Contents

[ViewModel 3](#_Toc118739769)

[Adding Constructor Parameters to ViewModel 3](#_Toc118739770)

[AndroidViewModel 4](#_Toc118739771)

[Sharing ViewModel Between Fragments 4](#_Toc118739772)

[Saving / Reading Files from ViewModel 4](#_Toc118739773)

[Setting Parent Activity (For Up Arrows) 5](#_Toc118739774)

[EditText 5](#_Toc118739775)

[Setting Text of an Edit Text 5](#_Toc118739776)

[Removing The Underline 5](#_Toc118739777)

[TextView 5](#_Toc118739778)

[Revert To Default TextView Color 5](#_Toc118739779)

[Set TextView Margins Dynamically 5](#_Toc118739780)

[Set Text Opacity 5](#_Toc118739781)

[Set Text Color Dynamically 5](#_Toc118739782)

[ImageView 6](#_Toc118739783)

[Setting The Color of an ImageView 6](#_Toc118739784)

[Android Studio 6](#_Toc118739785)

[Issue: Can’t Read R.() Files 6](#_Toc118739786)

[Spinner (Drop Down Selection) 6](#_Toc118739787)

[Dynamically created spinner 6](#_Toc118739788)

[Statically created spinner 7](#_Toc118739789)

[Spinner With Dynamically defined List of Items 7](#_Toc118739790)

[Custom Spinner Adapter 7](#_Toc118739791)

[Set Selected Item of a Spinner 8](#_Toc118739792)

[Parcelize (Creating A Parcelable OBject) 8](#_Toc118739793)

[Maps 8](#_Toc118739794)

[Finding If a Key Exists 8](#_Toc118739795)

[For Loops 9](#_Toc118739796)

[Generate a View Id 9](#_Toc118739797)

[ActionBar 9](#_Toc118739798)

[Creating an Action Bar 9](#_Toc118739799)

[Status Bar Icons Color 10](#_Toc118739800)

[Random Number Generator 10](#_Toc118739801)

[Table Layout Formatting 10](#_Toc118739802)

[Recycler View 10](#_Toc118739803)

[Creating Recycler View 10](#_Toc118739804)

[Creating Recycler View Items (CardView) 10](#_Toc118739805)

[Set Background Color of a CardView 12](#_Toc118739806)

[Getting Context Within a RecyclerView Adapter 13](#_Toc118739807)

[Calling a Method From Parent Activity 13](#_Toc118739808)

[Working with EditTexts in RecyclerView 13](#_Toc118739809)

[Checking For Duplicate EditTexts in RecyclerView 13](#_Toc118739810)

[Accessing Each Item In a Recycler View 15](#_Toc118739811)

[Setting The Background Color of a RecyclerView Item 15](#_Toc118739812)

[Issue: Recycler View Doesn’t Show Updated List Contents 15](#_Toc118739813)

[Issue: Recycler View Not Showing Contents 16](#_Toc118739814)

[issue: Crash When Removing a “Completed Task” 16](#_Toc118739815)

[ViewPager2 and TabLayout 17](#_Toc118739816)

[Using “Add On Tab Selected Listener” 17](#_Toc118739817)

[Setting Button Clickability Dynamically 17](#_Toc118739818)

[Using TextWatcher 17](#_Toc118739819)

[Changing Edit Text Without Triggering Text Watcher 18](#_Toc118739820)

[Getting Context In Text Watcher 18](#_Toc118739821)

[Using NumberPicker 19](#_Toc118739822)

[Setting a Calendar Date (DatePicker) 19](#_Toc118739823)

[Kotlin Lists 19](#_Toc118739824)

[Check If An Element Is In a List 19](#_Toc118739825)

[Combining Lists 19](#_Toc118739826)

[Check For Duplicates In a List 19](#_Toc118739827)

[Check if All Elements in a List Are Equal 20](#_Toc118739828)

[Sum of a List of Integers 20](#_Toc118739829)

[Convert String to List 20](#_Toc118739830)

[Kotlin Strings 20](#_Toc118739831)

[Trim a String 21](#_Toc118739832)

[Insert Characters Into String 21](#_Toc118739833)

[Send Data / Passing Values 21](#_Toc118739834)

[Passing Data Using Bundle (Activity to Activity) 21](#_Toc118739835)

[Passing Data Using Bundle (Activity to Fragment) 21](#_Toc118739836)

[Passing Data Using FragmentManager (Fragment Result API) 22](#_Toc118739837)

[Homework Log App – Specific Issue: RV Does Not Refresh When Items Are Swiped 22](#_Toc118739838)

[Dialog Fragments (eg. Popup Rating System) 23](#_Toc118739839)

[Alert Dialog 23](#_Toc118739840)

[Referencing R.attr.(color) 24](#_Toc118739841)

[Json / Klaxon 24](#_Toc118739842)

[Saving a Json File Using Klaxon 24](#_Toc118739843)

[Reading a Json File Using Klaxon 24](#_Toc118739844)

[Check If a File Exists 25](#_Toc118739845)

[Adding Custom Icons (Drawables) 25](#_Toc118739846)

## ViewModel

[tutorial](https://appdevnotes.com/android-viewmodel-tutorial-for-beginners-in-kotlin/), [starter github repo](https://github.com/AnushkaMadusanka/ViewModelDemo_starter)

**build.gradle (:app)**

dependencies **{** implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.0"

**MainActivityViewModel.kt**

class MainActivityViewModel: ViewModel() {  
  
 private var count = 0  
  
 fun getUpdatedCount(): Int {  
 return ++count  
 }  
  
 fun getCurrentCount(): Int {  
 return count  
 }  
}

**MainActivity.kt**

viewModel = ViewModelProvider(this).get(MainActivityViewModel::class.*java*)

binding.countText.*text* = viewModel.getCurrentCount().toString()

*Note: first line creates an instance of MainActivityViewModel; second line calls a method of MainActivityViewModel (getCurrentCount) and sets it as the text for a textView “countText”.*

### Adding Constructor Parameters to ViewModel

[final github repo (tutorial)](https://github.com/AnushkaMadusanka/ViewModelDemo_final2)

