ANDROID

User Interface

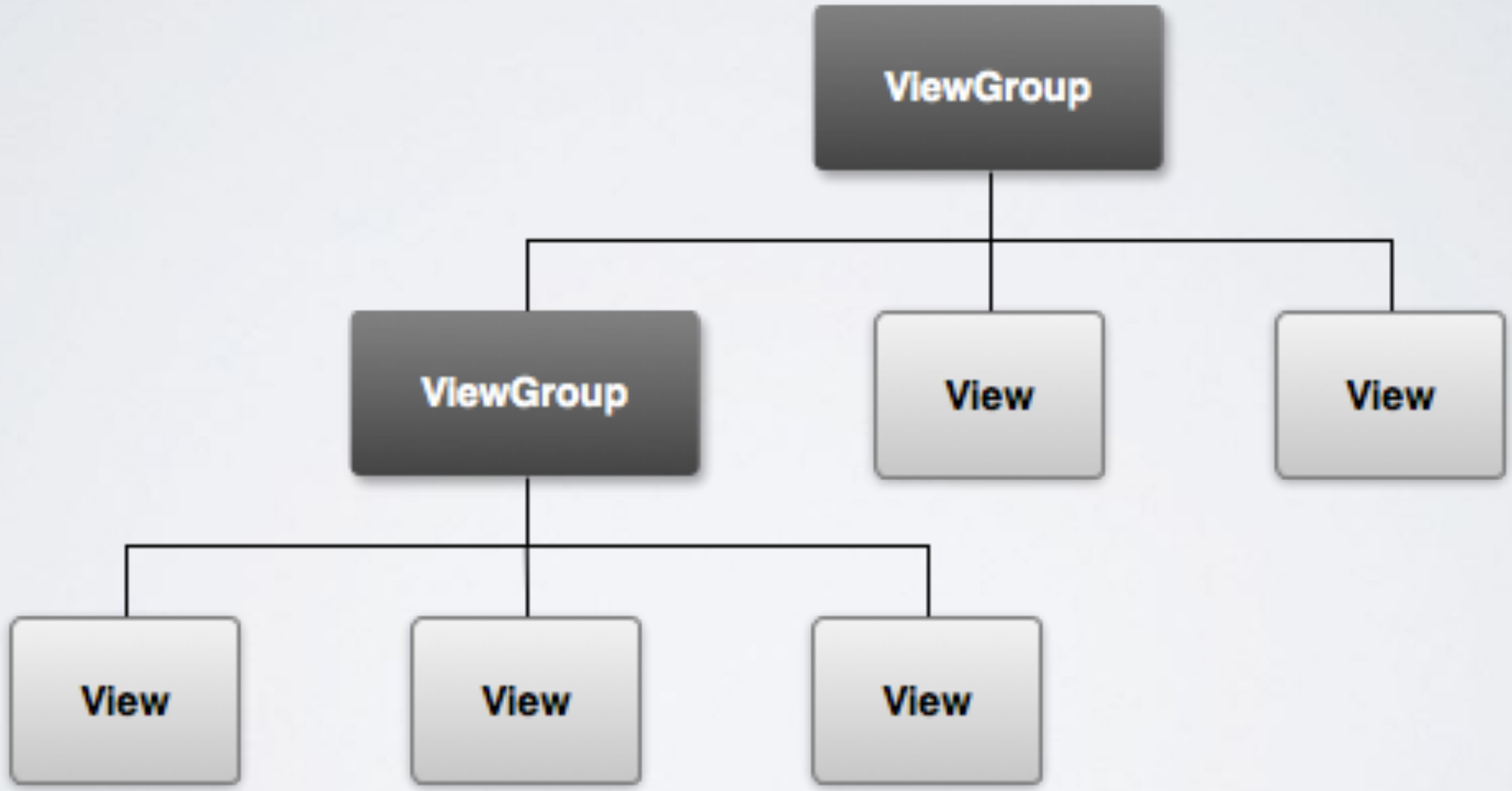
08.12.2016

MOBILE APPLICATIONS PROGRAMMING

Krzysztof Pawłowski

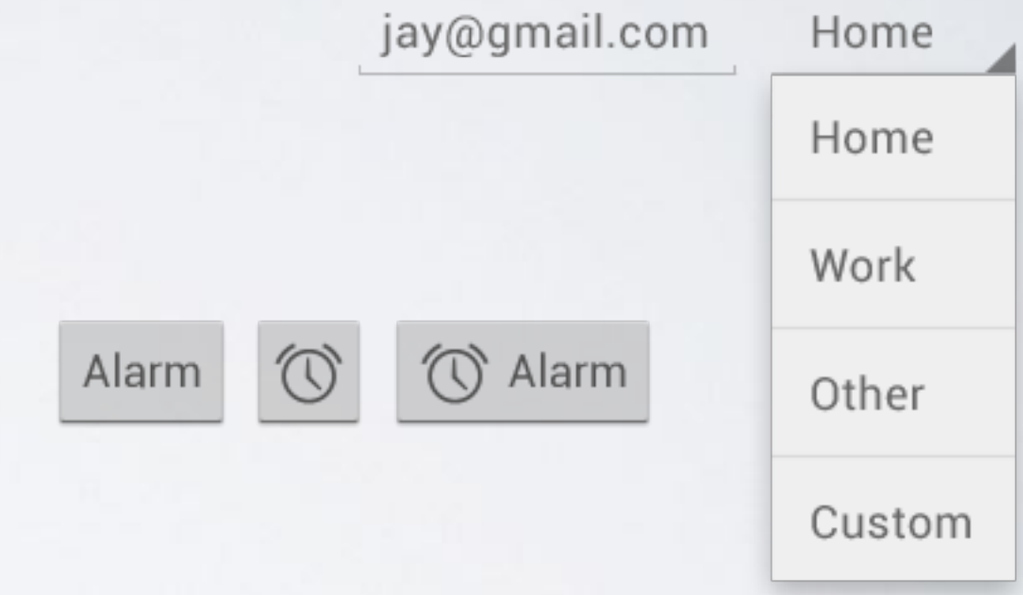
Polsko-Japońska Akademia Technik Komputerowych

USER INTERFACE LAYOUT



VIEW

- basic building block for user interface components
- responsible for event handling
- base class for widgets e.g. button, text field etc.



VIEWGROUP

- subclass of View
- contains other views (in particular other ViewGroups)
- its subclasses represent layouts

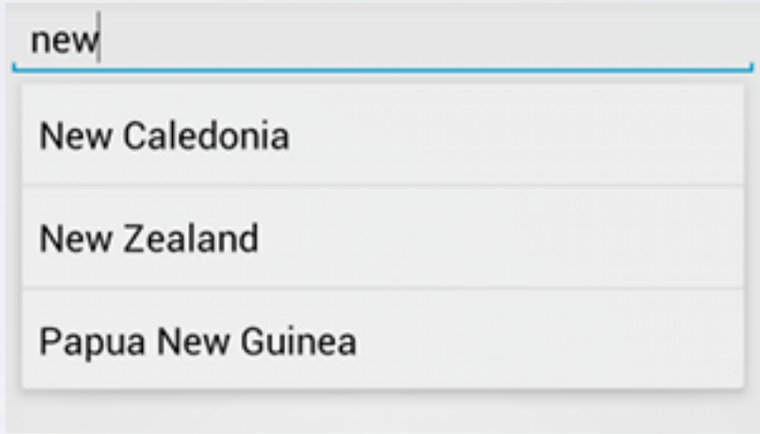
VIEWS AND VIEWGROUP - CODE

```
<LinearLayout xmlns:android=“http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  tools:context=".MainActivity"
  android:orientation="vertical">

  <Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button 1"/>
  <Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button 2"/>
  <Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button 3"/>
  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="This is vertical orientation"
    android:gravity="center"/>

</LinearLayout>
```

