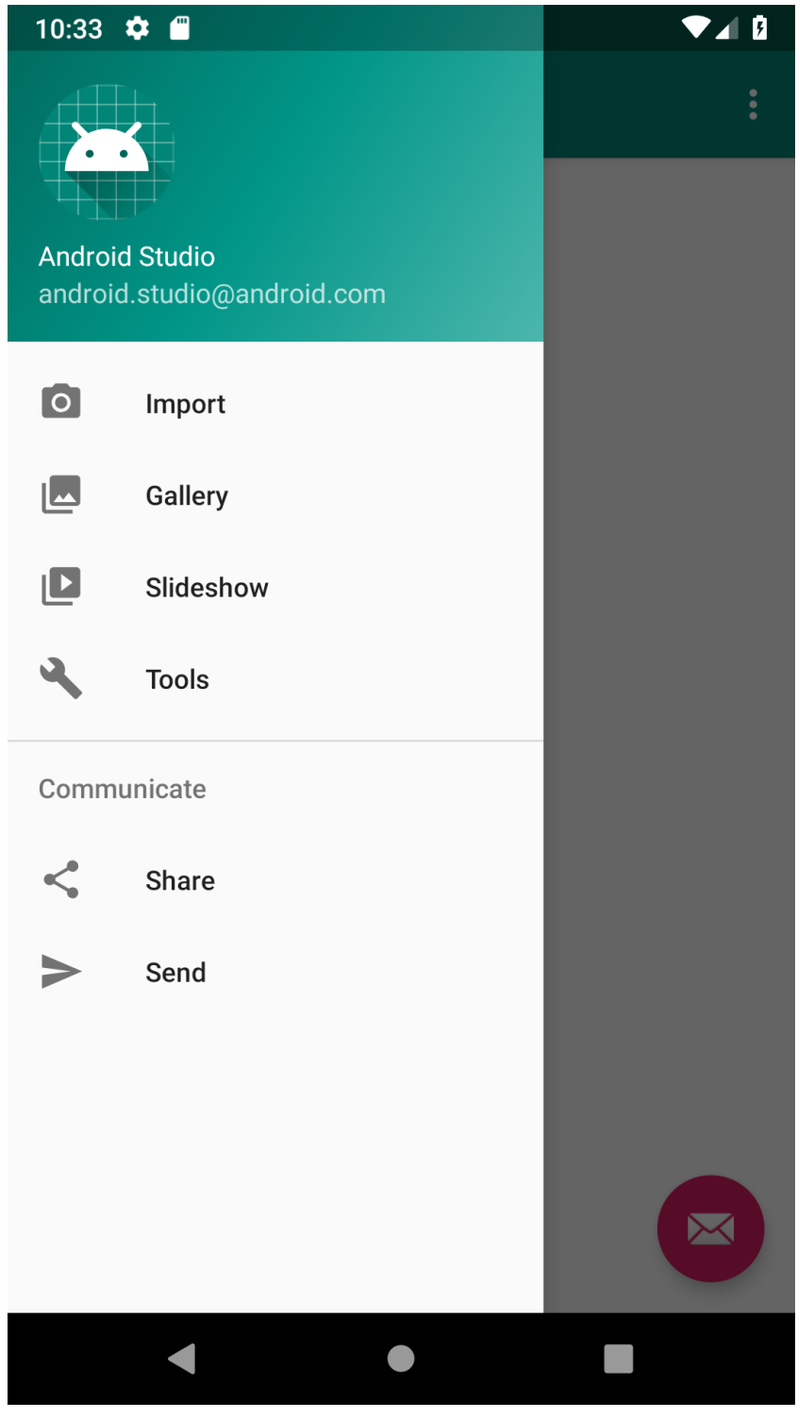
Lecture #3
Navigation and Rest Resources

Mobile Applications 2018-2019

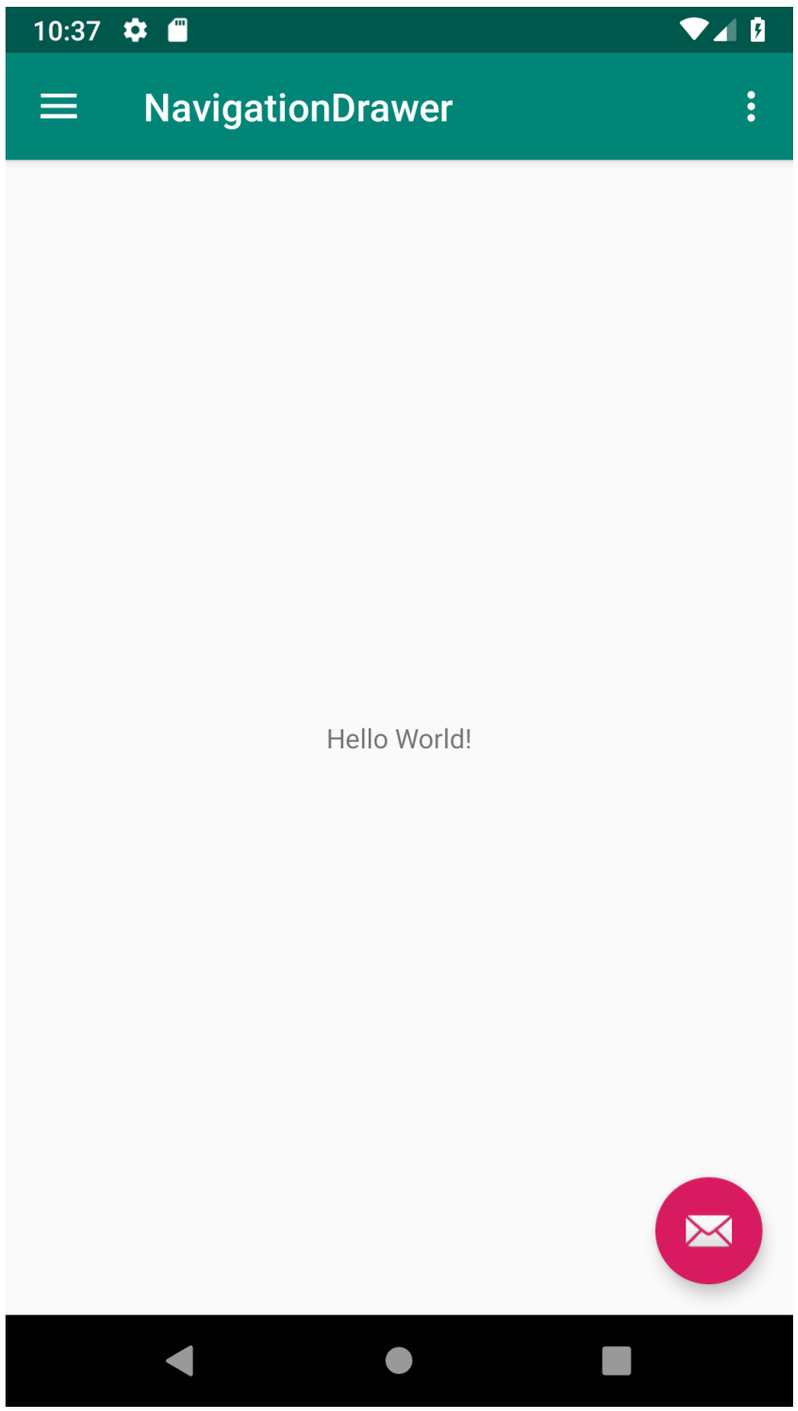
Navigation Drawer

- App main navigation menu.
- Hidden when not in use.
- Appears:
  - with a left swipe from the screen edge
  - when the user touches the drawer icon in the app bar



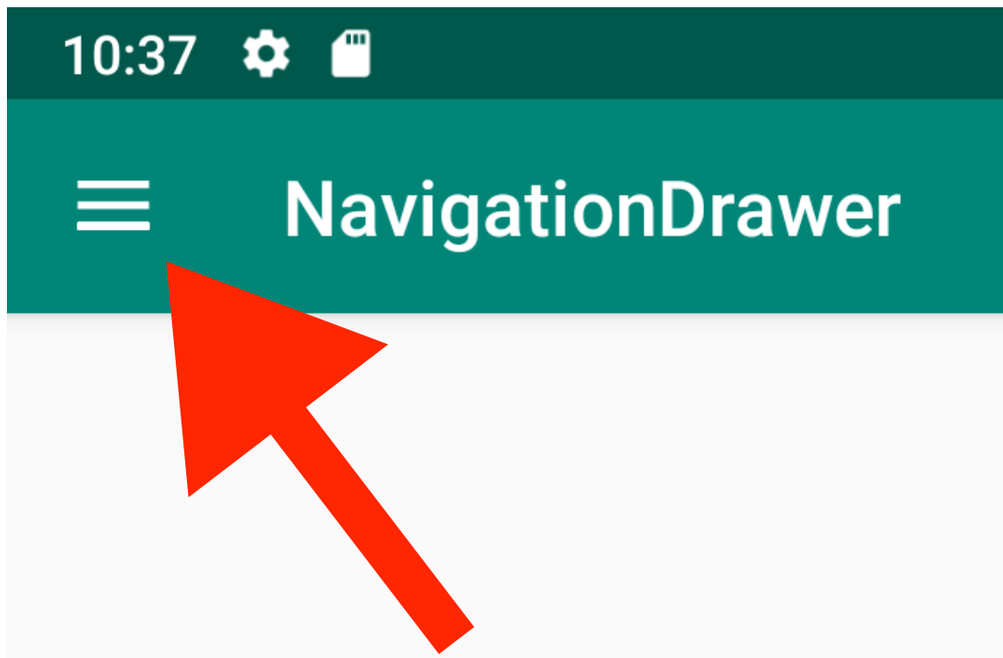
Navigation Drawer

- App main navigation menu.
- Hidden when not in use.
- Appears:
  - with a left swipe from the screen edge
  - when the user touches the drawer icon in the app bar



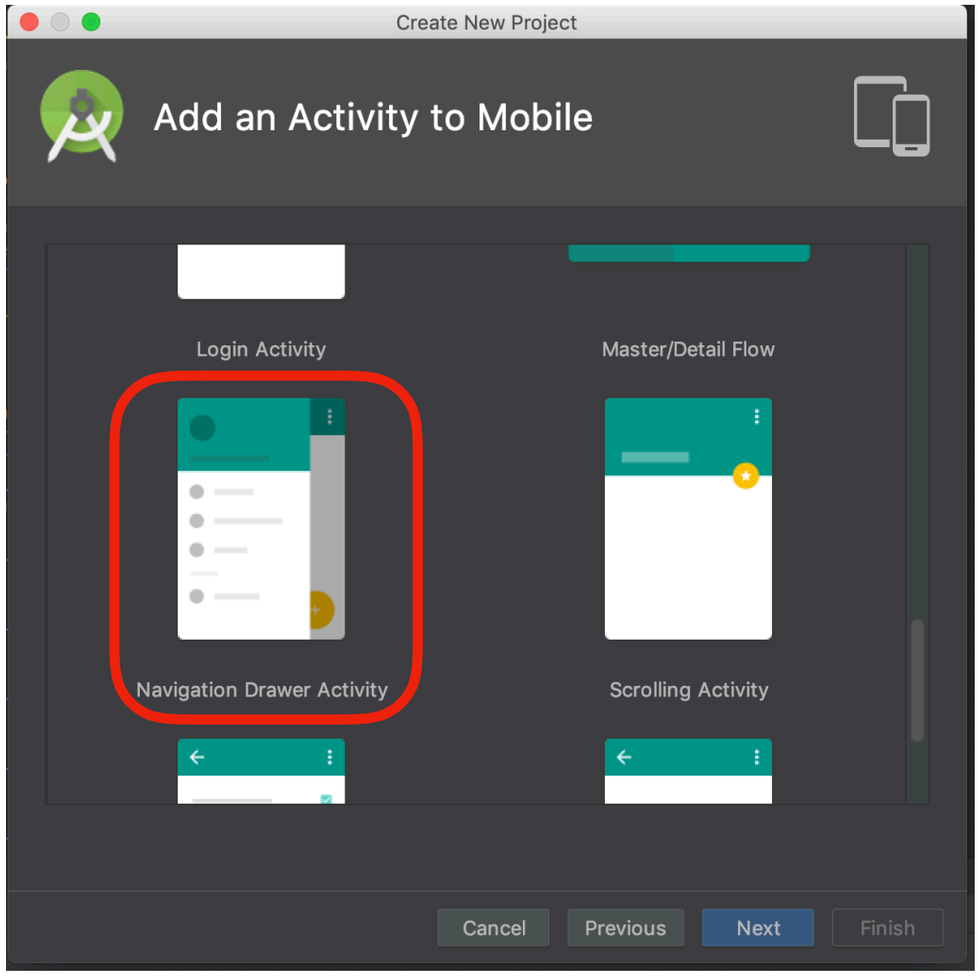
Navigation Drawer

- App main navigation menu.
- Hidden when not in use.
- Appears:
  - with a left swipe from the screen edge
  - when the user touches the drawer icon in the app bar



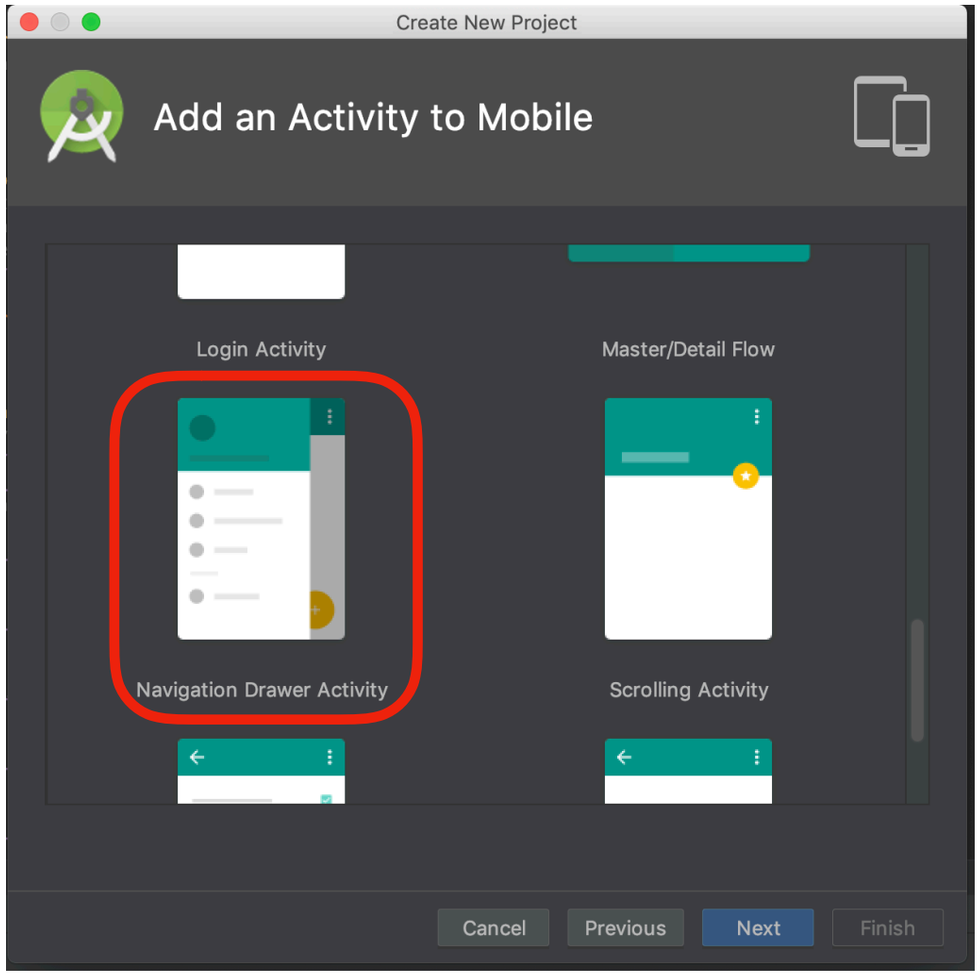
https://developer.android.com/training/implementing-navigation/nav-drawer