A default ViewModel on its own can’t take constructor parameters like other classes can. To achieve this, a ViewModel Factory class is required:

**MainActivityViewModel.kt**

class MainActivityViewModel(startingCount: Int): ViewModel() {  
  
 private var count = startingCount  
  
 fun getUpdatedCount(): Int {  
 return ++count  
 }  
  
 fun getCurrentCount(): Int {  
 return count  
 }

*Note the constructor parameter “startingCount”*

**MainActivityViewModelFactory.kt**

class MainActivityViewModelFactory(private val startingCount: Int): ViewModelProvider.Factory {  
 override fun <T : ViewModel?> create(modelClass: Class<T>): T {  
 if (modelClass.isAssignableFrom(MainActivityViewModel::class.*java*)) {  
 return MainActivityViewModel(startingCount) as T  
 }   
 throw IllegalArgumentException("Unknown View Model Class")  
 }   
}

*Note: the above code is reusable for any ViewModel, just need to change the class name, ViewModel’s name, and the list of constructor parameters.*

**MainActivity.kt**

viewModelFactory = MainActivityViewModelFactory(125) // set 125 as *startingCount*  
  
viewModel = ViewModelProvider(this, viewModelFactory).get(MainActivityViewModel::class.*java*)

*Note: above code is used when instantiating viewModel. Instead of just writing “this” in the bracket (as shown in previous section when creating a standard ViewModel), I need to write “this, viewModelFactory” if I want to pass constructor parameters.*

Calling methods within the ViewModel is the same as with a standard ViewModel (shown in previous section).

### AndroidViewModel

AndroidViewModel is an application context aware ViewModel. It is used when we need to use “context” inside the ViewModel.

**MainActivityViewModel.kt**

class MainActivityViewModel(val app:Application) : AndroidViewModel(app) {

    private var count = 0

    fun getCurrentCount():Int{

        return count

    }

    fun getUpdatedCount():Int{

        return ++count

    }

}

*Note how “AndroidViewModel(app)” was extended instead of “ViewModel()”; Note the constructor parameter app: Application.*

### Sharing ViewModel Between Fragments

When sharing a ViewModel between fragments contained in the same activity, the ViewModel has to be instantiated as follows:

**Activity.kt**

viewModel = ViewModelProvider(this).get(ViewModel::class.*java*)

**Fragment.kt**

viewModel = ViewModelProvider(requireActivity()).get(ViewModel::class.*java*)

*Note the owner is “this” in the parent activity, but “requireActivity()” in the child fragment. This is to ensure the ViewModels are instantiated within the same activity – since passing “requireActivity()” in the child fragment refers to the same owner as passing “this” in the parent activity.*

### Saving / Reading Files from ViewModel

**ViewModel.kt**

class ViewModel(val app: Application): AndroidViewModel(app) {

Note that class ViewModel extends AndroidViewModel(app). AndroidViewModel takes the constructor parameter ‘app’ so it’s able to call methods that require a context, such as saving a file locally as shown below:

// save listCardColors  
val fileListCardColors = Klaxon().toJsonString(listCardColors)  
  
app.openFileOutput("fileListCardColors", Context.MODE\_PRIVATE).use **{** it.write(fileListCardColors.toByteArray())  
**}**

Note that instead of *“this.openFileOutput…”* as typically done when saving a file from an Activity, saving a file from a ViewModel requires *“app.openFileOutput…”*

val file = File(app.*filesDir*, "fileAssignment")

Similarly, when reading a file, *“app.filesDir”* is called instead of *“this.filesDir”*

## Setting Parent Activity (For Up Arrows)

**AndroidManifest.xml**

<activity  
 android:name=".ChildActivity"  
 android:exported="false"  
 android:parentActivityName=".ParentActivity" />

*Note the “parentActivityName” attribute*

## EditText

### Setting Text of an Edit Text

**Activity.kt**

editText.setText(string)

*use above instead of editText.text = string as this throws error: Expected “Editable” not “String”*

### Removing The Underline

**activity.xml**

android:background="@android:color/transparent"

*Above code is an attribute of the EditText*

## TextView

### Revert To Default TextView Color

**Activity.kt**

val defaultColor = textView.*currentTextColor*

*// change text color to something else*

textView.setTextColor(defaultColor) // revert to default color

### Set TextView Margins Dynamically

tvSubject.setPadding(30, 10, 10, 10)

*Note: this isn’t margins but I couldn’t figure out how to do margins so padding was the next best thing.*

### Set Text Opacity

**Activity.kt**

tvNotes.*alpha* = 0.65F // set opacity to 65%

### Set Text Color Dynamically

textView.setTextColor(Color.parseColor("#FF0000"))

**OR:**

val intColor = selectedColorCode.color // color in integer format   
val actualColor = ContextCompat.getColor(requireContext(), intColor) // convert into usable color format   
textview.setTextColor(actualColor)

*Note: replace “requireContext()” with “this” if above is called in an activity.*

## ImageView

### Setting The Color of an ImageView

*Note: the circle is a drawable resource.*

**In the xml file:**

app:tint="@color/teal\_700"

|  |  |
| --- | --- |
| Icon  Description automatically generated with medium confidence  Figure : Before (Default Gray) | Icon  Description automatically generated with medium confidence  Figure : After (Teal) |

**Dynamically:**

ivColor.setColorFilter(ContextCompat.getColor(*context*, R.color.YOUR\_COLOR), android.graphics.PorterDuff.Mode.*SRC\_IN*)

*ivColor is the ImageView which contains the above circle drawable.   
Note: not sure why the* [*stackoverflow*](https://stackoverflow.com/questions/20121938/how-to-set-tint-for-an-image-view-programmatically-in-android) *response included the android.graphics.PorterDuff… but it was unnecessary for mine.*

## Android Studio

### Issue: Can’t Read R.() Files

I’ve got a layout file spinner\_item but when I tried to reference R.layout.spinner\_item, “spinner\_item” is written in red and can’t be found.