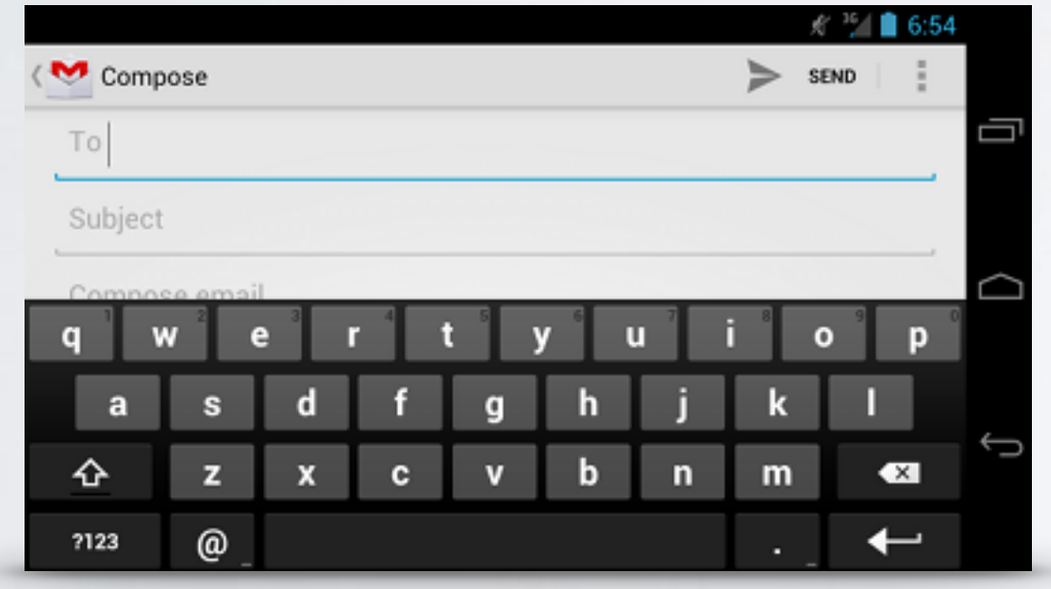
INPUT CONTROLS

Button

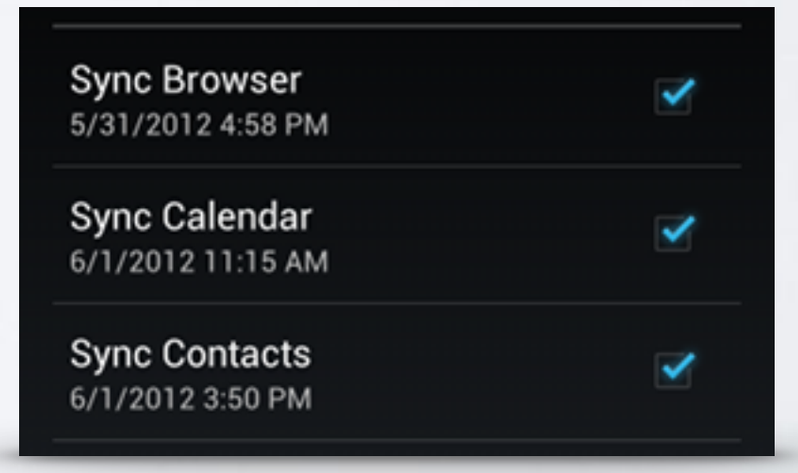
AutoCompleteEditText



EditText

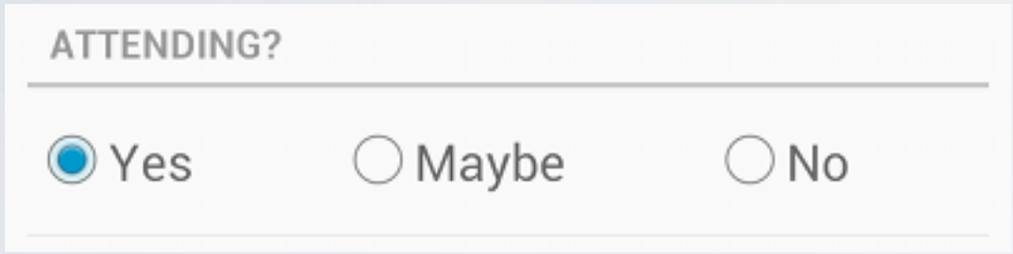


CheckBoxes



INPUT CONTROLS

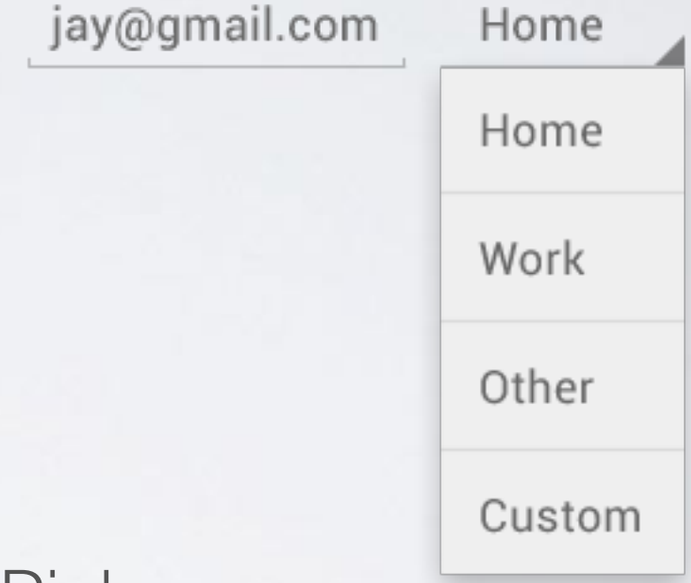
RadioBoxes



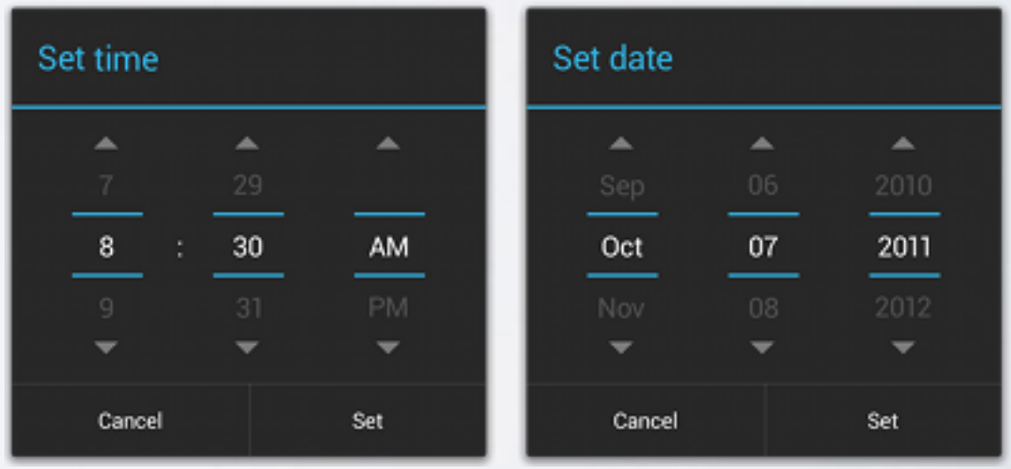
ToggleButton

Switch

Spinner



Picker



INPUT CONTROL - CODE

```xml
<?xml version="1.0" encoding="utf-8"?>
<AutoCompleteTextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/autocomplete_country"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="countries_array">
        <item>Afghanistan</item>
        <item>Albania</item>
        <item>Algeria</item>
        <item>American Samoa</item>
        <item>Andorra</item>
        <item>Angola</item>
        <item>Anguilla</item>
        <item>Antarctica</item>
        ...
    </string-array>
</resources>
```

INPUT CONTROL - CODE

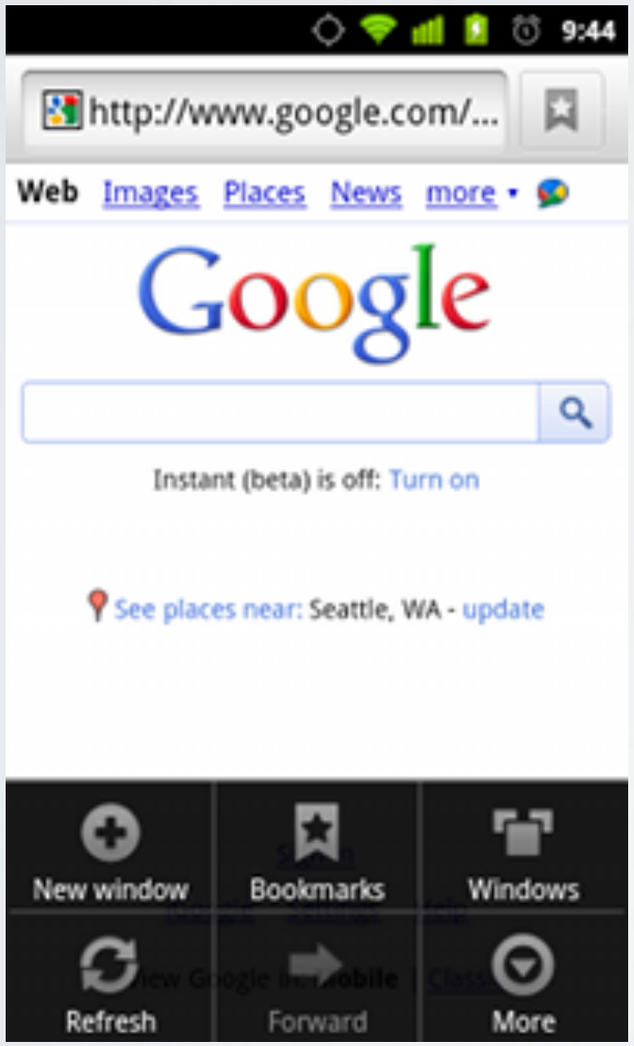
```
// Get a reference to the AutoCompleteTextView in the layout
AutoCompleteTextView textView =
     (AutoCompleteTextView) findViewById(R.id.autocomplete_country);
// Get the string array
String[] countries = getResources().getStringArray(R.array.countries_array);
// Create the adapter and set it to the AutoCompleteTextView
ArrayAdapter<String> adapter =
     new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
countries);
textView.setAdapter(adapter);
```

