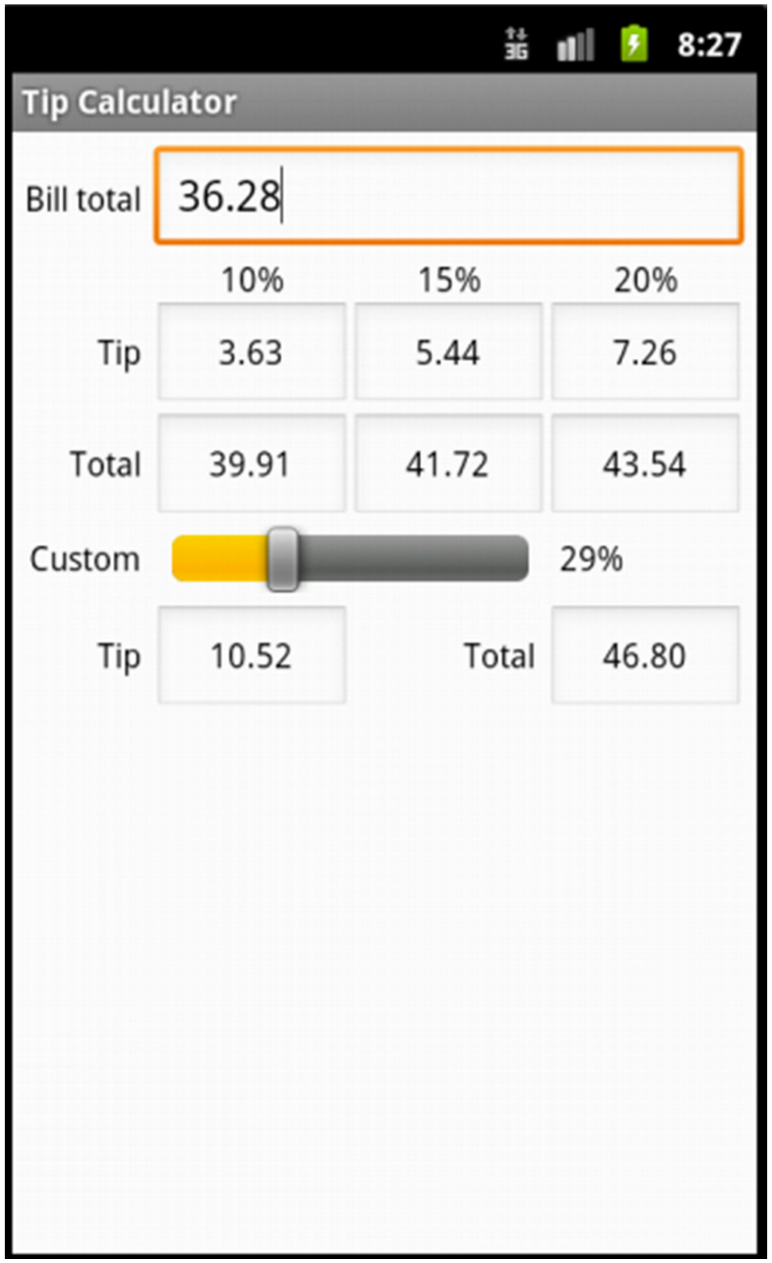
CS378 - Mobile Computing

More UI

Concrete Example

- Tip Calculator
- What kind of layout to use?
- Widgets:
  - TextView
  - EditText
  - SeekBar