**Solution:**

import android.R

Removed above line from the class which I wanted to reference spinner.item

## Spinner (Drop Down Selection)

[Tutorial](https://tutorial.eyehunts.com/android/android-spinner-with-example-in-kotlin/), [my github repo](https://github.com/emm-an-uel/spinner)

***Using a string resource as the options for the spinners*** *(below code goes in strings.xml)*

<string-array name="city\_list">  
 <item>Bangkok</item>  
 <item>London</item>  
 <item>Paris</item>  
 <item>Singapore</item>  
 <item>New York</item>  
 <item>Istanbul</item>  
 <item>Dubai</item>  
 <item>Kuala Lumpur</item>  
 <item>Hong Kong</item>  
 <item>Barcelona</item>  
</string-array>

### Dynamically created spinner

**activity\_main.xml**

<Spinner  
 android:id="@+id/spinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_margin="10dp" />

**MainActivity.kt**

// create an ArrayAdapter  
val adapter = ArrayAdapter.createFromResource(this,  
R.array.*city\_list*, android.R.layout.*simple\_spinner\_item*)  
  
// specify the layout to use when the list of choices appears  
adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*)  
  
// apply adapter to the spinner  
spinner.*adapter* = adapter

val spinnerValue = spinner.*selectedItem*.toString()

### Statically created spinner

**activity\_main.xml**

<Spinner  
 android:id="@+id/spinner2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/spinner"  
 android:layout\_margin="10dp"  
 android:entries="@array/city\_list" />

**MainActivity.kt**

val spinner2Value = spinner2.*selectedItem*.toString()

*Note: entries is pre-set to @array/city\_list, whereas dynamically created spinner has it set using the adapter.*

### Spinner With Dynamically defined List of Items

**Activity.kt**

val listSubjects = *intent*.getStringArrayListExtra("listSubjects") // listSubjects: ArrayList<String>  
  
if (listSubjects != null) {  
 val adapter = ArrayAdapter(this, android.R.layout.*simple\_spinner\_item*, listSubjects)  
 adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*)  
}

Note: spinner xml code is the same as before, but when creating the adapter, don’t call *ArrayAdapter.createFromResource* as done in the previous section. Instead, do as shown above if I want to put in a ArrayList<String> defined in the activity itself.

### Custom Spinner Adapter

[youtube tutorial](https://www.youtube.com/watch?v=sqilqsxeiwY) / [my github repo](https://github.com/emm-an-uel/color-picker) (checkout id 34fa183)

**ColorCode.kt**

class ColorCode (  
 val code: String,  
 val color: Int  
 )

**MainActivity.kt**

spinnerColor = findViewById(R.id.*spinner*)  
val adapter = SpinnerAdapter(this, colorCodeList)  
spinnerColor.*adapter* = adapter

**SpinnerAdapter.kt**

class SpinnerAdapter(context: Context, colorCodeList: ArrayList<ColorCode>)  
 : ArrayAdapter<ColorCode>(context, 0, colorCodeList) {  
  
 override fun getView(position: Int, convertView: View?, parent: ViewGroup): View { // required method   
 return myView(position, convertView, parent)  
 }  
  
 override fun getDropDownView(position: Int, convertView: View?, parent: ViewGroup): View { // required method   
 return myView(position, convertView, parent)  
 }  
  
 private fun myView(position: Int, convertView: View?, parent: ViewGroup): View {  
  
 val colorCode = getItem(position)   
 val view = convertView ?: LayoutInflater.from(*context*).inflate( // inflate the view  
 R.layout.*spinner\_item*,  
 parent,  
 false  
 )  
  
 colorCode?.*let* **{** val tvCode = view.findViewById<TextView>(R.id.*tvCode*)  
 val ivColor = view.findViewById<ImageView>(R.id.*ivColor*)  
  
 if (colorCode.code != null) {  
   
 // populate spinner\_item TextView and ImageView with corresponding text and color   
 tvCode.*text* = colorCode.code  
 ivColor.setColorFilter(ContextCompat.getColor(*context*, colorCode.color))  
 }  
 **}** return view  
 }  
}

### Set Selected Item of a Spinner

spinnerObject.setSelection(INDEX)

## Parcelize (Creating A Parcelable OBject)

Build.gradle (app):

plugins **{** id 'org.jetbrains.kotlin.android.extensions'  
**}**

Kotlin class:

@Parcelize  
class Task(  
 val id: String,  
 val subject: String,  
 val task: String,  
 val dueDate: String,  
 val dateInt: Int,  
 var status: Boolean,  
 val notes: String  
 ) : Parcelable

*Note the “@Parcelize” and return a “Parcelable”*

## Maps

### Finding If a Key Exists

map.containsKey(key)

### For Loops

for ((subjectID, colorID) in idMap) {  
 // do something  
}

*where subjectID is the key, colorID is the corresponding value.*

## Generate a View Id

etSubject.*id* = View.generateViewId()

## ActionBar

### Creating an Action Bar

[Youtube tutorial](https://www.youtube.com/watch?v=pYBsbsasZwo)

[my github repo](https://github.com/emm-an-uel/action-bar)

* ActionBar is created as a menu resource file (eg. custom\_menu.xml)
* <?xml version="1.0" encoding="utf-8"?>  
  <menu xmlns:android="http://schemas.android.com/apk/res/android"  
   xmlns:app="http://schemas.android.com/apk/res-auto">  
    
   <item  
   android:id="@+id/Search"  
   android:title="Search"  
   android:icon="@drawable/ic\_search"  
   app:showAsAction="always"/>  
    
   <item  
   android:id="@+id/Favourite"  
   android:title="Favourite"  
   app:showAsAction="never"/>  
  </menu>
* The menu resource file (xml) is inflated in MainActivity (or wherever the menu is hosted) – onCreateOptionsMenu:
* override fun onCreateOptionsMenu(menu: Menu?): Boolean {  
   *menuInflater*.inflate(R.menu.*custom\_menu*, menu)  
   return true  
  }
* onOptionsItemSelected is called when a menu item is clicked:

override fun onOptionsItemSelected(item: MenuItem): Boolean {  
 return when(item.*itemId*) {  
 R.id.*Search* -> {  
 Toast.makeText(this,"You clicked Search", Toast.*LENGTH\_LONG*).show()  
 return true  
 }  
  