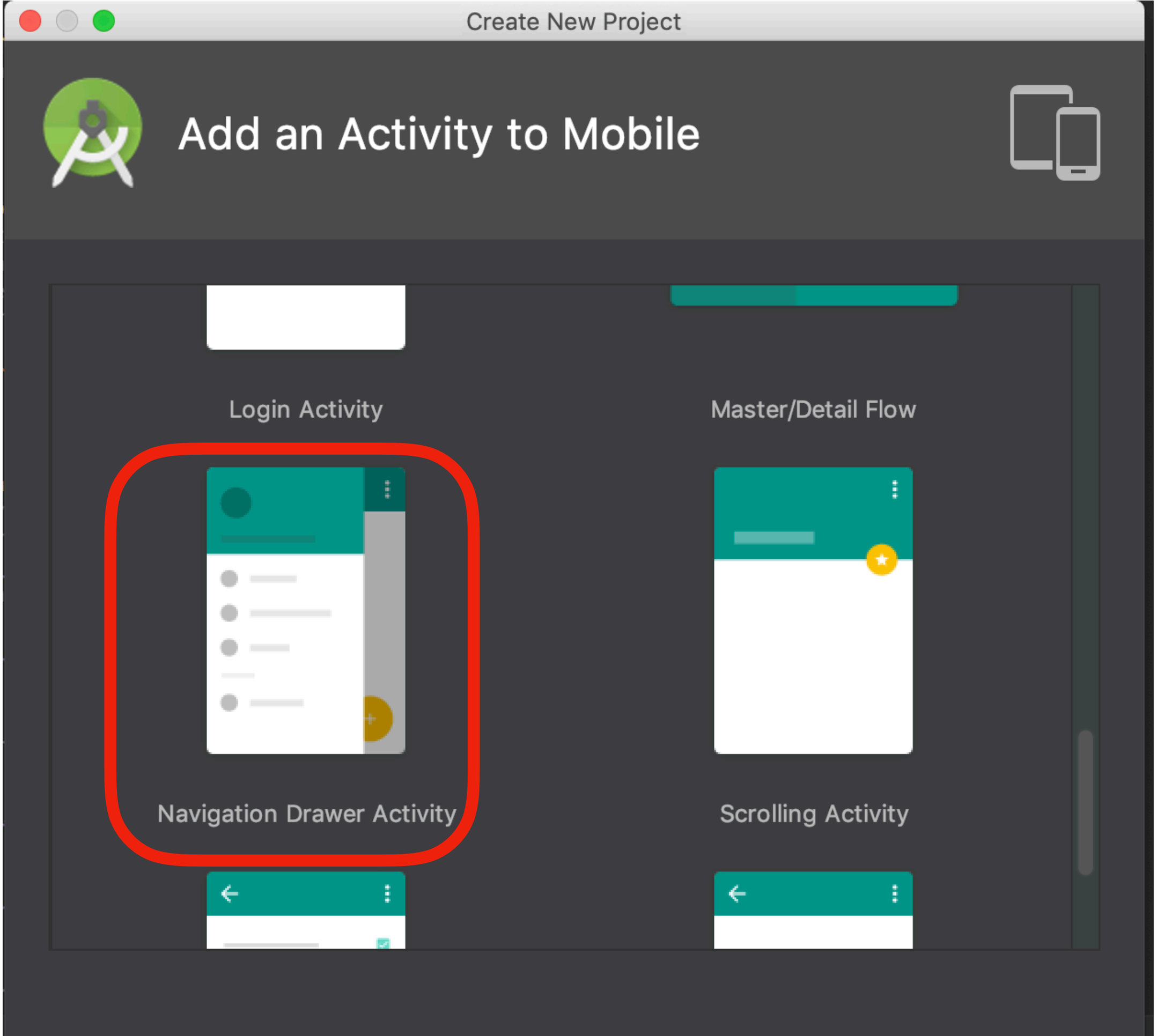
Create Options



Create Options

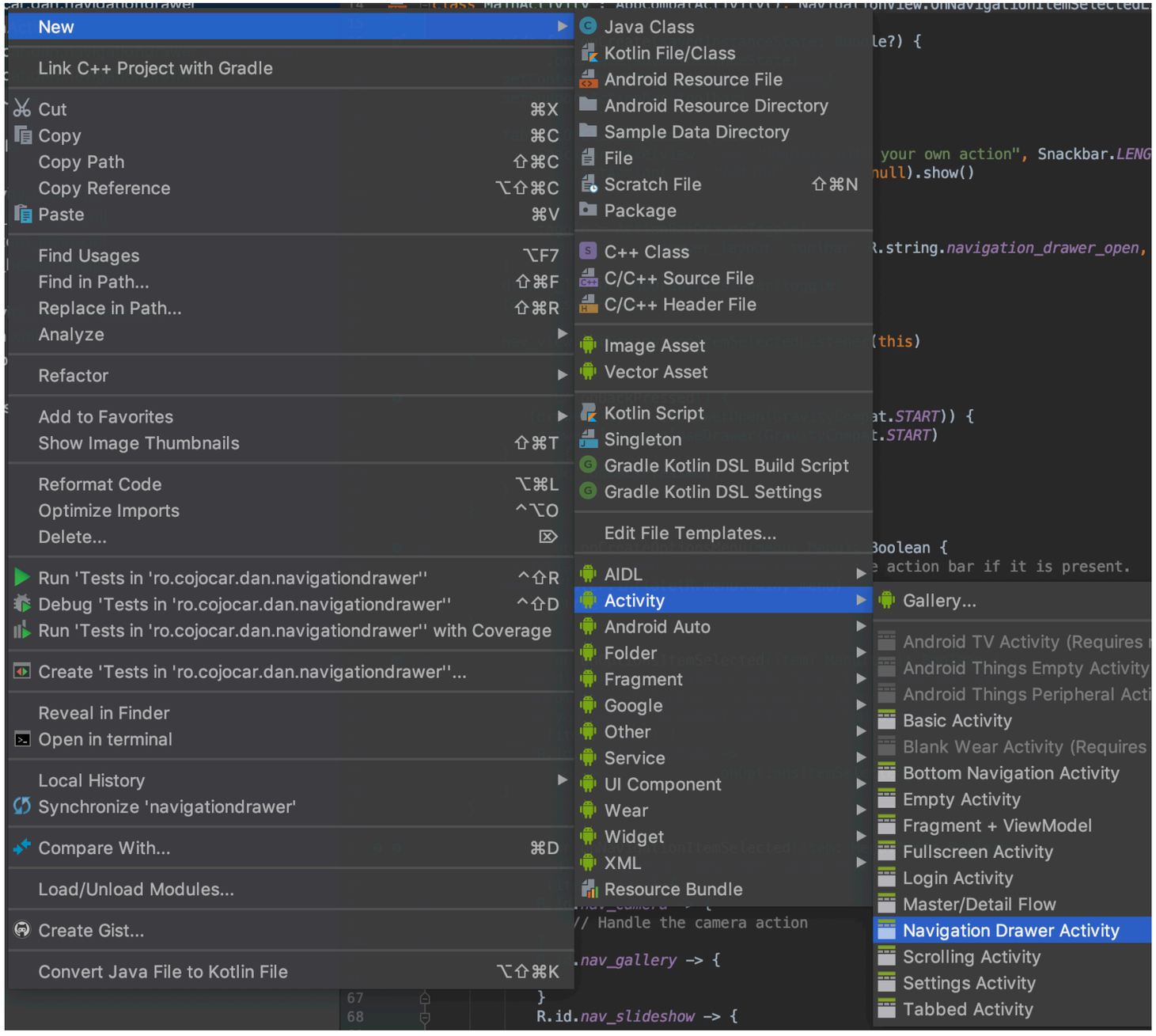
- New app wizard.

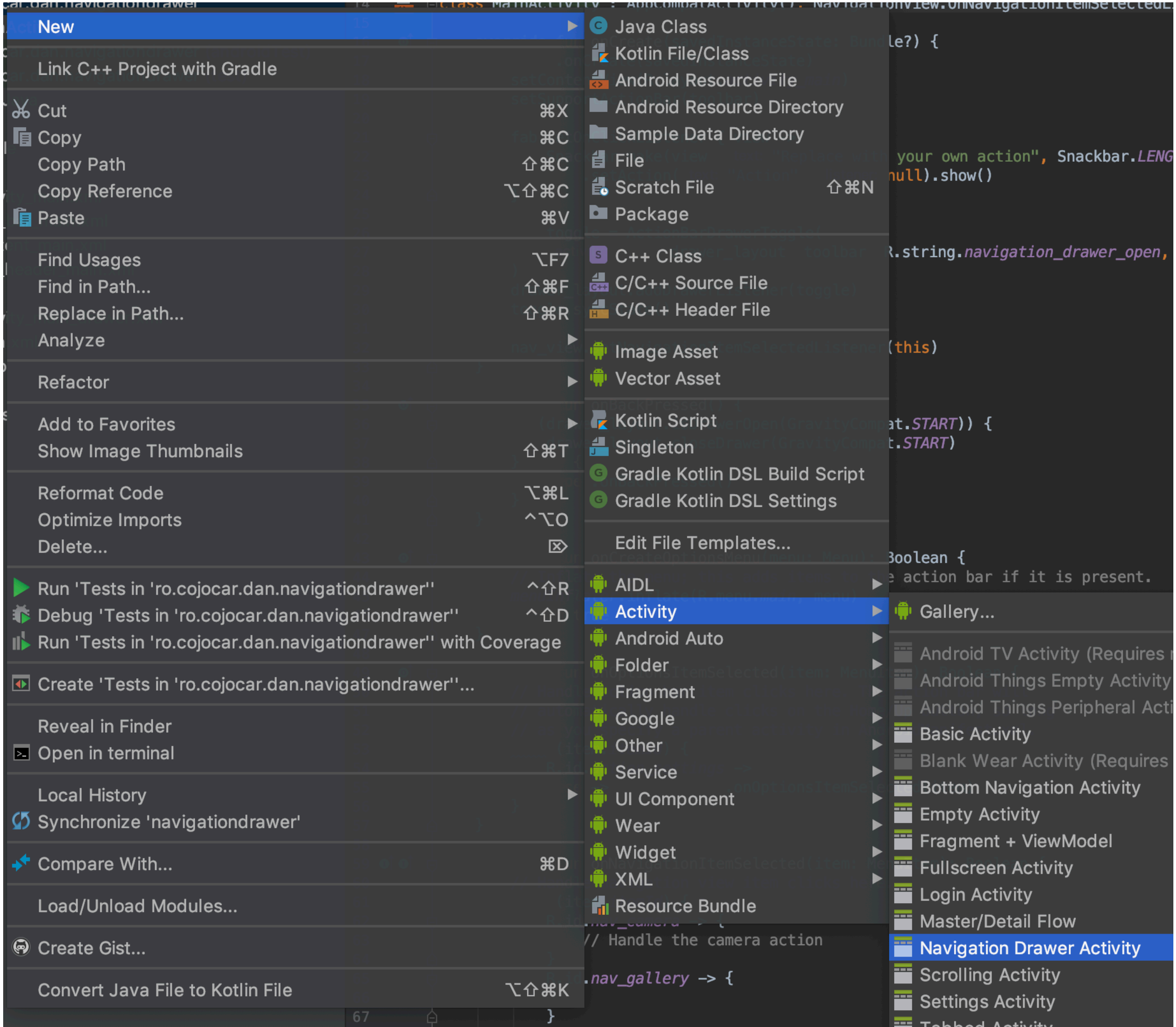



Create New Project
Add an Activity to Mobile
Login Activity
Master/Detail Flow
Navigation Drawer Activity
Scrolling Activity

Create Options

- New app wizard.
- New activity wizard.

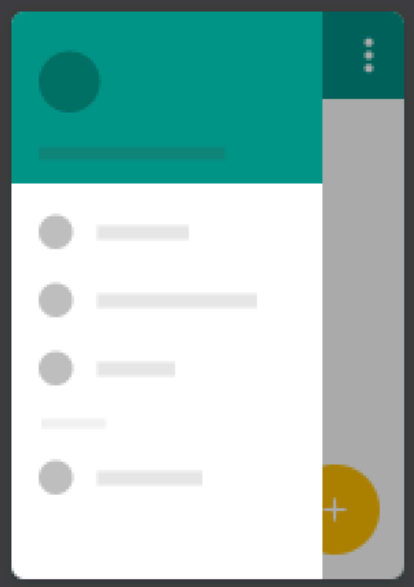



New
Link C++ Project with Gradle
Cut ⌘X
Copy ⌘C
Copy Path ⇧⌘C
Copy Reference ⌥⇧⌘C
Paste ⌘V
Find Usages ⌥F7
Find in Path... ⇧⌘F
Replace in Path... ⇧⌘R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails ⇧⌘T
Reformat Code ⌥⌘L
Optimize Imports ^⌥O
Delete...
Run 'Tests in 'ro.cojocar.dan.navigationdrawer'' ^⇧R
Debug 'Tests in 'ro.cojocar.dan.navigationdrawer'' ^⇧D
Run 'Tests in 'ro.cojocar.dan.navigationdrawer'' with Coverage
Create 'Tests in 'ro.cojocar.dan.navigationdrawer''...
Reveal in Finder
Open in terminal
Local History
Synchronize 'navigationdrawer'
Compare With... ⌘D
Load/Unload Modules...
Create Gist...
Convert Java File to Kotlin File ⌥⇧⌘K
Java Class
Kotlin File/Class
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File ⇧⌘N
Package
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Kotlin Script
Singleton
Gradle Kotlin DSL Build Script
Gradle Kotlin DSL Settings
Edit File Templates...
AIDL
Activity
Android Auto
Folder
Fragment
Google
Other
Service
UI Component
Wear
Widget
XML
Resource Bundle
Gallery...
Android TV Activity (Requires
Android Things Empty Activity
Android Things Peripheral Acti
Basic Activity
Blank Wear Activity (Requires
Bottom Navigation Activity
Empty Activity
Fragment + ViewModel
Fullscreen Activity
Login Activity
Master/Detail Flow
Navigation Drawer Activity
Scrolling Activity
Settings Activity

New Android Activity

Configure Activity

Android Studio



Creates a new Activity with a Navigation Drawer.

Activity Name: MainActivity

Layout Name: activity_main

Title: MainActivity

Launcher Activity

Hierarchical Parent: ...

