Contents

[Publish Library to Jitpack 4](#_Toc125717943)

[Setting Up View Binding 4](#_Toc125717944)

[Haptic Feedback (Vibrations) 6](#_Toc125717945)

[Google API 6](#_Toc125717946)

[Error: unable to create directory: /tokens 6](#_Toc125717947)

[Error: The Application Default Credentials are not available 6](#_Toc125717948)

[Generate SHA1 Key 6](#_Toc125717949)

[Snackbar 6](#_Toc125717950)

[Custom Snackbar 6](#_Toc125717951)

[Splash Screen 7](#_Toc125717952)

[Switch 7](#_Toc125717953)

[setOnCheckedChangeListener 7](#_Toc125717954)

[Issue: ActivityMainLog Doesn’t Immediately Reflect Changes In Settings 7](#_Toc125717955)

[Themes.xml 8](#_Toc125717956)

[Colors 8](#_Toc125717957)

[LinearLayout 8](#_Toc125717958)

[Setting Background Color of a LinearLayout 8](#_Toc125717959)

[ViewModel 8](#_Toc125717960)

[Adding Constructor Parameters to ViewModel 9](#_Toc125717961)

[AndroidViewModel 9](#_Toc125717962)

[Sharing ViewModel Between Fragments 10](#_Toc125717963)

[Saving / Reading Files from ViewModel 10](#_Toc125717964)

[Setting Parent Activity (For Up Arrows) 10](#_Toc125717965)

[EditText 10](#_Toc125717966)

[Set Margins Dynamically 10](#_Toc125717967)

[Convert From dp To px (Set Margins Dynamically) 11](#_Toc125717968)

[Setting Text of an Edit Text 11](#_Toc125717969)

[Removing The Underline 11](#_Toc125717970)

[TextView 11](#_Toc125717971)

[Set Rounded Background (Or Custom Background) 11](#_Toc125717972)

[Set Background Tint Dynamically 12](#_Toc125717973)

[Revert To Default TextView Color 12](#_Toc125717974)

[Set TextView Margins Dynamically 12](#_Toc125717975)

[Set Text Opacity 12](#_Toc125717976)

[Set Text Color Dynamically 12](#_Toc125717977)

[ImageView 12](#_Toc125717978)

[Setting The Color of an ImageView 12](#_Toc125717979)

[Android Studio 13](#_Toc125717980)

[Issue: Can’t Read R.() Files 13](#_Toc125717981)

[Spinner (Drop Down Selection) 13](#_Toc125717982)

[Dynamically created spinner 13](#_Toc125717983)

[Statically created spinner 13](#_Toc125717984)

[Spinner With Dynamically defined List of Items 14](#_Toc125717985)

[Custom Spinner Adapter 14](#_Toc125717986)

[Set Selected Item of a Spinner 15](#_Toc125717987)

[Parcelize (Creating A Parcelable OBject) 15](#_Toc125717988)

[Maps 15](#_Toc125717989)

[Finding If a Key Exists 15](#_Toc125717990)

[For Loops 15](#_Toc125717991)

[Generate a View Id 16](#_Toc125717992)

[ActionBar / Menu 16](#_Toc125717993)

[Creating an Action Bar 16](#_Toc125717994)

[Creating a Menu 17](#_Toc125717995)

[Action Bar Title Text Color 17](#_Toc125717996)

[Customise Menu Icon Colors 17](#_Toc125717997)

[Status Bar Icons Color 17](#_Toc125717998)

[Random Number Generator 17](#_Toc125717999)

[Table Layout Formatting 17](#_Toc125718000)

[Navigation Drawer 18](#_Toc125718001)

[Customize Navigation Drawer Colors 21](#_Toc125718002)

[Change Navigation Drawer Hamburger Icon Color 22](#_Toc125718003)

[Customize Navigation Drawer Icons (Replacing the Hamburger Icon) 23](#_Toc125718004)

[Issue: Error – Activity Does Not Have a NavController Set On … 23](#_Toc125718005)

[Issue: Dark Theme Drawer Item Background remains Black, Can’t Be Customised 23](#_Toc125718006)

[Recycler View 23](#_Toc125718007)

[Creating a Recycler View With Interface Click LIstener 24](#_Toc125718008)

[Grid Layout RecyclerView 26](#_Toc125718009)

[Search RecyclerView (Filter Results) 26](#_Toc125718010)

[Sectioned RecyclerView 27](#_Toc125718011)

[Sectioned RecyclerView – Prevent Certain ViewHolders From Getting Swiped 27](#_Toc125718012)

[Sectioned RecyclerView – Map of <Position, actualIndex> 28](#_Toc125718013)

[Issue: Sectioning by Date Doesn’t Work All The Time 29](#_Toc125718014)

[Issue: CardView Items Not Taking Up Full Width 30](#_Toc125718015)

[OnScrollListener – Set Floating Action Button (FAB) Visiblity 31](#_Toc125718016)

[RecyclerView Item Divider 31](#_Toc125718017)

[Set Background Color of a CardView 31](#_Toc125718018)

[Getting Context Within a RecyclerView Adapter 31](#_Toc125718019)

[Calling a Method From Parent Activity 31](#_Toc125718020)

[Working with EditTexts in RecyclerView 31](#_Toc125718021)

[Checking For Duplicate EditTexts in RecyclerView 32](#_Toc125718022)

[Accessing Each Item In a Recycler View 33](#_Toc125718023)

[Setting The Background Color of a RecyclerView Item 33](#_Toc125718024)

[Change Item Background Color On Click (Selected / Unselected) 34](#_Toc125718025)

[Issue: Recycler View Doesn’t Show Updated List Contents 34](#_Toc125718026)

[Issue: onCreateViewHolder and onBindViewHolder Not Called 35](#_Toc125718027)

[Issue: Recycler View Not Showing Contents 35](#_Toc125718028)

[issue: Crash When Removing a “Completed Task” 36](#_Toc125718029)

[ViewPager2 and TabLayout 36](#_Toc125718030)

[Using “Add On Tab Selected Listener” 36](#_Toc125718031)

[Setting Button Clickability Dynamically 36](#_Toc125718032)

[Using TextWatcher 37](#_Toc125718033)

[Changing Edit Text Without Triggering Text Watcher 37](#_Toc125718034)

[Getting Context In Text Watcher 37](#_Toc125718035)

[Using NumberPicker 38](#_Toc125718036)

[Calendar 38](#_Toc125718037)

[Compare Calendar Dates 38](#_Toc125718038)

[Convert Month - From Int to Month Name 38](#_Toc125718039)

[Converting Calendar to Integer 39](#_Toc125718040)

[Converting Integer to Calendar 39](#_Toc125718041)

[Convert Day of Week Int to String Names 39](#_Toc125718042)

[CalendarView 39](#_Toc125718043)

[Get Day of Week From a Date 40](#_Toc125718044)

[Convert Date (Individual Integers) into Date: Int For Sorting By Date 40](#_Toc125718045)

[prolificinteractive Material Calendar View 40](#_Toc125718046)

[Change Calendar Text Color 41](#_Toc125718047)

[Add Labels to Calendar Days 42](#_Toc125718048)

[Issue: “Cannot Access Class ‘org.threeten.bp.Localdate’” 42](#_Toc125718049)

[hugoandrade CalendarView Widget 42](#_Toc125718050)

[Adding Events to Calendar (CalendarObject) 43](#_Toc125718051)

[Editing Events in Calendar (CalendarObject) 43](#_Toc125718052)

[Setting a Calendar Date (DatePicker) 45](#_Toc125718053)

[Setting Minimum Date 45](#_Toc125718054)

[PagerAdapter (ViewPager Swipe Views) 45](#_Toc125718055)

[Adjust Scale and Alpha Of Pages Not In Focus 47](#_Toc125718056)

[Kotlin Lists 48](#_Toc125718057)

[Convert Resource Array (r.array) To List 48](#_Toc125718058)

[Check If An Element Is In a List 48](#_Toc125718059)

[Combining Lists 48](#_Toc125718060)

[Check For Duplicates In a List 48](#_Toc125718061)

[Check if All Elements in a List Are Equal 49](#_Toc125718062)

[Sum of a List of Integers 49](#_Toc125718063)

[Convert String to List 49](#_Toc125718064)

[Kotlin Strings 49](#_Toc125718065)

[Trim a String 50](#_Toc125718066)

[Insert Characters Into String 50](#_Toc125718067)

[Send Data / Passing Values 50](#_Toc125718068)

[Passing Data Using Bundle (Activity to Activity) 50](#_Toc125718069)

[Passing Data Using Bundle (Activity to Fragment) 50](#_Toc125718070)

[Passing Data Using FragmentManager (Fragment Result API) 51](#_Toc125718071)

[Homework Log App – Specific Issue: RV Does Not Refresh When Items Are Swiped 51](#_Toc125718072)