 R.id.*Favourite* -> {  
 Toast.makeText(this,"You clicked Favourite", Toast.*LENGTH\_LONG*).show()  
 return true  
 }  
  
 R.id.*Share* -> {  
 Toast.makeText(this,"You clicked Share", Toast.*LENGTH\_LONG*).show()  
 return true  
 }  
  
 R.id.*whatsapp* -> {  
 Toast.makeText(this,"You clicked Whatsapp", Toast.*LENGTH\_LONG*).show()  
 return true  
 }  
  
 R.id.*instagram* -> {  
 Toast.makeText(this,"You clicked Instagram", Toast.*LENGTH\_LONG*).show()  
 return true  
 } else -> super.onOptionsItemSelected(item)  
 }  
}

*Note the placement of the “else” line is within the “return when” loop.*

### Status Bar Icons Color

[stackoverflow](https://stackoverflow.com/questions/30075827/android-statusbar-icons-color)

## Random Number Generator

val num = (0..6).*random*()

## Table Layout Formatting

**Getting TextViews to occupy the full width of a TableRow**

* TableLayout to have width = 0dp, height = wrap\_content
* TextView to be inside a TableRow (which is itself inside a TableLayout) with the following code (done dynamically):

tvDie.*layoutParams* = TableRow.LayoutParams(  
 TableRow.LayoutParams.*WRAP\_CONTENT*,  
 TableRow.LayoutParams.*WRAP\_CONTENT*,  
 1f  
)

*Note: “1f” refers to weight of the TextView*

## Recycler View

### Creating Recycler View

**activity.xml**

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".Activity">  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/recyclerView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager" />  
  
</RelativeLayout>

*Note the app:layoutManager attribute – this is needed for the recycler view to work*

### Creating Recycler View Items (CardView)

**task\_rv\_item.xml**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:layout\_margin="5dp"  
 app:cardCornerRadius="5dp"  
 app:cardElevation="4dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <TextView  
 // fill with code />

<TextView  
 // fill with code />  
  
  
 </LinearLayout>  
  
</androidx.cardview.widget.CardView>

*Above: a recycler view item with two textviews.*

**RVAdapter.kt**

class RVAdapter (  
  
private val taskList: ArrayList<Task>, // list of items to populate recycler view with   
 ): RecyclerView.Adapter<RVAdapter.NewViewHolder>() {  
   
 override fun onCreateViewHolder(  
 parent: ViewGroup,  
 viewType: Int  
 ): NewViewHolder { // inflate the layout for task\_rv\_item.xml   
 val itemView = LayoutInflater.from(parent.*context*).inflate(  
 R.layout.*task\_rv\_item*,  
 parent, false  
 )  
  
 return NewViewHolder(itemView, mListener)  
 }  
  
 class NewViewHolder(itemView: View, listener: onItemClickListener) :   
 RecyclerView.ViewHolder(itemView) { // initialize views   
 val tvSubject: TextView = itemView.findViewById(R.id.*tvSubject*)  
 val tvTask: TextView = itemView.findViewById(R.id.*tvTask*)  
 val tvDueDate: TextView = itemView.findViewById(R.id.*tvDueDate*)  
  
 init {  
 itemView.setOnClickListener() **{** listener.onItemClick(*adapterPosition*)  
 **}** }  
 }  
  
 override fun onBindViewHolder(holder: NewViewHolder, position: Int) { // populate views with data from list   
 holder.tvSubject.*text* = taskList[position].subject  
 holder.tvTask.*text* = taskList[position].task  
 holder.tvDueDate.*text* = taskList[position].dueDate  
 }  
  
 override fun getItemCount(): Int { // this function is required   
 return taskList.size  
 }  
  
 // click listener  
  
 private lateinit var mListener: onItemClickListener  
  
 interface onItemClickListener {  
 fun onItemClick(position: Int)  
 }  
  
 fun setOnItemClickListener(listener: onItemClickListener) {  
 mListener = listener  
 }  
}

*note: above adapter has a click listener to respond to user clicking on the item*

**Fragment.kt** – setting up recycler view

RVTodo = binding.rvTodo  
RVAdapter = RVAdapter(todoList)  
  
// set adapter to recycler view  
RVTodo.*adapter* = RVAdapter  
  
swipeFunctions()

**Fragment.kt** – swipe functions

private fun swipeFunctions() {  
 ItemTouchHelper(object : ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.*RIGHT*) {  
 override fun onMove(  
 recyclerView: RecyclerView,  
 viewHolder: RecyclerView.ViewHolder,  
 target: RecyclerView.ViewHolder  
 ): Boolean {  
 // this method is called  
 // when the item is moved.  
 return false  
 }  
  
 override fun onSwiped(viewHolder: RecyclerView.ViewHolder, direction: Int) {  
  
 // this method is called when item is swiped.  
 // below line is to remove item from our array list.  
 todoList.removeAt(viewHolder.*adapterPosition*)  
  
 // below line is to notify our item is removed from adapter.  
 RVAdapter.notifyItemRemoved(viewHolder.*adapterPosition*)  
 }

// at last we are adding this to recycler view   
 }).attachToRecyclerView(RVTodo)  
}

**Fragment.kt** – item click listener

RVAdapter.setOnItemClickListener(object: RVAdapter.onItemClickListener {  
 override fun onItemClick(position: Int) {  
  
 val selectedTask = todoList[position]  
 // do something with selectedTask  
})

### Set Background Color of a CardView

**RVAdapter.kt** (onBindViewHolder)

holder.cardView.setCardBackgroundColor(ContextCompat.getColor(context, color))

### Getting Context Within a RecyclerView Adapter

[stackoverflow](https://stackoverflow.com/questions/32136973/how-to-get-a-context-in-a-recycler-view-adapter)

tvSubject.*context*

*Above: call for context of a view within the recycler view item*

### Calling a Method From Parent Activity

**Adapter.kt**

(context as MainActivity).updateFruit()

Where “updateFruit” is the name of a public method (ie public fun updateFruit() in Activity.kt)

### Working with EditTexts in RecyclerView

Typically, RecyclerViews are used to display static items – ie they remain unchanged as long as the RV is being displayed. When working with RVs with EditTexts in their items though, the following have to be implemented to prevent the views from being “recycled” – ie keeps the EditText content and position from getting jumbled up due to “recycling”.