INPUT LISTENERS

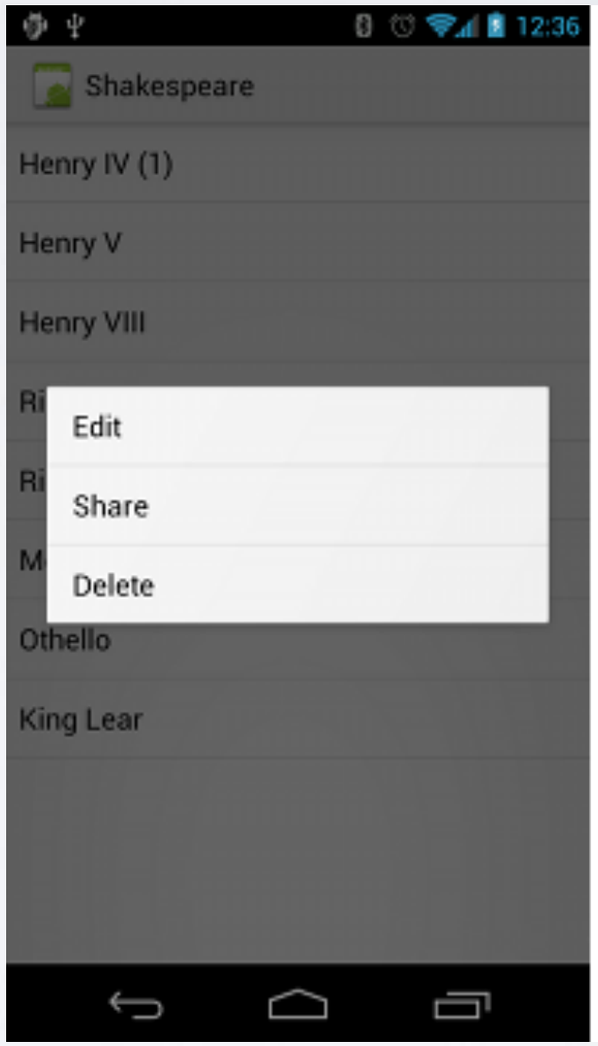
- onClick()
- onLongClick()
- onFocusChanged()
- onKey()
- onTouch()
- onCreateContextMenu()

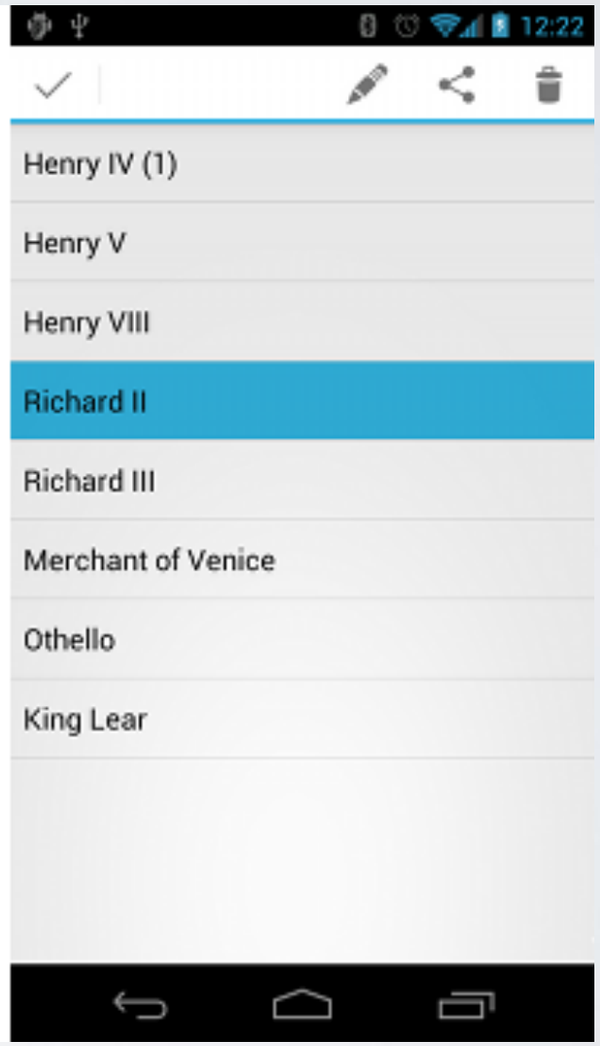
MENU

Options Menu



Context Menu





OPTIONS MENU - DEFINITION

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/
android">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game"
          android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```

OPTIONS MENU - CODE

```
// In Activity or Fragment
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.game_menu, menu);
    return true;
}
```

OPTIONS MENU - HANDLING CLICK EVENTS

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    switch (item.getItemId()) {
        case R.id.new_game:
            newGame();
            return true;
        case R.id.help:
            showHelp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

OPTIONS MENU - MODIFY IN RUNTIME

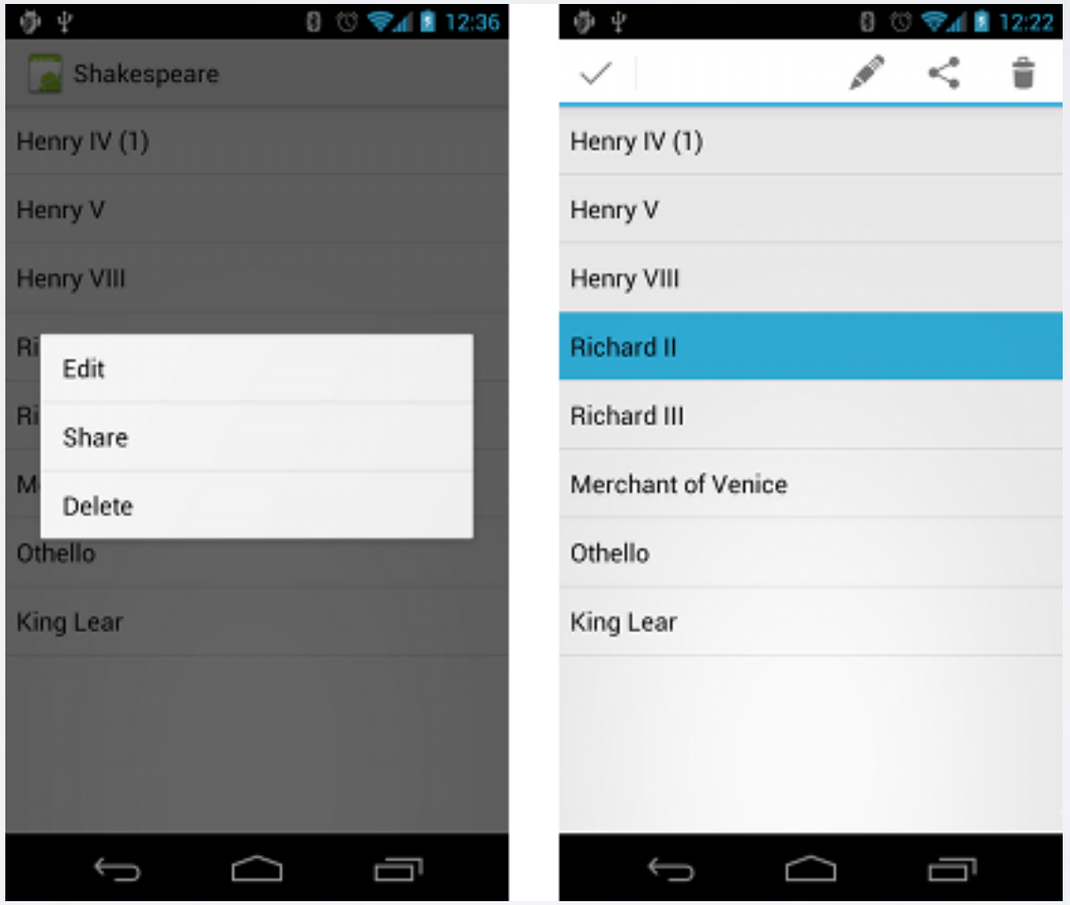
- invalidateOptionsMenu()
- onPrepareOptionsMenu()