[Dialog Fragments (eg. Popup Rating System) 52](#_Toc125718073)

[Custom AlertDialog 52](#_Toc125718074)

[Referencing R.attr.(color) 53](#_Toc125718075)

[Json / Klaxon 53](#_Toc125718076)

[Saving a Json File Using Klaxon 53](#_Toc125718077)

[Reading a Json File Using Klaxon 53](#_Toc125718078)

[Check If a File Exists 53](#_Toc125718079)

[Issue: Expected a [ but read { 54](#_Toc125718080)

[Adding Custom Icons (Drawables) 54](#_Toc125718081)

## Publish Library to Jitpack

[tutorial](https://dev.to/vtsen/how-to-publish-android-library-on-jitpackio-with-github-50n1), [jitpack.io](https://jitpack.io/)

## Setting Up View Binding

**build.gradle (app)**

android **{** ...buildFeatures **{** viewBinding true  
 **}  
}**

**MainActivity.kt**

private lateinit var binding: ActivityMainBinding  
  
override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 binding = ActivityMainBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
}

**FirstFragment.kt**

private var \_binding: FragmentFirstBinding? = null  
private val binding get() = \_binding!!  
  
override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
): View? {  
 \_binding = FragmentFirstBinding.inflate(inflater, container, false)  
 return binding.*root*}

## Haptic Feedback (Vibrations)

**Fragment.kt**

requireView().performHapticFeedback(HapticFeedbackConstants.*LONG\_PRESS*)

Note: different *HapticFeedbackConstants* can be used

## Google API

[my github repo](https://github.com/emm-an-uel/sheets-api-dummy) – learning how to use Sheets API to pull data from Google Sheets and populate a recyclerview

### Error: unable to create directory: /tokens

[solution](https://www.appsloveworld.com/kotlin/100/132/java-io-ioexception-unable-to-create-directory-storage-emulated-0-tokens)

### Error: The Application Default Credentials are not available

**Solution:** [stackoverflow](https://stackoverflow.com/questions/65060664/how-to-point-google-application-credentials-to-my-json-file)

**Steps I did to solve this:**

1. Installed and initialized the gcloud CLI, created credential file (following this [tutorial](https://cloud.google.com/docs/authentication/provide-credentials-adc#local-dev))
2. Imported credential file (json) into Android Studio’s assets folder
3. Defined where to get “GOOGLE\_APPLICATION\_CREDENTIALS” as below:

val GOOGLE\_APPLICATION\_CREDENTIALS: String = "application\_default\_credentials.json"

1. Created GoogleCredentials by getting credentials stuff from above json file as such:

val stream: InputStream = *resources*.*assets*.open(GOOGLE\_APPLICATION\_CREDENTIALS)  
  
val credentials: GoogleCredentials = GoogleCredentials.fromStream(stream)  
 .createScoped(Collections.singleton(SheetsScopes.*SPREADSHEETS*))

## Generate SHA1 Key

[Method 1](https://www.youtube.com/watch?v=dwmMTsHFkbc)

[Method 2](https://www.youtube.com/watch?v=Ah1wX26lmxY) (to use if method 1 doesn’t work)

## Snackbar

### Custom Snackbar

**Fragment.kt**

private fun createSnackbar(deletedTaskItem: TaskItem, pos: Int) {  
 val snack = Snackbar.make(rvDone, "", Snackbar.*LENGTH\_LONG*)  
  
 val customSnackView = *layoutInflater*.inflate(R.layout.*snackbar\_undo\_delete*, null, false) // inflate custom snackbar layout  
  
 if (snack.*view*.*background* != null) { // set default background to transparent  
 snack.*view*.setBackgroundColor(ContextCompat.getColor(requireContext(), com.google.android.material.R.color.*mtrl\_btn\_transparent\_bg\_color*))  
 }  
   
 // add custom view   
 val snackbarLayout: Snackbar.SnackbarLayout = snack.*view* as Snackbar.SnackbarLayout  
 snackbarLayout.setPadding(5, 0, 5, 15)  
 snackbarLayout.addView(customSnackView)  
  
 // btnUndo functionality  
 val btnUndo: Button = snackbarLayout.findViewById(R.id.*btnUndo*)  
 btnUndo.setOnClickListener **{** cancelDelete(deletedTaskItem, pos)  
 snack.dismiss()  
 **}** snack.show()  
}

*Note: R.layout.snackbar\_undo\_delete is a custom xml layout file which consists of a CardView with a TextView and Button.*

## Splash Screen

A splash screen (also known as launch screen) is the screen first shown when an app is launched. By default, the splash screen shows the app’s icon in the centre of the screen. The splash screen can be modified as shown in this [tutorial](https://proandroiddev.com/splash-screen-in-android-3bd9552b92a5). (*Use option 3: Splash Screen API)*

Note the following:

**AndroidManifest.xml**

<application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/logo"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/logo\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.AppSplash"  
 tools:targetApi="31">

*android:theme* attribute is set to the SplashScreen theme, but for every other activity, it is necessary to specify the theme as the app’s default theme – Theme.HomeworkLogApp in this case, otherwise it’ll crash when trying to launch another activity (one without its theme defined as the app’s default theme).

<activity  
 android:name=".ActivityAllSettings"  
 android:exported="false"  
 android:label="Settings"  
 android:parentActivityName=".ActivityMainLog"  
 android:screenOrientation="portrait"  
 android:theme="@style/Theme.HomeworkLogApp" />

## Switch

### setOnCheckedChangeListener

Listener to determine if the switch has been checked / unchecked

[tutorial](https://medium.com/@hasperong/switch-button-kotlin-android-59e1a08b1fdf)

**RVAdapter.kt**

setOnCheckedChangeListener **{** \_, isChecked **->** if (isChecked) {  
 // do something   
 } else {  
 // do something   
 }  
**}**

### Issue: ActivityMainLog Doesn’t Immediately Reflect Changes In Settings

**Issue description:** After changing switchSettings from checked to unchecked (or vice versa) – where checked = glow, unchecked = no glow, the rv items in MainLog still reflect the previous setting (ie if user changes from glow to no glow, rv item will still have glow). Rv item only reflects changes if the user goes back into settings and back out.

**Cause of issue:** User preferences in Settings are saved in the onDestroy method. User preferences are called in the onResume method in MainLog. MainLog’s onResume is called before Settings’ onDestroy. This means that MainLog calls up the previous settings before the new ones are saved.

**The solution:** User preferences will now be saved as soon as they are changed – saveSettings method will be called in the onCheckedChangeListener of switchSettings.

## Themes.xml

### Colors

|  |  |  |
| --- | --- | --- |
| **Item name** | **What it is** | **Example** |
| colorSurface | Color of menu bar |  |
| android:colorBackground | Background color in general |  |
|  |  |  |
|  |  |  |
|  |  |  |

## LinearLayout

### Setting Background Color of a LinearLayout

**Activity.kt**

val backgroundColor = ContextCompat.getColor(context, R.color.red)  
  
linearLayout.setBackgroundColor(backgroundColor)

## 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:2.5.1"

**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.*

Also note that any changes to variables in the ViewModel is reflected in the Fragments which show its data. eg if ‘todoList’ is a variable instantiated in the ViewModel and called upon in the Fragment, any changes to todoList (eg if an item is added) in the ViewModel will be reflected in todoList in the Fragment.

### 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

### Set Margins Dynamically

**Activity.kt** – where etTask is an EditText defined dynamically.

val layoutParams: RelativeLayout.LayoutParams = RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.*WRAP\_CONTENT*, RelativeLayout.LayoutParams.*WRAP\_CONTENT*)  
layoutParams.setMargins(30, 5, 30, 0)  
  
linearLayout.addView(etTask, layoutParams)

### Convert From dp To px (Set Margins Dynamically)

[stackoverflow](https://stackoverflow.com/questions/12728255/in-android-how-do-i-set-margins-in-dp-programmatically)

private fun dpToPx(context: Context, dp: Float): Int {  
 val r: Resources = context.*resources* return TypedValue.applyDimension(  
 TypedValue.*COMPLEX\_UNIT\_DIP*,  
 dp,  
 r.*displayMetrics* ).toInt()  
}

Note that when setting margins statically in xml, I use ‘dp’, but when setting margins dynamically in Kotlin, I’m working with px. Use the above method to convert from dp to px as shown below:

val layoutParams = LinearLayout.LayoutParams(LinearLayout.LayoutParams.*MATCH\_PARENT*, LinearLayout.LayoutParams.*WRAP\_CONTENT*)  
layoutParams.setMargins(dpToPx(context, 5F), dpToPx(context, 10F), dpToPx(context, 5F), dpToPx(context, 10F))  
holder.itemView.*layoutParams* = layoutParams

In the above example, 5F, 10F etc are *dp* values of the margins that I want. In order to set these *dp* margins dynamically though, I have to convert them into *px* by calling on *dpToPx()* method.

