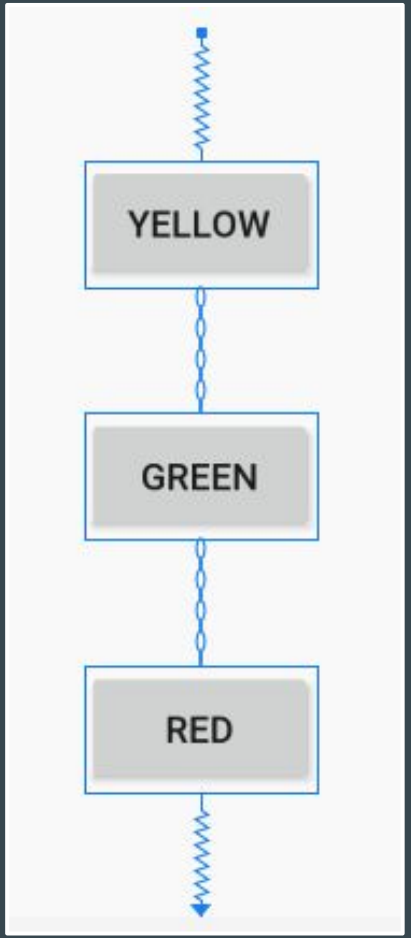
Can be linked horizontally or vertically

Provide much of LinearLayout functionality

Create a Chain in Layout Editor

Three steps to create a chain

- Select the objects you want to be in the chain.
- Right-click and select Chains.
- Create a horizontal or vertical chain.



Chain styles: Overview

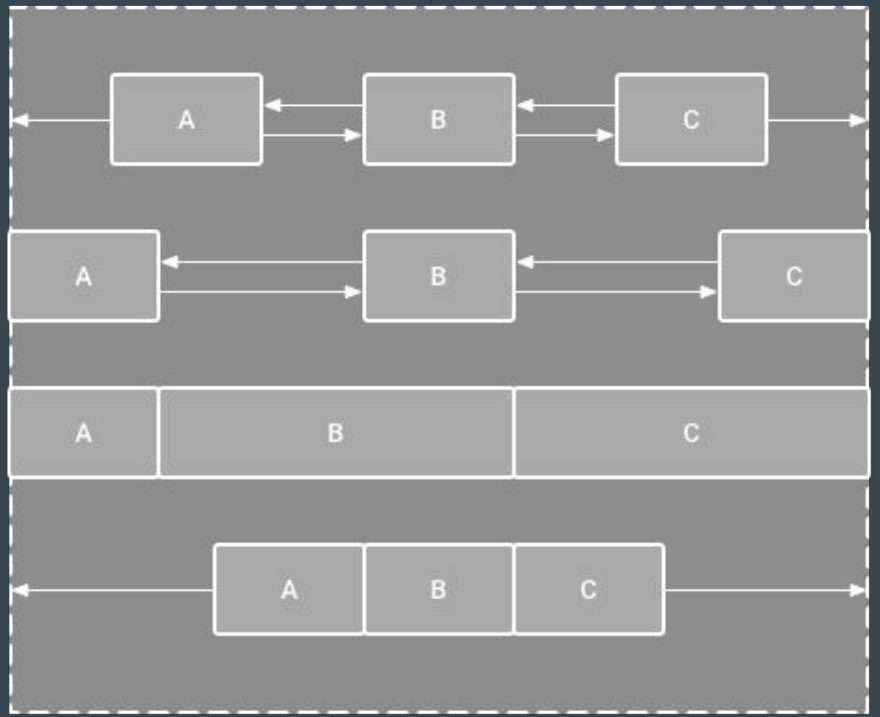
Chains can be styled in a number of ways