**RVAdapter.kt**

override fun onBindViewHolder(holder: NewViewHolder, position: Int) {  
  
 holder.setIsRecyclable(false) // prevent "recycling" the views - keeps EditText content/position from being jumbled up  
}

override fun getItemViewType(position: Int): Int {  
 return position  
}  
  
override fun getItemId(position: Int): Long {  
 return position.toLong()  
}

### Checking For Duplicate EditTexts in RecyclerView

[my github repo](https://github.com/emm-an-uel/identify-duplicates)

**The objective:** I’ve got a RecyclerView populated with items, each with an EditText. Users can type into any of these items’ EditTexts. I want the EditText text color to change to red if there’s another EditText with the same text in it. I also want the existing EditText (ie the duplicate which isn’t currently being edited) to change to red too.

**TLDR:** Use textwatcher to detect changes to any item’s edittext. After detecting a text change, run a method in parent activity which accesses each recyclerview item in the parent activity and check if it’s a duplicate. If it is, change color to red.

**Full solution:**

1. Have listFruits defined in Activity.kt **(Activity.kt)**
2. Use a TextWatcher and update listFruits[position] when an EditText is changed (in **Adapter.kt**, call a method in **Activity.kt**, as shown in previous section)
3. Call the method checkDuplicates(), belonging to Activity.kt, to handle color changes (**Adapter.kt**)

override fun afterTextChanged(p0: Editable?) {  
 val input = p0.*toString*().*trim*()  
  
 (context as MainActivity).updateFruit(input, position) // update old fruit to new fruit  
  
 (context as MainActivity).checkDuplicates() // checkDuplicates() is run in MainActivity so it can iterate through all items in rvList  
}

1. The method checkDuplicates(), in Actvity.kt, is shown below: (**Activity.kt**)

fun checkDuplicates() { // checkDuplicates() is run in MainActivity so it can iterate through all items in rvList  
  
 val itemCount = rvListFruits.*adapter*!!.*itemCount* for (i in 0 *until* itemCount) {  
 val holder = rvListFruits.findViewHolderForAdapterPosition(i)  
 if (holder != null) {  
 val etInput = holder.itemView.findViewById<EditText>(R.id.*etInput*)  
 val fruit = etInput.*text*.toString().*trim*()  
  
 val count = listFruits.*count* **{ it** == fruit **}** if (count > 1) {  
 etInput.setTextColor(ContextCompat.getColor(this, R.color.*red*))  
 } else {  
 etInput.setTextColor(ContextCompat.getColor(this, R.color.*white*))  
 }  
 }  
 }  
}

*Code commentary: The above code iterates through all items in the RecyclerView and counts the number of times its corresponding “fruit” appears in the listFruits. If it appears more than once (ie there is a duplicate), it changes the EditText color to red. Otherwise, it changes the EditText color to white.*

The above method will only be called as the EditText is being changed. Thus, a similar checkDuplicates() method has to be called upon initialization / updates of the RecyclerView, shown below: (written in **Adapter.kt**)

private fun checkDuplicatesOnStart(input: String, etInput: EditText) {  
  
 val context = etInput.*context* val count = listFruits.*count* **{ it** == input **}** if (count > 1) {  
 etInput.setTextColor(ContextCompat.getColor(context, R.color.*red*))  
 } else {  
 etInput.setTextColor(ContextCompat.getColor(context, R.color.*white*))  
 }  
}

*Note: The above method is called within the override method onBindViewHolder, as shown below:*

override fun onBindViewHolder(holder: NewViewHolder, position: Int) {  
  
 holder.setIsRecyclable(false) // prevent "recycling" the views - keeps EditText content/position from being jumbled up  
  
 val etInput = holder.etInput  
 val fruit = listFruits[position]  
  
 etInput.setText(fruit)  
 etInput.addTextChangedListener(textWatcher(etInput, position))  
  
 checkDuplicatesOnStart(fruit, etInput)  
}

*Note: TextWatcher is also added in the onBindViewHolder method.*

### Accessing Each Item In a Recycler View

[tutorial from github](https://gist.github.com/dominicthomas/1a0d6d7c81eb69e5ad56a62cb7bfd11d)

**Activity.kt**

val newListSubjectColor = *arrayListOf*<SubjectColor>()  
  
val itemCount = rvSettings.*adapter*!!.*itemCount*for (i in 0 *until* itemCount) { // add all subjectColor to newListSubjectColor  
 val holder = rvSettings.findViewHolderForAdapterPosition(i)  
 if (holder != null) {  
 val etSubject = holder.itemView.findViewById<EditText>(R.id.*etSubject*)  
 val subject = etSubject.*text*.toString()  
  
 val spinnerColor = holder.itemView.findViewById<Spinner>(R.id.*spinnerColor*)  
 val colorIndex = spinnerColor.*selectedItemPosition* val subjectColor = SubjectColor(subject, colorIndex)  
 newListSubjectColor.add(subjectColor)  
 }  
}

*etSubject and spinnerColor are views within the recycler view item*

### Setting The Background Color of a RecyclerView Item

**RVAdapter.kt**

holder.cardView.setCardBackgroundColor(backgroundColor)

### Issue: Recycler View Doesn’t Show Updated List Contents

I’ve got a button that adds a new item to listSubjectColor when clicked. If I try updating the contents of listSubjectColor (to reflect the contents shown in the EditTexts) then add a new “blank subject” to the list, the recycler view doesn’t show these changes, and instead reverts back to showing the original list.