CONTEXTUAL MENU

- Register View (UI component) for contextual menu by calling **`registerForContextMenu()`**
- Implement **`onCreateContextMenu()`**
- Implement **`onContextItemSelected`**
- Two **`ActionModes:`**
  - floating context menu
  - contextual action mode

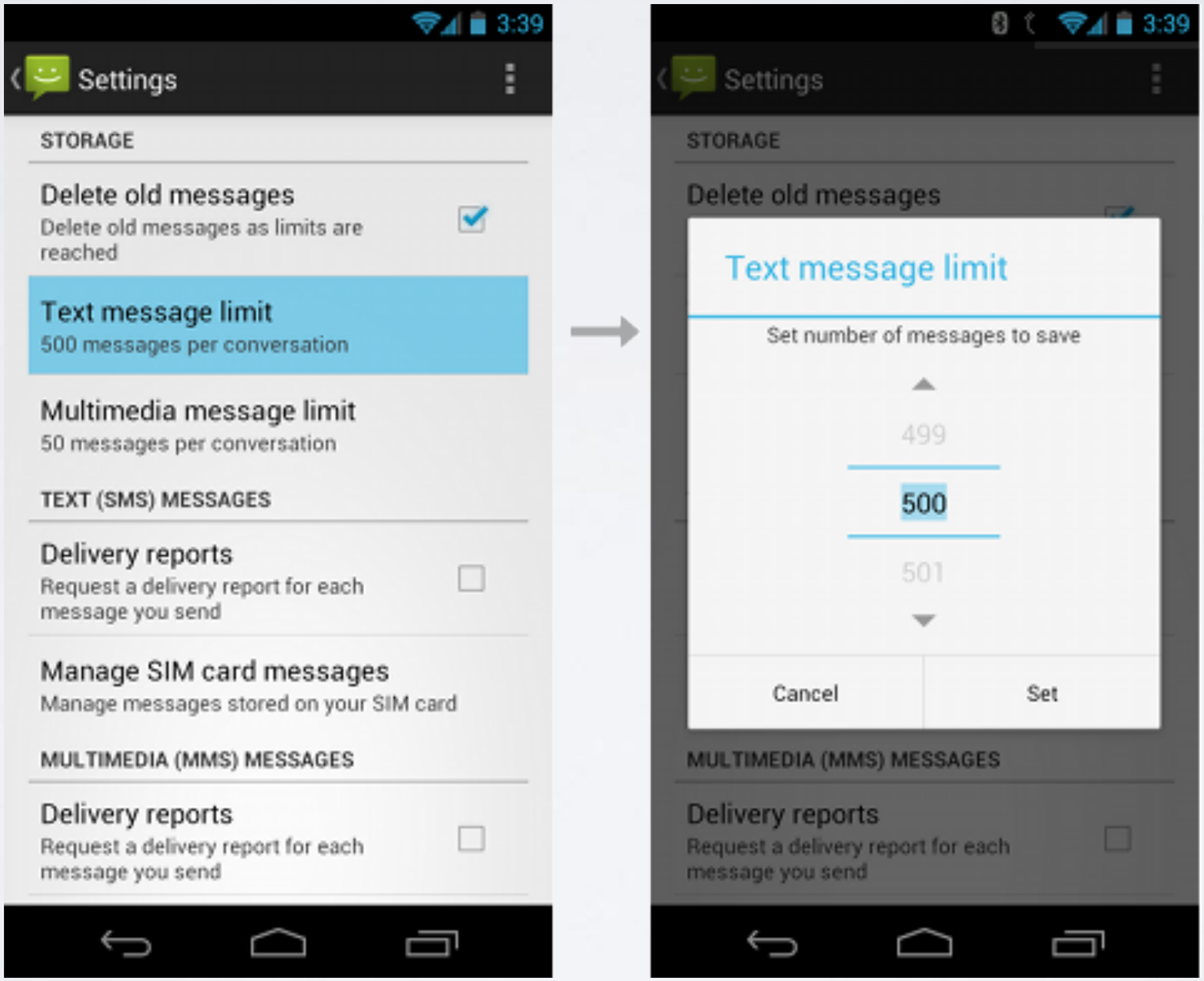
CONTEXTUAL MENU

floating context menu



contextual action mode

SETTINGS


3:39
Settings
STORAGE
Delete old messages
Delete old messages as limits are reached
Text message limit
500 messages per conversation
Multimedia message limit
50 messages per conversation
TEXT (SMS) MESSAGES
Delivery reports
Request a delivery report for each message you send
Manage SIM card messages
Manage messages stored on your SIM card
MULTIMEDIA (MMS) MESSAGES
Delivery reports
Request a delivery report for each message you send
3:39
Settings
STORAGE
Delete old messages
Text message limit
Set number of messages to save
499
500
501
Cancel
Set
MULTIMEDIA (MMS) MESSAGES
Delivery reports
Request a delivery report for each message you send

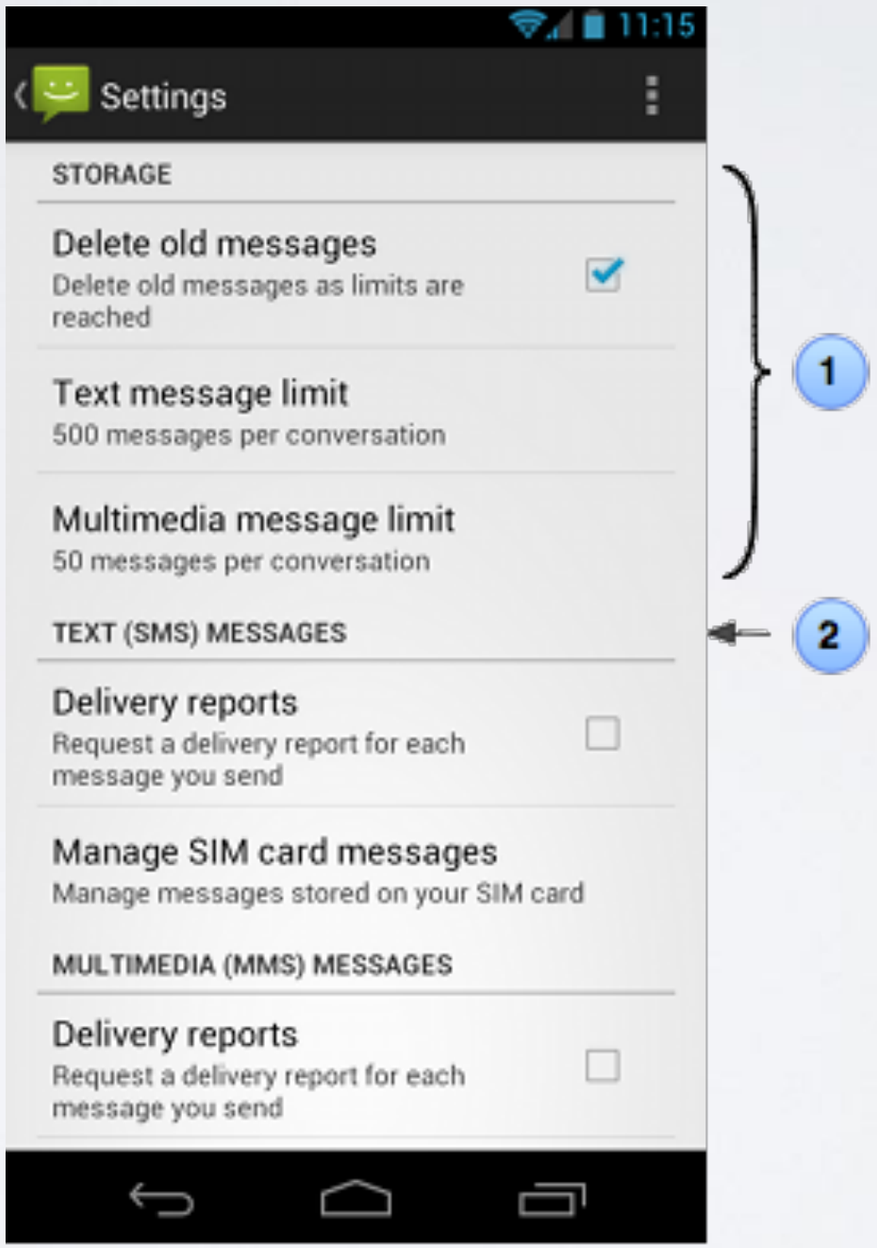
SETTINGS

- Preference instead of View
  - CheckboxPreference
  - ListPreference
  - EditTextPreference
- Settings are stores in SharedPreferences

