Contents

[Setting Parent Activity (For Up Arrows) 2](#_Toc116647966)

[EditText 2](#_Toc116647967)

[Removing The Underline 2](#_Toc116647968)

[TextView 2](#_Toc116647969)

[Set TextView Margins Dynamically 2](#_Toc116647970)

[Set Text Color Dynamically 2](#_Toc116647971)

[ImageView 3](#_Toc116647972)

[Setting The Color of an ImageView 3](#_Toc116647973)

[Android Studio 3](#_Toc116647974)

[Issue: Can’t Read R.() Files 3](#_Toc116647975)

[Spinner (Drop Down Selection) 3](#_Toc116647976)

[Dynamically created spinner 3](#_Toc116647977)

[Statically created spinner 4](#_Toc116647978)

[Custom Spinner Adapter 4](#_Toc116647979)

[Parcelize (Creating A Parcelable OBject) 5](#_Toc116647980)

[Maps 5](#_Toc116647981)

[Finding If a Key Exists 5](#_Toc116647982)

[For Loops 5](#_Toc116647983)

[Generate a View Id 5](#_Toc116647984)

[ActionBar 5](#_Toc116647985)

[Random Number Generator 6](#_Toc116647986)

[Table Layout Formatting 6](#_Toc116647987)

[Recycler View 7](#_Toc116647988)

[Creating Recycler View 7](#_Toc116647989)

[Creating Recycler View Items (CardView) 7](#_Toc116647990)

[Getting Context Within a RecyclerView Adapter 9](#_Toc116647991)

[Issue: Recycler View Doesn’t Show Updated List Contents 9](#_Toc116647992)

[Issue: Recycler View Not Showing Contents 10](#_Toc116647993)

[issue: Crash When Removing a “Completed Task” 10](#_Toc116647994)

[ViewPager2 and TabLayout 11](#_Toc116647995)

[Using “Add On Tab Selected Listener” 11](#_Toc116647996)

[Setting Button Clickability Dynamically 11](#_Toc116647997)

[Using TextWatcher 12](#_Toc116647998)

[Using NumberPicker 12](#_Toc116647999)

[Setting a Calendar Date (DatePicker) 12](#_Toc116648000)

[Kotlin Lists 12](#_Toc116648001)

[Combining Lists 12](#_Toc116648002)

[Check if All Elements in a List Are Equal 12](#_Toc116648003)

[Sum of a List of Integers 12](#_Toc116648004)

[Convert String to List 13](#_Toc116648005)

[Kotlin Strings 13](#_Toc116648006)

[Trim a String 13](#_Toc116648007)

[Insert Characters Into String 13](#_Toc116648008)

[Send Data / Passing Values 13](#_Toc116648009)

[Passing Data Using Bundle (Activity to Activity) 13](#_Toc116648010)

[Passing Data Using Bundle (Activity to Fragment) 14](#_Toc116648011)

[Passing Data Using FragmentManager (Fragment Result API) 14](#_Toc116648012)

[Dialog Fragments (eg. Popup Rating System) 15](#_Toc116648013)

[Referencing R.attr.(color) 15](#_Toc116648014)

[Json / Klaxon 15](#_Toc116648015)

[Saving a Json File Using Klaxon 15](#_Toc116648016)

[Reading a Json File Using Klaxon 16](#_Toc116648017)

[Adding Custom Icons (Drawables) 16](#_Toc116648018)

## Setting Parent Activity (For Up Arrows)

**AndroidManifest.xml**

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

*Note the “parentActivityName” attribute*

## EditText

### Setting Text of an Edit Text

**Activity.kt**

editText.setText(string)

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

### Removing The Underline

**activity.xml**

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

*Above code is an attribute of the EditText*

## TextView

### Revert To Default TextView Color

**Activity.kt**

val defaultColor = textView.*currentTextColor*

*// change text color to something else*

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

### Set TextView Margins Dynamically

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

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

### Set Text 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.*

### Custom Spinner Adapter

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

**ColorCode.kt**

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

**MainActivity.kt**

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

**SpinnerAdapter.kt**

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

### Set Selected Item of a Spinner

spinnerObject.setSelection(INDEX)

## Parcelize (Creating A Parcelable OBject)

Build.gradle (app):

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

Kotlin class:

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

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

## Maps

### Finding If a Key Exists

map.containsKey(key)

### For Loops

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

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

## Generate a View Id

etSubject.*id* = View.generateViewId()

## ActionBar

### Creating an Action Bar

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

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

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

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

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

### Status Bar Icons Color

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

## Random Number Generator

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

## Table Layout Formatting

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

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

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

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

## Recycler View

### Creating Recycler View

**activity.xml**

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

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

### Creating Recycler View Items (CardView)

**task\_rv\_item.xml**

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

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

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

**RVAdapter.kt**

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

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

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

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

**Fragment.kt** – swipe functions

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

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

**Fragment.kt** – item click listener

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

### 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*

### 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*

### 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  
}

### 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()

      }

    }

}

### Getting Context In Text Watcher

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

**SettingsRVAdapter.kt (main section)**

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

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

**SettingsRVAdapter.kt (textWatcher section)**

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

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

## Using NumberPicker

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

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

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

*(Getting value from numberPicker)*

val numDice = numberPicker.*value*

## Setting a Calendar Date (DatePicker)

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

## Kotlin Lists

### Check If An Element Is In a List

if (list.contains(element)) {

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

} else {

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

}

### Combining Lists

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

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

### Check For Duplicates In a List

[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  
 }  
}

### 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).*

## Dialog Fragments (eg. Popup Rating System)

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

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

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

**Activity.kt**

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

**ColorDialogFragment.kt**

import androidx.fragment.app.DialogFragment

class ColorDialogFragment : DialogFragment()

*Note: “DialogFragment” class*

### Alert Dialog

**Fragment.kt**

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

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

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

## Referencing R.attr.(color)

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

**Implementation**

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

## Json / Klaxon

### Saving a Json File Using Klaxon

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

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

### Reading a Json File Using Klaxon

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

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

### Check If a File Exists

**Activity.kt**

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

// do something

## Adding Custom Icons (Drawables)

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

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