Android Studio User Interface

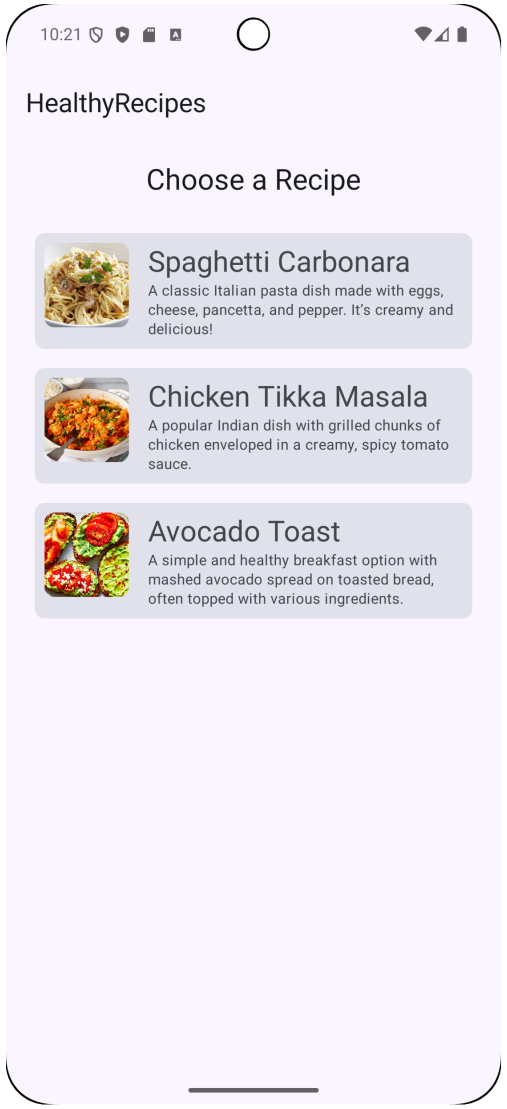
```
Course Code: ELEE1146

Course Name: Mobile Applications for Engineers

Credits: 15

Module Leader: Seb Blair BEng(H) PGCAP MIET MIHEEM FHEA
```

Download as a PDF

Designing an Android App

- Designing apps is like constructing a building
- **The Big Picture**
- Follow these steps:
  - JetPack Composeable elements
  - Create an Activity (Kotlin class) for every screen
  - Code each Kotlin class with appropriate objects and actions
  - Test the application in the emulator

Designing an Android App



10:21

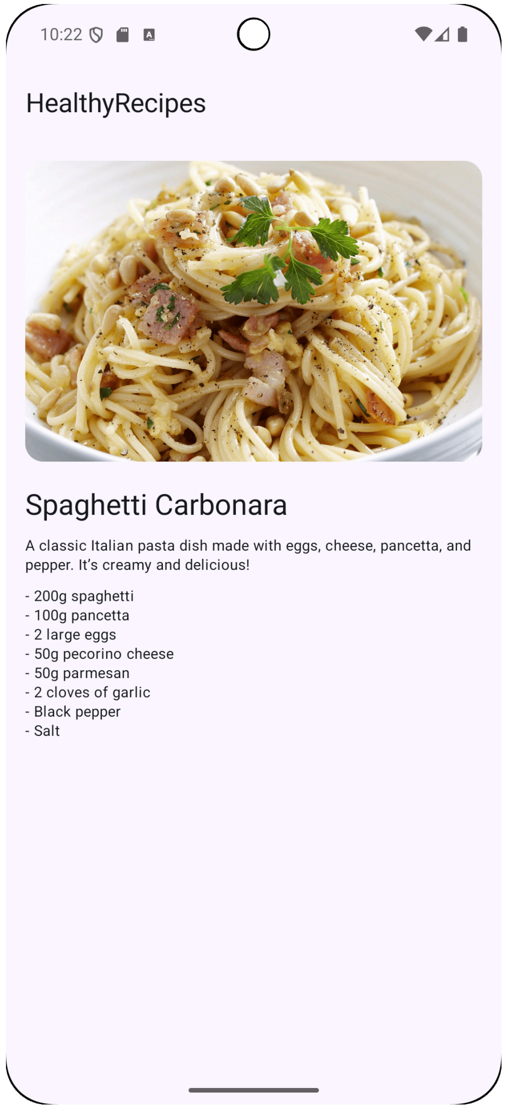
HealthyRecipes

Choose a Recipe

Spaghetti Carbonara
A classic Italian pasta dish made with eggs, cheese, pancetta, and pepper. It's creamy and delicious!