PREFERENCE

```
<?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>
```

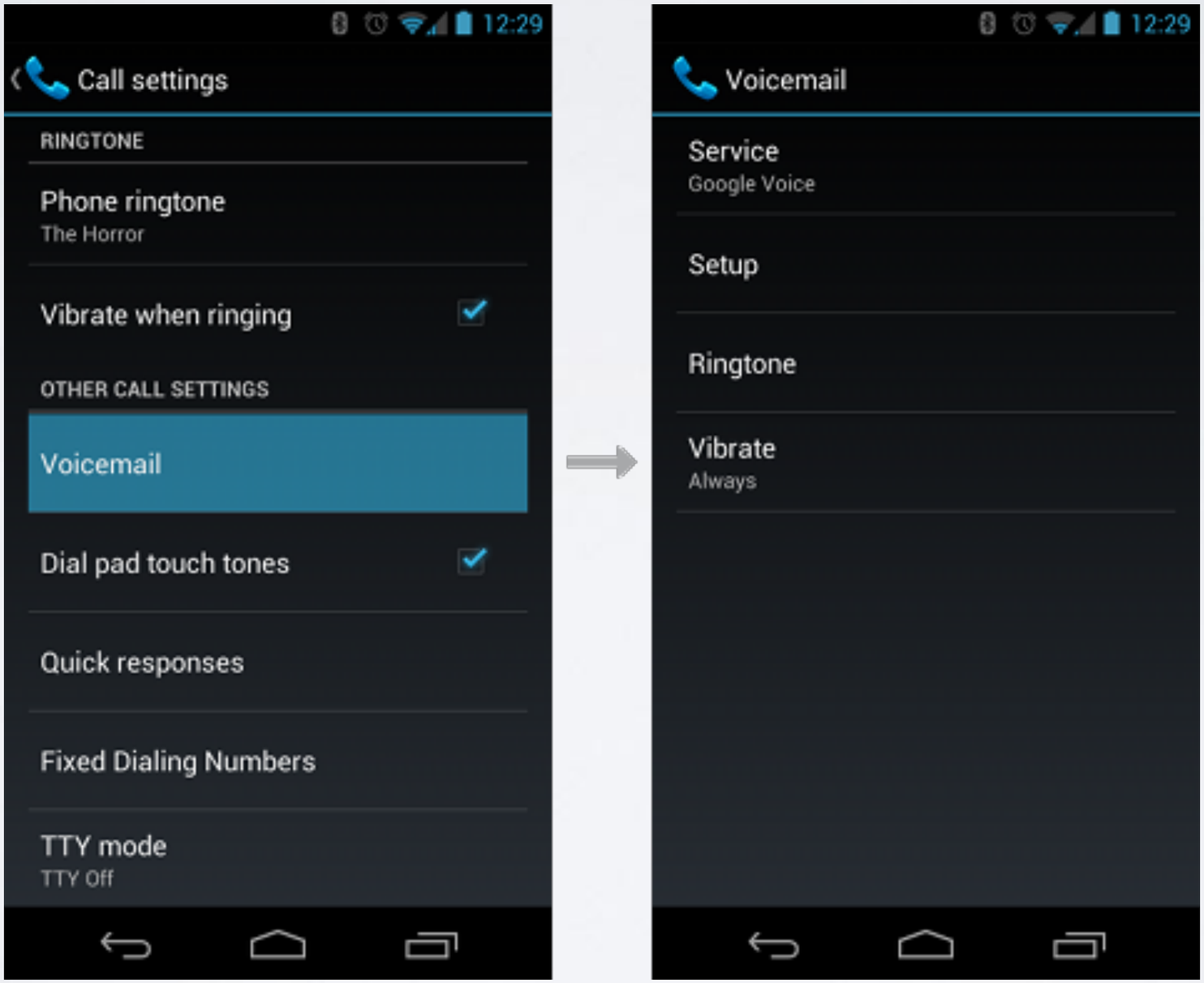
SETTINGS GROUPS



SETTINGS GROUPS

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory
        android:title="@string/pref_sms_storage_title"
        android:key="pref_key_storage_settings">
        <CheckBoxPreference
            android:key="pref_key_auto_delete"
            android:summary="@string/pref_summary_auto_delete"
            android:title="@string/pref_title_auto_delete"
            android:defaultValue="false"... />
        <Preference
            android:key="pref_key_sms_delete_limit"
            android:dependency="pref_key_auto_delete"
            android:summary="@string/pref_summary_delete_limit"
            android:title="@string/pref_title_sms_delete"... />
        <Preference
            android:key="pref_key_mms_delete_limit"
            android:dependency="pref_key_auto_delete"
            android:summary="@string/pref_summary_delete_limit"
            android:title="@string/pref_title_mms_delete" ... />
    </PreferenceCategory>
    ...
</PreferenceScreen>
```

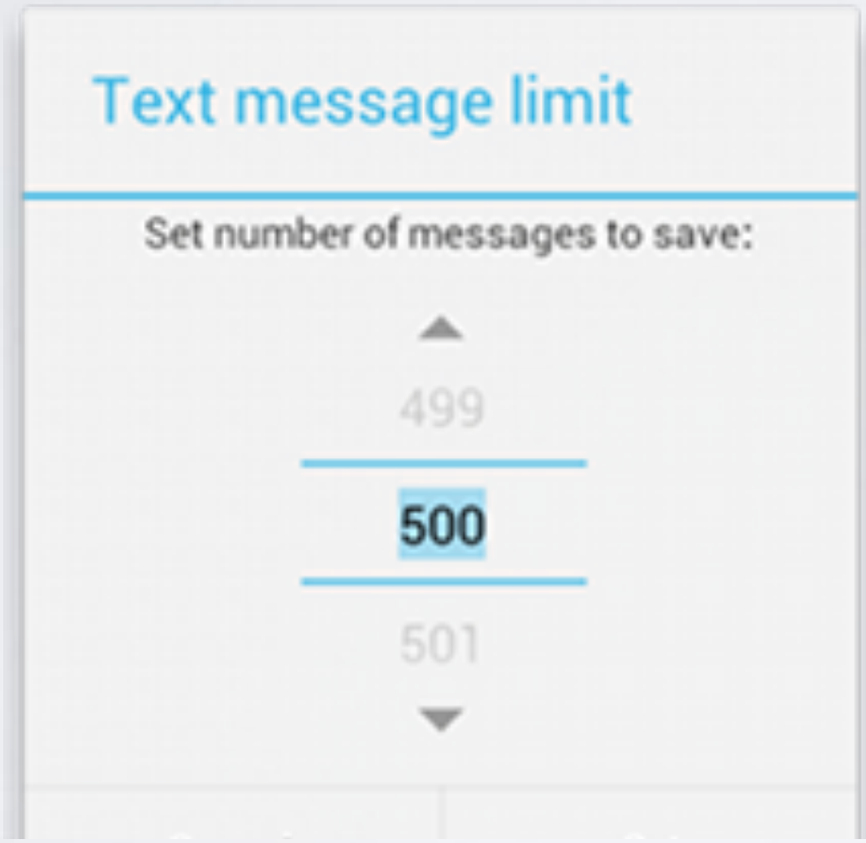
SETTINGS SUBSCREENS

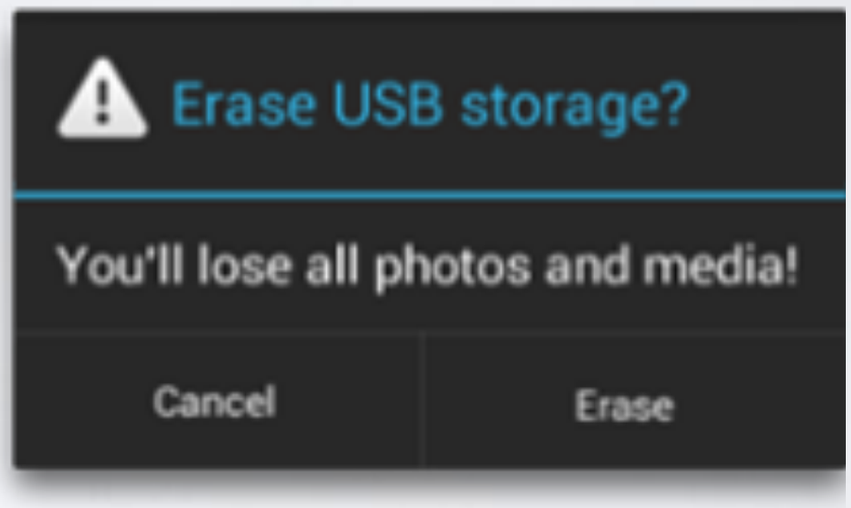


SETTINGS SUBSCREENS