Package name: ro.cojocar.dan.myapplication

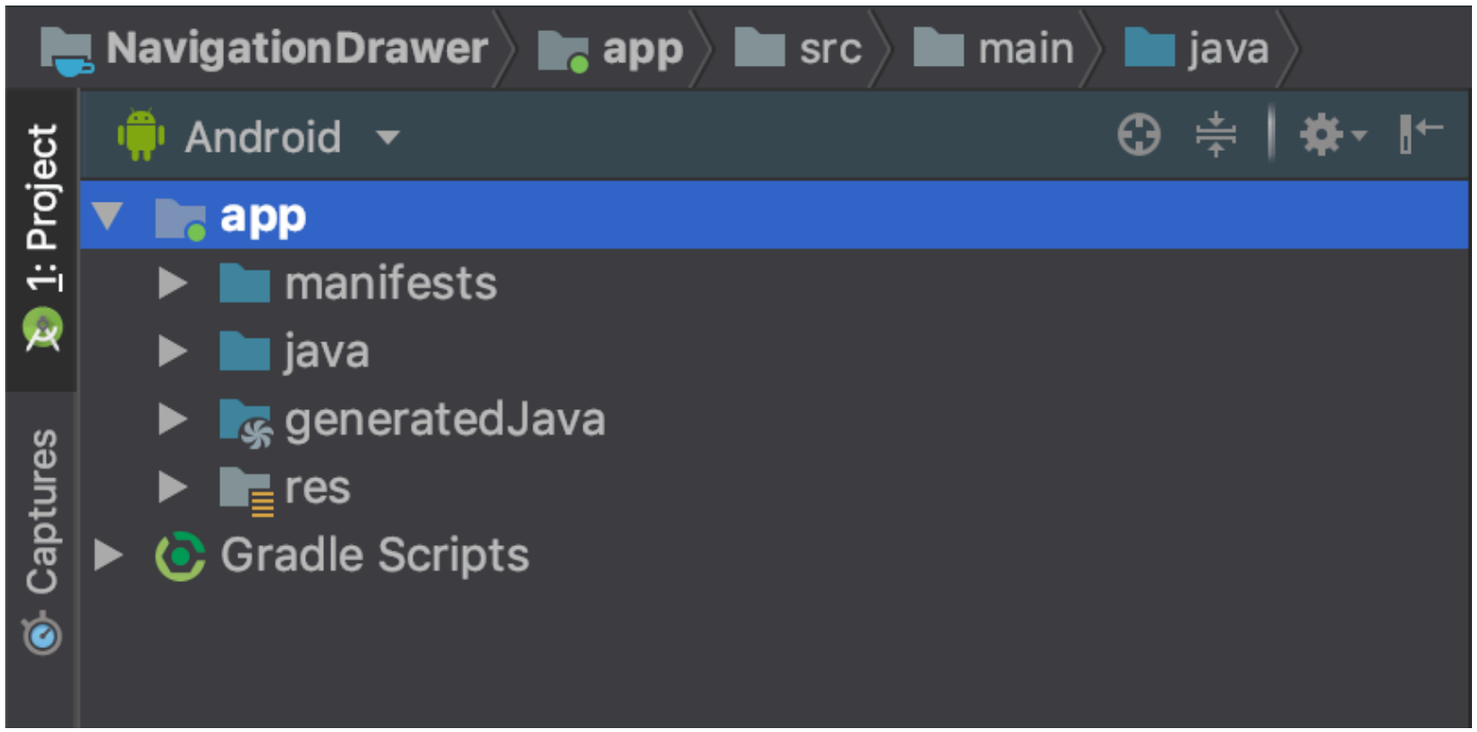
Source Language: Kotlin

The name of the activity class to create

Cancel | Previous | Next | Finish

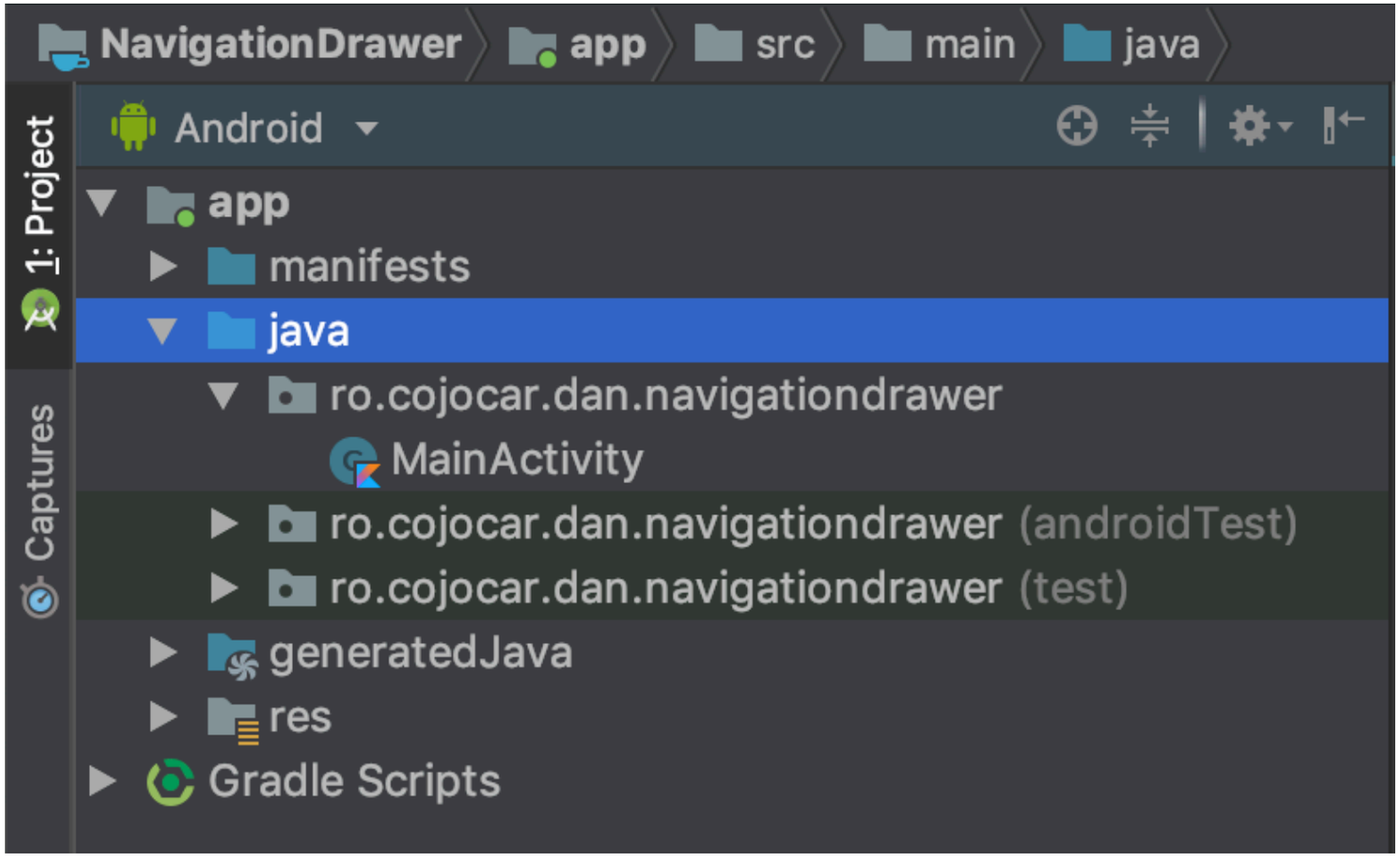
Generated Artifacts

Generated Artifacts



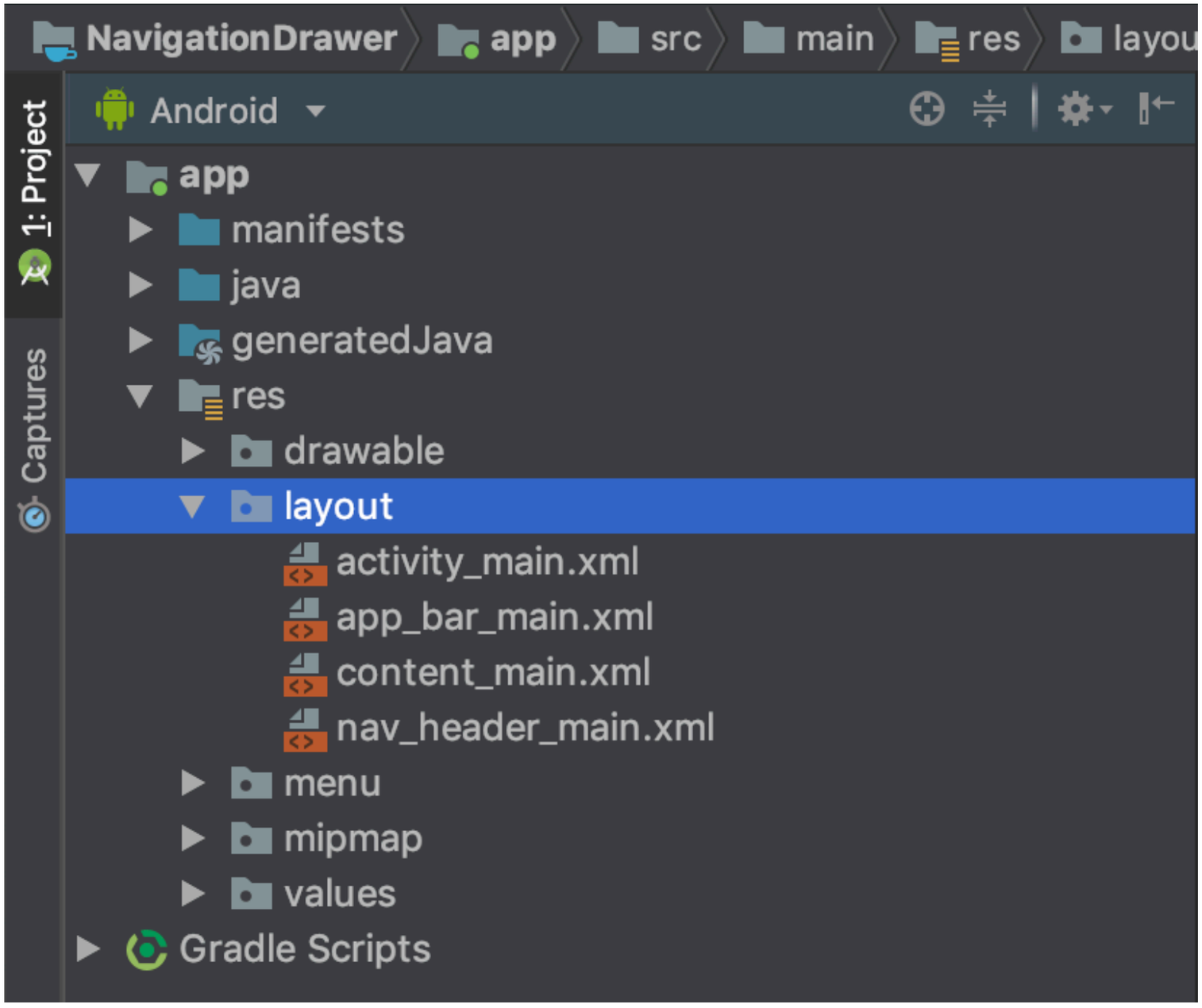
Generated Artifacts

- Sources



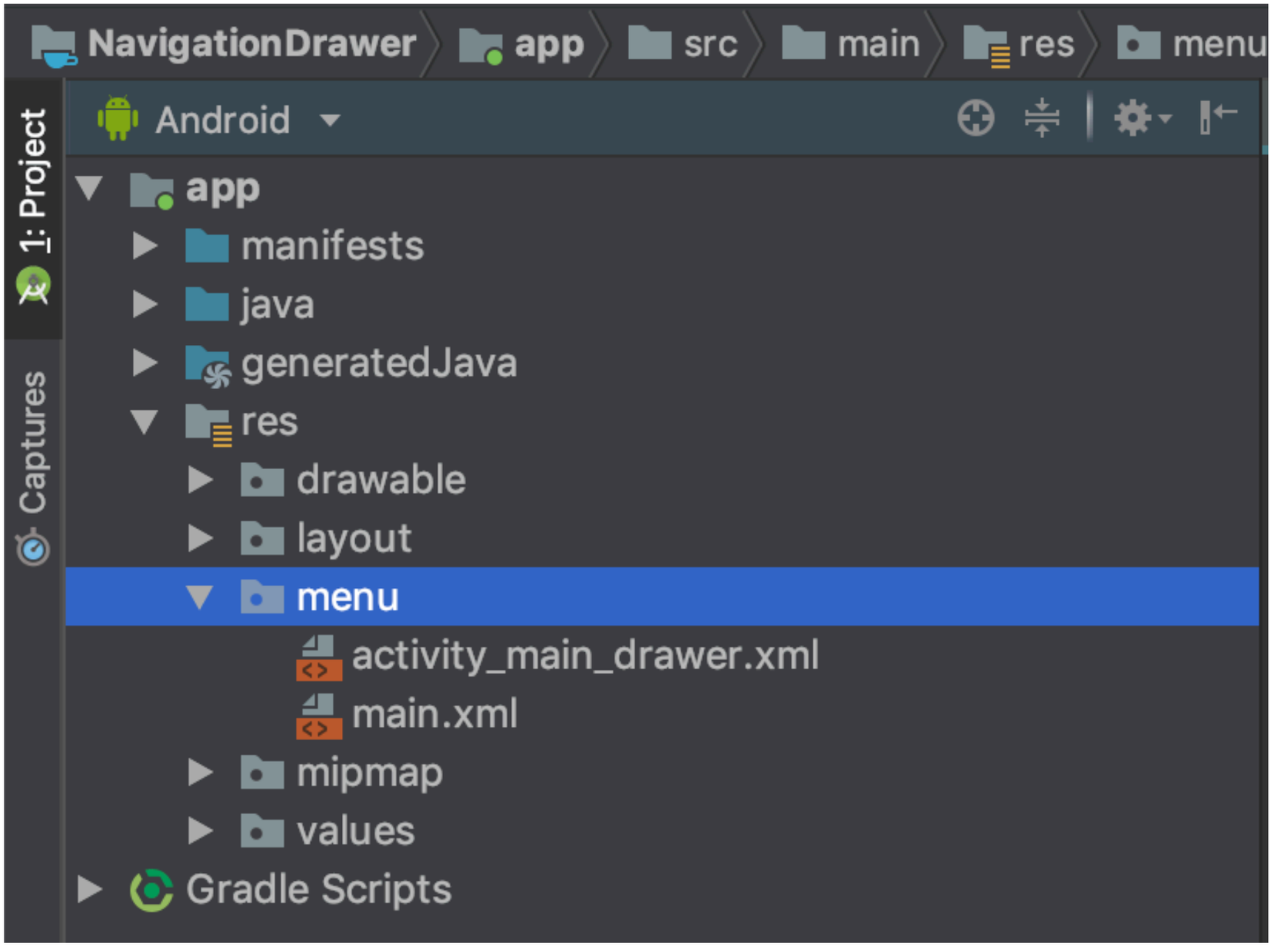
Generated Artifacts

- Sources
- Layouts



Generated Artifacts

- Sources
- Layouts
- Menus



Dependencies

```
dependencies {
  implementation 'com.android.support:appcompat-v7:28.0.0'
  implementation 'com.android.support:design:28.0.0'
}
```

Add a drawer to a layout

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Use DrawerLayout as root container for activity -->
<android.support.v4.widget.DrawerLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true">

    <!-- Layout to contain contents of main body of screen
         (drawer will slide over this) -->
    <FrameLayout
        android:id="@+id/content_frame"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!-- Container for contents of drawer
          - use NavigationView to make configuration easier -->
    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true" />

</android.support.v4.widget.DrawerLayout>
```

Add a drawer to a layout

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Use DrawerLayout as root container for activity -->
<android.support.v4.widget.DrawerLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true">

    <!-- Layout to contain contents of main body of screen
         (drawer will slide over this) -->
    <FrameLayout
        android:id="@+id/content_frame"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!-- Container for contents of drawer
          - use NavigationView to make configuration easier -->
    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true" />

</android.support.v4.widget.DrawerLayout>
```