Chicken Tikka Masala
A popular Indian dish with grilled chunks of chicken enveloped in a creamy, spicy tomato sauce.

Avocado Toast
A simple and healthy breakfast option with mashed avocado spread on toasted bread, often topped with various ingredients.



10:22

HealthyRecipes

Spaghetti Carbonara

A classic Italian pasta dish made with eggs, cheese, pancetta, and pepper. It's creamy and delicious!

- 200g spaghetti
- 100g pancetta
- 2 large eggs
- 50g pecorino cheese
- 50g parmesan
- 2 cloves of garlic
- Black pepper
- Salt

Using the Android User Interface

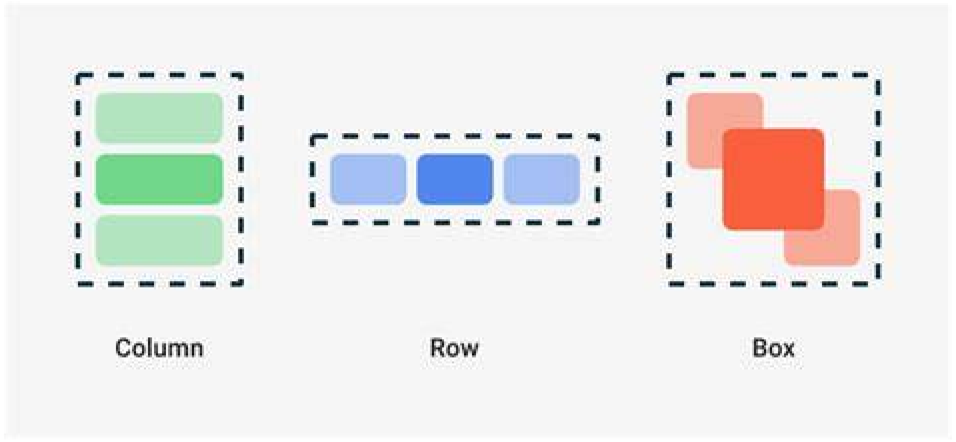
- The interface is a window on the screen of any mobile device
- Android apps run on various **form factors** such as smartphones, smart watches, tablets, televisions, etc.
  Form factor refers to the screen size, configuration, or physical layout of a device
- The layout is designed with the Composable class