Spread Chain

Spread Inside Chain

Weighted Chain

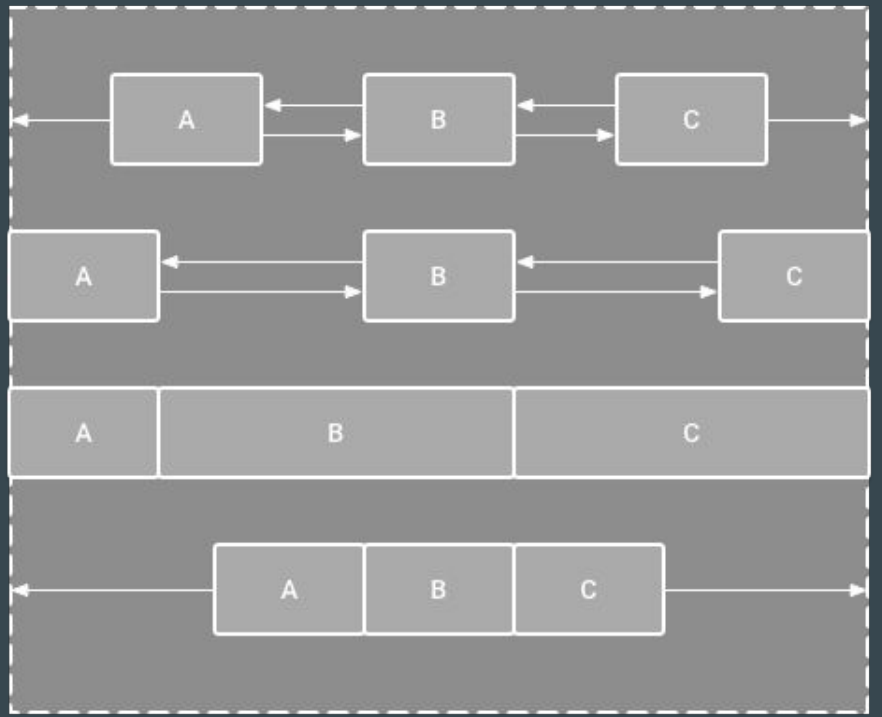
Packed Chain



Chain styles : Spread

the views are evenly distributed after margins are accounted for.

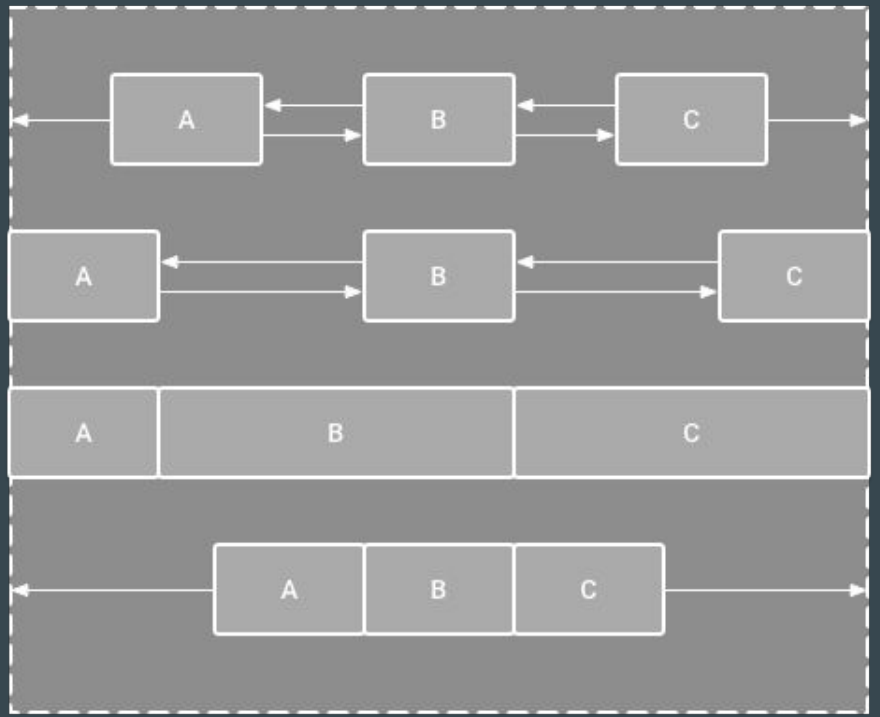
This is the default.