Add a drawer to a layout

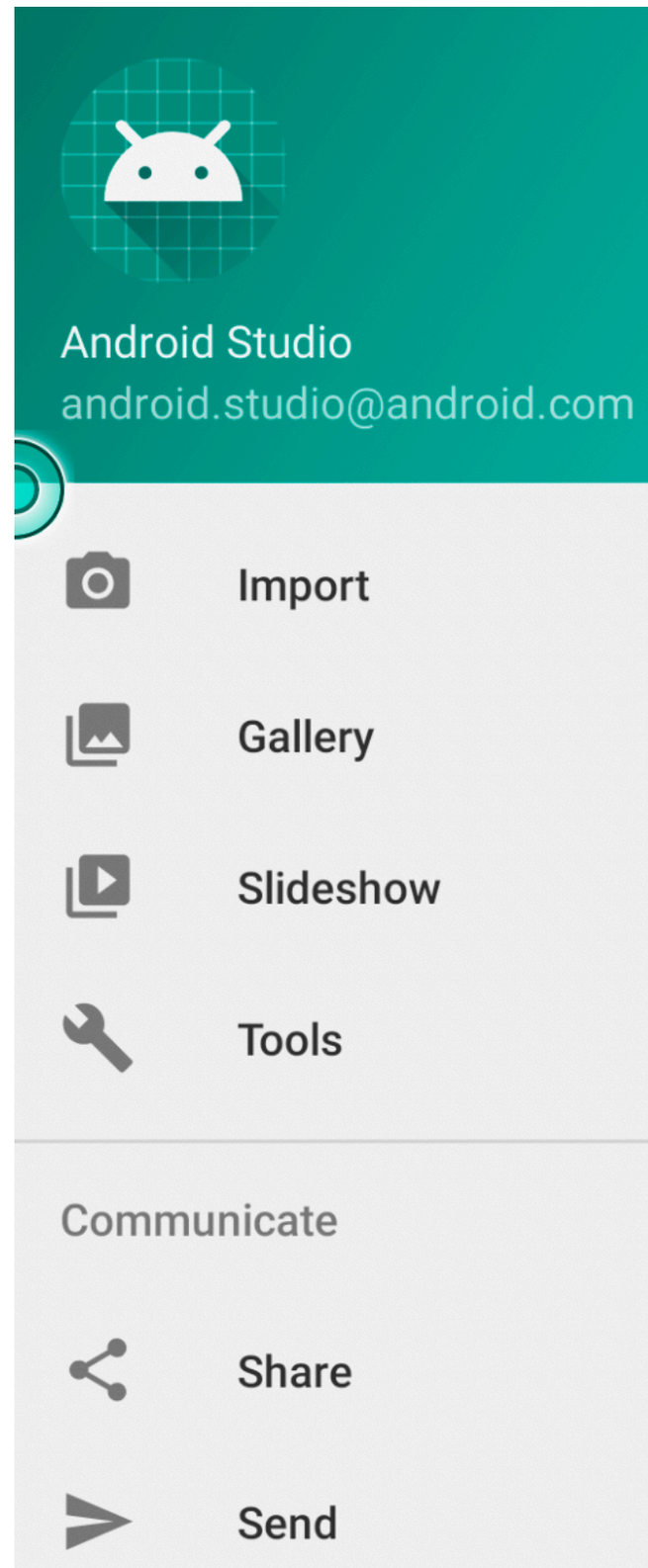
```
<?xml version="1.0" encoding="utf-8"?>
<!-- Use DrawerLayout as root container for activity -->
<android.support.v4.widget.DrawerLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true">

    <!-- Layout to contain contents of main body of screen
         (drawer will slide over this) -->
    <FrameLayout
        android:id="@+id/content_frame"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!-- Container for contents of drawer
          - use NavigationView to make configuration easier -->
    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true" />

</android.support.v4.widget.DrawerLayout>
```

Add a drawer to a layout

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Use DrawerLayout as root container for activity -->
<android.support.v4.widget.DrawerLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true">

    <!-- Layout to contain contents of main body of screen
         (drawer will slide over this) -->
    <FrameLayout
        android:id="@+id/content_frame"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!-- Container for contents of drawer
          - use NavigationView to make configuration easier -->
    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true" />

</android.support.v4.widget.DrawerLayout>
```

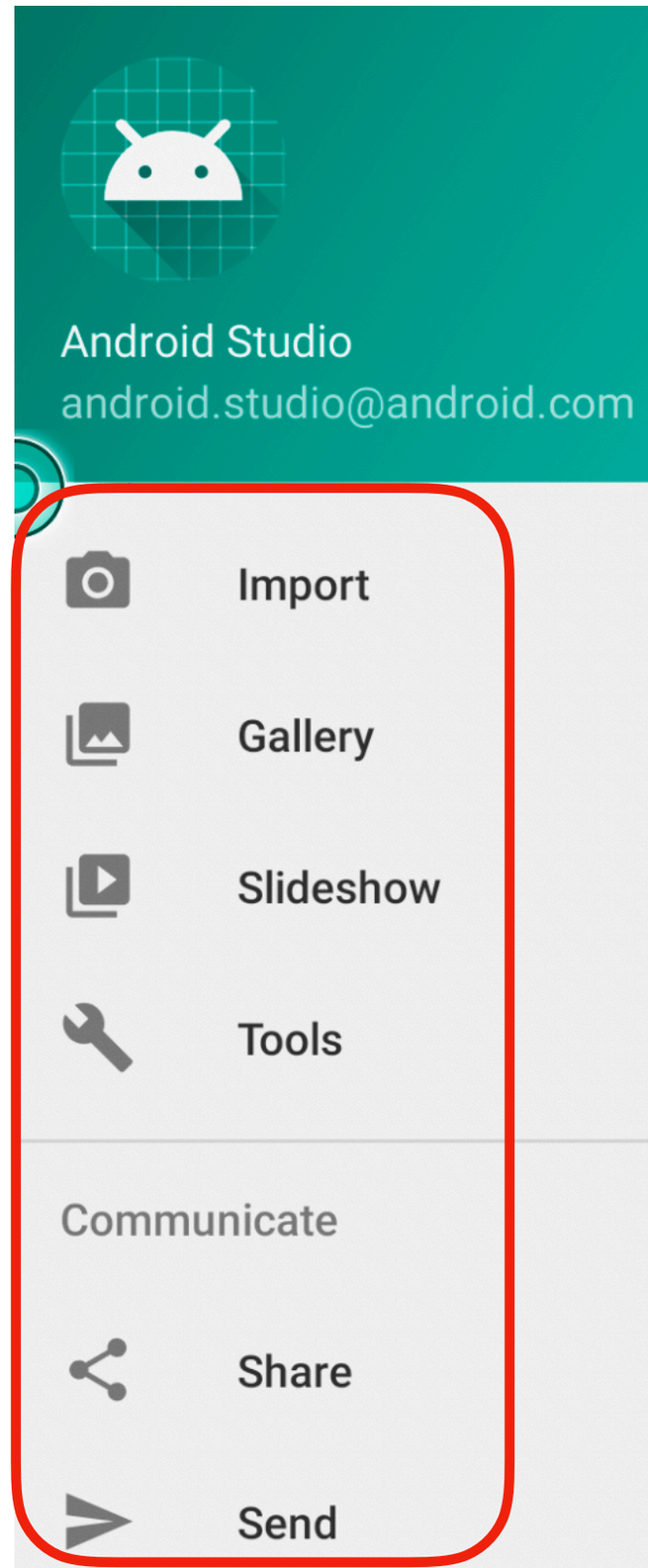
Declare the menu items

```
<android.support.design.widget.NavigationView
android:id="@+id/nav_view"
android:layout_width="wrap_content"
android:layout_height="match_parent"
android:layout_gravity="start"
android:fitsSystemWindows="true"
```



Declare the menu items

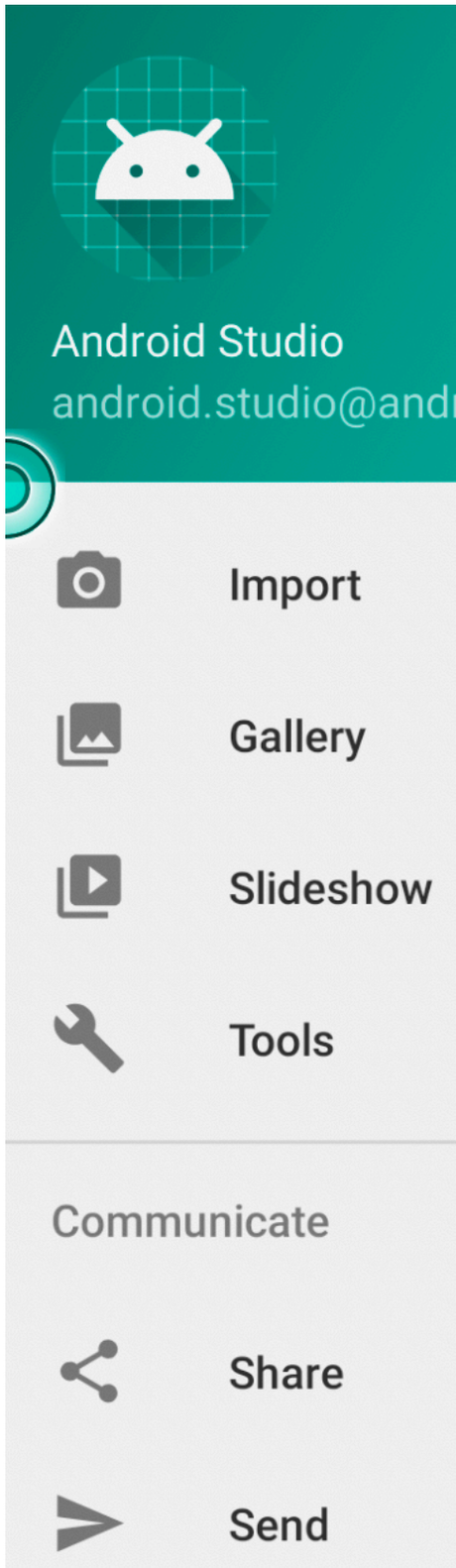
```
<android.support.design.widget.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:menu="@menu/drawer_view" />
```



Declare the menu items

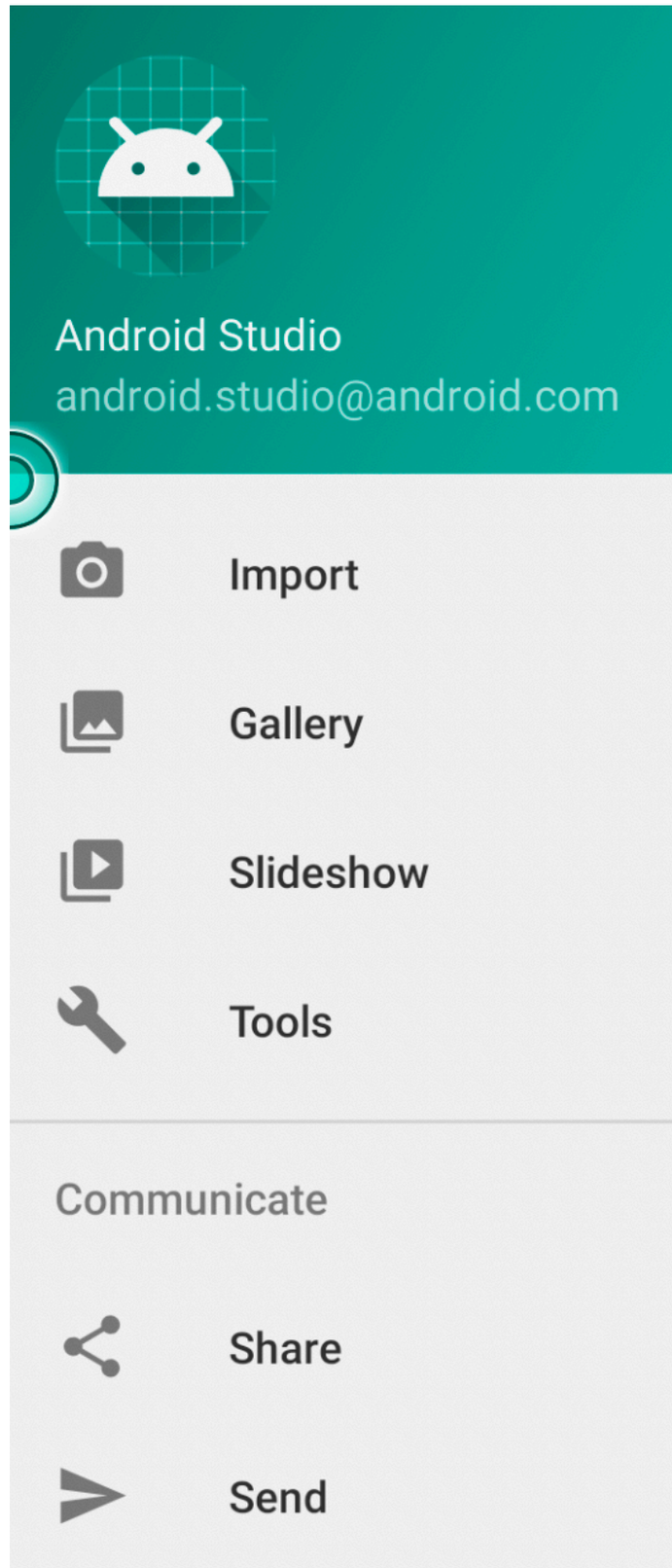
```
<android.support.design.widget.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:menu="@menu/drawer_view" />
```

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_camera"
            android:icon="@drawable/ic_menu_camera"
            android:title="@string/import" />
        <item
            android:id="@+id/nav_gallery"
            android:icon="@drawable/ic_menu_gallery"
            android:title="@string/gallery" />
        <item
            android:id="@+id/nav_slideshow"
            android:icon="@drawable/ic_menu_slideshow"
            android:title="@string/slideshow" />
        ...
    </group>
</menu>
```



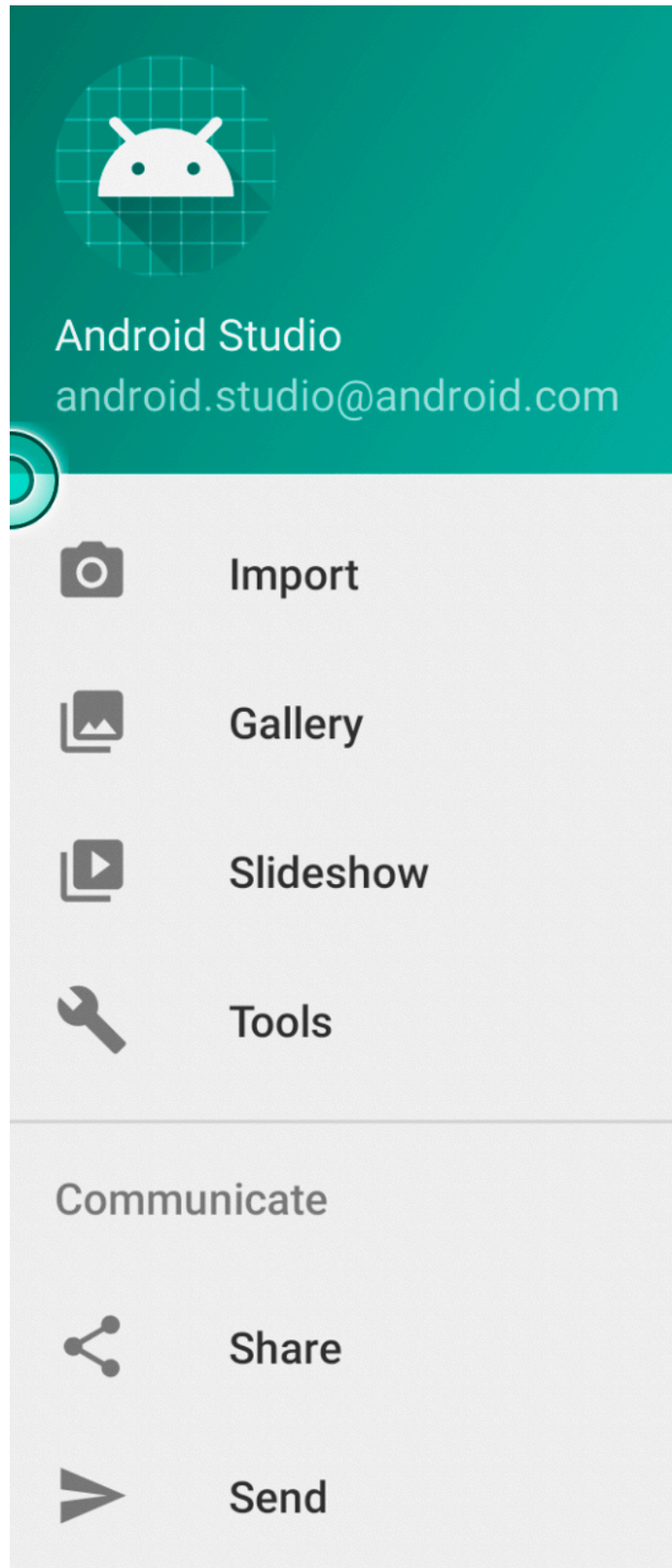
Add a header to the nav drawer

```
<android.support.design.widget.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:menu="@menu/drawer_view" />
```