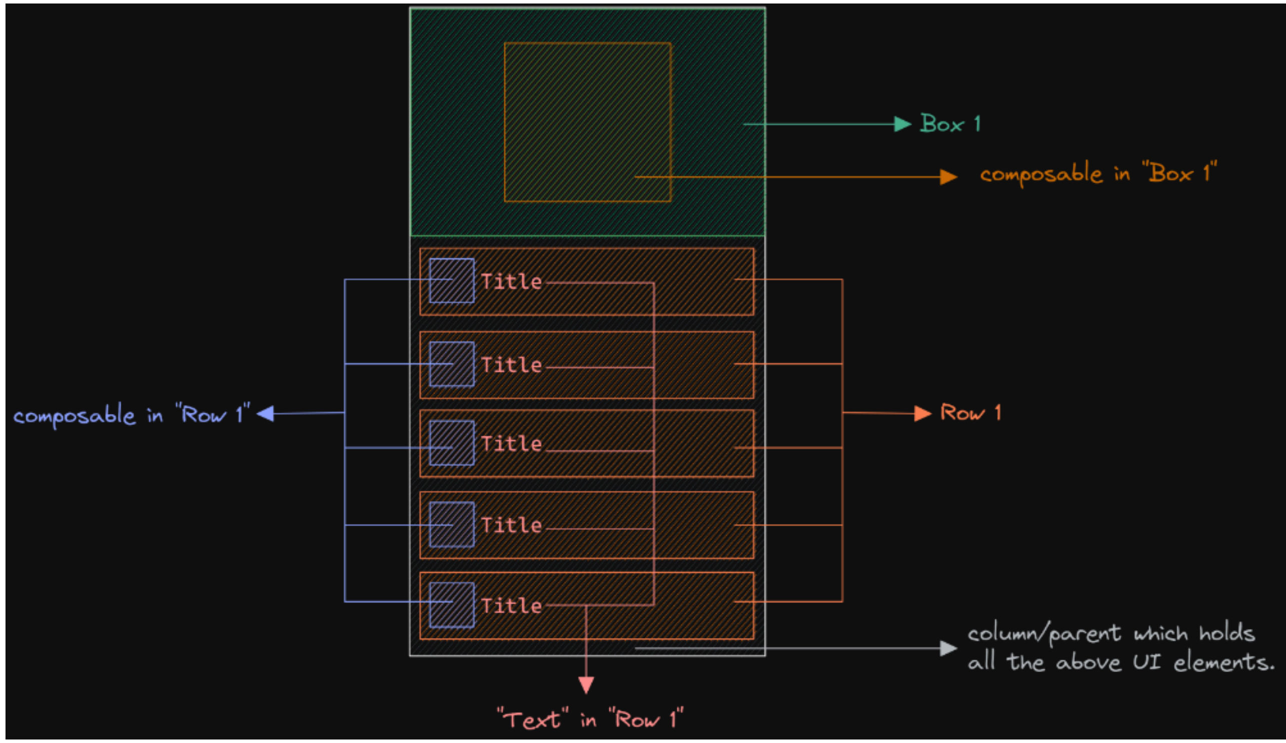
Resources: String

- `strings.xml` is a default file that is part of every Android application and contains commonly used strings for an application. It is located in `res/values` folder