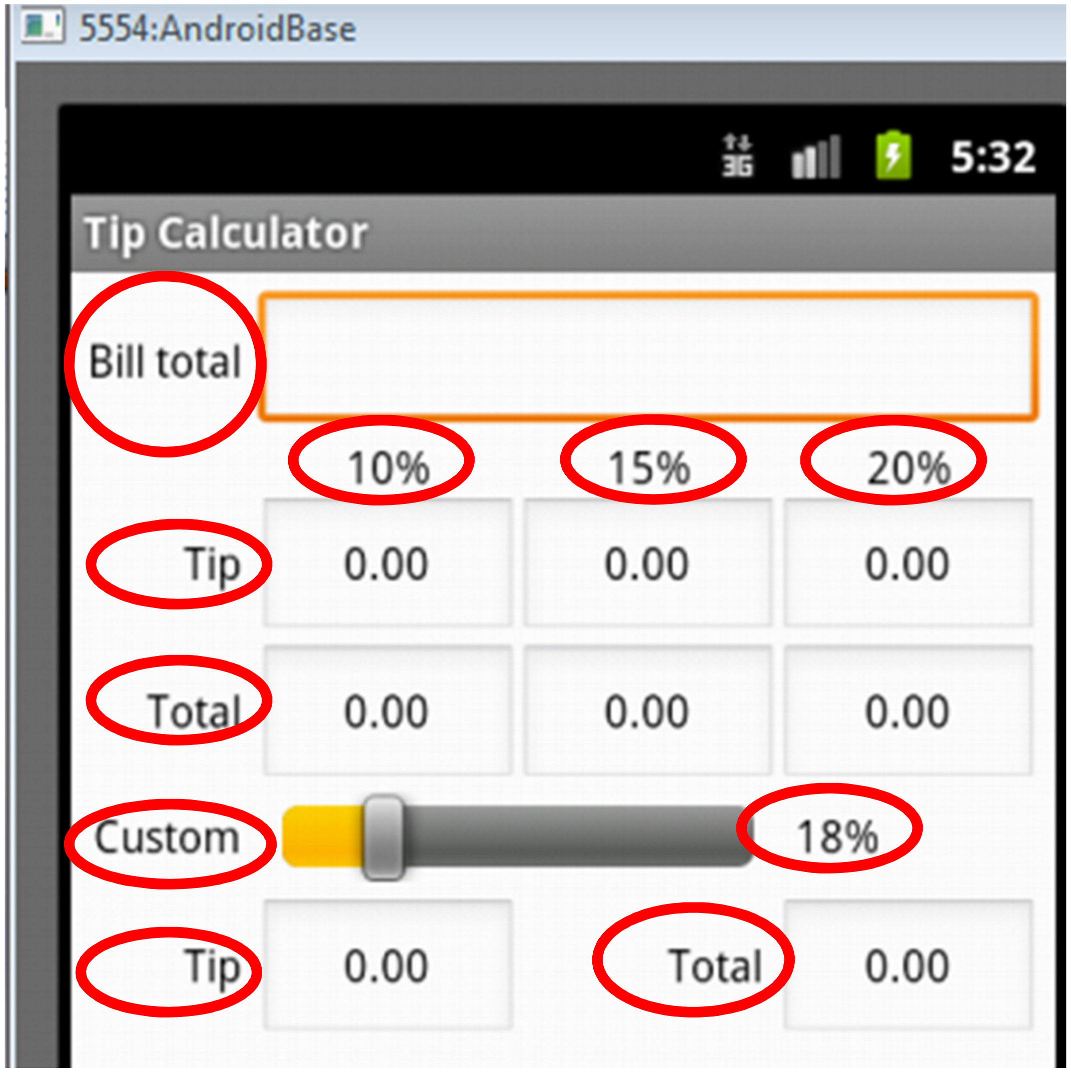
Bill total 36.28

	10%	15%	20%
Tip	3.63	5.44	7.26
Total	39.91	41.72	43.54

Custom 29%

Tip	10.52	Total	46.80

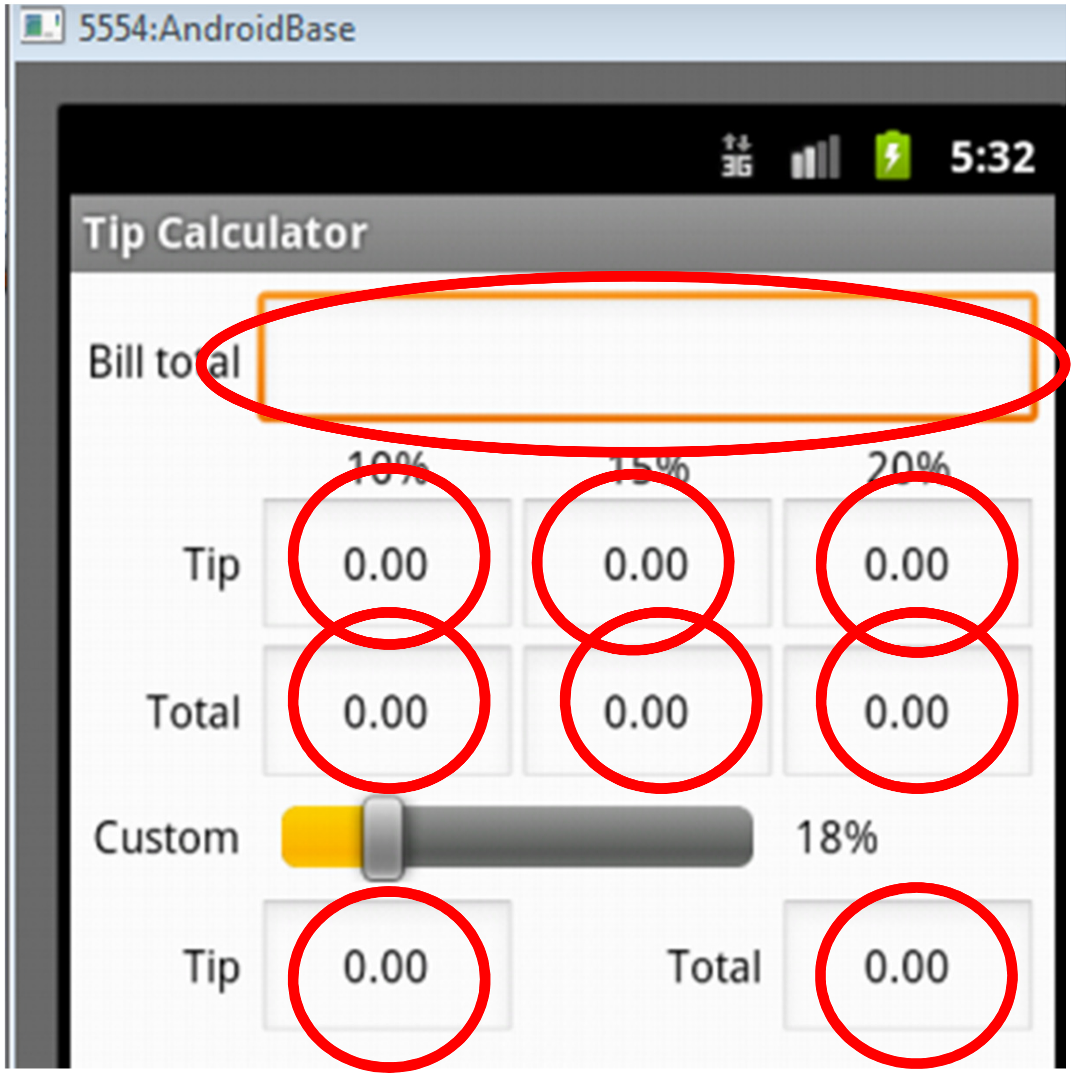
TextViews


5554:AndroidBase
5:32
Tip Calculator
Bill total
10%
15%
20%
Tip
0.00
0.00
0.00
Total
0.00
0.00
0.00
Custom
18%
Tip
0.00
Total
0.00