### 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

### Set Rounded Background (Or Custom Background)

**bg\_rounded.xml** (in Drawables folder)

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<stroke

android:width="10dp"

android:color="#f00" />

<solid android:color="#aaa" />

<corners

android:radius="5dp"

android:topRightRadius="100dp" />

</shape>

**TextView in Fragment.xml**

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="@drawable/bg\_rounded"

android:text="Text"

android:padding="20dp"

android:layout\_margin="10dp"

/>

### Set Background Tint Dynamically

**Activity.kt**

val bgColor = getColor(context, com.google.android.material.R.attr.*colorSecondaryContainer*)

tvDueDate.*backgroundTintList* = ColorStateList.valueOf(bgColor)

Use the *backgroundTIntList* attribute as shown above. This maintains any custom backgrounds used.

The alternative method below overwrites any custom backgrounds used and replaces them with a simple rectangle instead.

tvDueDate.setBackgroundColor(getColor(context, com.google.android.material.R.attr.*colorSecondaryContainer*))

### 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

spinnerColor.setPopupBackgroundResource(android.R.color.*transparent*) // remove default black background if I’m using spinner items with rounded corners

**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  
 }  
}

#### Remove Default Black Background

[stackoverflow](https://stackoverflow.com/a/32139693/19750288)

**Activity.kt**

spinnerSubject.setPopupBackgroundResource(android.R.color.*transparent*) // remove default black background

### 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 / Menu

### 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.*

### Creating a Menu

Note that the method covered in the above section has been deprecated. Use the method shown in [stackoverflow](https://stackoverflow.com/questions/71917856/sethasoptionsmenuboolean-unit-is-deprecated-deprecated-in-java) instead.

### Action Bar Title Text Color

[youtube tutorial](https://www.youtube.com/watch?v=2saHwKHxpyk)

**themes.xml**

<item name="titleTextColor">@color/black</item>

Note that the parent attribute should be defined as follows: (instead of the default parent)

<style name="Theme.MarcellinaPizzas" parent="Theme.AppCompat.Light.DarkActionBar">

The parent attribute of the “night\themes” xml file should also be changed as follows, otherwise the app will crash when changing between light / dark themes.

<style name="Theme.MarcellinaPizzas" parent="Theme.AppCompat.DayNight.DarkActionBar">

### Customise Menu Icon Colors

**menu.xml** *(in the menu directory)*

<item  
 android:id="@+id/actionSearch"  
 android:icon="@drawable/ic\_action\_search"  
 android:title="Search"  
 app:actionViewClass="android.widget.SearchView"  
 app:iconTint="?attr/searchIconColor"  
 app:showAsAction="always" />

Note the *iconTint* attribute has an *app* namespace declaration, not *android*.

If the above method doesn’t work, try setting icon tint dynamically as follows:

### Status Bar Icons Color

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

**themes.xml**

<item name="android:windowLightStatusBar">true</item>

## Random Number Generator

val randomGenerator = Random(System.currentTimeMillis())  
val randomNumber = randomGenerator.nextInt(50) // generates a random number between 0 - 49 (not including 50)

## 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*

## Navigation Drawer

[my github repo](https://github.com/emm-an-uel/marcellina-pizzas) – Marcellina Pizzas app which uses a navigation drawer.

**AndroidManifest.xml**

<application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/app\_icon"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/app\_icon\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.MarcellinaPizzas.NoActionBar"  
 tools:targetApi="31">

Note that the theme attribute must be defined as above (this *style* will be defined in themes.xml) for the nav drawer to go above the action bar. Otherwise, the nav drawer will be behind the action bar and won’t take up the full screen – doesn’t look as clean.

**MainActivity.kt**

// view binding  
private lateinit var appBarConfiguration: AppBarConfiguration  
private lateinit var binding: ActivityMainBinding

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

...  
  
 binding = ActivityMainBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
  
 setSupportActionBar(binding.appBarMain.toolbar)  
  
 // setup navigation drawer  
 val drawerLayout: DrawerLayout = binding.drawerLayout  
 val navView: NavigationView = binding.navView  
 val navController = *findNavController*(R.id.*nav\_host\_fragment\_content\_main*)  
 appBarConfiguration = *AppBarConfiguration*(  
 *setOf*(  
 R.id.*nav\_quiz*, R.id.*nav\_solutions* ), drawerLayout  
 )  
 *setupActionBarWithNavController*(navController, appBarConfiguration)  
 navView.*setupWithNavController*(navController)  
}  
  
override fun onSupportNavigateUp(): Boolean {  
 val navController = *findNavController*(R.id.*nav\_host\_fragment\_content\_main*)  
 return navController.*navigateUp*(appBarConfiguration) || super.onSupportNavigateUp()  
}

**activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.drawerlayout.widget.DrawerLayout 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:id="@+id/drawer\_layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:fitsSystemWindows="true"  
 tools:openDrawer="start">  
  
 <include  
 android:id="@+id/app\_bar\_main"  
 layout="@layout/app\_bar\_main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
  
 <com.google.android.material.navigation.NavigationView  
 android:id="@+id/nav\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start"  
 android:backgroundTint="?attr/drawerBackground"  
 android:fitsSystemWindows="true"  
 app:headerLayout="@layout/nav\_header\_main"  
 app:itemBackground="@drawable/drawer\_item\_selector"  
 app:itemTextColor="@color/drawer\_item\_text\_color"  
 app:menu="@menu/activity\_main\_drawer" />  
  
</androidx.drawerlayout.widget.DrawerLayout>

The activity\_main.xml file is just to define how the navigation drawer will look like. This file *includes* the app\_bar\_main.xml file, which defines how the action bar will look like, and *includes* the content\_main.xml file, which handles the actual fragments.

**app\_bar\_main.xml**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.coordinatorlayout.widget.CoordinatorLayout 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=".MainActivity">  
  
 <com.google.android.material.appbar.AppBarLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:theme="@style/Theme.MarcellinaPizzas.AppBarOverlay">  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="?attr/colorPrimary"  
 app:popupTheme="@style/Theme.MarcellinaPizzas.PopupOverlay" />  
  
 </com.google.android.material.appbar.AppBarLayout>  
  
 <include layout="@layout/content\_main" />  
  
</androidx.coordinatorlayout.widget.CoordinatorLayout>

**content\_main.xml**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout 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"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"  
 tools:showIn="@layout/app\_bar\_main">  
  
 <fragment  
 android:id="@+id/nav\_host\_fragment\_content\_main"  
 android:name="androidx.navigation.fragment.NavHostFragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:defaultNavHost="true"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:navGraph="@navigation/nav\_graph" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

Note: the highlighted line – defining the *layout behaviour* – ensures the fragment doesn’t get cut off by the action bar.

**nav\_header\_main.xml**

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="?attr/drawerBackground"  
 android:gravity="bottom"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="@dimen/status\_bar\_height"  
 android:contentDescription="Navigation Header"  
 android:paddingTop="10dp"  
 android:scaleX="2"  
 android:scaleY="2"  
 android:src="@mipmap/logo\_letters\_foreground" />  
  
</LinearLayout>

The nav\_header\_main.xml file defines how the navigation drawer header looks like. In this example, I’ve simply got the Marcellina logo against a solid color background, defined by *drawerBackground* color attribute.

**main\_activity\_drawer.xml** (in the *menu* directory)

<?xml version="1.0" encoding="utf-8"?>  
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 tools:showIn="navigation\_view">  
  
 <group android:checkableBehavior="single">  
 <item  
 android:id="@+id/nav\_quiz"  
 android:title="Quiz" />  
 <item  
 android:id="@+id/nav\_solutions"  
 android:title="Solutions" />  
 </group>  
</menu>

The main\_activity\_drawer.xml file is located in the *menu* directory, and it defines the items in the navigation drawer. In this example, I’ve only got two fragments, Quiz and Solutions, so I only need two items as shown above.

**themes.xml**

<resources xmlns:tools="http://schemas.android.com/tools">  
 <!-- Base application theme. -->  
 <style name="Theme.MarcellinaPizzas" parent="Theme.AppCompat.Light.DarkActionBar">  
 <!-- Primary brand color. -->

...  
 </style>  
  
 <style name="Theme.MarcellinaPizzas.NoActionBar">  
 <item name="windowActionBar">false</item>  
 <item name="windowNoTitle">true</item>  
 </style>  
  
 <style name="Theme.MarcellinaPizzas.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar" />  
  
 <style name="Theme.MarcellinaPizzas.PopupOverlay" parent="ThemeOverlay.AppCompat.Light" />  
</resources>

Note the three additional *styles* – NoActionBar, AppBarOverlay, and PopupOverlay. These are referred to in the above files when creating the nav drawer and are necessary for the nav drawer to appear above the action bar (for improved aesthetic).