- Every string is composed of a **key**, also called the **id property**, which is the name of a control, and a **default value**, which is the text associated with the control

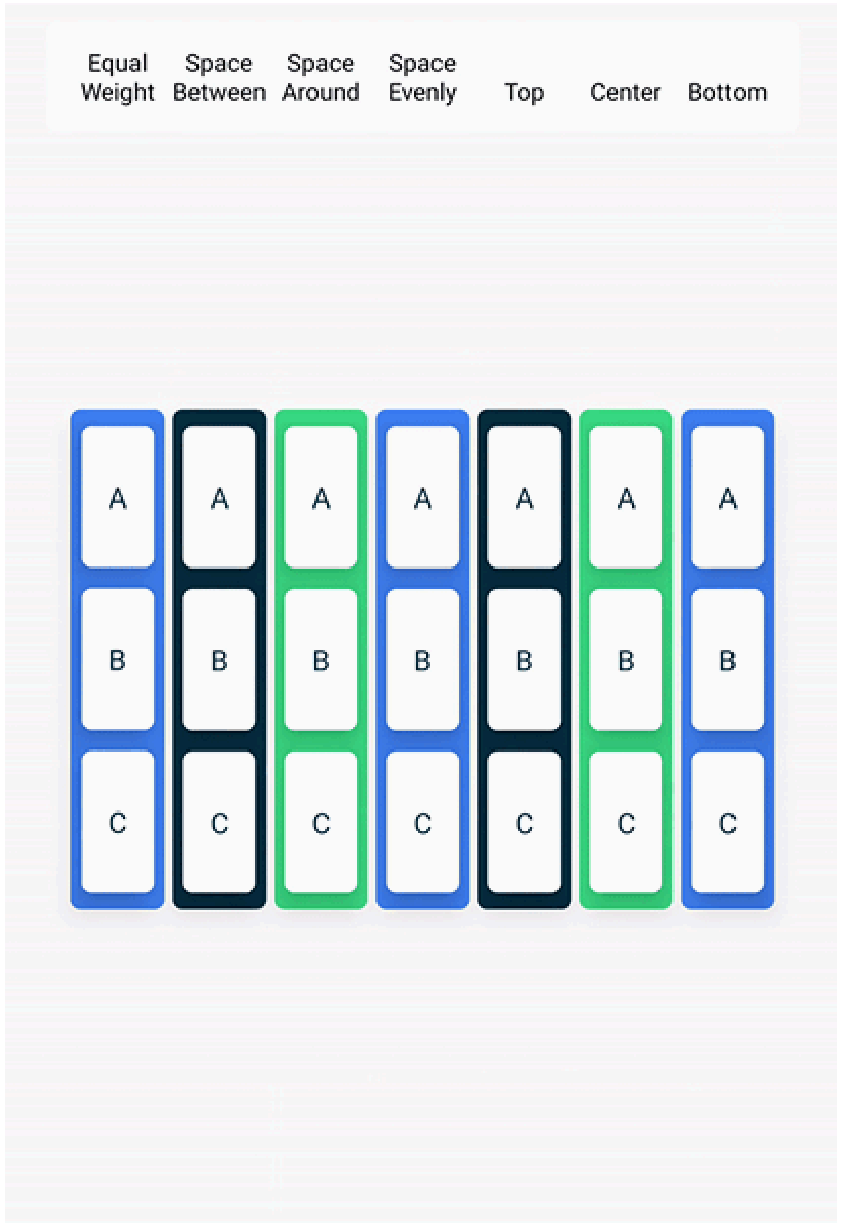
```
<resources>
    <string name='app_name'>Your apps name</string>
</resources>
```