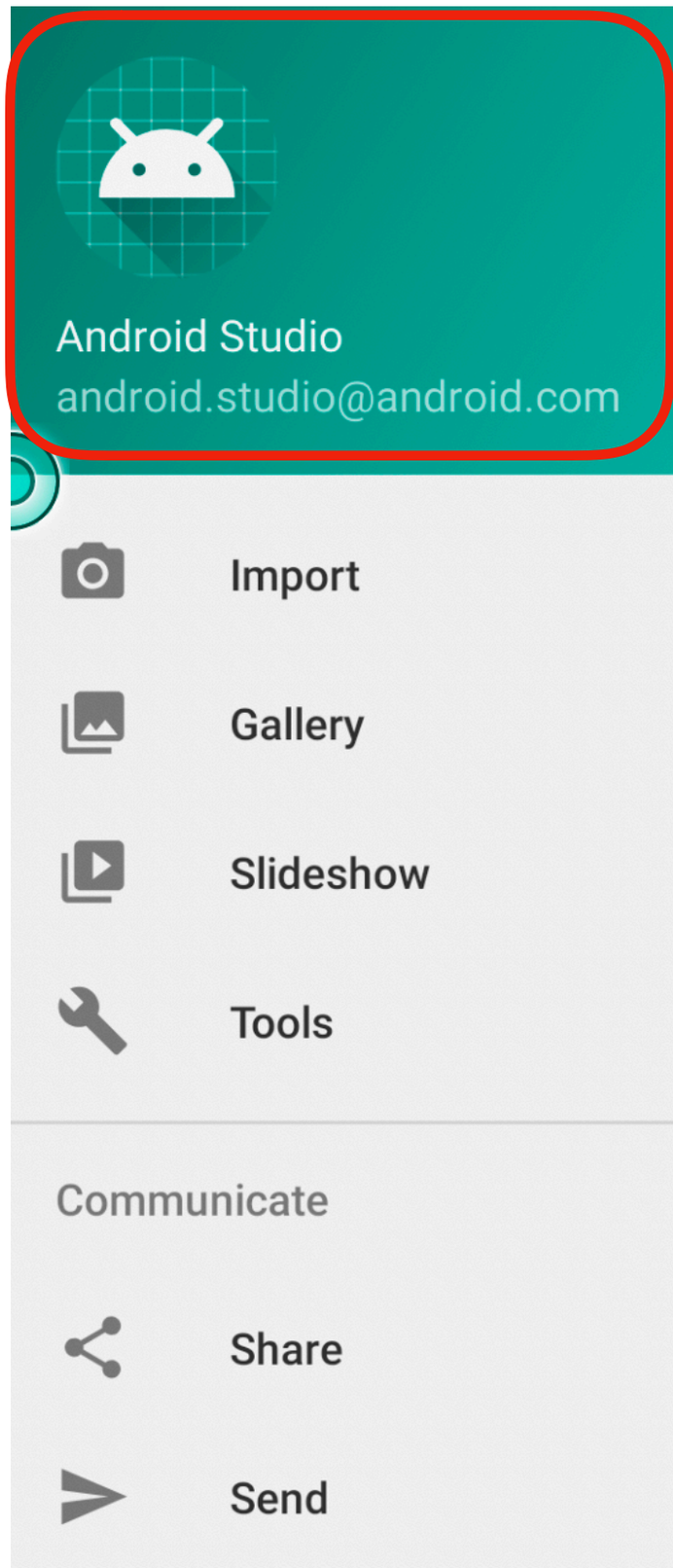
Add a header to the nav drawer

```
<android.support.design.widget.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:menu="@menu/drawer_view"
```



Add a header to the nav drawer

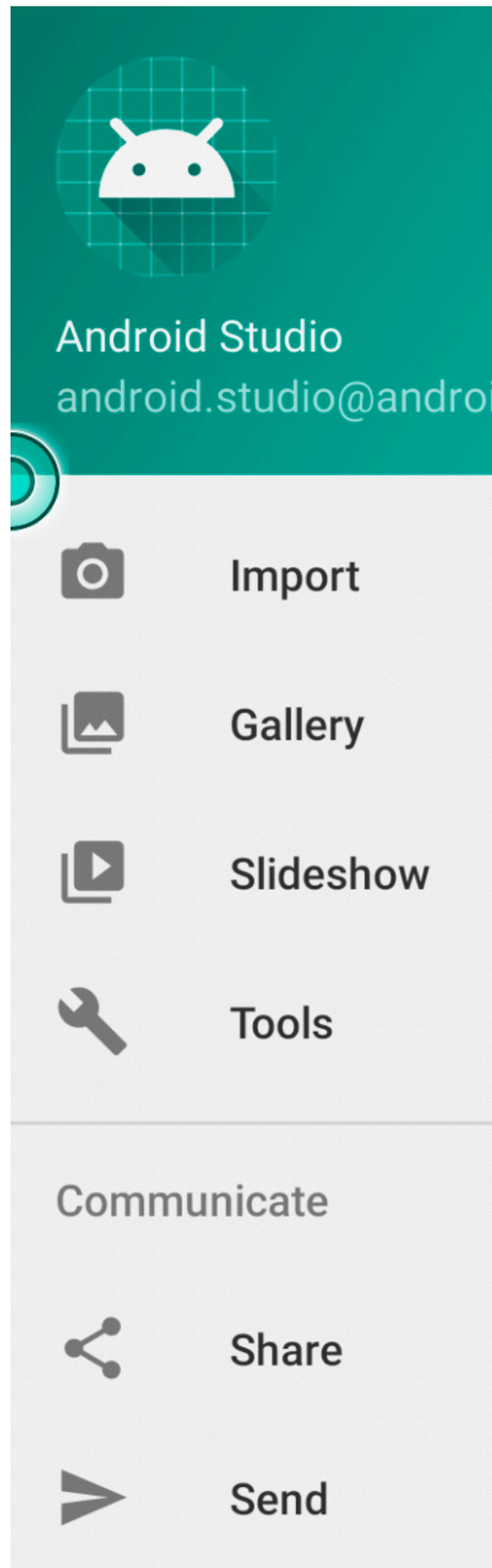
```
<android.support.design.widget.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:menu="@menu/drawer_view"
    app:headerLayout="@layout/nav_header" />
```



Add a header to the nav drawer

```
<android.support.design.widget.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:menu="@menu/drawer_view"
    app:headerLayout="@layout/nav_header" />
```

```
<?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="192dp"
    android:background="?attr/colorPrimaryDark"
    android:padding="16dp"
    android:theme="@style/ThemeOverlay.AppCompat.Dark"
    android:orientation="vertical"
    android:gravity="bottom">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="My header title"
        android:textAppearance=
           "@style/TextAppearance.AppCompat.Body1"/>
</LinearLayout>
```



Handle navigation events

```
class MainActivity : AppCompatActivity() {
    private lateinit var mDrawerLayout: DrawerLayout

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

    }
}
```

Handle navigation events

```
class MainActivity : AppCompatActivity() {

    private lateinit var mDrawerLayout: DrawerLayout

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

        val toggle = ActionBarDrawerToggle(
            this, drawer_layout, toolbar,
            R.string.navigation_drawer_open,
            R.string.navigation_drawer_close)

        drawer_layout.addDrawerListener(toggle)
        toggle.syncState()
    }
}
```

Handle navigation events

```
class MainActivity : AppCompatActivity() {

    private lateinit var mDrawerLayout: DrawerLayout

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

        val toggle = ActionBarDrawerToggle(
            this, drawer_layout, toolbar,
            R.string.navigation_drawer_open,
            R.string.navigation_drawer_close)

        drawer_layout.addDrawerListener(toggle)
        toggle.syncState()
    }
}
```

Handle navigation events

```
class MainActivity : AppCompatActivity(),
    NavigationView.OnNavigationItemSelectedListener {
    private lateinit var mDrawerLayout: DrawerLayout

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        ...
        nav_view.setNavigationItemSelectedListener(this)
    }
    override fun onNavigationItemSelected(item: MenuItem): Boolean {
        // Handle navigation view item clicks here.
        when (item.itemId) {
            R.id.nav_camera -> {
                // Handle the camera action
            }
            R.id.nav_gallery -> {
                // Handle the gallery action
            }
        }

        drawer_layout.closeDrawer(GravityCompat.START)
        return true
    }
}
```

Add a toolbar


```
<android.support.v4.widget.DrawerLayout ...>

    <FrameLayout
        android:id="@+id/content_frame"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            android:theme="@style/AppTheme.PopupOverlay" />

    </FrameLayout>
    ...

</android.support.v4.widget.DrawerLayout>
```

Add a toolbar


```
<manifest ...>
  <application ...
      <activity ...
          android:theme="@style/Theme.NoActionBar" >
  ...
</manifest>
```

Add a toolbar


```
class MainActivity : AppCompatActivity() {
    private lateinit var mDrawerLayout: DrawerLayout

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

        val toggle = ActionBarDrawerToggle(
            this, drawer_layout, toolbar,
            R.string.navigation_drawer_open,
            R.string.navigation_drawer_close)

        drawer_layout.addDrawerListener(toggle)
        toggle.syncState()
    }
}
```

Add a toolbar


```
class MainActivity : AppCompatActivity() {
    private lateinit var mDrawerLayout: DrawerLayout

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

        val toggle = ActionBarDrawerToggle(
            this, drawer_layout, toolbar,
            R.string.navigation_drawer_open,
            R.string.navigation_drawer_close)

        drawer_layout.addDrawerListener(toggle)
        toggle.syncState()
    }
}
```

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(
         savedInstanceState: Bundle?) {
        ...
        setSupportActionBar(toolbar)
        ...
    }
}
```

Add a toolbar

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(
            savedInstanceState: Bundle?) {
            ...
            setSupportActionBar(toolbar)
            ...
    }
}
```

Add a toolbar

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(
        savedInstanceState: Bundle?) {
        ...
        setSupportActionBar(toolbar)
        val actionbar: ActionBar? = supportActionBar
        actionbar?.apply {
            setDisplayHomeAsUpEnabled(true)
            setHomeAsUpIndicator(R.drawable.ic_menu)
        }
        ...
    }
}
```

Add a toolbar


```
class MainActivity : AppCompatActivity() {

    override fun onCreate(
        savedInstanceState: Bundle?) {
        ...
    }
    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        return when (item.itemId) {
            android.R.id.home -> {
                mDrawerLayout.openDrawer(GravityCompat.START)
                true
            }
            else -> super.onOptionsItemSelected(item)
        }
    }
}
```

Other State Changes

```
drawer_layout.addDrawerListener(
  object : DrawerLayout.DrawerListener {
    override fun onDrawerSlide(drawerView: View, slideOffset: Float) {
      // Respond when the drawer's position changes
    }

    override fun onDrawerOpened(drawerView: View) {
      // Respond when the drawer is opened
    }

    override fun onDrawerClosed(drawerView: View) {
      // Respond when the drawer is closed
    }

    override fun onDrawerStateChanged(newState: Int) {
      // Respond when the drawer motion state changes
    }
  }
)
```

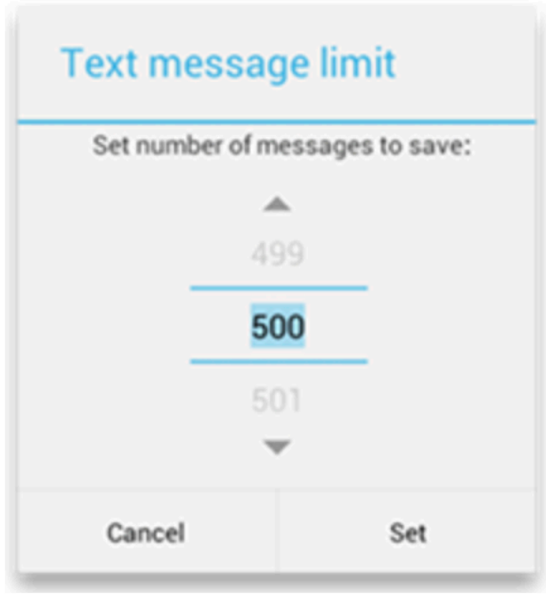
DEMO

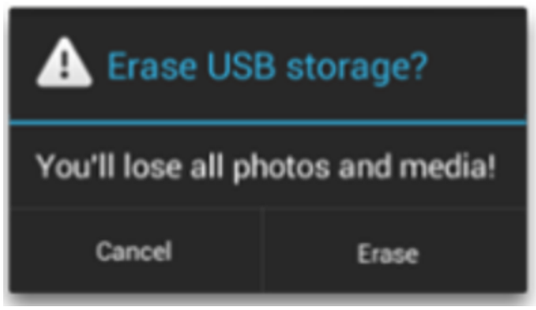
Other State Changes

```
drawer_layout.addDrawerListener(
  object : DrawerLayout.DrawerListener {
    override fun onDrawerSlide(drawerView: View, slideOffset: Float) {
      // Respond when the drawer's position changes
    }

    override fun onDrawerOpened(drawerView: View) {
      // Respond when the drawer is opened
    }

    override fun onDrawerClosed(drawerView: View) {
      // Respond when the drawer is closed
    }

    override fun onDrawerStateChanged(newState: Int) {
      // Respond when the drawer motion state changes
    }
  }
)
```

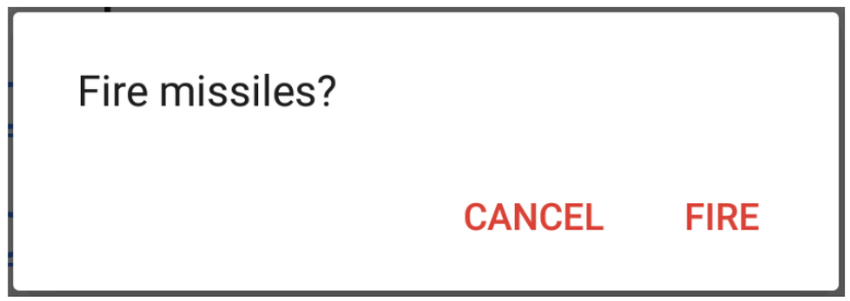
Dialogs

- Small window.
- Prompts the user to take a decision.
- Modal, by default.



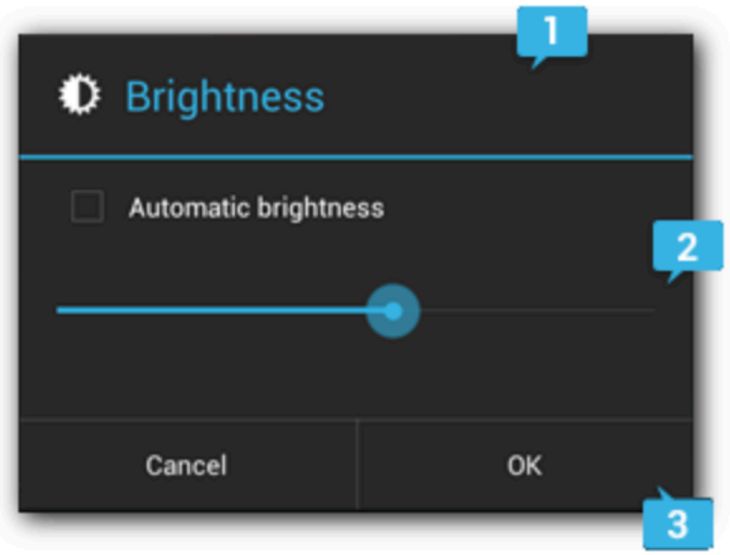


Dialogs



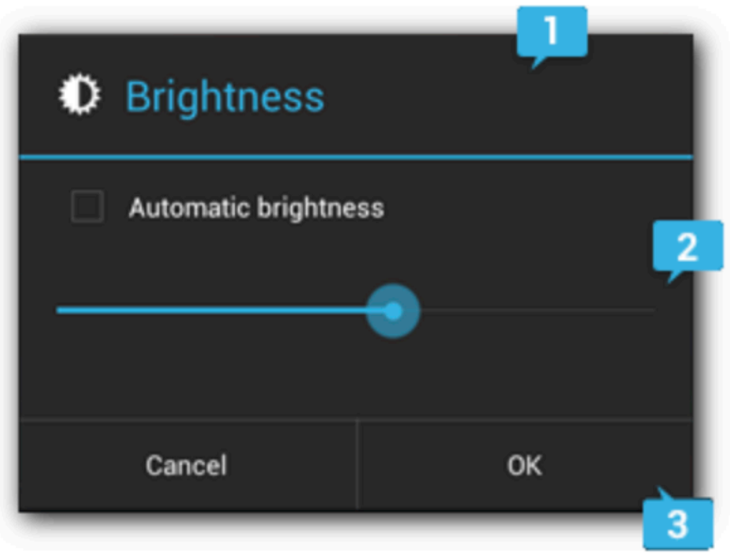