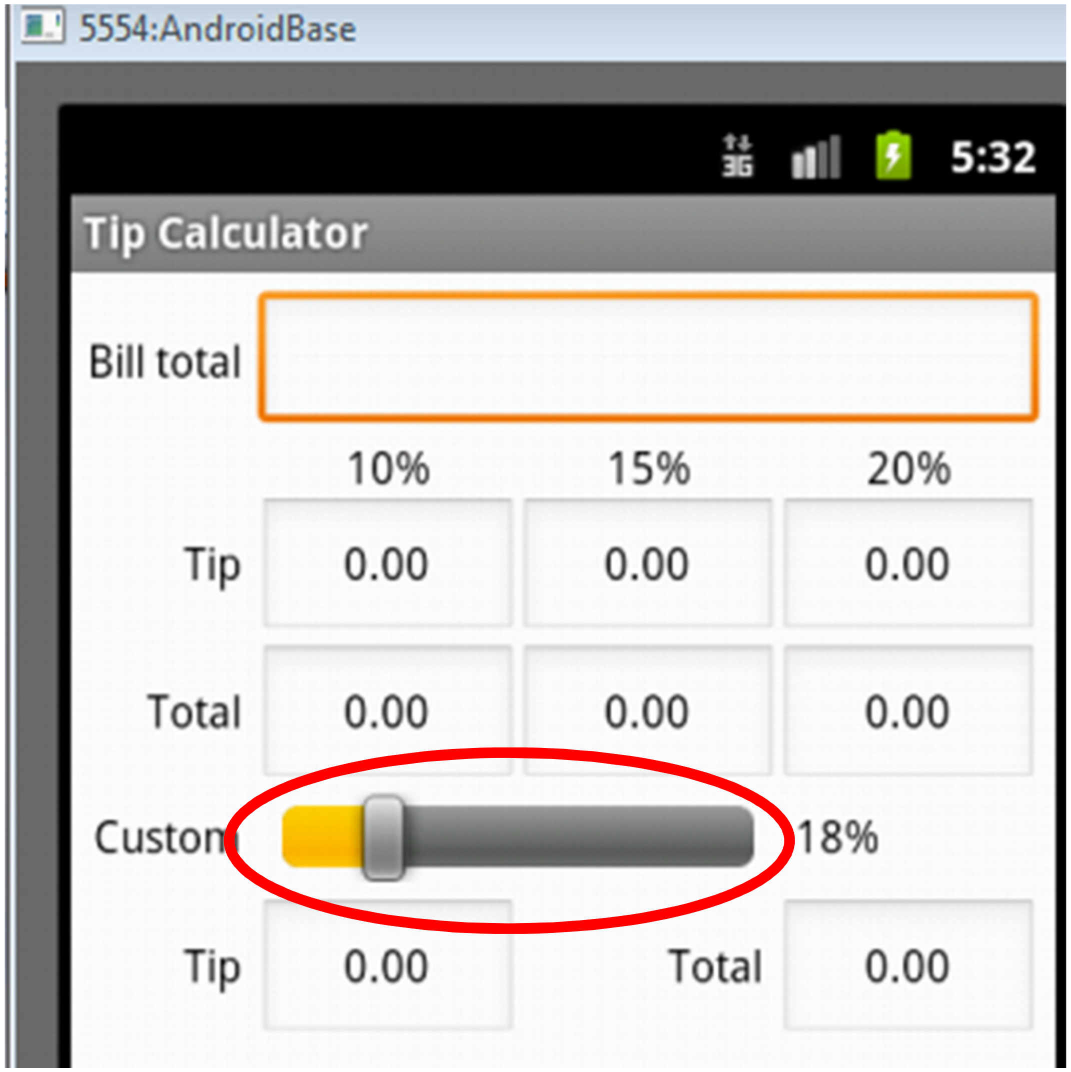
EditText

All but top EditText are uneditable

Alternative? TextViews?