**nav\_graph.xml** (in the *navigation* directory)

<?xml version="1.0" encoding="utf-8"?>  
<navigation 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:id="@+id/nav\_graph"  
 app:startDestination="@id/nav\_quiz">  
  
 <fragment  
 android:id="@+id/nav\_quiz"  
 android:name="com.example.marcellinapizzas.FragmentQuiz"  
 android:label="Quiz"  
 tools:layout="@layout/fragment\_quiz" />  
  
 <fragment  
 android:id="@+id/nav\_solutions"  
 android:name="com.example.marcellinapizzas.FragmentSolutions"  
 android:label="Solutions"  
 tools:layout="@layout/fragment\_solutions" />  
  
</navigation>

This is required for nav\_host in content\_main.xml to function.

**Build.gradle (app)**

dependencies **{** // navigation  
 implementation 'androidx.navigation:navigation-fragment-ktx:2.5.3'  
 implementation 'androidx.navigation:navigation-ui-ktx:2.5.3'

These dependencies are required for the navigation (nav graph, nav host etc) to function.

### Customize Navigation Drawer Colors

[youtube tutorial](https://www.youtube.com/watch?v=FxYBxVPaK1w&t=574s), [github repo (tutorial)](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbGFrTmJmalc2UF9ZWGg0c0xFVllOOWhNc2Y3UXxBQ3Jtc0tuSVhfZEVSWDc1M2Ztd2oxUVZPSmxzd0JoMXR2bFV2QnYtZXRjQWZCNkZXVzc5b18tVE5SMjlqeTRRcU1nUmJ1YV9FejJQQ3UzTmo4aXJVSG9ZcGl2bDZCeTN6OGpsSDVvU1Yxc0hWN0p2bUtXU2Rsaw&q=https%3A%2F%2Fgithub.com%2FAppDevAssist%2FCustomNavigationDrawer&v=FxYBxVPaK1w), [my github repo](https://github.com/emm-an-uel/marcellina-pizzas)

**drawer\_item\_text\_color.xml (res -> color directory)**

<?xml version="1.0" encoding="utf-8"?>  
<selector xmlns:android="http://schemas.android.com/apk/res/android">  
 <item android:color="?attr/drawerItemColorChecked" android:state\_checked="true" />  
 <item android:color="?attr/drawerItemColorNormal" />  
</selector>

This defines the text colors when the drawer item is checked/unchecked.

**drawer\_item\_bg.xml (res -> drawable directory)**

<?xml version="1.0" encoding="utf-8"?>  
<shape xmlns:android="http://schemas.android.com/apk/res/android"  
 android:shape="rectangle">  
 <solid android:color="?attr/drawerItemBackground" />  
</shape>

|  |  |
| --- | --- |
| This defines the background of the drawer item when it’s checked – in this example, it’s a solid rectangle background, shown alongside (the light blue). |  |

**drawer\_item\_selector.xml (res -> drawable directory)**

<?xml version="1.0" encoding="utf-8"?>  
<selector xmlns:android="http://schemas.android.com/apk/res/android">  
 <item android:drawable="@drawable/drawer\_item\_bg" android:state\_checked="true" />  
</selector>

This checks if the drawer item is checked, and if it is, sets the drawer item background as drawer\_item\_bg.xml, as defined above.

**activity\_main.xml**

<com.google.android.material.navigation.NavigationView  
 android:id="@+id/nav\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start"  
 android:backgroundTint="?attr/drawerBackground"  
 android:fitsSystemWindows="true"  
 app:headerLayout="@layout/nav\_header\_main"  
 app:itemBackground="@drawable/drawer\_item\_selector"  
 app:itemTextColor="@color/drawer\_item\_text\_color"  
 app:menu="@menu/activity\_main\_drawer" />

The highlighted attributes are those used to customise drawer colors.

**app\_attrs.xml (res -> values directory)**

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
 <attr name="drawerItemColorChecked" format="color" />  
 <attr name="drawerItemColorNormal" format="color" />  
 <attr name="drawerItemBackground" format="color" />  
  
 <attr name="drawerBackground" format="color" />  
</resources>

This allows you to add custom attributes, such as the color attributes added in this example.

**themes.xml**

<item name="drawerItemColorNormal">@color/black</item>  
<item name="drawerItemColorChecked">@color/white</item>  
<item name="drawerItemBackground">@color/dark\_blue</item>  
  
<item name="drawerBackground">@color/white</item>

The custom attributes added above in app\_attrs.xml can be defined in themes.xml as you would with the more standard colors (eg colorPrimary, colorPrimaryVariant etc).

### Change Navigation Drawer Hamburger Icon Color

**themes.xml** [(stackoverflow)](https://stackoverflow.com/questions/31870132/how-to-change-color-of-hamburger-icon-in-material-design-navigation-drawer)

<style name="Theme.MarcellinaPizzas" parent="Theme.AppCompat.Light.DarkActionBar">  
 <!-- Primary brand color. -->  
 <item name="colorPrimary">@color/white</item>  
 <item name="colorPrimaryVariant">@color/m\_red</item>  
 <item name="colorOnPrimary">@color/white</item>

...  
  
 <!-- hamburger icon -->  
 <item name="drawerArrowStyle">@style/DrawerArrowStyle</item>  
</style>  
  
<style name="DrawerArrowStyle" parent="@style/Widget.AppCompat.DrawerArrowToggle">  
 <item name="spinBars">true</item>  
 <item name="color">@android:color/black</item>  
</style>

### Customize Navigation Drawer Icons (Replacing the Hamburger Icon)

[youtube tutorial](https://www.youtube.com/watch?v=biUaIO-N7Ew)

### Issue: Error – Activity Does Not Have a NavController Set On …

**Description:**

I’m using a FragmentContainerView in the xml file (instead of *fragment*), and initially used the following code to get the navController but the app would crash with the above error message.

val navController = *findNavController*(R.id.*nav\_host\_fragment\_content\_main*)

[**Solution**](https://stackoverflow.com/questions/65170700/activity-does-not-have-a-navcontroller-set-on)**: MainActivity.kt**

val navHostFragment = *supportFragmentManager*.findFragmentById(R.id.*nav\_host\_fragment\_content\_main*)!!  
val navController = navHostFragment.*findNavController*()

### Issue: Dark Theme Drawer Item Background remains Black, Can’t Be Customised

|  |  |
| --- | --- |
| **Description of the issue:**  The *drawerItemBackground* color for dark theme can’t be changed. While the light theme *drawerItemBackground* color can be changed by modifying the color attribute in the ‘themes.xml’ file, any changes made to this attribute in the night ‘themes.xml’ file is not reflected (shown alongside – the *drawerItemBackground* color for “Quiz” is black, although I specified it to be light blue in the night ‘themes.xml’ file). |  |

**Solution:**

I found that I added the following code into the night ‘themes.xml’ file to change the background color from the default gray to black. However, it seems that by default, the selected item background color will be same color as the *android:backgroundTint* attribute, so this attribute will override the *drawerItemBackground* that I’ve defined in my night ‘themes.xml’ file.

<item name="android:backgroundTint">@color/black</item>

To solve this, I removed the above line from the night ‘themes.xml’ file, and instead defined the background color in the individual fragment xml files as shown below:

**fragment\_quiz.xml**

<androidx.constraintlayout.widget.ConstraintLayout 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"  
 android:background="?attr/colorPrimary"  
 tools:context=".FragmentQuiz">

Note the highlighted ‘background’ attribute.

## Recycler View

### Creating a Recycler View With Interface Click LIstener

**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*

**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  
})

### Grid Layout RecyclerView

**MainActivity.kt**

val layoutManager = GridLayoutManager(this, 2) // 2 columns

rv.*layoutManager* = layoutManager

**main\_activity.xml** (properties of the recyclerview)

app:layoutManager="androidx.recyclerview.widget.GridLayoutManager"

### Search RecyclerView (Filter Results)

[tutorial](https://www.geeksforgeeks.org/android-searchview-with-recyclerview-using-kotlin/), [my github repo - marcellina pizzas](https://github.com/emm-an-uel/marcellina-pizzas), [stackoverflow](https://stackoverflow.com/questions/71917856/sethasoptionsmenuboolean-unit-is-deprecated-deprecated-in-java) *(the onCreateOptionsMenu method described in the first tutorial has been deprecated for Fragments. My code below uses the stackoverflow method instead).*

**FragmentSolutions.kt**

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
 ...  
  