```
<PreferenceScreen  xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- opens a subscreen of settings -->
    <PreferenceScreen
        android:key="button_voicemail_category_key"
        android:title="@string/voicemail"
        android:persistent="false">
        <ListPreference
            android:key="button_voicemail_provider_key"
            android:title="@string/voicemail_provider" ... />
        <!-- opens another nested subscreen -->
        <PreferenceScreen
            android:key="button_voicemail_setting_key"
            android:title="@string/voicemail_settings"
            android:persistent="false">
            ...
        </PreferenceScreen>
        <RingtonePreference
            android:key="button_voicemail_ringtone_key"
            android:title="@string/voicemail_ringtone_title"
            android:ringtoneType="notification" ... />
        ...
    </PreferenceScreen>
    ...
</PreferenceScreen>
```

DIALOGS


Text message limit
Set number of messages to save:
499
500
501

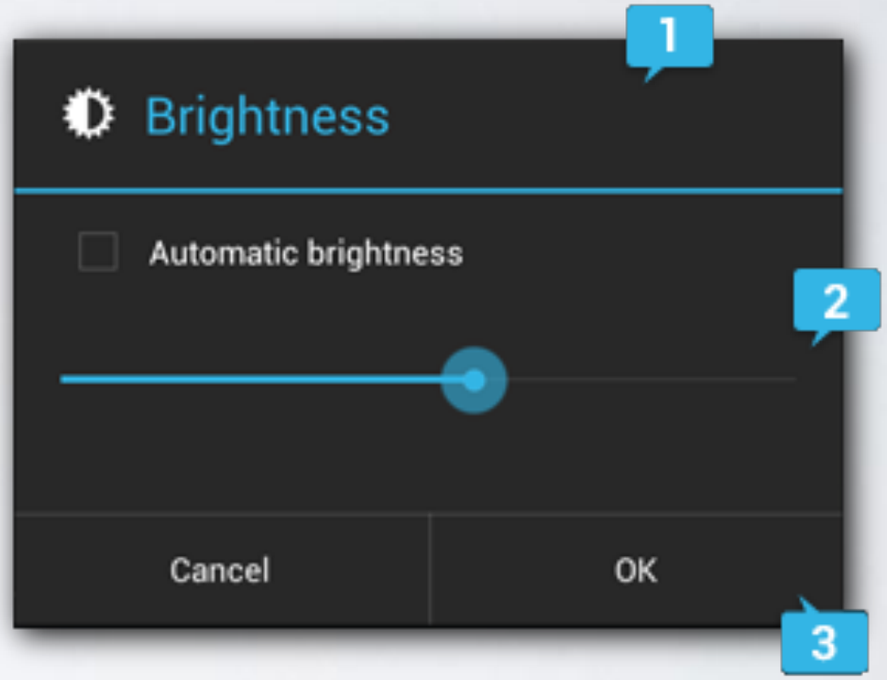

Erase USB storage?
You'll lose all photos and media!
Cancel
Erase

CREATING A DIALOG

```
public class FireMissilesDialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Use the Builder class for convenient dialog construction
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        builder.setMessage(R.string.dialog_fire_missiles)
               .setPositiveButton(R.string.fire, new DialogInterface.OnClickListener()
{
                   public void onClick(DialogInterface dialog, int id) {
                       // FIRE ZE MISSILES!
                   }
               })
               .setNegativeButton(R.string.cancel, new
                   DialogInterface.OnClickListener() {
                       public void onClick(DialogInterface dialog, int id) {
                           // User cancelled the dialog
                       }
                   });
        // Create the AlertDialog object and return it
        return builder.create();
    }
}
```

BUILDING AN ALERT DIALOG

```
// 1. Instantiate an AlertDialog.Builder with
// its constructor
AlertDialog.Builder builder = new
    AlertDialog.Builder(getActivity());

// 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()
AlertDialog dialog = builder.create();
```

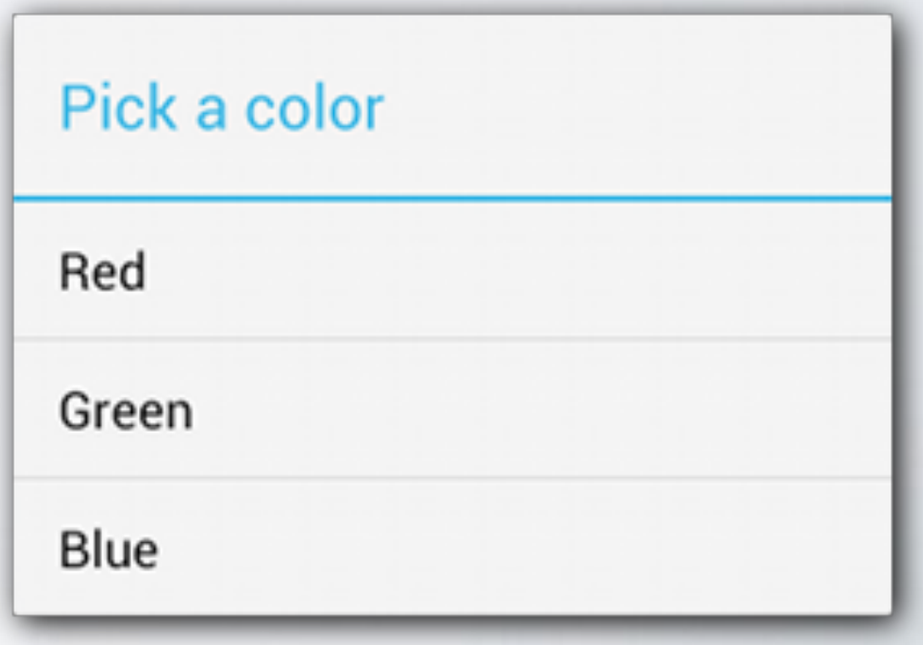


BUILDING AN ALERT DIALOG ADDING BUTTONS

```
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
// Add the buttons
builder.setPositiveButton(R.string.ok, new
DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int id) {
               // User clicked OK button
           }
       });
builder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int id) {
               // User cancelled the dialog
           }
       });
// Set other dialog properties
...

// Create the AlertDialog
AlertDialog dialog = builder.create();
```

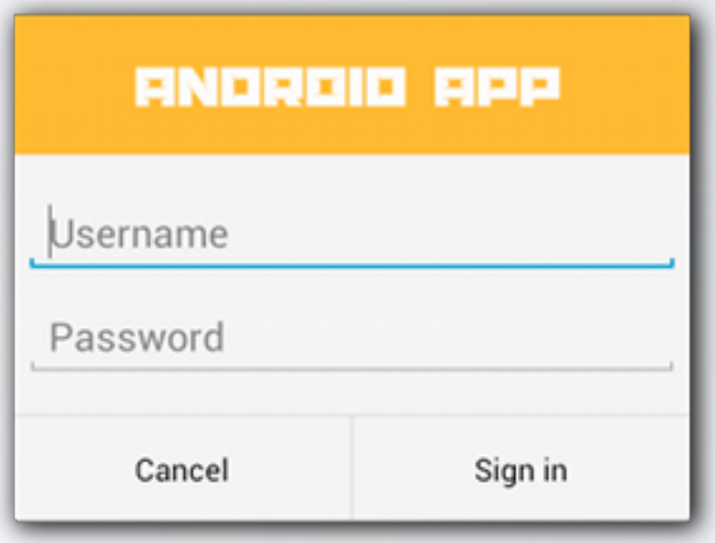
BUILDING AN ALERT DIALOG ADDING A LIST

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder = new
        AlertDialog.Builder(getActivity());
    builder.setTitle(R.string.pick_color)
           .setItems(R.array.colors_array, new
        DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int
                   which) {
                   // The 'which' argument contains the
                   // index position
                   // of the selected item
               }
           });
    return builder.create();
}
```