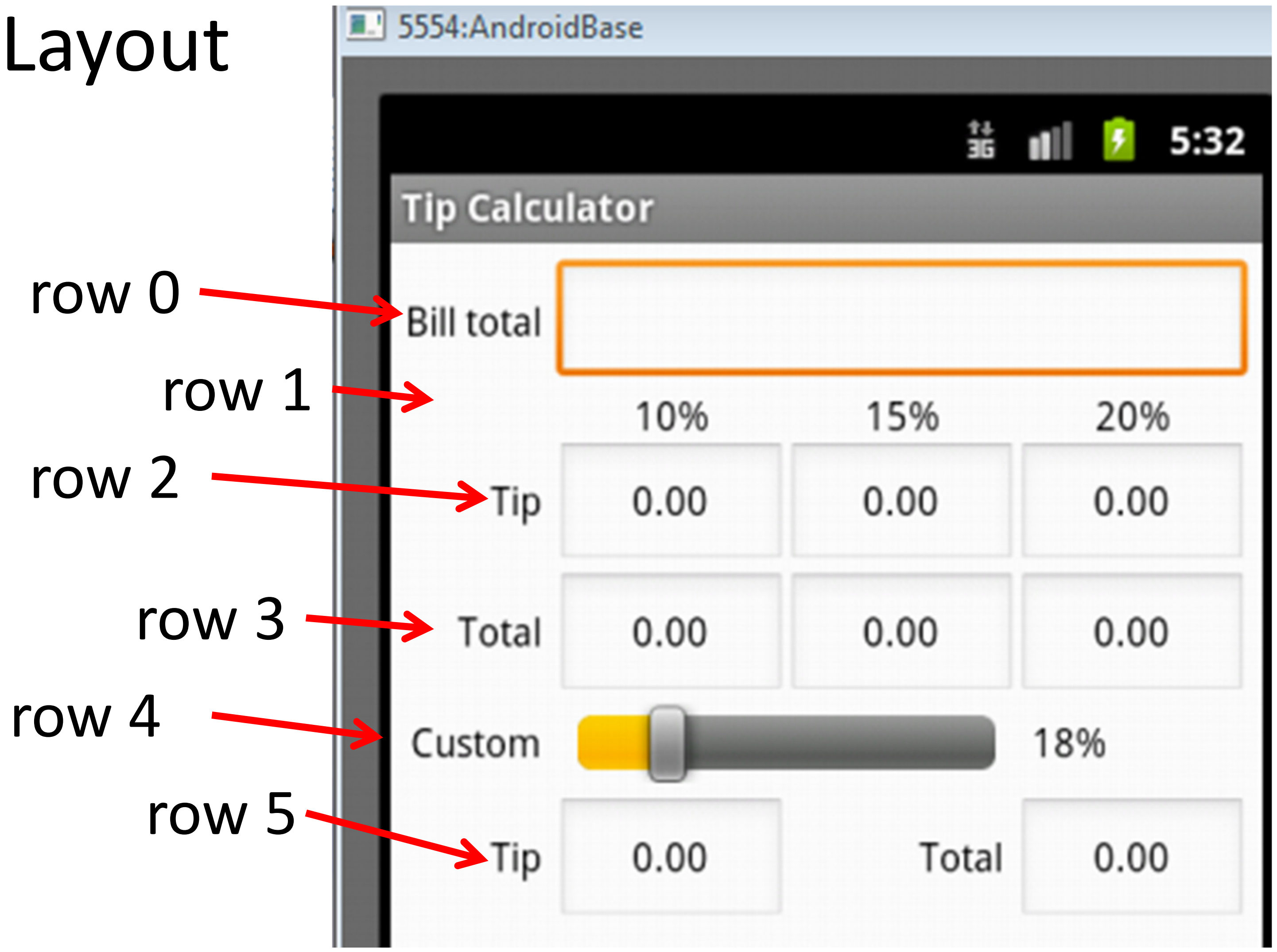


SeekBar


5554:AndroidBase
5:32
Tip Calculator
Bill total
10%
15%
20%
Tip
0.00
0.00
0.00
Total
0.00
0.00
0.00
Custom
18%
Tip
0.00
Total
0.00

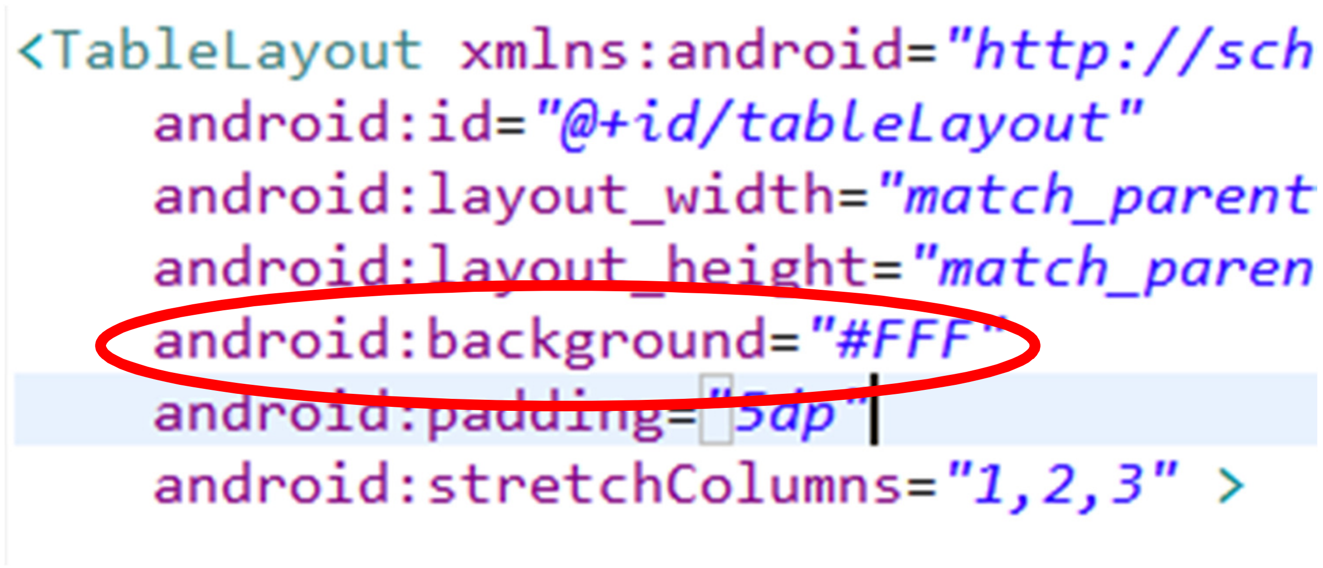
Layout

- TableLayout



Bill total			
	10%	15%	20%
Tip	0.00	0.00	0.00
Total	0.00	0.00	0.00
Custom			18%
Tip	0.00	Total	0.00