```
class FireMissilesDialogFragment : DialogFragment() {

    override fun onCreateDialog(savedInstanceState: Bundle): Dialog {
        return activity?.let {
            // Use the Builder class for convenient dialog construction
            val builder = AlertDialog.Builder(it)
            builder.setMessage(R.string.dialog_fire_missiles)
                    .setPositiveButton(R.string.fire,
                            DialogInterface.OnClickListener { dialog, id ->
                                // FIRE ZE MISSILES!
                            })
                    .setNegativeButton(R.string.cancel,
                            DialogInterface.OnClickListener { dialog, id ->
                                // User cancelled the dialog
                            })
            // Create the AlertDialog object and return it
            builder.create()
        } ?: throw IllegalStateException("Activity cannot be null")
    }
}
```

Dialogs




```
// 1. Instantiate an AlertDialog.Builder with its constructor
val builder: AlertDialog.Builder? = activity?.let {
    AlertDialog.Builder(it)
}

// 2. Chain together various setter methods to set the dialog characteristics
builder?.setMessage(R.string.dialog_message)
        .setTitle(R.string.dialog_title)

// 3. Get the AlertDialog from create()
val dialog: AlertDialog? = builder?.create()
```

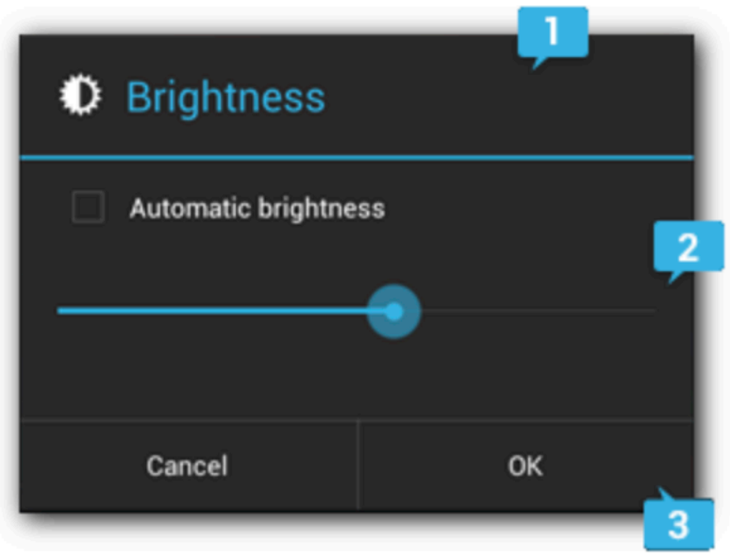
Dialogs




```
// 1. Instantiate an AlertDialog.Builder with its constructor
val builder: AlertDialog.Builder? = activity?.let {
    AlertDialog.Builder(it)
}

// 2. Chain together various setter methods to set the dialog characteristics
builder?.setMessage(R.string.dialog_message)
        .setTitle(R.string.dialog_title)

// 3. Get the AlertDialog from create()
val dialog: AlertDialog? = builder?.create()
```

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

Adding actions

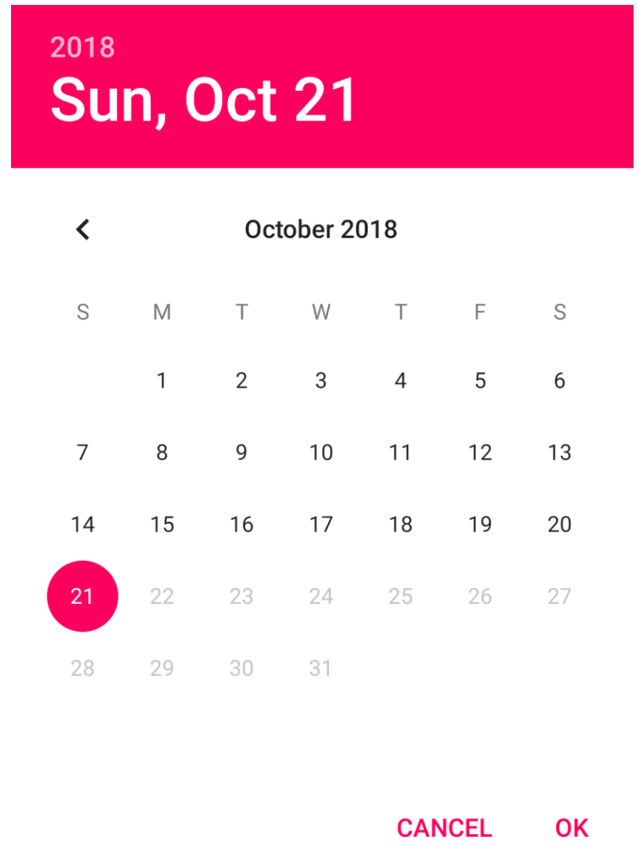



```
val alertDialog: AlertDialog? = activity?.let {
    val builder = AlertDialog.Builder(it)
    builder.apply {
        setPositiveButton(R.string.ok,
                DialogInterface.OnClickListener { dialog, id ->
                    // User clicked OK button
                })
        setNegativeButton(R.string.cancel,
                DialogInterface.OnClickListener { dialog, id ->
                    // User cancelled the dialog
                })
    }
    // Set other dialog properties
    ...

    // Create the AlertDialog
    builder.create()
}
```

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

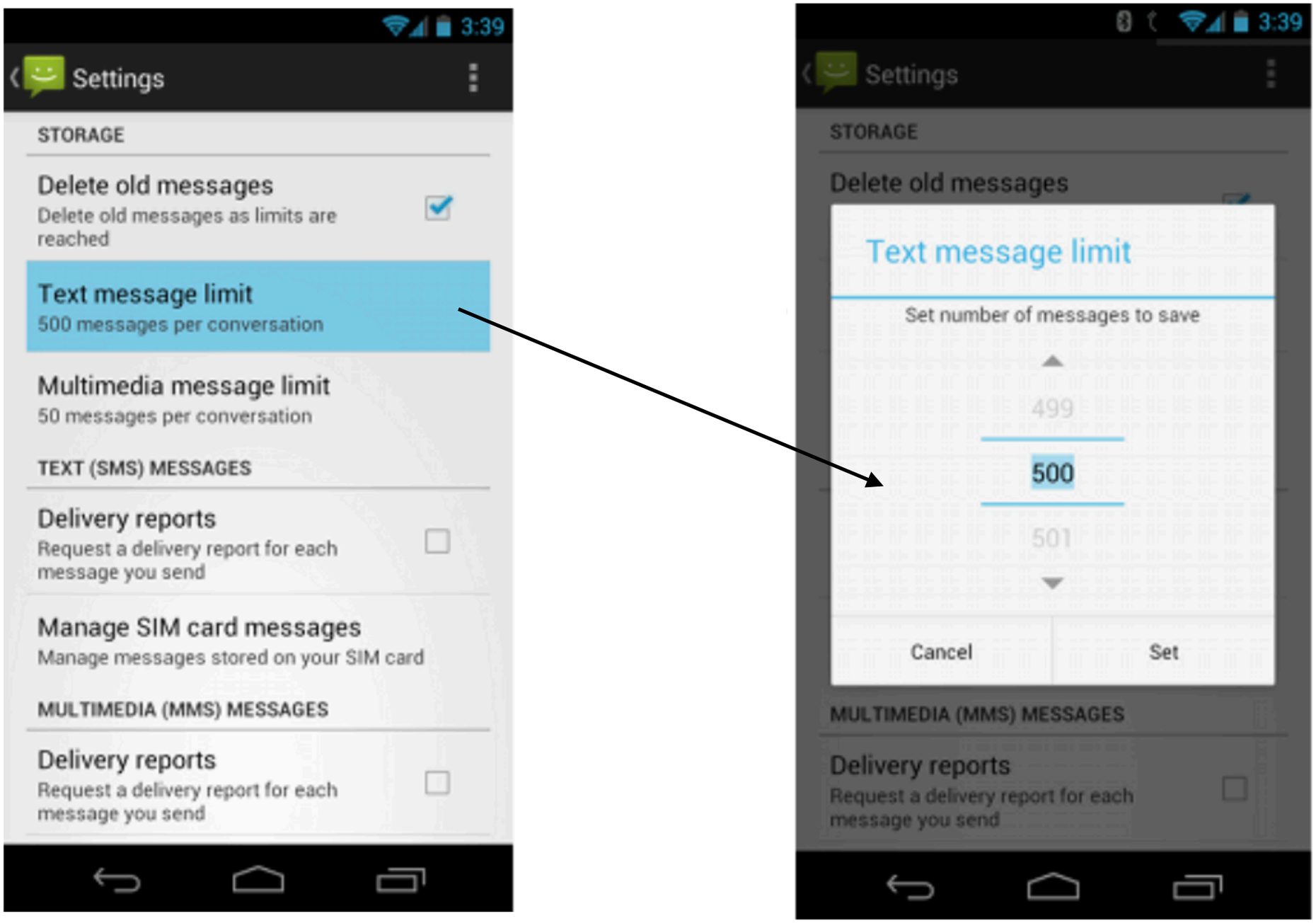
Using Anko