**Solution:** after updating listSubjectColor, instead of calling notifyDataSetChanged(), I initialized the recycler view from scratch. This way, I’m able to pass the updated listSubjectColor into the adapter. *Not sure if this is the best way though.*

**Activity.kt**

private fun setupRecyclerView() {  
 rvSettings = findViewById(R.id.*rvSettings*)  
 rvAdapter = SettingsRVAdapter(listSubjectColor)  
 rvSettings.*adapter* = rvAdapter  
}  
  
private fun addSubjectColor() {  
  
 updateList()  
  
 // add new item in recycler view  
 val newSubjectColor = SubjectColor("", R.color.*blue*) // adds an empty subject string  
 listSubjectColor.add(newSubjectColor)  
  
 setupRecyclerView()  
}  
  
private fun updateList() {  
  
 val newListSubjectColor = *arrayListOf*<SubjectColor>()  
  
 val itemCount = rvSettings.*adapter*!!.*itemCount* for (i in 0 *until* itemCount) { // add all subjectColor to newListSubjectColor  
 val holder = rvSettings.findViewHolderForAdapterPosition(i)  
 if (holder != null) {  
 val etSubject = holder.itemView.findViewById<EditText>(R.id.*etSubject*)  
 val subject = etSubject.*text*.toString()  
  
 val spinnerColor = holder.itemView.findViewById<Spinner>(R.id.*spinnerColor*)  
 val spinnerColorIndex = spinnerColor.*selectedItemPosition* val color = listColors[spinnerColorIndex]  
  
 val subjectColor = SubjectColor(subject, color)  
 newListSubjectColor.add(subjectColor)  
 }  
 }  
  
 listSubjectColor = newListSubjectColor  
}

**Alternative solution:** *the following method clears old list and adds all elements of updatedList before notifying data set changed. This gets the same result without having to re-initalize the entire recycler view.* [*stackoverflow*](https://stackoverflow.com/questions/52294428/how-to-update-recyclerview-adapter-data-while-adding-new-item-in-list)

**Activity.kt**

list.clear()

list.addAll(yourUpdatedList)

adapter.notifyDataSetChanged()

### Issue: Recycler View Not Showing Contents

**Issue:**

*Setting up RecyclerView and RVAdapter*

RVTodo = binding.rvTodo  
todoList = ArrayList()  
RVAdapter = RVAdapter(todoList)  
  
// set adapter to recycler view  
RVTodo.*adapter* = RVAdapter

*Initializing todoList, which is passed into RVAdapter*

*setFragmentResultListener*("rqTodoList") **{** requestKey, bundle **->** todoList = bundle.getParcelableArrayList("todoList")!!  
**}**

When the fragment is launched, RVAdapter is not called, and thus does not display the items in todoList.

**Solution:**

todoList is initialized only after the setFragmentResultListener gets a result. So until then, todoList is empty, and an empty todoList is passed to RVAdapter, so it does nothing. createRV() should be called only after todoList has been populated (shown below).

*setFragmentResultListener*("rqTodoList") **{** requestKey, bundle **->** todoList = bundle.getParcelableArrayList("todoList")!!  
 createRV()  
**}**

### issue: Crash When Removing a “Completed Task”

**The issue:** completedTask is deleted from todoList, app crashes when running line “todoList.removeAt (viewHolder.adapterPosition)”

**Solution:** if completedTask is deleted before “todoList.removeAt(viewHolder.adapterPosition)”, the item to be removed in todoList is null, so app crashes. completedTask should be deleted after the todoList.remove… line, as shown below.

override fun onSwiped(viewHolder: RecyclerView.ViewHolder, direction: Int) {  
 // change task status  
 val completedTask: Task = todoList[viewHolder.*adapterPosition*]  
 // *todo: implement completed task functionality* // this method is called when item is swiped.  
 // below line is to remove item from our array list.  
 todoList.removeAt(viewHolder.*adapterPosition*)  
  
 // below line is to notify our item is removed from adapter.  
 RVAdapter.notifyItemRemoved(viewHolder.*adapterPosition*)  
  
 taskCompleted(completedTask)  
}

## ViewPager2 and TabLayout

[Tutorial](https://medium.com/busoft/how-to-use-viewpager2-with-tablayout-in-android-eaf5b810ef7c)

### Using “Add On Tab Selected Listener”

**Example code:** (Using addOnTabSelectedListener to change fab visibility)

[Stack Overflow link](https://stackoverflow.com/questions/37235125/how-to-get-tab-click-event-in-activity-on-tablayout-android)

tabLayout.addOnTabSelectedListener(object : TabLayout.OnTabSelectedListener {  
 override fun onTabSelected(tab: TabLayout.Tab?) {  
 val position = tab?.*position* if (position == 0) {  
 fabTask.*visibility* = View.*VISIBLE* } else {  
 fabTask.*visibility* = View.*INVISIBLE* }  
 }  
  
 override fun onTabUnselected(tab: TabLayout.Tab?) {  
 }  
  
 override fun onTabReselected(tab: TabLayout.Tab?) {  
 }  
})

## Setting Button Clickability Dynamically

|  |  |
| --- | --- |
| **Clickable**  button.isEnabled = true | **Unclickable**  button.isEnabled = false |

**Example code:** (including setting button opacity, where alpha ranges from 0 to 255)

private fun btnDisabled() {  
 btnConfirm.*isEnabled* = false  
 btnConfirm.*background*.*alpha* = 45  
}  
  
private fun btnEnabled() {  
 btnConfirm.*isEnabled* = true  
 btnConfirm.*background*.*alpha* = 255  
}

## Using TextWatcher

<https://www.tutorialspoint.com/how-to-use-the-textwatcher-class-in-kotlin>

**Activity.kt**

val input: EditText = findViewById(R.id.etInput)

val output: TextView = findViewById(R.id.textView)

input.addTextChangedListener(textWatcher)

private val textWatcher = object : TextWatcher {

      override fun afterTextChanged(s: Editable?) {

    }

    override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {

    }

    override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {

      output.text = s

      if (start == 12) {

          Toast.makeText(applicationContext, "Maximum Limit Reached", Toast.LENGTH\_SHORT)

          .show()

      }

    }

}

### Changing Edit Text Without Triggering Text Watcher

[stackoverflow](https://stackoverflow.com/questions/9385081/how-can-i-change-the-edittext-text-without-triggering-the-text-watcher)

### Getting Context In Text Watcher

[My github repo](https://github.com/emm-an-uel/homework-log) – commit e3497bd

**SettingsRVAdapter.kt (main section)**

holder.etSubject.addTextChangedListener(  
 textWatcher(  
 holder.etSubject,  
 listSubject  
 )  
) // to watch for duplicate subject entries

*adds textWatcher (defined below) and passes parameters etSubject – EditText, and listSubject – ArrayList<String>*