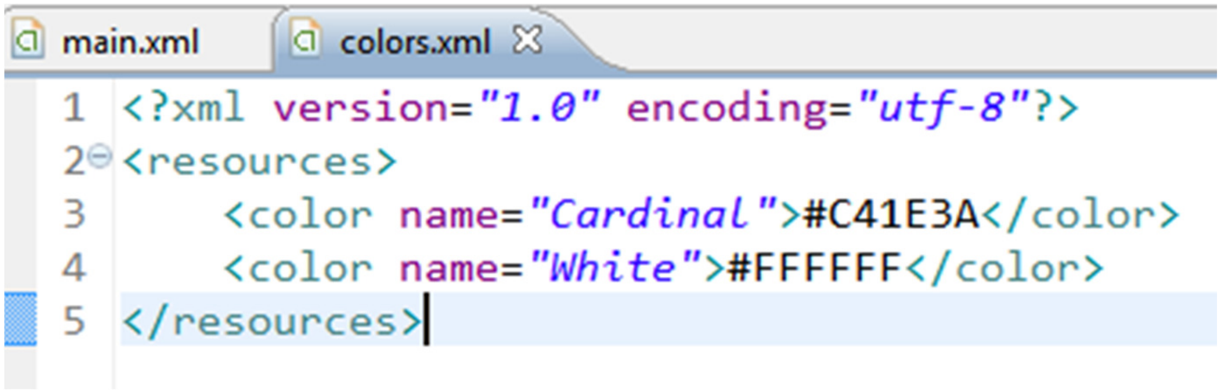
Layout Attributes



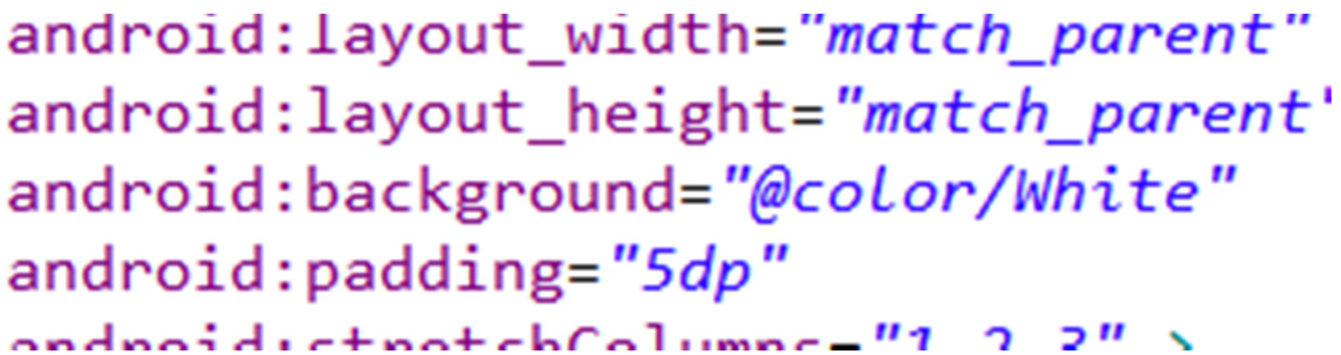
```
<TableLayout xmlns:android="http://sch
    android:id="@+id/tableLayout"
    android:layout_width="match_parent
    android:layout_height="match_paren
    android:background="#FFF"
    android:padding="5dp"
    android:stretchColumns="1,2,3" >
```

- android:background
  - #RGB, #ARGB, #RRGGBB, #AARRGGBB
  - can place colors in res/values/colors.xml

Color Resources



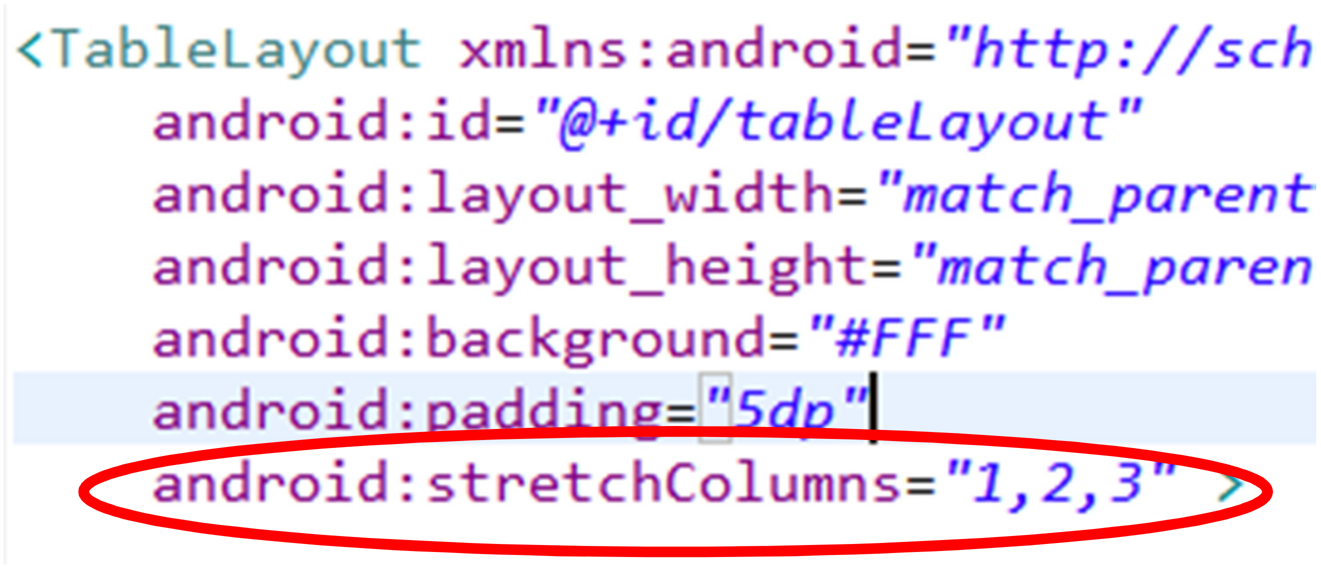
```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="Cardinal">#C41E3A</color>
    <color name="White">#FFFFFF</color>
</resources>
```



```xml
android:layout_width="match_parent"
android:layout_height="match_parent'
android:background="@color/White"
android:padding="5dp"
android:stretchColumns="1, 2, 3" >
```

- Good Resource / W3C colors
  – http://tinyurl.com/6py9huk

StretchColumns

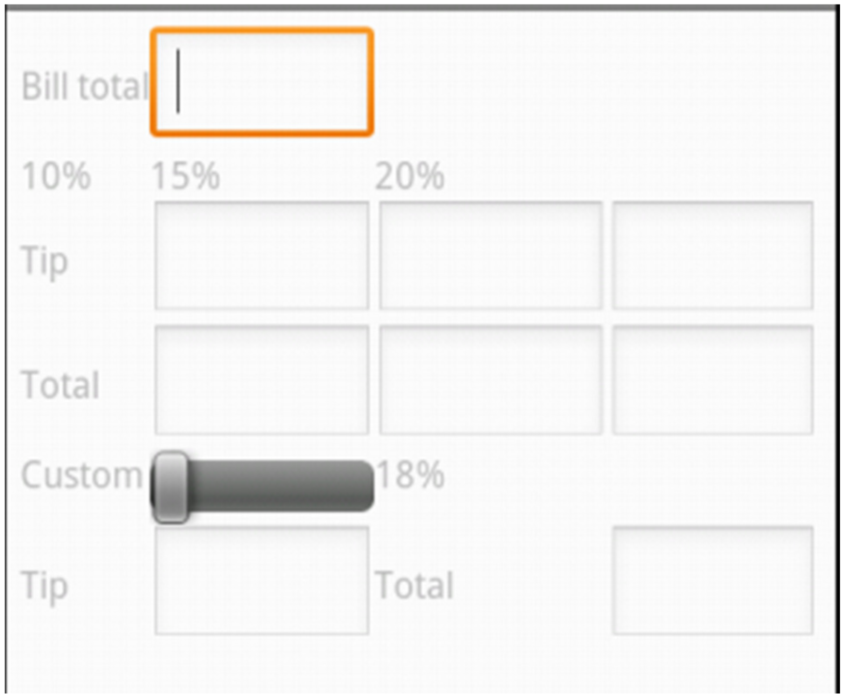


```
<TableLayout xmlns:android="http://sch
    android:id="@+id/tableLayout"
    android:layout_width="match_parent
    android:layout_height="match_paren
    android:background="#FFF"
    android:padding="5dp"
    android:stretchColumns="1,2,3" >
```