Compose Layouts

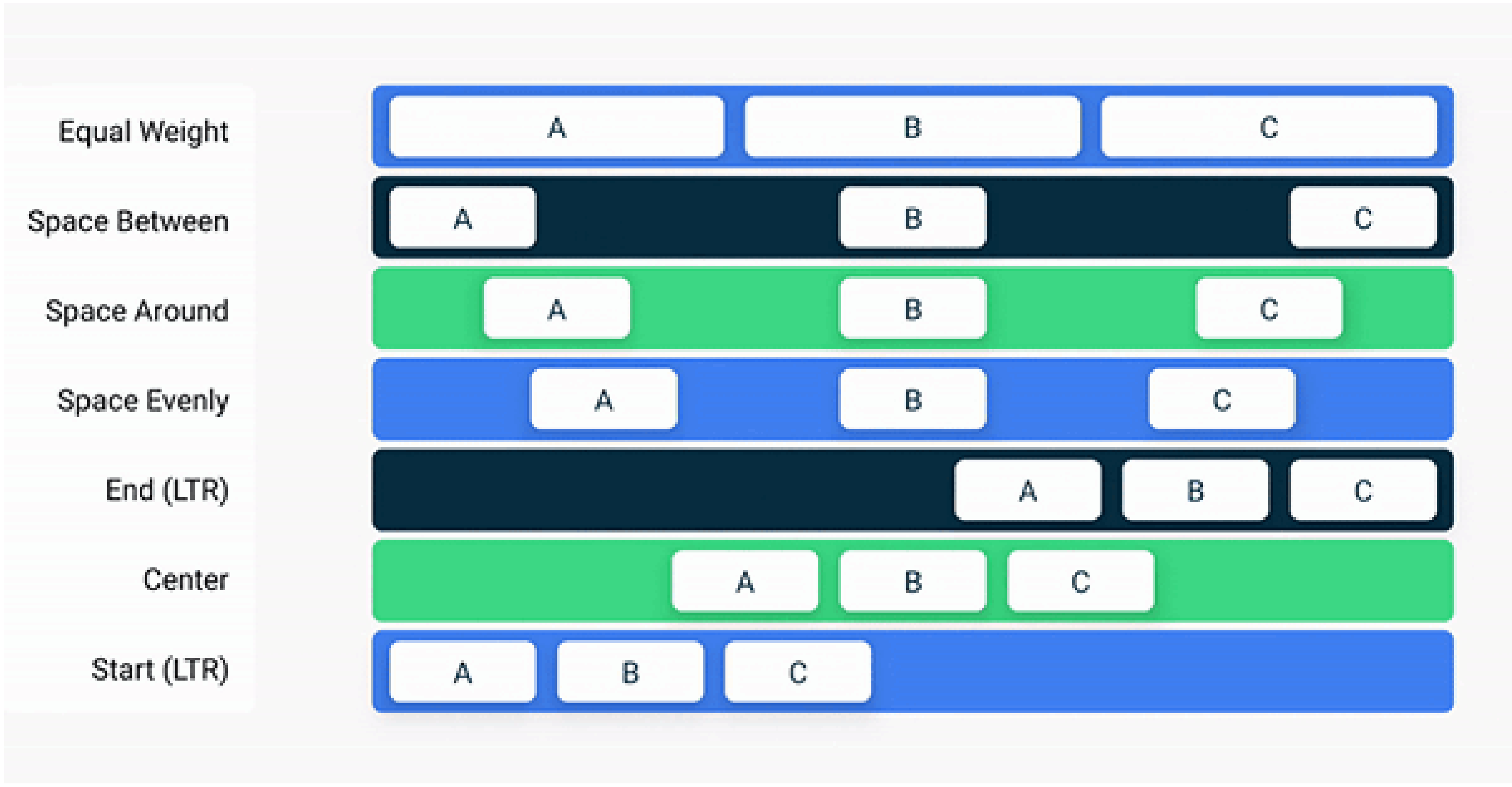




Columns


Equal Weight
Space Between
Space Around
Space Evenly
Top
Center
Bottom
A
B
C

Rows


Equal Weight
A
B
C
Space Between
A
B
C
Space Around
A
B
C
Space Evenly
A
B
C
End (LTR)
A
B
C
Center
A
B
C
Start (LTR)
A
B
C

Units of Measurement

The **text property** uses text from a string resource to display within the control.

The **textSize property** can use various units of measurement, as shown below

Unit of Measure	Abbreviation	Example
Inches	in	"0.5in"
Millimeters	mm	"20mm"
Pixels	px	"100px"
Density-independent pixels	dp or dip	"100dp" or "100dip"
Scaled-independent pixels	sp	"10sp"

The preferred unit of measurement is often `sp`, which stands for scaled-independent pixels

Adding a File to the Resources Folder

- Android creates a **Drawable** resource for any of these images when you save them in the `res/drawable` folder

Name	Description
drawable	Generic resources for any screen
mipmap - xxhdpi	Resources for extra, extra high-density screens
mipmap - xhdpi	Resources for extra high-density screens
mipmap - hdpi	Resources for high-density screens
mipmap - mdpi	Resources for extra, medium-density screens
mipmap - ldpi	Resources for low-density screens

- Android projects have multiple drawable folders that are named after the resolution of the device

Creating Activities

- An **Activity** is one of the core components of an Android application
- Each screen is considered an **activity**
  - The point at which the application makes contact with users
- Planning a program
  i. Gather and analyze the program requirements
  ii. Design the user interface
  iii. Design the program processing objects
  iv. Code the program
  v. Test the program