CREATING A CUSTOM LAYOUT

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <ImageView
        android:src="@drawable/header_logo"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:scaleType="center"
        android:background="#FFFFBB33"
        android:contentDescription="@string/app_name" />
    <EditText
        android:id="@+id/username"
        android:inputType="textEmailAddress"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="4dp"
        android:layout_marginBottom="4dp"
        android:hint="@string/username" />
    <EditText
        android:id="@+id/password"
        android:inputType="textPassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="4dp"
        android:layout_marginBottom="16dp"
        android:fontFamily="sans-serif"
        android:hint="@string/password"/>
</LinearLayout>
```



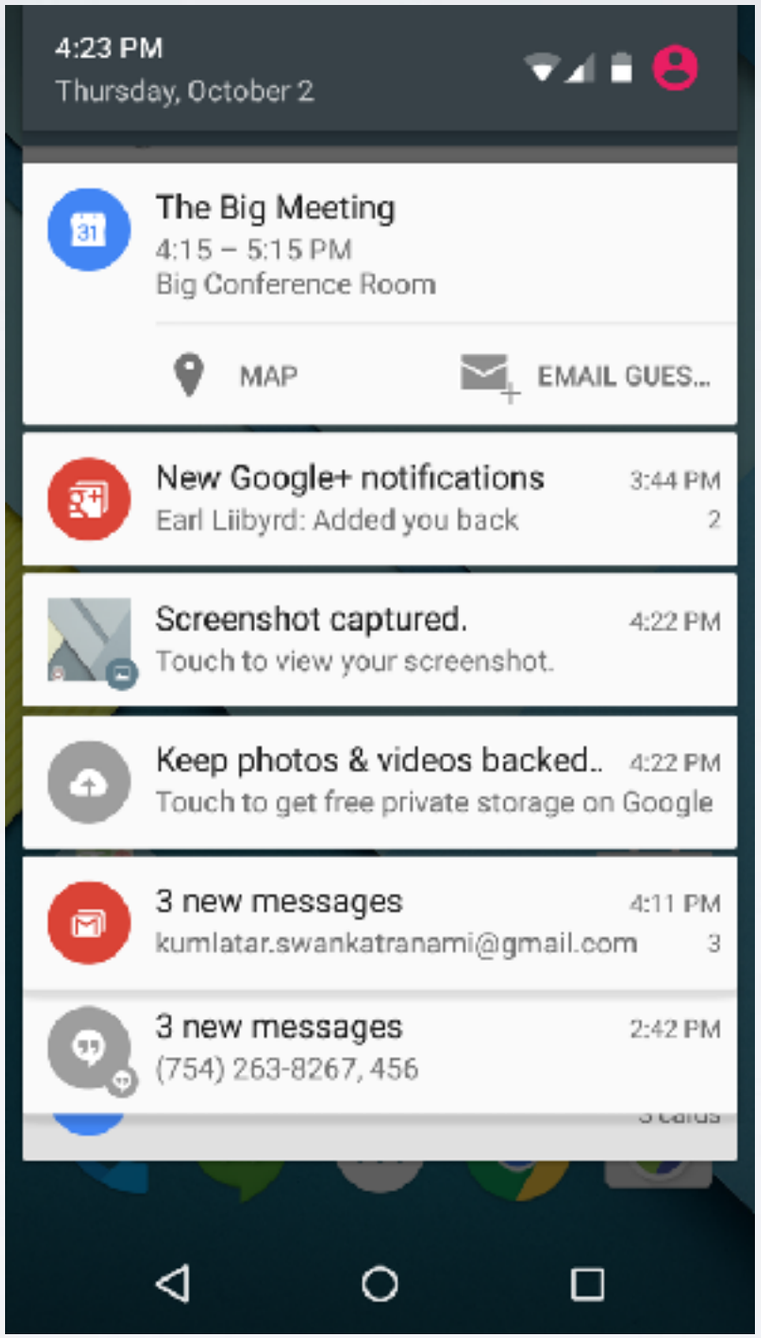
CREATING A CUSTOM LAYOUT

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    // Get the layout inflater
    LayoutInflater inflater = getActivity().getLayoutInflater();

    // Inflate and set the layout for the dialog
    // Pass null as the parent view because its going in the dialog layout
    builder.setView(inflater.inflate(R.layout.dialog_signin, null))
    // Add action buttons
           .setPositiveButton(R.string.signin, new DialogInterface.OnClickListener() {
               @Override
               public void onClick(DialogInterface dialog, int id) {
                   // sign in the user ...
               }
           })
           .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int id) {
                   LoginDialogFragment.this.getDialog().cancel();
               }
           });
    return builder.create();
}
```

SHOWING A DIALOG

```
public void confirmFireMissiles() {
    DialogFragment newFragment = new
        FireMissilesDialogFragment();
    newFragment.show(getSupportFragmentManager(),
        "missiles");
}
```

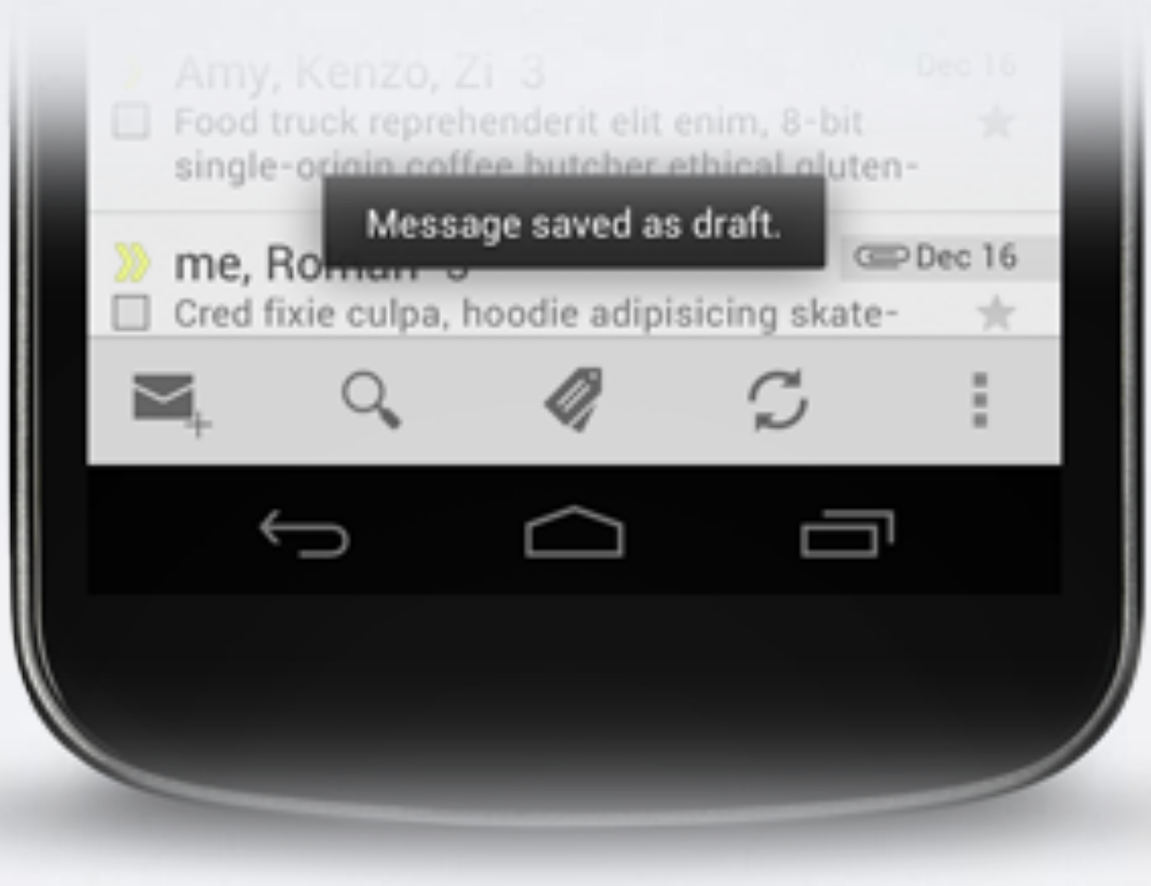
NOTIFICATIONS



NOTIFICATIONS

```
NotificationCompat.Builder mBuilder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("My notification")
        .setContentText("Hello World!");
// Creates an explicit intent for an Activity in your app
Intent resultIntent = new Intent(this, ResultActivity.class);

// The stack builder object will contain an artificial back stack for the
// started Activity.
// This ensures that navigating backward from the Activity leads out of
// your application to the Home screen.
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
// Adds the back stack for the Intent (but not the Intent itself)
stackBuilder.addParentStack(ResultActivity.class);
// Adds the Intent that starts the Activity to the top of the stack
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent =
        stackBuilder.getPendingIntent(
            0,
            PendingIntent.FLAG_UPDATE_CURRENT
        );
mBuilder.setContentIntent(resultPendingIntent);
NotificationManager mNotificationManager =
    (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
// mId allows you to update the notification later on.
mNotificationManager.notify(mId, mBuilder.build());
```