- columns 0 indexed
- columns 1, 2, 3 stretch to fill layout width
- column 0 wide as widest element, plus any padding for that element

Initial UI

- Done via some Drag and Drop, Outline view, and editing XML
- Demo outline view
  - properties

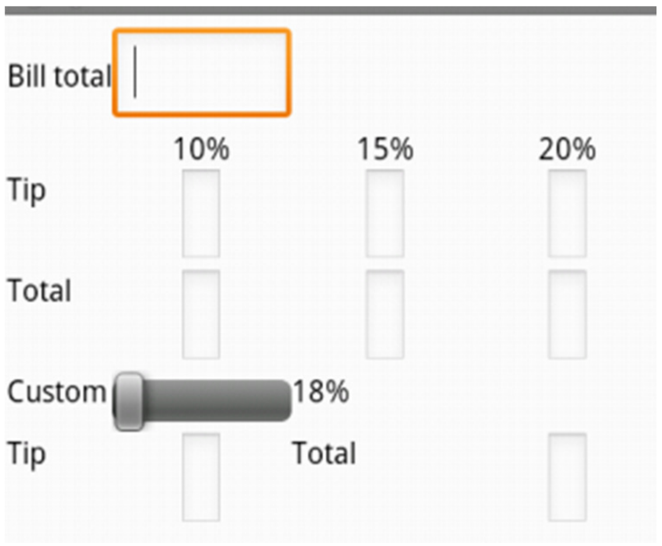


Changes to UI

- Outline multiple select properties
  - all TextViews' textColor set to black #000000
- change column for %DD labels

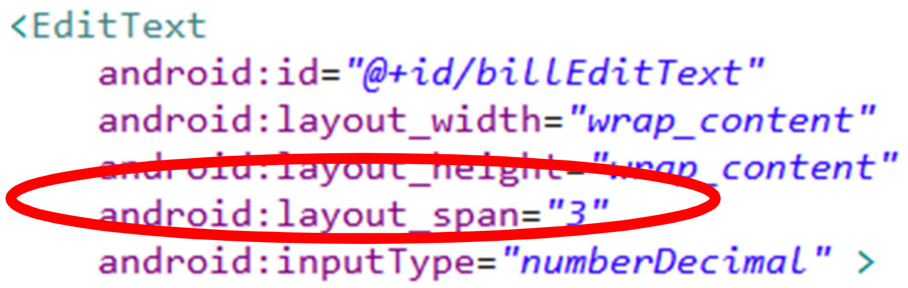
```
android:text="10%"
android:layout_column="1"
android:textColor="#000000" />
```

- use center gravity for components



Changes to UI

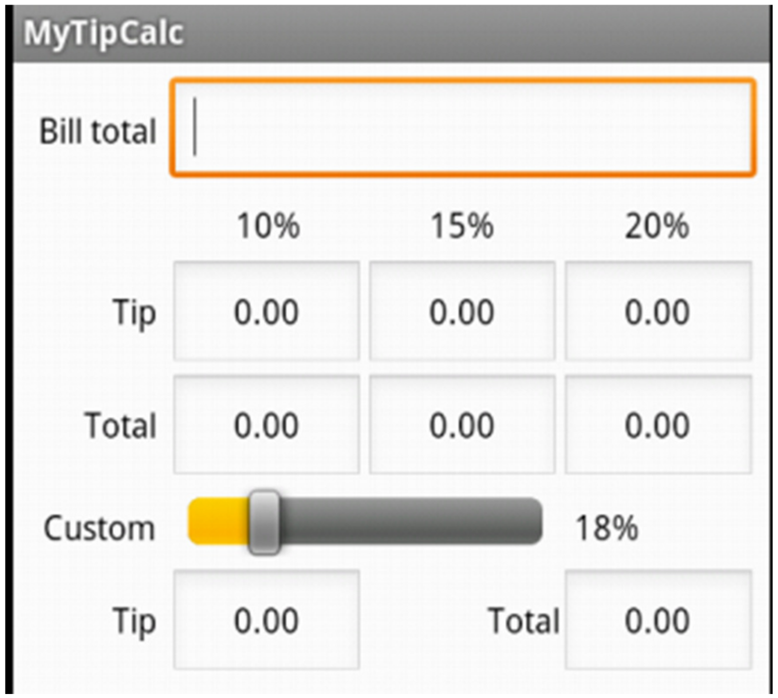
- change bill total and seekbar to span more columns
- gravity and padding for text in column 0
- align text with seekBar
- set seekBar progress to 18
- set seekBar focusable to false - keep keyboard on screen



```
<EditText
    android:id="@+id/billEditText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_span="3"
    android:inputType="numberDecimal" >
```