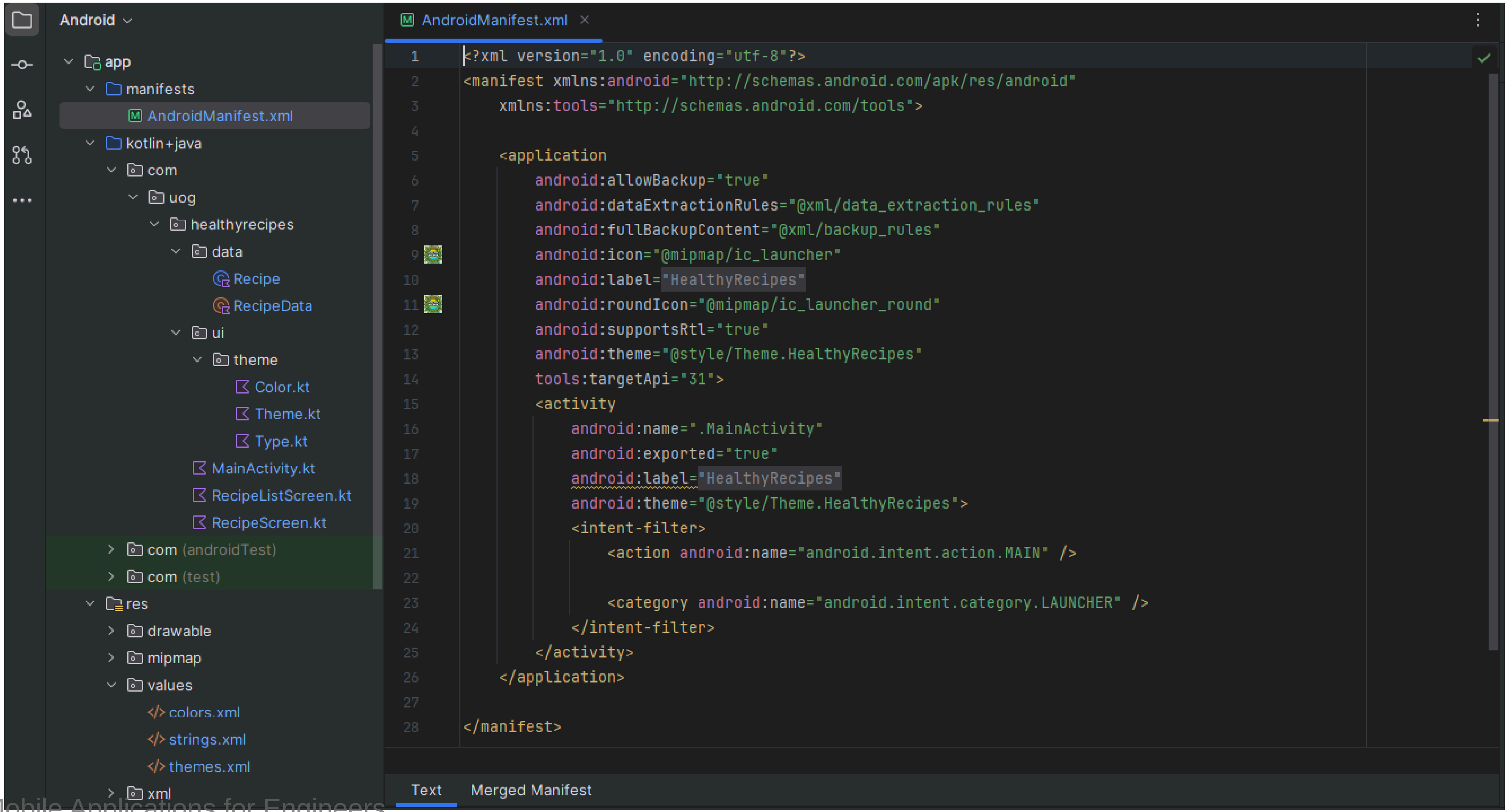
Creating Activities

- Adding a Class File
    - A **class** describes a group of objects and serves as a blueprint, or template, for creating those objects
    - An **object** is a specific, concrete instance of a class
    - When you create an object, you **instantiate** it; meaning you define one particular variation of the object

The Android Manifest File

- The **Android Manifest** file contains:
  - the name of the Java application
  - a listing of each activity
  - permissions needed to access other Android functions (like accessing the Internet)
  - the minimum level of the Android APL
- Adding an Activity to the Android Manifest
  - When applications have more than one activity, the Manifest must have an **intent** to navigate among multiple activities
  - An intent in Android is an abstract description of an operation to be performed

The Android Manifest File



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="HealthyRecipes"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.HealthyRecipes"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="HealthyRecipes"
            android:theme="@style/Theme.HealthyRecipes">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Coding the Kotlin Activity

- A **method** is a set of Kotlin statements that can be included inside a Kotlin class to perform specific tasks
- The **method body** contains a collection of statements that define what the method does.
- Coding an `onCreate` Method
- Requires corresponding `setContent` Kotlin code to display a specific screen