```
alert {
    isCancelable = false
    lateinit var datePicker: DatePicker
    customView {
        verticalLayout {
            datePicker = datePicker {
                maxDate = System.currentTimeMillis()
            }
        }
    }
    yesButton {
        val parsedDate =
          "${datePicker.dayOfMonth}/${datePicker.month + 1}/${datePicker.year}"
        toast("Selected date: $parsedDate")
    }
    noButton { }
}.show()
```



https://github.com/Kotlin/anko/wiki/Anko-Commons---Dialogs

Preferences

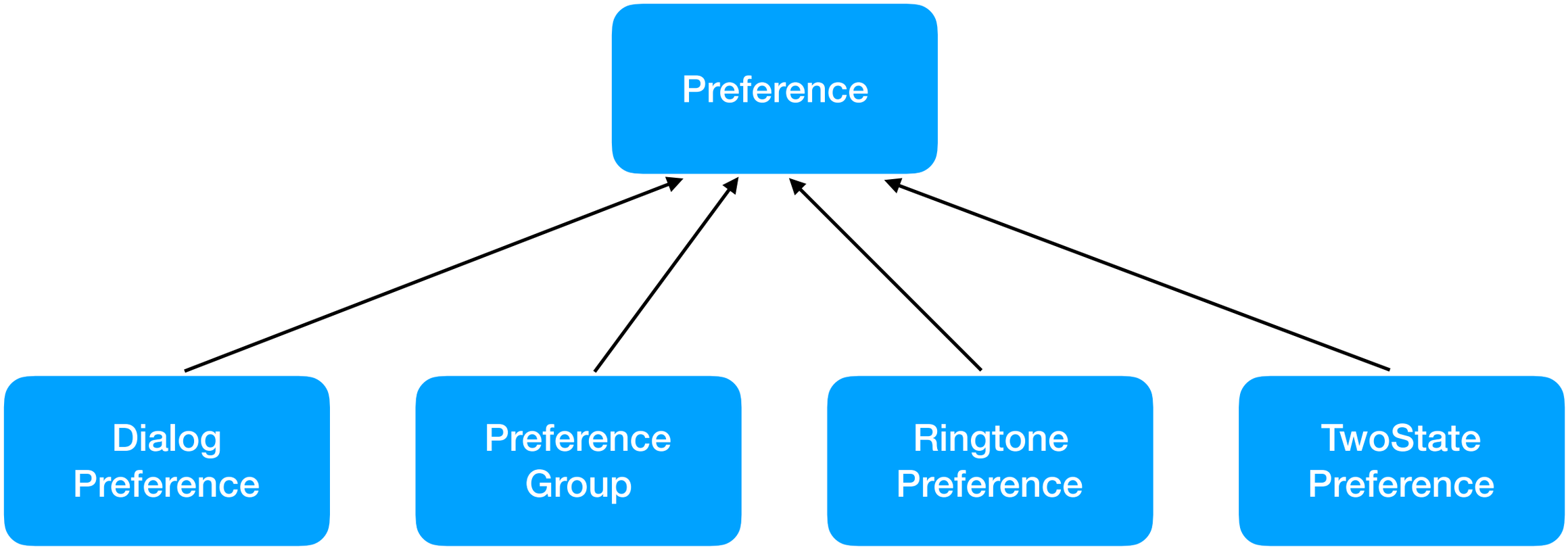


Preferences

View

Preference

Preferences



Preferences

Dialog Preference

Preference Group

Ringtone Preference

TwoState Preference

Preferences

Dialog Preference

Preference Group

Ringtone Preference

TwoState Preference

EditText Preference

List Preference

MultiSelectList Preference

Preferences

Dialog Preference

Preference Group

Ringtone Preference

TwoState Preference

EditText Preference

Preference Category

List Preference

Preference Screen

MultiSelectList Preference

Preferences

Dialog Preference

Preference Group

Ringtone Preference

TwoState Preference

EditText Preference

Preference Category

CheckBox Preference

List Preference

Preference Screen

Switch Preference

MultiSelectList Preference

Preferences

Dialog Preference

Preference Group

Ringtone Preference

TwoState Preference

EditText Preference

Preference Category

CheckBox Preference

List Preference

Preference Screen

Switch Preference

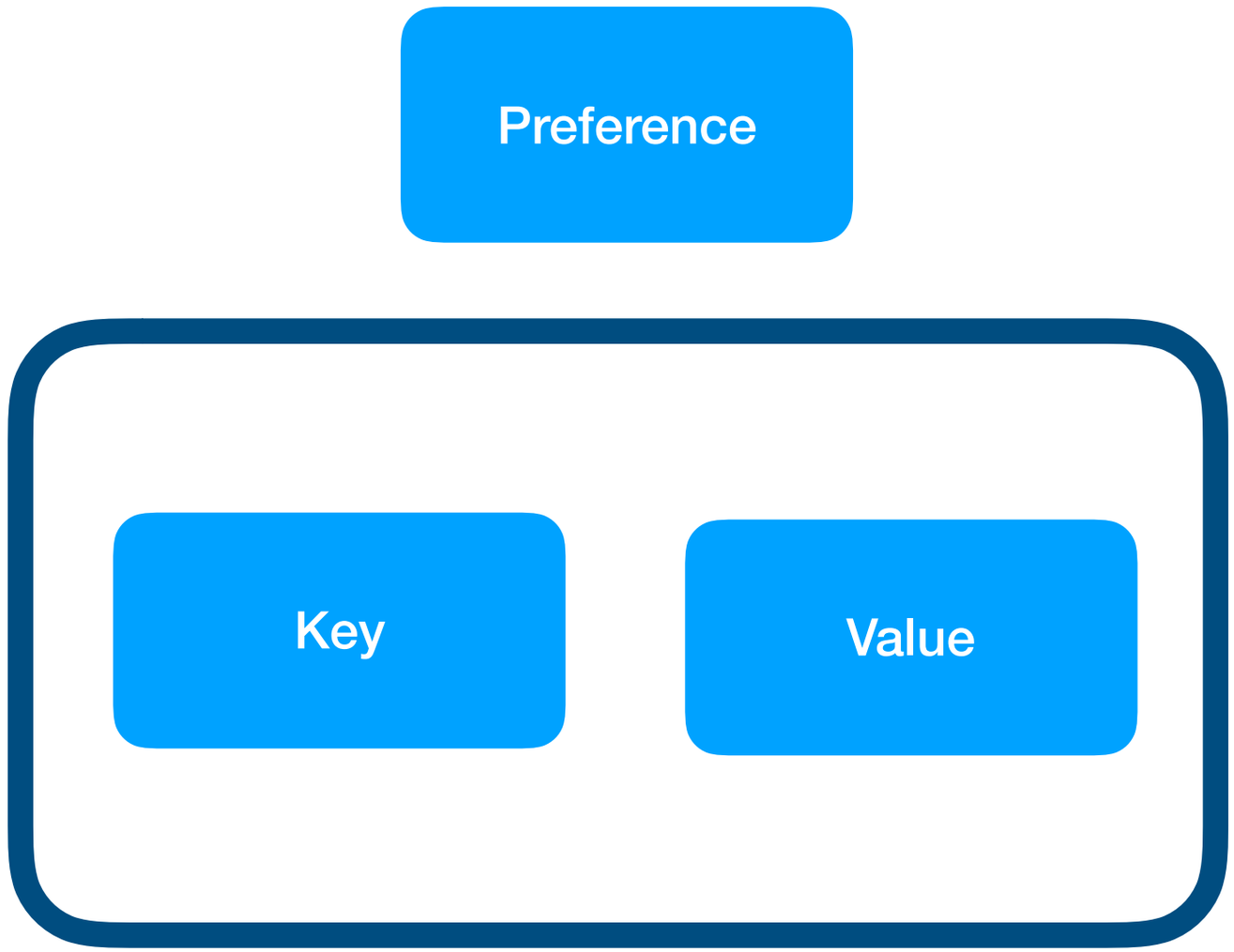
MultiSelectList Preference

https://developer.android.com/reference/android/preference/Preference

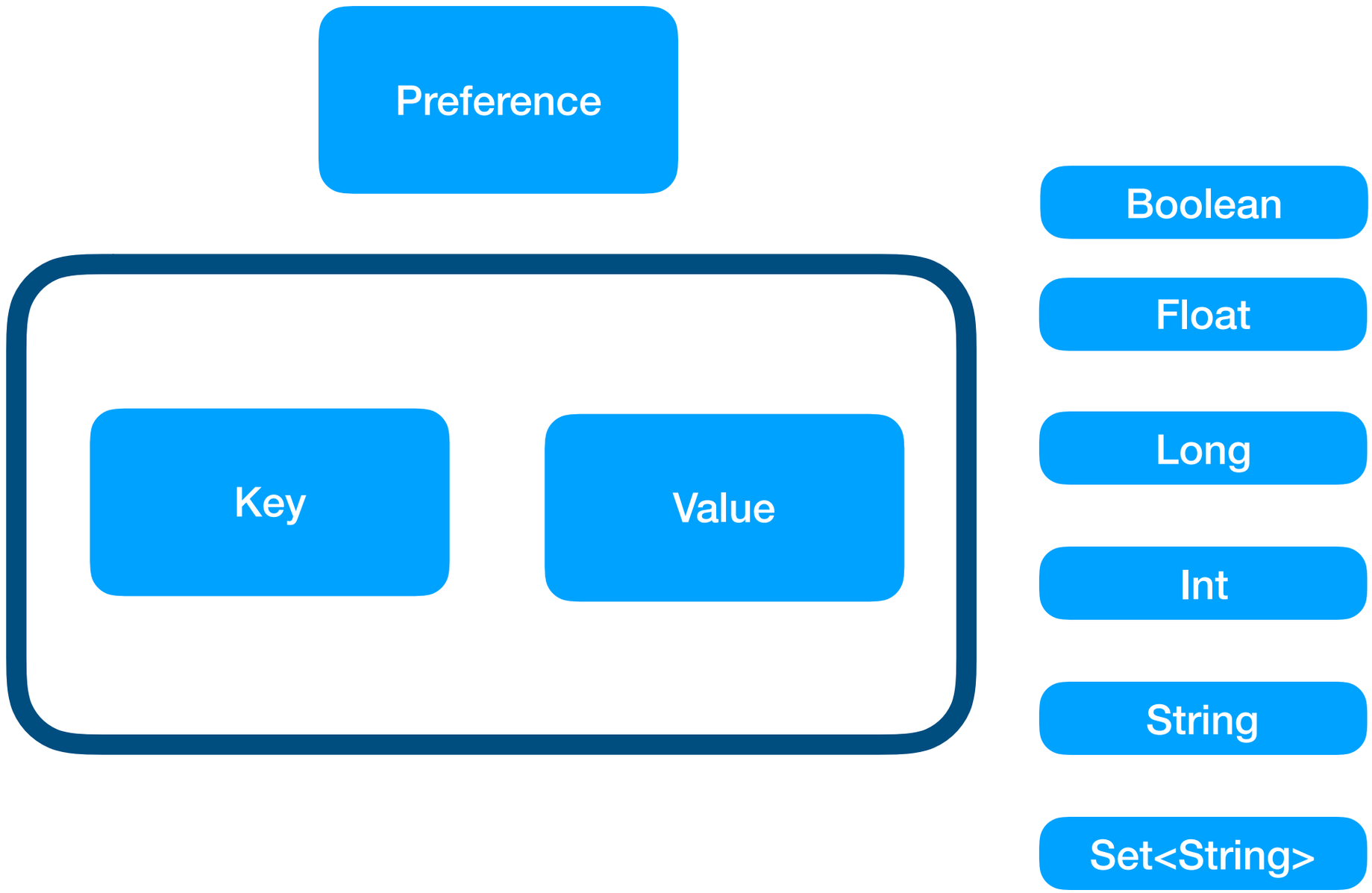
Preferences

Preference

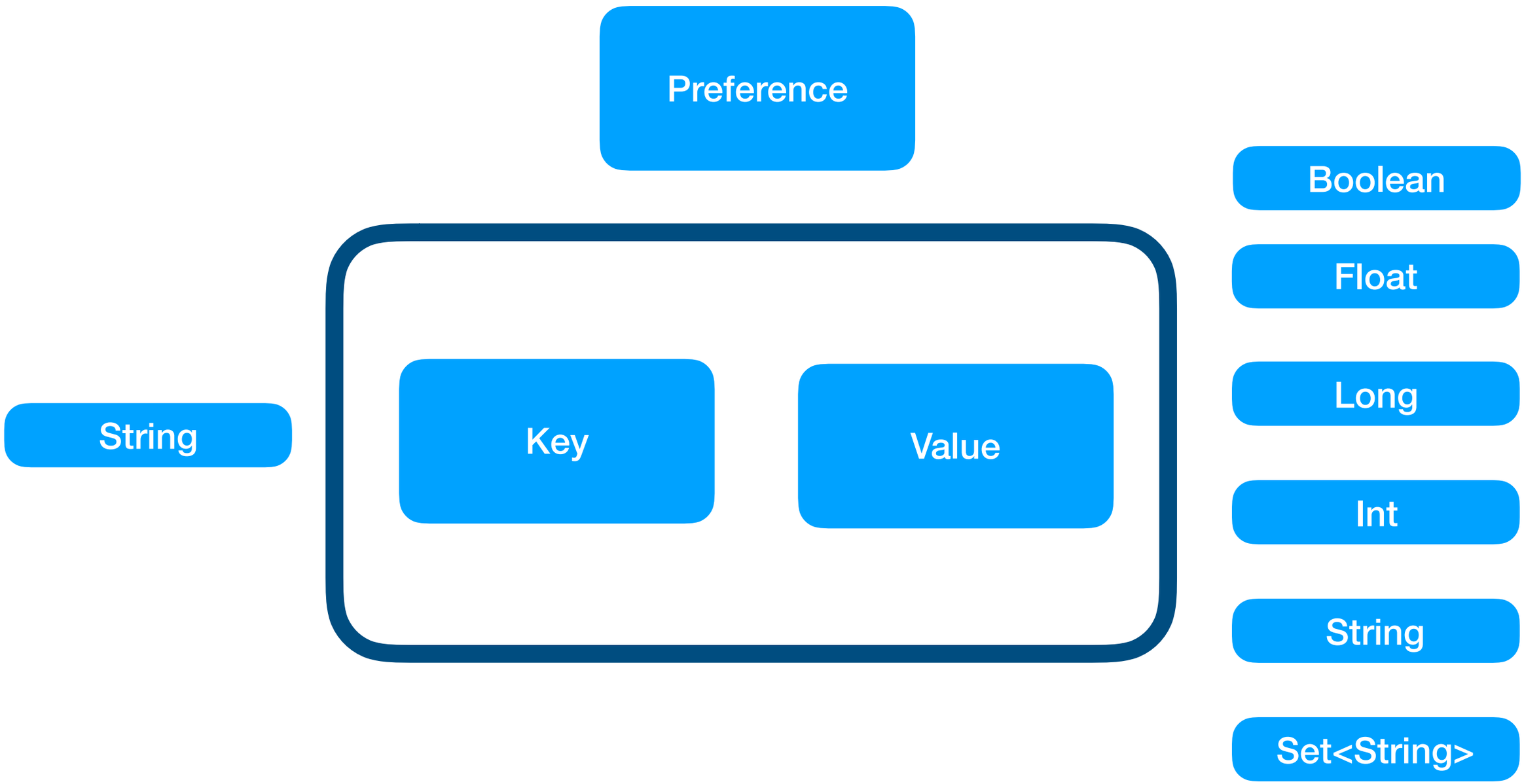
Preferences



Preferences

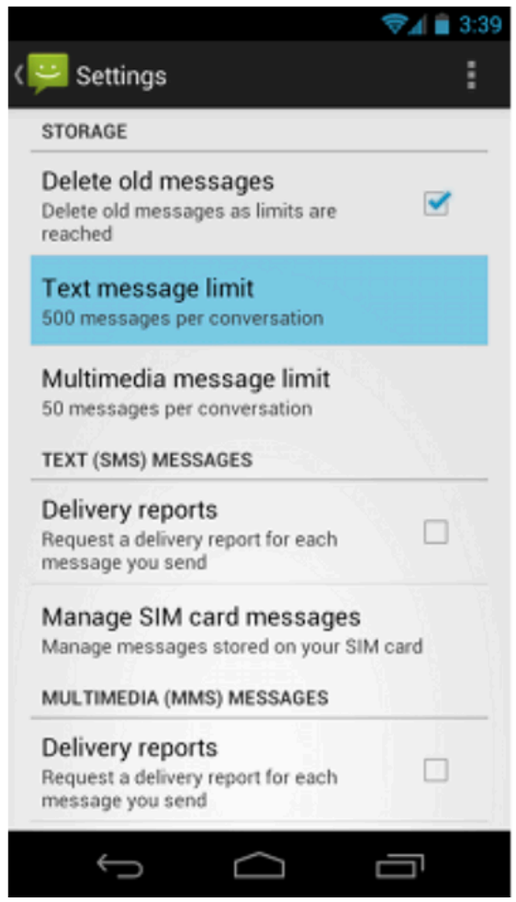


Preferences



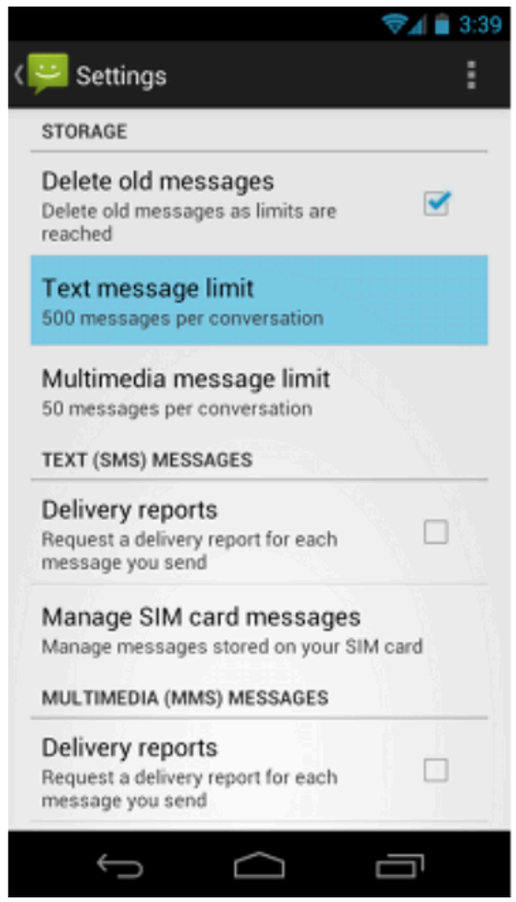
PreferenceScreen

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
   xmlns:android="http://schemas.android.com/apk/res/android">
    <CheckBoxPreference
        android:key="pref_sync"
        android:title="@string/pref_sync"
        android:summary="@string/pref_sync_summ"
        android:defaultValue="true" />
    <ListPreference
        android:dependency="pref_sync"
        android:key="pref_syncConnectionType"
        android:title="@string/pref_syncConnectionType"
        android:dialogTitle="@string/pref_syncConnectionType"
        android:entries="@array/pref_syncConnectionTypes_entries"
        android:entryValues="@array/pref_syncConnectionTypes_values"
        android:defaultValue="@string/pref_syncConnectionTypes_default" />
</PreferenceScreen>
```



PreferenceScreen

```xml
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">
     <CheckBoxPreference
          android:key="pref_sync"
          android:title="@string/pref_sync"
          android:summary="@string/pref_sync_summ"
          android:defaultValue="true" />
     <ListPreference
          android:dependency="pref_sync"
          android:key="pref_syncConnectionType"
          android:title="@string/pref_syncConnectionType"
          android:dialogTitle="@string/pref_syncConnectionType"
          android:entries="@array/pref_syncConnectionTypes_entries"
          android:entryValues="@array/pref_syncConnectionTypes_values"
          android:defaultValue="@string/pref_syncConnectionTypes_default" />
 </PreferenceScreen>
```

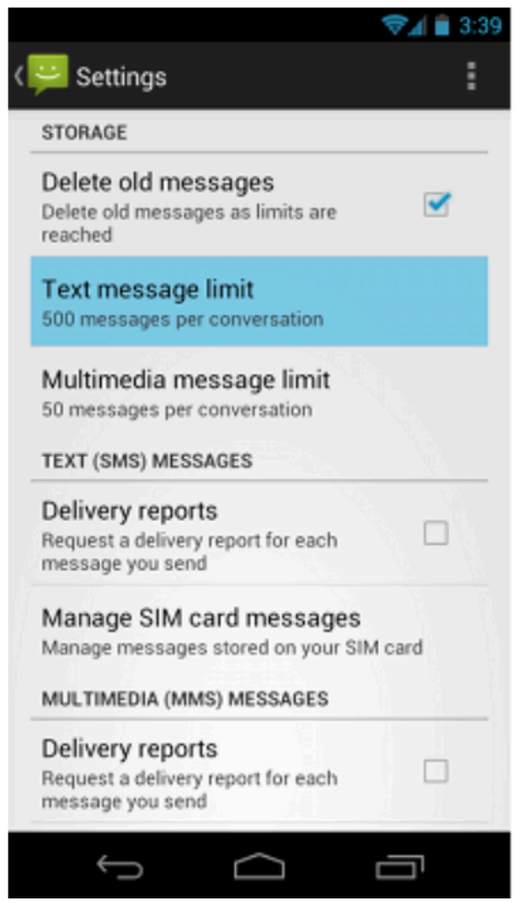



```kotlin
class SettingsActivity : PreferenceActivity() {

    override fun onCreate(savedInstanceState: Bundle) {
        super.onCreate(savedInstanceState)
        addPreferencesFromResource(R.xml.preferences)
    }
}
```

DEMO

PreferenceScreen

```xml
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">
     <CheckBoxPreference
          android:key="pref_sync"
          android:title="@string/pref_sync"
          android:summary="@string/pref_sync_summ"
          android:defaultValue="true" />
     <ListPreference
          android:dependency="pref_sync"
          android:key="pref_syncConnectionType"
          android:title="@string/pref_syncConnectionType"
          android:dialogTitle="@string/pref_syncConnectionType"
          android:entries="@array/pref_syncConnectionTypes_entries"
          android:entryValues="@array/pref_syncConnectionTypes_values"
          android:defaultValue="@string/pref_syncConnectionTypes_default" />
 </PreferenceScreen>
```




```
class SettingsActivity : PreferenceActivity() {

    override fun onCreate(savedInstanceState: Bundle) {
        super.onCreate(savedInstanceState)
        addPreferencesFromResource(R.xml.preferences)
    }
}
```

Saving & Reading Local Files

- Internal storage
  - Internal cache files
- External storage
- Shared preferences
- Databases

https://developer.android.com/guide/topics/data/

Internal Storage

- It's always available.
- Available only to your app.
- On uninstall everything is removed.