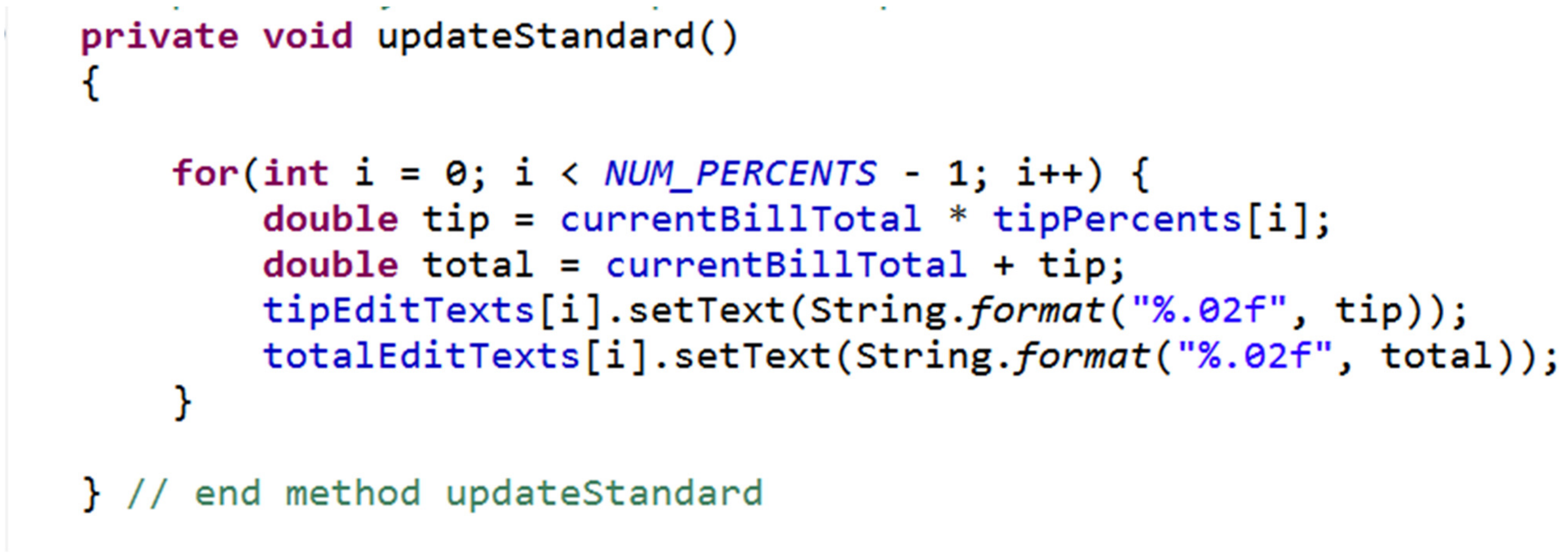
Changes to UI

- Prevent Editing in EditText
  - focusable, long clickable, and cursor visible properties to false
- Set text in EditText to 0.00
- Change weights to 1 to spread out



Functionality

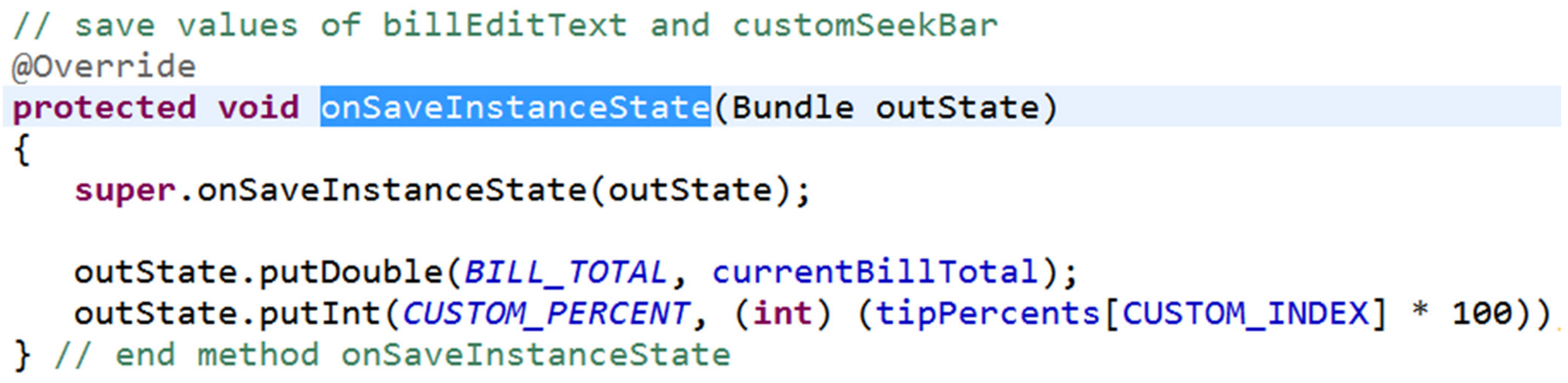
- onCreate instance variables assigned to components found via ids
- update standard percents:



```
private void updateStandard()
{

    for(int i = 0; i < NUM_PERCENTS - 1; i++) {
        double tip = currentBillTotal * tipPercents[i];
        double total = currentBillTotal + tip;
        tipEditTexts[i].setText(String.format("%.02f", tip));
        totalEditTexts[i].setText(String.format("%.02f", total));
    }

} // end method updateStandard
```

Functionality - Saving State

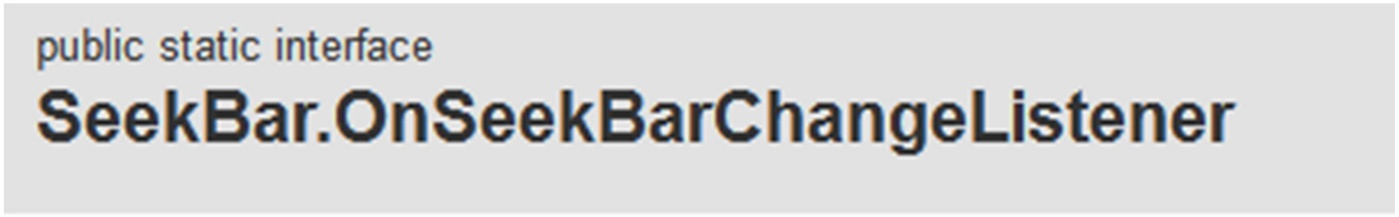
- onSaveInstance
  - save BillTotal and CustomPercent to the Bundle
  - check for these in onCreate