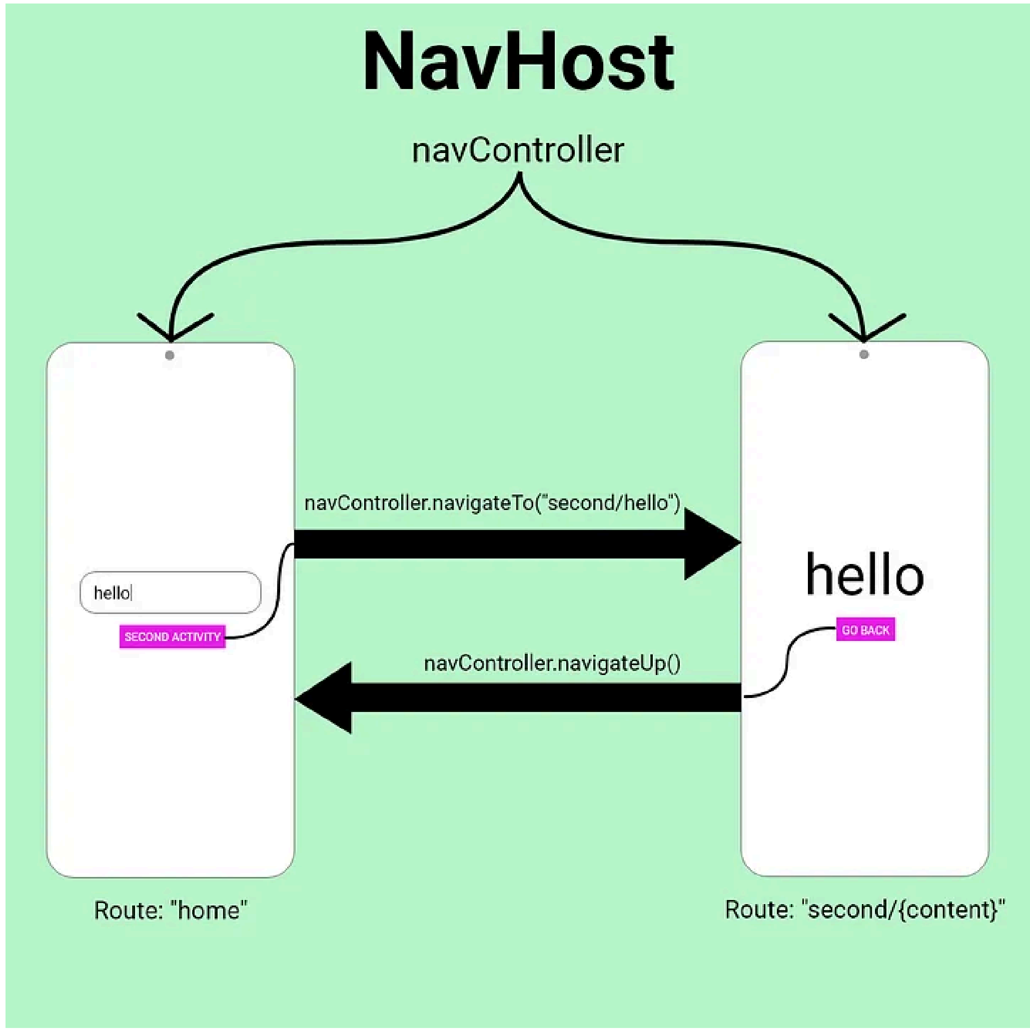
```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
          ...
        }
    }
```

Moving between composables

`NavController`: Handles the app's navigation and manages the back stack of composable screens.

`NavHost`: Hosts the navigation graph, specifying the composable destinations.

```
val navController = rememberNavController()
NavHost(
    navController = navController,
    startDestination = "home"
) {
    composable("home") { HomeScreen(navController) }
    composable("details") { DetailsScreen(navController) }
}
```


NavHost
navController
navController.navigateTo("second/hello")
hello
SECOND ACTIVITY
hello
GO BACK
navController.navigateUp()
Route: "home"
Route: "second/{content}"

@Composable and composable

- `@Composable`: This is an annotation used to mark a function as composable. It tells the compiler that this function can be used in the UI hierarchy to build the UI in a declarative way. Any function marked with @Composable can define part of the UI.

```
@Composable
fun MyScreen() {
    Text("Hello")
}
```

- `composable()`: This is a function used within a NavHost to declare a navigation destination. It defines which composable function should be rendered when a user navigates to that destination.

```
NavHost(navController, startDestination = "home") {
    composable("home") { HomeScreen() }
}
```