 // menu - for filtering pizzas  
 val menuHost: MenuHost = requireActivity()  
 menuHost.addMenuProvider(object: MenuProvider {  
 override fun onCreateMenu(menu: Menu, menuInflater: MenuInflater) {  
 menuInflater.inflate(R.menu.*search\_menu*, menu)  
  
 val searchItem: MenuItem = menu.findItem(R.id.*actionSearch*)  
 val searchView: SearchView = searchItem.*actionView* as SearchView  
 searchView.setOnQueryTextListener(object: SearchView.OnQueryTextListener {  
 override fun onQueryTextSubmit(p0: String?): Boolean {  
 return false  
 }  
 override fun onQueryTextChange(p0: String?): Boolean {  
 filter(p0)  
 return false  
 }  
 private fun filter(p0: String?) {  
 val filteredList: ArrayList<Pizza> = *arrayListOf*()  
 if (p0 != null) {  
 for (item in listOfPizzas) {  
 if (item.name.*contains*(p0, true)) {  
 filteredList.add(item)  
 }  
 }  
 }  
 rvAdapter.filterList(filteredList)  
 }  
 })  
 }  
  
 override fun onMenuItemSelected(menuItem: MenuItem): Boolean {  
 return when (menuItem.*itemId*) {  
 R.id.*actionSearch* -> {  
 true  
 } else -> false  
 }  
 }  
 }, *viewLifecycleOwner*, Lifecycle.State.*RESUMED*)  
}

Note the purpose of the last line above is to ensure that the search icon (and, by extension, the entire menu) is shown only when FragmentSolutions is in state *resumed*. This prevents a) search icon from being shown in other fragments, and b) duplicate search icons – which happened without this line –> I could navigate from FragmentSolutions to FragmentQuiz and there would still be a search icon. When I went back to FragmentSolutions, there would be two search icons.

**RVAdapterSolutions.kt**

// filtering (searching)  
fun filterList(listOfFilteredPizzas: ArrayList<Pizza>) {  
 listOfPizzas = listOfFilteredPizzas  
 notifyDataSetChanged()  
}

**search\_menu.xml** *(in menu directory)*

<?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/actionSearch"  
 android:icon="@drawable/ic\_action\_search"  
 android:title="Search"  
 app:actionViewClass="android.widget.SearchView"  
 app:showAsAction="always" />  
</menu>

### Sectioned RecyclerView

|  |  |
| --- | --- |
| **Description:** recyclerview with dividers, eg grouped by date  [stackoverflow](https://stackoverflow.com/questions/41447044/divide-elements-on-groups-in-recyclerview-or-grouping-recyclerview-items-say-by), [my github repo](https://github.com/emm-an-uel/sectioned-recyclerview) |  |

### Sectioned RecyclerView – Prevent Certain ViewHolders From Getting Swiped

[stackoverflow](https://stackoverflow.com/questions/30713121/disable-swipe-for-position-in-recyclerview-using-itemtouchhelper-simplecallback)

**MainActivity.kt – private fun swipeFunctions()**

override fun getSwipeDirs (recyclerView: RecyclerView, viewHolder: RecyclerView.ViewHolder): Int {  
 if (viewHolder is Adapter.DateViewHolder) return 0 // prevents DateViewHolders from getting swiped   
 return super.getSwipeDirs(recyclerView, viewHolder)  
}

### Sectioned RecyclerView – Map of <Position, actualIndex>

**The Issue:** in a sectioned recyclerview, there are 2 view types – TaskItem and DateItem. TaskItem is displayed in cardview with details like ‘subject’, ‘task’, ‘due date’ etc, while DateItem is just a textview with a date on it. When a consolidatedList containing both TaskItems and DateItems are passed into the rvAdapter, the indices of the TaskItems no longer match those of the corresponding Tasks in todoList. Thus, a map of <position, actualIndex> needs to be created. The table below visualizes the data this map keeps:

|  |  |
| --- | --- |
| Item position in consolidatedList | Actual index of corresponding Task in todoList |
| 1 | 0 |
| 2 | 1 |
| 4 | 2 |
| 5 | 3 |
| 6 | 4 |

In the above example, positions 0 and 3 contain a DateItem, which is why the map does not store these keys – since the map’s purpose is to track only TaskItems. The ‘actual index’ column is a continuous count from 0 to 4, representing the index of these TaskItems’ corresponding Tasks in todoList.

In the following code, whenever a Task is called from todoList, I first need to find its actualIndex by looking up the TaskItem position in mapOfIndex.

private fun createMapOfIndex() {  
 mapOfIndex = *mutableMapOf*()  
 var index = 0  
 for (n in 0 *until* consolidatedList.size) {  
 if (consolidatedList[n].type == ListItem.TYPE\_TASK) {  
 mapOfIndex[n] = index  
 index++  
 }  
 }  
}

private fun updateMap(pos: Int, indexChanged: Boolean) {  
 mapOfIndex.remove(pos) // remove the key-value pair of the swiped item  
  
 // adjust the following key-value pairs  
 if (indexChanged) { // TaskItem got removed  
 for (p in pos+1 *until* consolidatedList.size+1) {  
 if (mapOfIndex.containsKey(p)) { // if it doesn't contain p, that means there is a DateItem in that position (not a TaskItem)  
 val oldValue = mapOfIndex[p]!!  
 mapOfIndex.remove(p)  
 mapOfIndex[p-1] = oldValue-1  
 }  
 }  
 } else { // DateItem got removed  
 for (p in pos+1 *until* consolidatedList.size+1) {  
 if (mapOfIndex.containsKey(p)) {  
 val actualIndex = mapOfIndex[p]!!  
 mapOfIndex.remove(p)  
 mapOfIndex[p-1] = actualIndex // actualIndex of TaskItems remains unchanged  
 }  
 }  
 }  
}  
  
private fun checkForDoubleDate(removedIndex: Int) {  
 if (removedIndex < consolidatedList.size) {  
 if (consolidatedList[removedIndex].type == ListItem.TYPE\_DATE) {  
 if (consolidatedList[removedIndex-1].type == ListItem.TYPE\_DATE) {  
 // if both a) the item which has replaced the one just removed, and b) the previous item are DateItems  
 consolidatedList.removeAt(removedIndex-1) // remove the double date (ie the one that has no TaskItems below it)  
 rvAdapter.notifyItemRemoved(removedIndex-1)  
 updateMap(removedIndex-1, false)  
 }  
 }  
 } else { // if item removed was the last item in list  
 if (consolidatedList[removedIndex-1].type == ListItem.TYPE\_DATE) {  
 consolidatedList.removeAt(removedIndex-1)  
 rvAdapter.notifyItemRemoved(removedIndex-1)  
 updateMap(removedIndex-1, false)  
 }  
 }  
}

The method *updateMap* above is called when a) a TaskItem is removed (this would have *indexChanged* be true) – all ‘position’ and ‘actualIndex’ following the one which has been removed will have its numbers reduced by 1. An example is shown below, where the item in position 4 of consolidatedList has been removed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item position in consolidatedList | Actual index of corresponding Task in todoList |  | Item position in consolidatedList | Actual index of corresponding Task in todoList |
| 1 | 0 |  | 1 | 0 |
| 2 | 1 |  | 2 | 1 |
| 4 | 2 |  | *Note: this row was left blank for illustration purposes.* | |
| 5 | 3 |  | 4 | 2 |
| 6 | 4 |  | 5 | 3 |

In the method *updateMap*, if a DateItem was removed due to it being a doubleDate (ie it does not contain any TaskItems below it, determined via the method *checkForDoubleDate* above), only the ‘positions’ are edited, illustrated below, assuming a DateItem in position 3 was removed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item position in consolidatedList | Actual index of corresponding Task in todoList |  | Item position in consolidatedList | Actual index of corresponding Task in todoList |
| 1 | 0 |  | 1 | 0 |
| 2 | 1 |  | 2 | 1 |
| 4 | 2 |  | 3 | 2 |
| 5 | 3 |  | 4 | 3 |
| 6 | 4 |  | 5 | 4 |

### Issue: Sectioning by Date Doesn’t Work All The Time

|  |  |
| --- | --- |
| **The goal:**  Have GeneralItems in the list be sectioned according to its date, where “overdue” is when its date is less than today’s date, “due today” is when its date == today’s date etc. | **The issue:**  I attempted to achieve this by comparing the Calendar object stored in each item against the Calendar objects which represented “today”, “tomorrow” etc., but this resulted in sectioning which was sometimes wrong – items which were meant to be “due today” was placed under “overdue”, items “due tomorrow” was placed under “due next week”.  **The solution:**  I couldn’t figure out why comparing Calendar objects produced inconsistent results, but I found that converting the Calendar objects to Int objects before doing the comparisons solved the issue. I used the following code to convert from Calendar to Int:  private fun dateToInt(date: Calendar): Int {  val year = date.get(Calendar.*YEAR*)  val month = date.get(Calendar.*MONTH*)+1  val day = date.get(Calendar.*DAY\_OF\_MONTH*)   var monthString = month.toString()  var dayString = day.toString()   // ensure proper MM format  if (month < 10) {  monthString = "0$month" // eg convert "8" to "08"  }   // ensure proper DD format  if (day < 10) {  dayString = "0$day"  }   // convert to YYYYMMDD format  val dateString = "$year$monthString$dayString"  val dateInt = dateString.*toInt*() // return integer so it can be sorted   return(dateInt) } |