Chain styles : Spread Inside

the first and last views are affixed to the constraints on each end of the chain

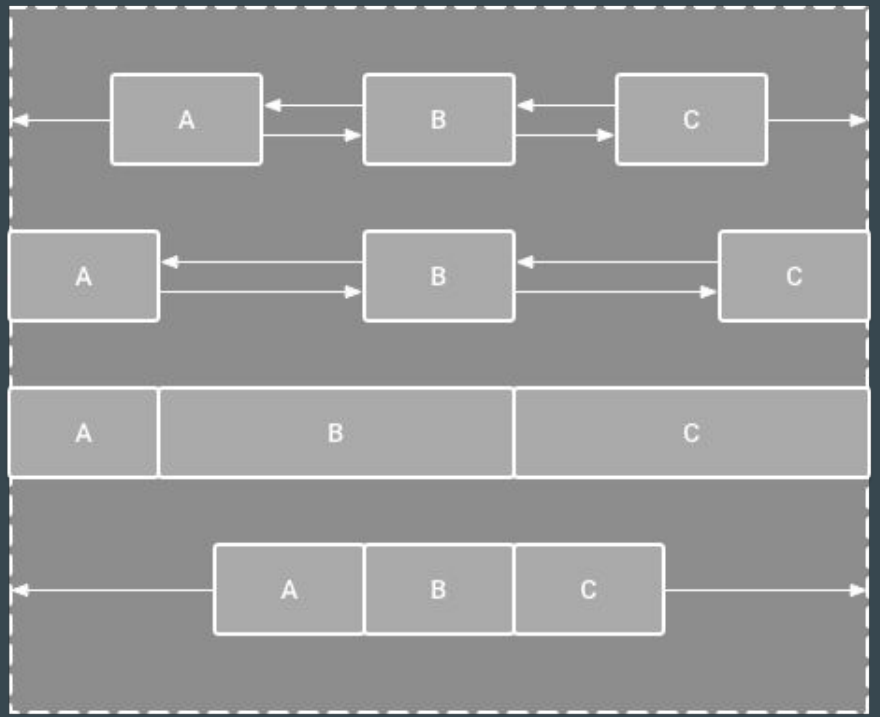
the rest are evenly distributed.



Chain styles : Weighted

Default: the space is evenly distributed between each view that's set to "match constraints,

can assign a weight of importance to each view using the layout_constraintHorizontal_weight and layout_constraintVertical_weight attributes

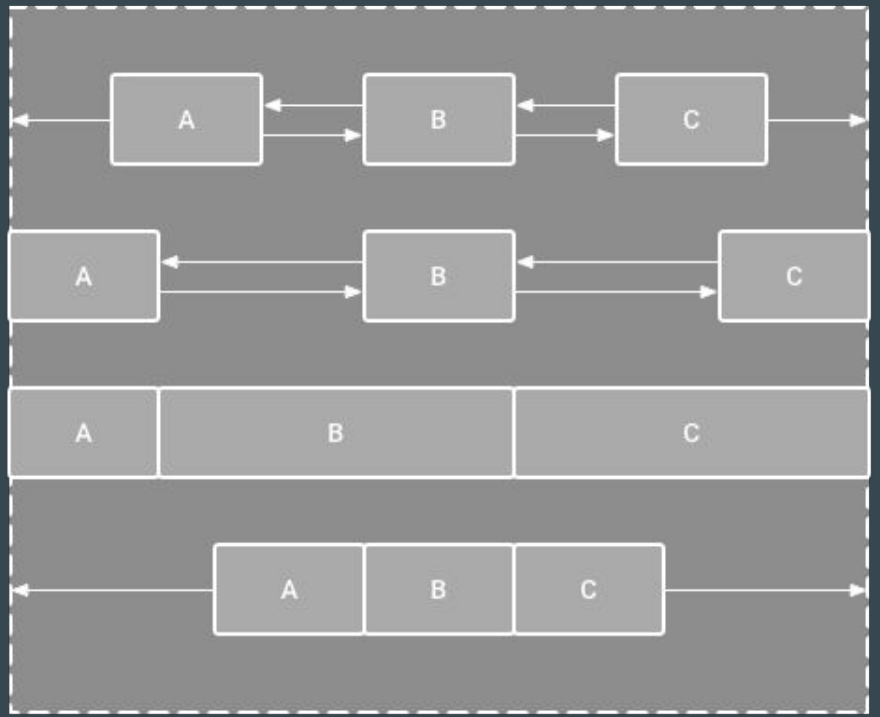


Chain styles : Packed

the views are packed together after margins are accounted for

can adjust the whole chain's bias by changing the chain's "head" view bias

- left or right
- up or down



Groups in ConstraintLayout

Control the visibility of a set of widgets

Group visibility can be toggled in code