**SettingsRVAdapter.kt (textWatcher section)**

class textWatcher(val view: TextView, val listSubject: ArrayList<String>) : TextWatcher {  
   
 val context = view.*context* override fun afterTextChanged(p0: Editable?) {  
 }  
  
 override fun beforeTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {  
 }  
  
 override fun onTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {  
 if (listSubject.contains(p0.*toString*())) {  
  
 Toast.makeText(context, "Duplicate", Toast.*LENGTH\_SHORT*).show()  
  
 // *TODO: prevent duplicate subjects* }  
 }  
}

*gets etSubject’s context by calling view.context*

## Using NumberPicker

[YouTube tutorial](https://www.youtube.com/watch?v=kSDMe9wnx9s)

**My code:** *(Setting min and max values)*

numberPicker.*minValue* = 1  
numberPicker.*maxValue* = 6

*(Getting value from numberPicker)*

val numDice = numberPicker.*value*

## Setting a Calendar Date (DatePicker)

// datePicker stuff  
val dateList = currentTask.dueDate.*split*(" ").*toList*()  
  
val year = dateList[2].*toInt*()  
val month = dateList[1].*toInt*() - 1  
val day = dateList[0].*toInt*()  
  
today = Calendar.getInstance()  
today.set(year, month, day) // convert to dueDate if there's a task being edited

## Kotlin Lists

### Check If An Element Is In a List

if (list.contains(element)) {

print("Element: $element is present in the list: $list.")

} else {

print("Element: $element is not present in the list: $list.")

}

### Combining Lists

// combine todoList and doneList  
val allList: ArrayList<Task> = ArrayList()  
allList.addAll(todoList)  
allList.addAll(doneList)

Use: *combinedList*.addAll (*List)* to combine multiple lists.

### Check For Duplicates In a List

##### if it doesn’t matter which elements are duplicates

[tutorial](https://www.techiedelight.com/check-duplicates-array-kotlin/)

**Activity.kt**

private fun noDuplicates(): Boolean {  
 val listSubjectDistinct = listSubject.*distinct*() // returns a list of distinct elements (ie duplicates removed)   
   
 if (listSubjectDistinct.size == listSubject.size) { // checks if size of original list = size of distinct list   
 return true  
 } else {  
 return false  
 }  
}

##### if it matters which elements are duplicates

**Activity.kt**

val count = myList.count **{** it == myString **}** // "count" returns the number of times myString appears in myList

if (count > 1) {  
 // do something if duplicates exist   
} else {  
 // do something if duplicates don't exist   
}

*Where myList is an arrayList<String> and myString is a String*

### Check if All Elements in a List Are Equal

// check if all dice have same number  
var isEqual = true   
for (num in diceList) {  
 if (num != diceList[0]) {  
 isEqual = false  
 }  
}

### Sum of a List of Integers

<https://www.techiedelight.com/calculate-sum-of-all-items-in-list-of-integers-in-kotlin/#:~:text=Using%20sum()%20function,%2C%20Double%20%2C%20Byte%20%2C%20Short%20.&text=Note%20that%20as%20of%20Kotlin%201.5%2C%20sumBy()%20function%20is%20deprecated>.

val diceSum = diceList.*sum*()

### Convert String to List

Text

Description automatically generated with medium confidence

## Kotlin Strings

### Trim a String

val subject = etSubject.*text*.toString().*trim*()  
val task = etTask.*text*.toString().*trim*()

### Insert Characters Into String

val dateString = currentTask.dateInt.toString()  
// insert "-" between year, month, day values (to allow parse to work)  
var dueDateYYYYMMDD = StringBuilder(dateString).insert(4, "-")  
dueDateYYYYMMDD = StringBuilder(dueDateYYYYMMDD).insert(7, "-")

## Send Data / Passing Values

<https://stackoverflow.com/questions/45157567/how-to-pass-the-values-from-activity-to-another-activity>

### Passing Data Using Bundle (Activity to Activity)

[My github repo](https://github.com/emm-an-uel/pass-data-activity-fragment)

**Example code:** (From activity to activity)

*First activity:*

val bundle = Bundle()  
bundle.putString("id", etId.*text*.toString())  
bundle.putString("name", etName.*text*.toString())  
bundle.putString("roll", etRoll.*text*.toString())  
  
val intent = Intent(this, SecondActivity::class.*java*)  
intent.putExtras(bundle)  
startActivity(intent)

*Second activity:*

val bundle = *intent*.*extras*if (bundle != null) {  
 tvId.*text* = "id = ${bundle.getString("id")}"  
 tvName.*text* = "Name = ${bundle.getString("name")}"  
 tvRoll.*text* = "RollNo = ${bundle.getString("roll")}"  
}

Using bundle.getString(“*key”*) to get the data passed from first activity.

### Passing Data Using Bundle (Activity to Fragment)

**Example code:** (From activity to fragment)

***Activity:***

val fragment = FirstFragment()  
val bundle = Bundle()  
bundle.putString("string", sendText.*text*.toString())  
fragment.*arguments* = bundle  
  
*supportFragmentManager*.beginTransaction().replace(R.id.*frameLayout*, fragment).commit()

***Fragment:***

val data = *arguments*tvText.*text* = data!!.get("string").*toString*()

*Note: if passing a Parcelable Object, do getParcelable<Class Name>(key) [shown below]*

val selectedColorCode = data!!.getParcelable<ColorCode>("bundleColorCode")

### Passing Data Using FragmentManager (Fragment Result API)

[Android Developers Documentation](https://developer.android.com/guide/fragments/communicate#fragment-result)

[my github repo](https://github.com/emm-an-uel/pass-data-activity-fragment) – “completed: pass Person via FragmentManager”

**Receiving fragment**

*setFragmentResultListener*("requestKey") **{** requestKey, bundle **->** result = bundle.getString("bundleKey")!!  
 val tvName = view.findViewById<TextView>(R.id.*tvName*)  
 tvName.*text* = result  
**}**

*Note: I initially placed setFragmentResultListener in onCreate, and val tvName etc. in onViewCreated. This resulted in a crash since it tried running tvName.text = result before setFragmentResultListener got a value for result.*