### Issue: CardView Items Not Taking Up Full Width

**Issue:** CardView items in RecyclerView does not take up the full width of the screen.

|  |  |
| --- | --- |
| **What it currently looks like** | **What it’s meant to look like**  Graphical user interface, text, application  Description automatically generated |

**Solution:** ‘attachToRoot’ should be defined as ‘false’ when inflating the views [(stackoverflow)](https://stackoverflow.com/questions/40576472/android-full-width-cardview-in-recyclerview)

**Adapter.kt**

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder {  
 val layoutInflater = LayoutInflater.from(parent.*context*)  
 return when (viewType) {  
 ListItem.TYPE\_DATE ->  
 DateViewHolder(DateItemBinding.inflate(layoutInflater, parent, false))  
 else ->  
 GeneralViewHolder(GeneralItemBinding.inflate(layoutInflater, parent, false))  
 }  
}

### OnScrollListener – Set Floating Action Button (FAB) Visiblity

[github tutorial](https://gist.github.com/ar-android/58e60b23ff1e51f69e08a22cd6466038)

**Fragment.kt**

rvTodo.addOnScrollListener(object : RecyclerView.OnScrollListener() {  
 override fun onScrolled(recyclerView: RecyclerView, dx: Int, dy: Int) {  
 super.onScrolled(recyclerView, dx, dy)  
  
 if (dy > 0) { // scrolling down  
 (*context* as ActivityMainLog).hideFabAddTask()  
 } else { // scrolling up  
 (*context* as ActivityMainLog).showFabAddTask()  
 }  
 }  
})  
  
(*context* as ActivityMainLog).showFabAddTask() // show by default

### RecyclerView Item Divider

This adds a line after each rv item:

**Fragment.kt**

rvTodo.addItemDecoration(DividerItemDecoration(*context*, LinearLayoutManager.*VERTICAL*))

### 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)

### Change Item Background Color On Click (Selected / Unselected)

**Adapter.kt**

class NewViewHolder(itemView: View, listener: onItemClickListener) :  
 RecyclerView.ViewHolder(itemView) { // initialize views  
 val tvTopping: TextView = itemView.findViewById(R.id.*tvTopping*)  
 val context = tvTopping.*context* init {  
 itemView.setOnClickListener **{** listener.onItemClick(*adapterPosition*)  
  
 // color change based on selected/unselected  
 val colorSelected: ColorStateList = ColorStateList.valueOf(ContextCompat.getColor(context, R.color.*light\_blue*))  
 val colorUnselected: ColorStateList = ColorStateList.valueOf(getColor(context, com.google.android.material.R.attr.*colorPrimaryContainer*))  
  
 if (itemView.*backgroundTintList* == colorSelected) { // item selected  
 itemView.*backgroundTintList* = colorUnselected  
} else { // item unselected  
 itemView.*backgroundTintList* = colorSelected  
}  
 **}** }

note on the code after *listener.onItemClick* – it checks the itemView’s current background color and changes it accordingly when user clicks on it. The actual ‘memory’ side of the code (which saves the user selections) is done in MainActivity as follows:

rvAdapter.setOnItemClickListener(object: Adapter.onItemClickListener {  
 override fun onItemClick(position: Int) {  
 val topping = listOfToppings[position]  
 userMapOfToppings[topping] = userMapOfToppings[topping] != true // if currently true, set to false; vice versa  
 }  
})

### 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: onCreateViewHolder and onBindViewHolder Not Called

**Issue:** When rvAdapter is instantiated, onCreateViewHolder and onBindViewHolder are not called. This results in the entire recyclerview not being initialized.

**Solution:** Note that the rvAdapter class is only instantiated after the onCreate method in its parent activity has been finished. In the above scenario, I was trying to access recyclerview items within the onCreate method – thus all holders returned null since the recyclerview has not been initialized (due to the rvAdapter class not being instantiated since the onCreate method has not been finished).

### 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*

## Calendar

### Compare Calendar Dates

private fun isSameDate(date1: Calendar, date2: Calendar): Boolean {  
 if (date1.get(Calendar.*DAY\_OF\_MONTH*) != date2.get(Calendar.*DAY\_OF\_MONTH*)) {  
 return false  
 }  
 if (date1.get(Calendar.*MONTH*) != date2.get(Calendar.*MONTH*)) {  
 return false  
 }  
 return date1.get(Calendar.*YEAR*) == date2.get(Calendar.*YEAR*)  
}

Note that *if (date1 == date2)* won’t work – Calendar date comparisons have to be done as above, by comparing each constituent date element.

### Convert Month - From Int to Month Name

[stackoverflow](https://stackoverflow.com/questions/1038570/how-can-i-convert-an-integer-to-localized-month-name-in-java)

val month = DateFormatSymbols().*months*[currentMonth]

*where month: String (January – December); currentMonth: Int (0 - 11)*

### Converting Calendar to Integer

private fun calendarToInt(date: Calendar): Int {  
 val year = date.get(Calendar.*YEAR*)  
 val month = date.get(Calendar.*MONTH*)+1  
 val day = date.get(Calendar.*DAY\_OF\_MONTH*)  
  
 var monthString = month.toString()  
 var dayString = day.toString()  
  
 // ensure proper MM format  
 if (month < 10) {  
 monthString = "0$month" // eg convert "8" to "08"  
 }  
  
 // ensure proper DD format  
 if (day < 10) {  
 dayString = "0$day"  
 }  
  
 // convert to YYYYMMDD format  
 val dateString = "$year$monthString$dayString"  
 return (dateString.*toInt*())  
}

### Converting Integer to Calendar

private fun intToCalendar(int: Int): Calendar {  
 val string = int.toString()  
 val year = string.*take*(4).*toInt*()  
 val monthDay = string.*takeLast*(4)  
 val month = (monthDay.*take*(2).*toInt*() - 1) // Calendar months go from 0 to 11  
 val day = monthDay.*takeLast*(2).*toInt*()  
  
 val calendar = Calendar.getInstance()  
 calendar.set(year, month, day)  
 return calendar  
}

### Convert Day of Week Int to String Names

private fun getDayOfWeek(dayInt: Int): String {  
 return when (dayInt) {  
 2 -> "Monday"  
 3 -> "Tuesday"  
 4 -> "Wednesday"  
 5 -> "Thursday"  
 6 -> "Friday"  
 7 -> "Saturday"  
 else -> "Sunday"  
 }  
}

Note that in the above example, Sunday is the first day of week (ie 1 -> Sunday)

## CalendarView

[tutorial](https://www.geeksforgeeks.org/calendar-view-app-in-android-with-kotlin/)

**Activity.kt**

calendarView = findViewById(R.id.calendarView)

        // on below line we are adding set on

        // date change listener for calendar view.

        calendarView

            .setOnDateChangeListener(

                OnDateChangeListener { view, year, month, dayOfMonth ->

                    // In this Listener we are getting values

                    // such as year, month and day of month

                    // on below line we are creating a variable

                    // in which we are adding all the variables in it.

                    val Date = (dayOfMonth.toString() + "-"

                            + (month + 1) + "-" + year)

                    // set this date in TextView for Display

                    dateTV.setText(Date)

                })

### Get Day of Week From a Date

**Activity.kt**

val c = Calendar.getInstance()  
c.set(year, month, dayOfMonth)  
val dayOfWeek = c.get(Calendar.*DAY\_OF\_WEEK*)  
var actualDayOfWeek = ""  
when (dayOfWeek) {  
 1 -> actualDayOfWeek = "Sun"  
 2 -> actualDayOfWeek = "Mon"  
 3 -> actualDayOfWeek = "Tue"  
 4 -> actualDayOfWeek = "Wed"  
 5 -> actualDayOfWeek = "Thu"  
 6 -> actualDayOfWeek = "Fri"  
 7 -> actualDayOfWeek = "Sat"  
}

Note: year, month, and dayOfMonth are Integers defined in an earlier chunk of code.

### Convert Date (Individual Integers) into Date: Int For Sorting By Date

**Activity.kt**

private fun createDateInt(day: Int, month: Int, year: Int): Int {  
 // \* dueDateSort will be in format YYYYMMDD for easy sorting of due dates \*  
  
 var monthString = month.toString()  
 var dayString = day.toString()  
  
 // ensure proper MM format  
 if (month < 10) {  
 monthString = "0$month" // eg convert "8" to "08"  
 }  
  
 // ensure proper DD format  
 if (day < 10) {  
 dayString = "0$day"  
 }  
  
 // convert to YYYYMMDD format  
 val dateString = "$year$monthString$dayString"  
 val dateInt = dateString.*toInt*() // return integer so it can be sorted  
  
 return(dateInt)  
}

The above method converts day, month, year (Int values) into one “YYYYMMDD” Int value for easy sorting of dates.