TOASTS


Amy, Kenzo, Zi 3
Dec 16
Food truck reprehenderit elit enim, 8-bit
single-origin coffee butcher ethical gluten-
Message saved as draft.
me, Roman 3
Dec 16
Cred fixie culpa, hoodie adipisicing skate-

TOASTS

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

POSITIONING A TOAST

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

CREATING A CUSTOM TOAST

```
// toast_layout_root.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/toast_layout_root"
              android:orientation="horizontal"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:padding="8dp"
              android:background="#DAAA"
              >
    <ImageView android:src="@drawable/droid"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:layout_marginRight="8dp"
               />
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:textColor="#FFF"
              />
</LinearLayout>
```

CREATING A CUSTOM TOAST

```
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.custom_toast,
                               (ViewGroup)
findViewById(R.id.toast_layout_root));

TextView text = (TextView) layout.findViewById(R.id.text);
text.setText("This is a custom toast");

Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
toast.setView(layout);
toast.show();
```

LAYOUTS

- XML description of how the components are placed on the screen
- Android Activity Layouts:
  - Linear
  - Relative
  - List
  - Grid

LINEAR LAYOUT

RELATIVE LAYOUT

LIST LAYOUT

GRID LAYOUT

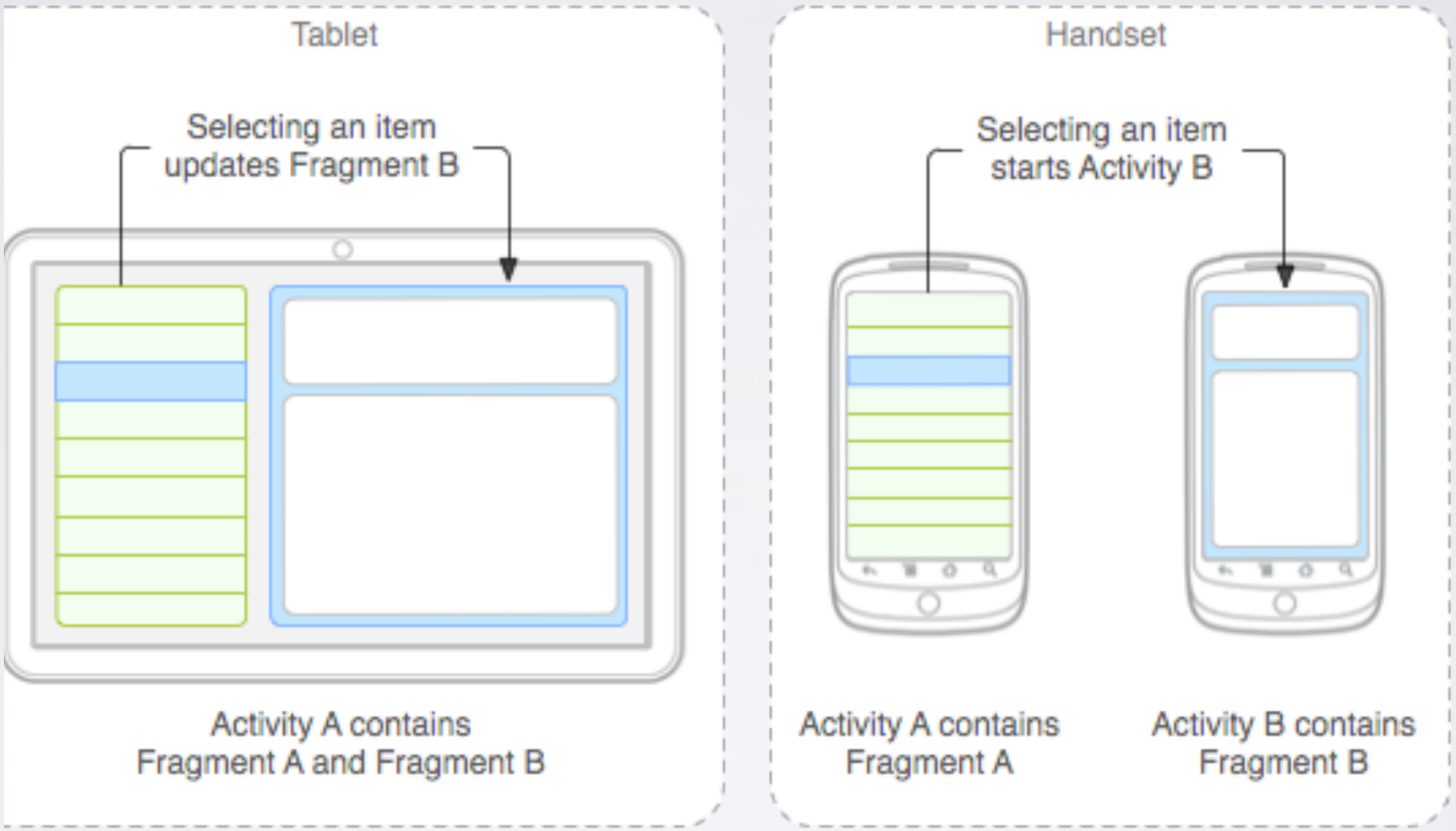
FRAGMENTS

- represents a part of UI
- has its own lifecycle
- is reusable
- is embedded in activity

FRAGMENTS

```
<fragment
android:name=”pl.edu.pjatk.FragmentExample”
        android:id="@+id/list”
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />


public class FragmentExample extends Fragment {
   //…
}
```

FRAGMENTS - EXAMPLE



FRAGMENTMANAGER

- allows to access fragment from the activity
- popBackStack() - simulation of "back button"
- starts the transaction on fragment

FRAGMENT TRANSACTIONS

- adding, removing, replacing the components in the fragment at once
- beginTransaction()
- commit

FRAGMENT TRANSACTIONS

```
Fragment newFragment = new ExampleFragment();
FragmentTransaction transaction =
        getFragmentManager().beginTransaction();

transaction.replace(R.id.fragment_container,
        newFragment);

transaction.commit();
```

PASSING THE EVENTS TO THE CALLER

```
public class NoticeDialogFragment extends DialogFragment {

    /* The activity that creates an instance of this dialog fragment must
     * implement this interface in order to receive event callbacks.
     * Each method passes the DialogFragment in case the host needs to query it. */
    public interface NoticeDialogListener {
        public void onDialogPositiveClick(DialogFragment dialog);
        public void onDialogNegativeClick(DialogFragment dialog);
    }

    // Use this instance of the interface to deliver action events
    NoticeDialogListener mListener;

    // Override the Fragment.onAttach() method to instantiate the NoticeDialogListener
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        // Verify that the host activity implements the callback interface
        try {
            // Instantiate the NoticeDialogListener so we can send events to the host
            mListener = (NoticeDialogListener) activity;
        } catch (ClassCastException e) {
            // The activity doesn't implement the interface, throw exception
            throw new ClassCastException(activity.toString()
                    + " must implement NoticeDialogListener");
        }
    }
    ...
}
```

PASSING THE EVENTS TO THE CALLER

```
public class MainActivity extends FragmentActivity
                          implements NoticeDialogFragment.NoticeDialogListener{
    ...

    public void showNoticeDialog() {
        // Create an instance of the dialog fragment and show it
        DialogFragment dialog = new NoticeDialogFragment();
        dialog.show(getSupportFragmentManager(), "NoticeDialogFragment");
    }

    // The dialog fragment receives a reference to this Activity through the
    // Fragment.onAttach() callback, which it uses to call the following methods
    // defined by the NoticeDialogFragment.NoticeDialogListener interface
    @Override
    public void onDialogPositiveClick(DialogFragment dialog) {
        // User touched the dialog's positive button
        ...
    }

    @Override
    public void onDialogNegativeClick(DialogFragment dialog) {
        // User touched the dialog's negative button
        ...
    }
}
```

PASSING THE EVENTS TO THE CALLER

```
public class NoticeDialogFragment extends DialogFragment {
    ...

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Build the dialog and set up the button click handlers
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        builder.setMessage(R.string.dialog_fire_missiles)
               .setPositiveButton(R.string.fire, new DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int id) {
                       // Send the positive button event back to the host activity
                       mListener.onDialogPositiveClick(NoticeDialogFragment.this);
                   }
               })
               .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int id) {
                       // Send the negative button event back to the host activity
                       mListener.onDialogNegativeClick(NoticeDialogFragment.this);
                   }
               });
        return builder.create();
    }
}
```

CREDITS

All the code snippets and pictures come from:

https://developer.android.com/guide/topics/ui/index.html