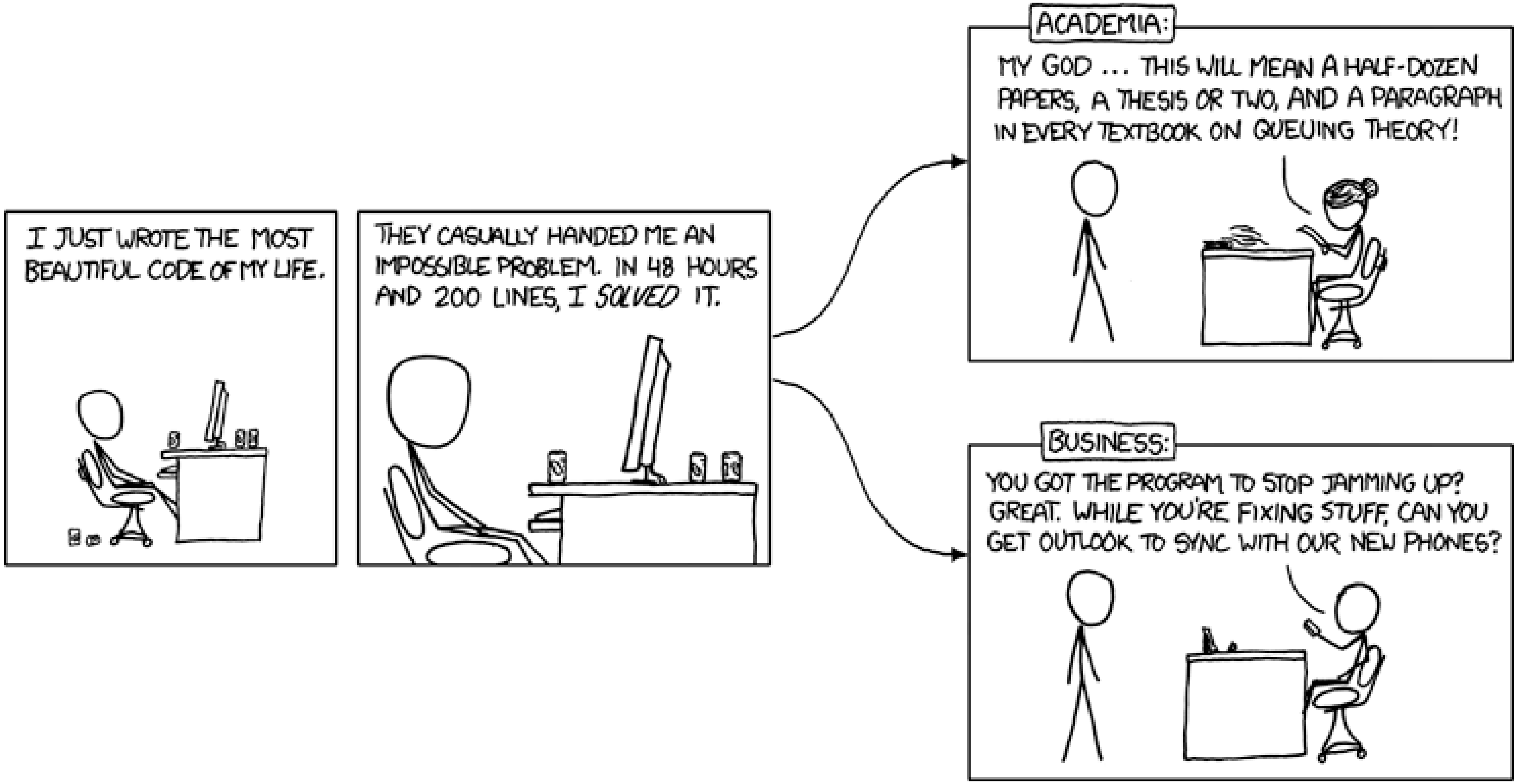
Resources

In Android Studio, resources (like `strings`, `layouts`, `drawables`, etc.) are referenced numerically because of how they are handled in the underlying Android system for efficiency and performance.

- **Efficiency/Performance**: The Android build system compiles all resources into a single class (R.java or R.kt) where each resource is assigned a unique integer identifier. Using integers for referencing resources allows for faster lookup during runtime compared to using text-based identifiers.
- **Naming Convention**: "File-based resource names must contain only lowercase a-z, 0-9, or underscore... must not start with a number" - source the Compiler

```
avocado_toast_1.png // Acceptable
1_Avocado-toast.png // Not Acceptable
```

Now do the lab!


I JUST WROTE THE MOST BEAUTIFUL CODE OF MY LIFE.
THEY CASUALLY HANDED ME AN IMPOSSIBLE PROBLEM. IN 48 HOURS AND 200 LINES, I SOLVED IT.
ACADEMIA:
MY GOD ... THIS WILL MEAN A HALF-DOZEN PAPERS, A THESIS OR TWO, AND A PARAGRAPH IN EVERY TEXTBOOK ON QUEUING THEORY!
BUSINESS:
YOU GOT THE PROGRAM TO STOP JAMMING UP? GREAT. WHILE YOU'RE FIXING STUFF, CAN YOU GET OUTLOOK TO SYNC WITH OUR NEW PHONES?