## prolificinteractive Material Calendar View

Using prolificinteractive’s [MaterialCalendarView](https://github.com/prolificinteractive/material-calendarview)

### Change Calendar Text Color

As of writing this, MaterialCalendarView does not have any built-in methods (or xml attributes) to change its calendar’s text color in a more straightforward way. This is the workaround. *Note that all LocalDates used here are org.threeten.bp.localdate, not java.time.localdate*

**Fragment.kt** *(main method called in onViewCreated)*

private fun setupCalendar() {  
 colorText(calendarView.*currentDate*) // first time coloring  
 calendarView.setOnMonthChangedListener **{** \_, date **->** // color months as user swipes through calendar  
 colorText(date)  
 **}**

**...**

*colorText method referenced above*

private fun colorText(date: CalendarDay) {  
 // set calendar text color  
 val localDate: LocalDate = date.*date* val firstDay = localDate.with(firstDayOfMonth())  
 val lastDay = localDate.with(lastDayOfMonth())  
 val minDate = CalendarDay.from(firstDay)  
 val maxDate = CalendarDay.from(lastDay)  
 calendarView.addDecorator(CurrentMonthTextDecorator(requireContext(), minDate, maxDate))  
 calendarView.addDecorator(OtherMonthTextDecorator(requireContext(), minDate, maxDate))  
}

*two DayViewDecorator classes, one to decorate the dates in the current month, one to reset to the default color for days not in current month*

inner class OtherMonthTextDecorator(  
 private val context: Context,  
 private val minDate: CalendarDay,  
 private val maxDate: CalendarDay  
) : DayViewDecorator { // decorate all dates not within current month with gray  
 override fun shouldDecorate(day: CalendarDay?): Boolean {  
 if (day != null) {  
 return !day.isInRange(minDate, maxDate)  
 }  
 return false  
 }  
  
 override fun decorate(view: DayViewFacade?) {  
 view?.addSpan(ForegroundColorSpan(getColor(context, R.attr.*defaultTextColor*)))  
 }  
  
 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  
 }  
}  
  
inner class CurrentMonthTextDecorator(  
 private val context: Context,  
 private val minDate: CalendarDay,  
 private val maxDate: CalendarDay  
) : DayViewDecorator { // decorate all dates within current month with primaryTextColor  
 override fun shouldDecorate(day: CalendarDay?): Boolean {  
 if (day != null) {  
 return day.isInRange(minDate, maxDate)  
 }  
 return false  
 }  
  
 override fun decorate(view: DayViewFacade?) {  
 view?.addSpan(ForegroundColorSpan(getColor(context, R.attr.*primaryTextColor*)))  
 }  
  
 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  
 }  
}

### Add Labels to Calendar Days

**The goal:** Add events to calendar, with labels similar to Samsung Calendar – will be used in Logit’s CalendarFragment.

[stackoverflow](https://stackoverflow.com/questions/61008523/add-text-to-dates-in-material-calendarview), [my github repo](https://github.com/emm-an-uel/calendar-events) (First Calendar)

*I couldn’t figure out a way to do this using MaterialCalendarView so I tried using hugomfandrade’s CalendarView Widget below.*

*This MaterialCalendarView is pretty good though, I like the look and the “DotSpan” feature. I would have used this if I could ‘decorate’ with custom decorations, not a single decoration which goes on every ‘shouldDecorate’ calendarDay, as they’ve provided.*

### Issue: “Cannot Access Class ‘org.threeten.bp.Localdate’”

[**Solution**](https://github.com/prolificinteractive/material-calendarview/issues/947)**:**

**Text

Description automatically generated**

## hugoandrade CalendarView Widget

Using hugomfandrade’s [CalendarView Widget](https://github.com/hugomfandrade/CalendarView-Widget), link to [my github repo](https://github.com/emm-an-uel/calendar-events) (Second Calendar)

**settings.gradle**

dependencyResolutionManagement **{** repositoriesMode.set(RepositoriesMode.FAIL\_ON\_PROJECT\_REPOS)  
 repositories **{** google()  
 mavenCentral()  
 maven **{** url 'https://jitpack.io' **}  
 }  
}**

note the *jetpack.io* repository (highlighted above)

**build.gradle (app)**

dependencies **{** // hugomfandrade CalendarView-Widget  
 implementation 'com.github.hugomfandrade:CalendarView-Widget:1.0.1'

**fragment.xml**

<org.hugoandrade.calendarviewlib.CalendarView  
 android:id="@+id/calendarView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />

### Adding Events to Calendar (CalendarObject)

**The goal:** Add events to calendar, with labels similar to Samsung Calendar – will be used in Logit’s CalendarFragment.

|  |  |
| --- | --- |
| **Documentation:** a CalendarObject is added to the CalendarView to represent an Event, as shown alongside (the teal bar below “5”): |  |

public CalendarObject(String id, Calendar datetime, int primaryColor, int secondaryColor) {  
 mID = id;  
 mDatetime = datetime;  
 mPrimaryColor = primaryColor;  
 mSecondaryColor = secondaryColor;  
}

**Fragment.kt**

private fun addEvents() {  
 val calObjectList = *arrayListOf*<CalendarView.CalendarObject>() // resets list to prevent duplicate Events  
 // previously, I looped through each event in events as below and added a CalendarObject each  
 // this caused duplicate events when the fragment is resumed since the CalendarObjects added before were not removed  
 for (event in events) {  
 calObjectList.add(  
 CalendarView.CalendarObject(  
 null,  
 event.date, // where 'date': Calendar  
 ContextCompat.getColor(requireContext(), R.color.*teal\_700*),  
 ContextCompat.getColor(requireContext(), R.color.*teal\_700*)  
 ))  
 }  
 calendarView.setCalendarObjectList(calObjectList)  
}

*Remarks: when this will be used in Logit, I’ll have to define the CalendarObject colors based on the task.subject – I could pass a map of ColorCodes to CalendarFragment and have it look up the subject-color pairs.*

### Editing Events in Calendar (CalendarObject)

**Goal:** User clicks on a date and the app will show a dialog with the events for that day. User can then select and edit events.

[my github repo](https://github.com/emm-an-uel/calendar-events) *Remarks: there’s too much code and too many classes involved to copy and paste all of that into this document. Refer to the code in SecondCalendarFragment if I ever need to make another calendar thing in future.*

#### Code Snapshot: Showing a Calendar Dialog using AlertDialog

**SecondCalendarFragment.kt**

private fun showCalendarDialog(selectedDate: Calendar) {  
 // inflate the view for the calendar dialog  
 mView = View.inflate(requireContext(), R.layout.*calendar\_dialog*, null)  
  
 // set up the ViewPager adapter  
 viewPagerAdapter = PagerAdapter(requireContext(), mapOfEvents, minDate, maxDate, selectedDate)  
  
 val index = ChronoUnit.*DAYS*.between(minDate.toInstant(), selectedDate.toInstant()).toInt() // corresponding index for the current date  
  
 viewPager = mView.findViewById(R.id.*viewPager*)  
 viewPager.*apply* **{** *offscreenPageLimit* = 3  
 *adapter* = viewPagerAdapter  
 *currentItem* = index  
 setPadding(100, 0, 100, 0)  
 **}** pagerSwipeAnimations()  
  
 // display mView in an AlertDialog  
 mAlertDialog = AlertDialog.Builder(requireContext()).create()  
 if (mAlertDialog.*window* != null) {  
 mAlertDialog.*window*!!.setBackgroundDrawable(ColorDrawable(Color.*TRANSPARENT*))  
 }  
 mAlertDialog.*apply* **{** setCanceledOnTouchOutside(true)  
 show()  
 setContentView(mView)  
 **}**}

The layout resource for *calendar\_dialog* is inflated and passed to an AlertDialog, which is displayed in the parent fragment. *Calendar\_dialog* contains a ViewPager, so an adapter is initialized in the parent fragment.