**Sending fragment**

val result = "result"  
*setFragmentResult*("requestKey", *bundleOf*("bundleKey" *to* result))

**Sending activity**

*supportFragmentManager*.setFragmentResult("requestPerson", *bundleOf*("bundlePerson" *to* person))

*Note: above is from a different project; sending a bundle with Person, not String (as in fragment example).*

*Note: when setFragmentResult is called from activity, supportFragmentManager needs to be called first (as shown above).*

**Receiving activity**

supportFragmentManager  
                .setFragmentResultListener("requestKey", this) { requestKey, bundle ->  
            // We use a String here, but any type that can be put in a Bundle is supported  
            val result = bundle.getString("bundleKey")  
            // Do something with the result  
        }

*Note: when setFragmentResultListener is called from activity, “LifecycleOwner” is set as “this” (as shown above).*

### Homework Log App – Specific Issue: RV Does Not Refresh When Items Are Swiped

**Issue:** createRV() is called only after the second setFragmentResultListener(), which doesn’t receive any data upon swipes.

**Solution:**

**FragmentTodo.kt**

private fun getFromBundle() {  
 *setFragmentResultListener*("rqTodoList") **{** requestKey, bundle **->** todoList = bundle.getParcelableArrayList("todoList")!!  
  
 if (mapSubjectColor.size > 0) { // if mapSubjectColor already exists (ie not the first time loading up this fragment)   
 createRV() // createRV() is called here to reflect changes when user swipes   
   
 } else { // if it is the first time loading up this fragment   
 *setFragmentResultListener*("rqMapSubjectColorTodo") **{** requestKey, bundle **->** mapSubjectColor = bundle.getSerializable("mapSubjectColor")!! as HashMap<String, Int>  
 createRV() // createRV() is called here only after getting mapSubjectColor (if it's the first time loading up this fragment)   
 **}** }  
 **}**

*Note: mapSubjectColor is initialized in onCreateView() as an empty HashMap.*

## Dialog Fragments (eg. Popup Rating System)

[Youtube tutorial](https://www.youtube.com/watch?v=SkFcDWt9GV4)

[My github repo](https://github.com/emm-an-uel/dialog-fragment)

|  |  |
| --- | --- |
|  | Note: for buttons to have bottom margins as shown below, buttons need to be constrained to the bottom of parent, then include marginBottom. |

**Activity.kt**

val dialog = CustomDialogFragment()  
  
 dialog.show(*supportFragmentManager*,"customDialog")

**ColorDialogFragment.kt**

import androidx.fragment.app.DialogFragment

class ColorDialogFragment : DialogFragment()

*Note: “DialogFragment” class*

### Alert Dialog

**Fragment.kt**

private fun confirmDelete(deletedTask: Task, position: Int) {  
 var touched = false  
  
 // alert dialog  
 val alertDialog: AlertDialog = requireContext().*let* **{** val builder = AlertDialog.Builder(**it**)  
 builder.*apply* **{** setPositiveButton("Confirm"  
 ) **{** dialog, id **->** deleteTask(deletedTask)  
 touched = true  
 **}** setNegativeButton("Cancel"  
 ) **{** dialog, id **->** doneList.add(position, deletedTask)  
 RVAdapter.notifyItemInserted(position)  
 touched = true  
 **}  
 }** builder.setMessage("Clear this task?")  
  
 builder.create()  
 **}** alertDialog.show()  
 val actualColorAccent = getColor(requireContext(), R.attr.*colorAccent*)  
  
 alertDialog.getButton(AlertDialog.*BUTTON\_POSITIVE*).setTextColor(actualColorAccent)  
 alertDialog.getButton(AlertDialog.*BUTTON\_NEGATIVE*).setTextColor(actualColorAccent)  
  
 alertDialog.setOnDismissListener **{** if (!touched) { // if touched == false (ie user touched outside dialog box)  
 doneList.add(position, deletedTask)  
 RVAdapter.notifyItemInserted(position)  
 }  
 **}**}

*Above: AlertDialog with buttons “Confirm” and “Cancel”, and if user clicks outside the dialog, program runs the same thing as if user had clicked “Cancel”.*

*Note: Above alertDialog also includes setting text color of “Confirm” and “Cancel” with a method “getColor”*

## Referencing R.attr.(color)

private fun getColor(context: Context, colorResId: Int): Int {  
  
 val typedValue = TypedValue()  
 val typedArray = context.obtainStyledAttributes(typedValue.data, *intArrayOf*(colorResId))  
 val color = typedArray.getColor(0, 0)  
 typedArray.recycle()  
 return color  
}

**Implementation**

val actualColorAccent = getColor(requireContext(), R.attr.*colorAccent*)

## Json / Klaxon

### Saving a Json File Using Klaxon

**Example code:** (creating a new file “fileAssignment”)

} else { // if "fileAssignment" does not exist  
  
 // new val listAssignment, add newAssignment and serialize listAssignments  
 val listAssignment = *mutableListOf*(newAssignment)  
 val updatedFile = Klaxon().toJsonString(listAssignment)  
  
 // store in local file  
 this.openFileOutput("fileAssignment", Context.*MODE\_PRIVATE*).*use* **{  
 it**.write(updatedFile.*toByteArray*())  
 **}**}

### Reading a Json File Using Klaxon

**Example code:** (adding each item in saved array to allList)

val file = File(requireContext().*filesDir*, "fileAssignment")  
  
// \* deserialize and read .json \*  
// read json file  
val fileJson = file.*readText*()  
  
// convert fileJson into list  
JsonReader(StringReader(fileJson)).*use* **{** reader **->** reader.beginArray **{** while (reader.hasNext()) {  
 val t = Klaxon().parse<Task>(reader)  
 allList.add(t!!) // add task to allList either way  
 }  
 **}  
}**

### Check If a File Exists

**Activity.kt**

val file = File(this.*filesDir*, "fileName")  
  
if (file.exists()) {

// do something

## Adding Custom Icons (Drawables)

[Manage your app's UI resources with Resource Manager](https://developer.android.com/studio/write/resource-manager)

[icons8 - source for icons](https://icons8.com/icons)