Internal Storage

- It's always available.
- Available only to your app.
- On uninstall everything is removed.

Neither the user nor other apps can access your files!

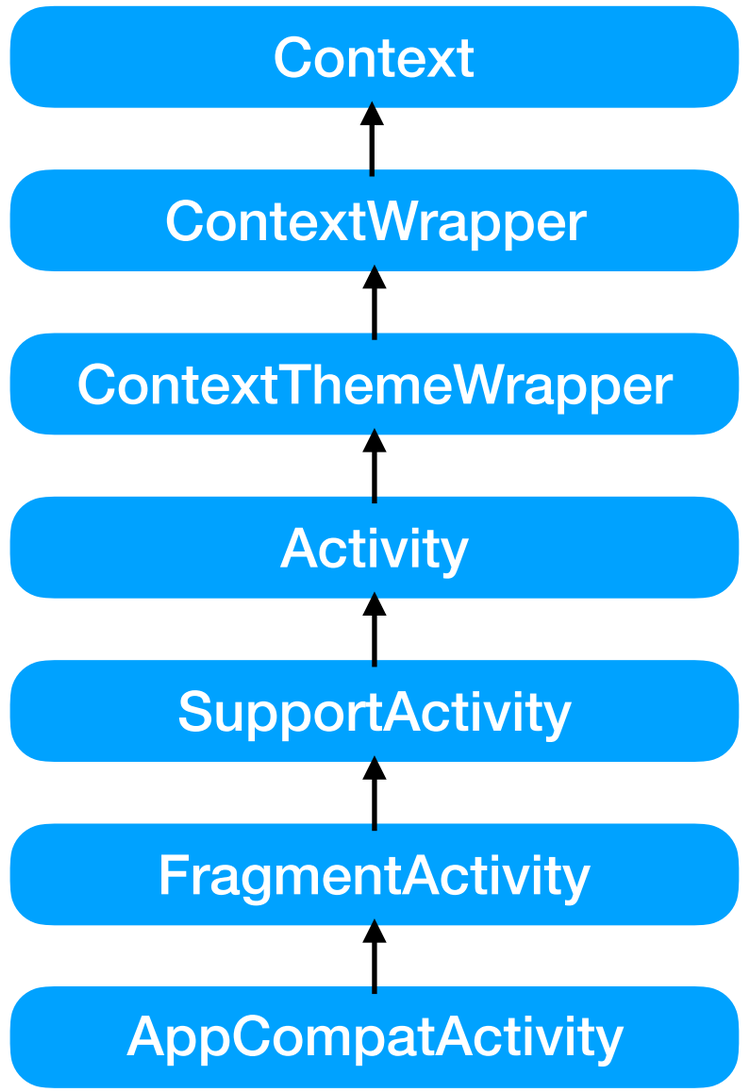
https://developer.android.com/guide/topics/data/data-storage#filesInternal

Internal Storage


```
public abstract class Context {
    ...
    public abstract File getFilesDir();
    ...
    public abstract File getCacheDir();
    ...
}
```

Internal Storage

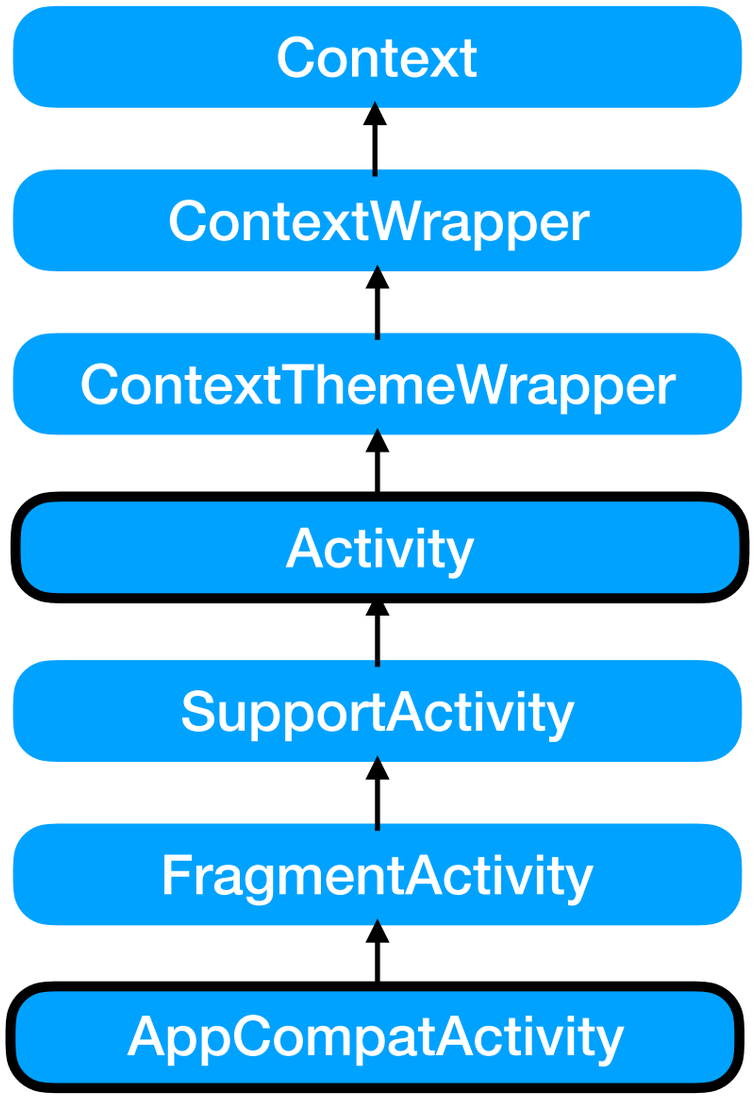

```
public abstract class Context {
    ...
    public abstract File getFilesDir();
    ...
    public abstract File getCacheDir();
    ...
}
```



Internal Storage


```
public abstract class Context {
    ...
    public abstract File getFilesDir();
    ...
    public abstract File getCacheDir();
    ...
}
```

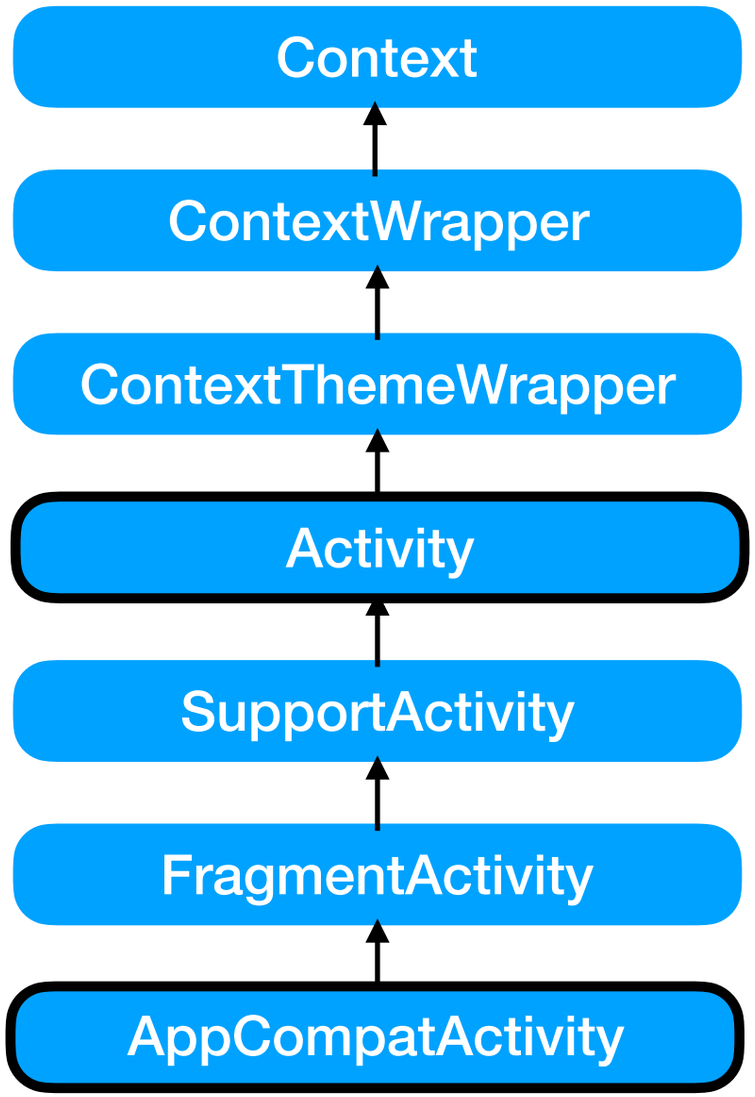
```
val file = File(context.filesDir, filename)
```



Internal Storage


```
public abstract class Context {
    ...
    public abstract File getFilesDir();
    ...
    public abstract File getCacheDir();
    ...
}
```

```
val file = File(context.filesDir, filename)


val filename = "myfile"
val fileContents = "Hello world!"
context.openFileOutput(filename,
    Context.MODE_PRIVATE).use {
      it.write(fileContents.toByteArray())
}
```



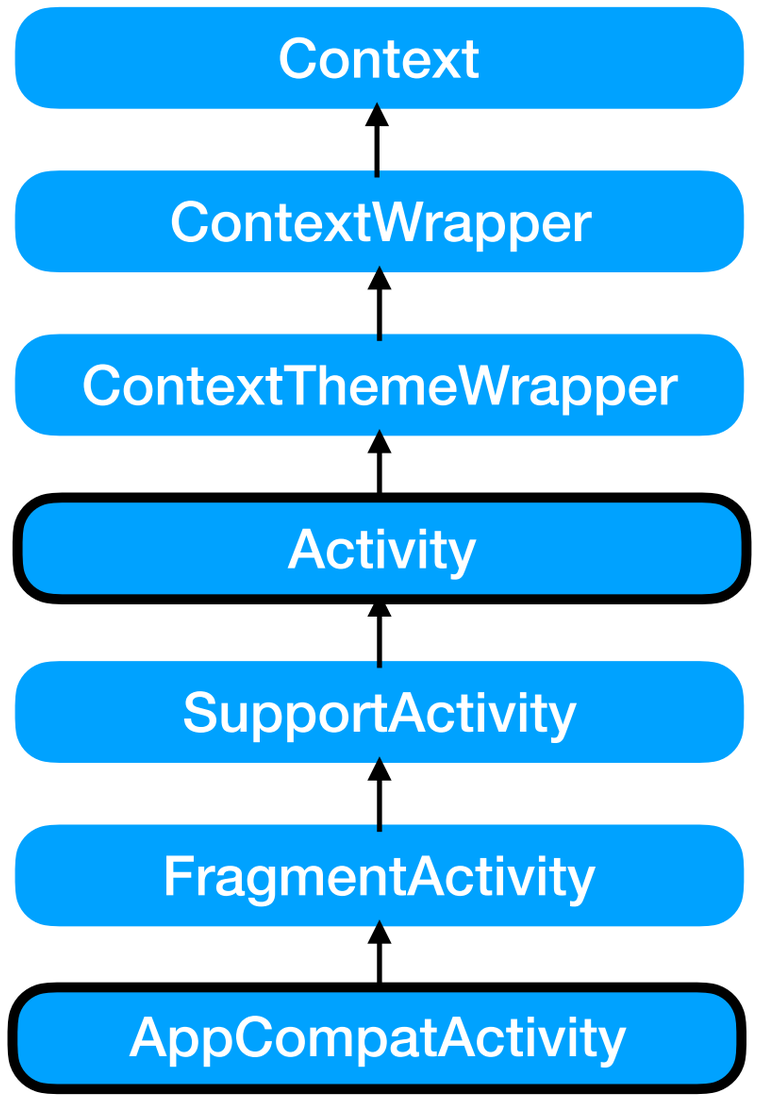
Internal Storage


```
public abstract class Context {
    ...
    public abstract File getFilesDir();
    ...
    public abstract File getCacheDir();
    ...
}
```

```
val file = File(context.filesDir, filename)
```

```
val filename = "myfile"
val fileContents = "Hello world!"
context.openFileOutput(filename,
    Context.MODE_PRIVATE).use {
      it.write(fileContents.toByteArray())
}
```

```
private fun getTempFile(
    context: Context, url: String): File? =
 Uri.parse(url)?.lastPathSegment?.let { filename ->
    File.createTempFile(filename, null, context.cacheDir)
 }
```



External Storage Permissions

```
<manifest ...>
    <uses-permission android:name=
       "android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

External Storage Permissions

```
<manifest ...>
    <uses-permission android:name=
       "android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name=
       "android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

< Android 4.4 (API level 19)

External Storage Permissions

```
<manifest ...>
    <uses-permission android:name=
       "android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name=
       "android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

< Android 4.4 (API level 19)


```
/* Checks if external storage is available for read and write */
fun isExternalStorageWritable(): Boolean {
    return Environment.getExternalStorageState() == Environment.MEDIA_MOUNTED
}

/* Checks if external storage is available to at least read */
fun isExternalStorageReadable(): Boolean {
     return Environment.getExternalStorageState() in
        setOf(Environment.MEDIA_MOUNTED, Environment.MEDIA_MOUNTED_READ_ONLY)
}
```

https://developer.android.com/training/data-storage/files#WriteInternalStorage

FileProvider

content://**com.example.myapp.fileprovider/myimages**/default_image.jpg

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp">
    <application
        ...>
        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="com.example.myapp.fileprovider"
            android:grantUriPermissions="true"
            android:exported="false">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/filepaths" />
        </provider>
        ...
    </application>
</manifest>
```

DEMO

FileProvider

content://**com.example.myapp.fileprovider/myimages**/default_image.jpg

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp">
    <application
        ...>
        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="com.example.myapp.fileprovider"
            android:grantUriPermissions="true"
            android:exported="false">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/filepaths" />
        </provider>
        ...
    </application>
</manifest>
```

```
<paths>
    <files-path path="images/" name="myimages" />
</paths>
```

https://developer.android.com/training/secure-file-sharing/setup-sharing

Lecture outcomes

- Navigate between screens/ views.
- Use dialogs and pickers.
- Manage files & preferences.