#### Code Snapshot: Creating a Map of <Calendar, List<Event>>

Since each ViewPager item has a RecyclerView displaying the events for the day, the ViewPager Adapter has to pass a list of Events to its RecyclerView Adapter. To determine which list to pass, ViewModel runs through a list of Events and sorts them into a Map with pairs of *key* – Calendar, *value* – list of Events on that date. The code which creates this map is as follows:

**ViewModel**

fun createMapOfEvents() {  
 var list: ArrayList<Event2> = *arrayListOf*()  
 var key: Calendar? = null  
 for (event in events) {  
 if (key != null) { // not the first item in list  
 if (isSameDate(key, event.date)) { // this event is on the same date as the other events in list  
 list.add(event)  
  
 } else { // this event is on a new date  
 mapOfEvents[key] = list // add list of events before this new event  
 key = event.date // set new key  
 list = *arrayListOf*() // reset list  
 list.add(event) // add new event to new list  
 }  
  
 } else { // first item in list  
 key = event.date  
 list.add(event)  
 }  
 }  
 if (key != null && list.*isNotEmpty*()) {  
 mapOfEvents[key] = list // save last-added <Calendar, List> pair  
 }  
}  
  
private fun isSameDate(date1: Calendar, date2: Calendar): Boolean {  
 if (date1.get(Calendar.*DAY\_OF\_MONTH*) != date2.get(Calendar.*DAY\_OF\_MONTH*)) {  
 return false  
 }  
 if (date1.get(Calendar.*MONTH*) != date2.get(Calendar.*MONTH*)) {  
 return false  
 }  
 return date1.get(Calendar.*YEAR*) == date2.get(Calendar.*YEAR*)  
}

Note that *events* is a list of Events, sorted in ascending order according to each event’s *date* value.

Also note the highlighted line – initially I did *list.clear()* but this would result in each saved list in mapOfEvents to be overwritten by the list of events for the latest date. Creating a new list (as I did in the highlighted code) solves this issue and creates the map as planned.

Also note that Calendar objects can’t be compared to each other using *if (date1 == date2)*, functions like *isSameDate*, which compare each consitutent item – day, month, year – have to be used for such comparisons.

## 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

### Setting Minimum Date

[tutorial](https://learntodroid.com/how-to-disable-dates-in-a-datepicker-for-android/)

**Activity.kt**

val today = Calendar.getInstance()  
val todayLong = today.*timeInMillis*dpDueDate.*minDate* = todayLong

dpDueDate is a DatePicker

Note that the attribute *minDate* is a Long, so I need to convert the variable ‘today’ into milliseconds – which is a Long type

## PagerAdapter (ViewPager Swipe Views)

|  |  |
| --- | --- |
| **Final product:** CardViews which can be swiped horizontally as shown alongside. [youtube tutorial](https://www.youtube.com/watch?v=MeG-0MVP3jw), [my github repo](https://github.com/emm-an-uel/SwipeViews-PagerAdapter)  **build.gradle**  dependencies **{** // cardview  implementation 'com.google.android.material:material:1.7.0'  **MainActivity.kt**  viewPager.addOnPageChangeListener(object: ViewPager.OnPageChangeListener {  override fun onPageScrolled(  position: Int,  positionOffset: Float,  positionOffsetPixels: Int  ) {  // do nothing  }   override fun onPageSelected(position: Int) {  // do nothing  }   override fun onPageScrollStateChanged(state: Int) {  // do nothing  } })  adapter = Adapter(this, modelList) viewPager = findViewById(R.id.*viewPager*) viewPager.*adapter* = adapter viewPager.setPadding(100, 0, 100, 0)  *remarks: modelList is an ArrayList of Model class, which has two String parameters – t1 and t2, referenced below in Adapter.kt* |  |

**Adapter.kt**

class Adapter(  
 private val context: Context,  
 private val modelArray: ArrayList<Model>  
) : PagerAdapter() {  
  
 override fun getCount(): Int {  
 return modelArray.size  
 }  
  
 override fun isViewFromObject(view: View, `object`: Any): Boolean {  
 return view == `object`  
 }  
  
 override fun instantiateItem(container: ViewGroup, position: Int): Any {  
 // inflate layout  
 val view = LayoutInflater.from(context).inflate(R.layout.*card\_item*, container, false)  
  
 // get data and populate views   
 ...

// item click  
 view.setOnClickListener **{** val t1 = model.t1  
 Toast.makeText(context, "You clicked on $t1", Toast.*LENGTH\_SHORT*).show()  
 **}** // add view to container  
 container.addView(view, position)  
  
 return view  
 }  
  
 override fun destroyItem(container: ViewGroup, position: Int, `object`: Any) {  
 container.removeView(`object` as View)  
 }  
}

**activity\_main.xml** (note: in this example, the parent layout is a RelativeLayout)

<androidx.viewpager.widget.ViewPager  
 android:id="@+id/viewPager"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:clipToPadding="false"  
 android:foregroundGravity="center"  
 android:overScrollMode="never" />

**card\_item.xml**

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:gravity="center\_vertical"  
 android:orientation="vertical">  
  
 <androidx.cardview.widget.CardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 app:cardCornerRadius="22dp">

*card\_item is just a normal CardView as would be used in a RecyclerView, for example.*

### Adjust Scale and Alpha Of Pages Not In Focus

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

**Adapter.kt**

override fun instantiateItem(container: ViewGroup, position: Int): Any {  
 // inflate layout  
 val view = LayoutInflater.from(context).inflate(R.layout.*card\_item*, container, false)  
 view.*tag* = position // tags each view with its position

...

**MainActivity.kt**

private val MIN\_OFFSET = 0f  
private val MAX\_OFFSET = 0.5f  
private val MIN\_ALPHA = 0.5f  
private val MIN\_SCALE = 0.8f

viewPager.*offscreenPageLimit* = 3 // loads (position + 2) below in time, prevents crash

viewPager.addOnPageChangeListener(object: ViewPager.OnPageChangeListener {  
 override fun onPageScrolled(  
 position: Int,  
 positionOffset: Float,  
 positionOffsetPixels: Int  
 ) {  
 // update view scale and alpha  
 updatePager(viewPager.findViewWithTag(position), 1f - positionOffset) // current page  
 if ((position + 1) < modelList.size) { // next page  
 updatePager(viewPager.findViewWithTag(position + 1), positionOffset)  
 }  
 if ((position + 2) < modelList.size) { // two pages in advance  
 // (so it's already made smaller before user can see it - smoother look)  
 updatePager(viewPager.findViewWithTag(position + 2), 0f)  
 }  
 if ((position - 1) >= 0) { // previous page  
 updatePager(viewPager.findViewWithTag(position - 1), 0f)  
 }  
 }

private fun updatePager(view: View, offset: Float) {  
 var adjustedOffset: Float =  
 (1.0f - 0.0f) \* (offset - MIN\_OFFSET) / (MAX\_OFFSET - MIN\_OFFSET) + 0.0f  
 adjustedOffset = if (adjustedOffset > 1f) 1f else adjustedOffset  
 adjustedOffset = if (adjustedOffset < 0f) 0f else adjustedOffset  
  
 val alpha: Float =  
 adjustedOffset \* (1f - MIN\_ALPHA) + MIN\_ALPHA  
 val scale: Float =  
 adjustedOffset \* (1f - MIN\_SCALE) + MIN\_SCALE  
  
 view.*alpha* = alpha  
 view.*scaleY* = scale  
}

The highlighted code above is needed to load the page at *position + 2*, otherwise it won’t have been loaded before *findViewWithTag* is called on it – returns null and causes crash.

## Kotlin Lists

### Convert Resource Array (r.array) To List

listOfToppings = *resources*.getStringArray(R.array.*listOfToppings*).*toMutableList*()

where ‘listOfToppings’ is a List<String>

### 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*

### Custom AlertDialog

**Fragment.kt**

val builder = AlertDialog.Builder(requireContext()).create()  
if (builder.*window* != null) { // set default background to transparent  
 builder.*window*!!.setBackgroundDrawable(ColorDrawable(Color.*TRANSPARENT*))  
}  
val view = *layoutInflater*.inflate(R.layout.*confirm\_delete\_all\_dialog*, null, false)  
val btnCancel: Button = view.findViewById(R.id.*btnCancel*)  
val btnConfirm: Button = view.findViewById(R.id.*btnConfirm*)  
  
btnCancel.setOnClickListener **{** builder.dismiss()  
**}**btnConfirm.setOnClickListener **{** deleteAll()  
 builder.dismiss()  
**}**builder.*apply* **{** setView(view)  
 setCanceledOnTouchOutside(true)  
 show()  
**}**

*Note: R.layout.confirm\_delete\_all\_dialog is a layout xml file containing a CardView with static TextViews and two Buttons.*

## 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

**build.gradle (app)**

dependencies **{** // klaxon  
 implementation 'com.beust:klaxon:5.5'

### 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

### Issue: Expected a [ but read {

|  |  |
| --- | --- |
| **Issue description** | I tried saving a list of Data<String, Int> objects, but the app would crash when I tried reading it, and throw this error message: *com.beust.klaxon.Klaxon Exception: Expected a [ but read {* |
| **Solution** | I had to convert Data class into a *parcelable* object (refer to notes above on *parcelize*) |

## 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)