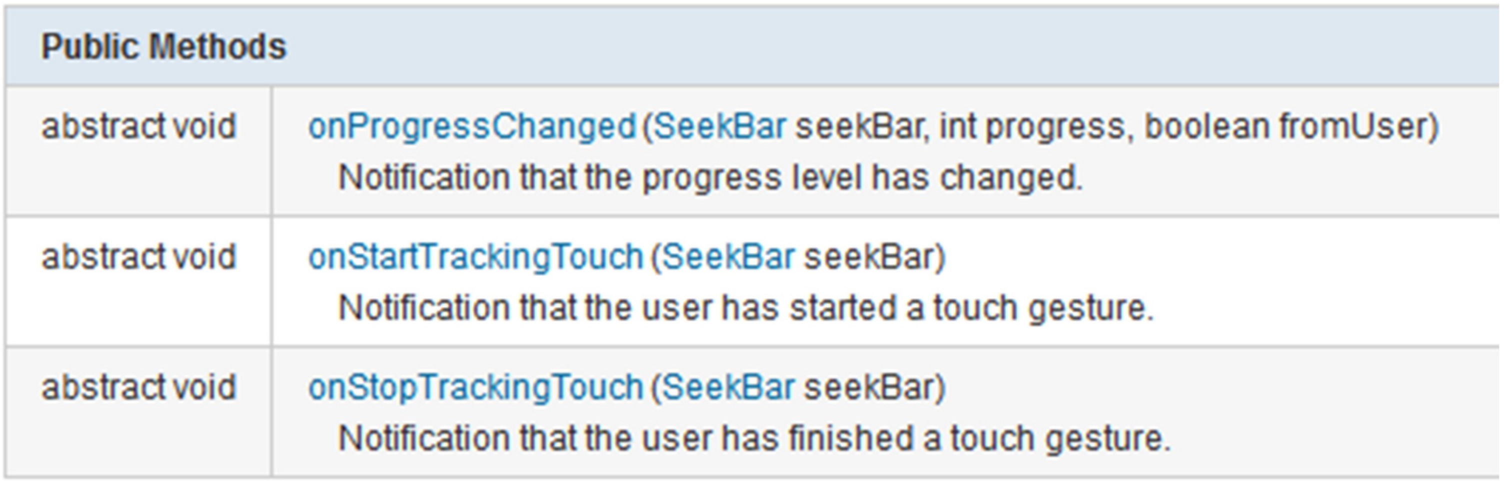
```java
// save values of billEditText and customSeekBar
@Override
protected void onSaveInstanceState(Bundle outState)
{
   super.onSaveInstanceState(outState);

   outState.putDouble(BILL_TOTAL, currentBillTotal);
   outState.putInt(CUSTOM_PERCENT, (int) (tipPercents[CUSTOM_INDEX] * 100))
} // end method onSaveInstanceState
```

Functionality Responding to SeekBar

- customSeekBarListener instance variable
- Of type OnSeekBarChangeListener



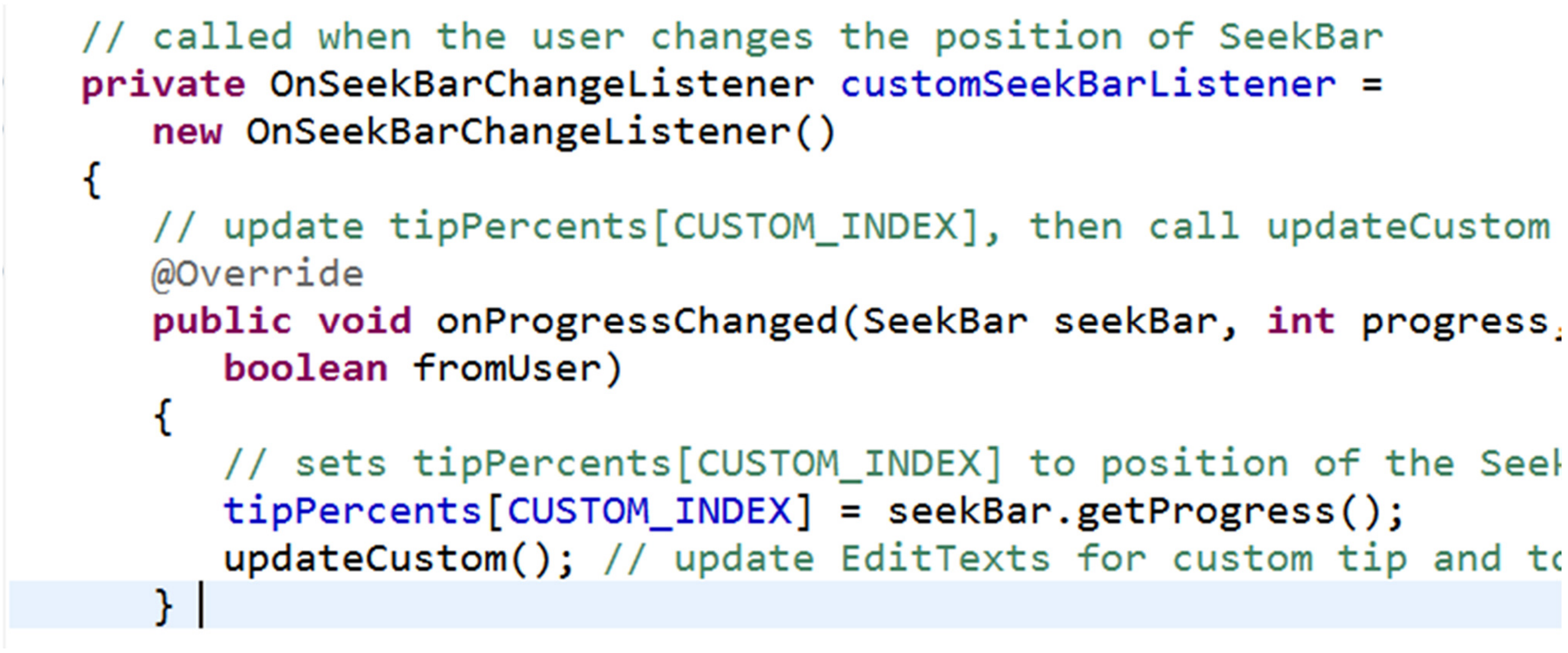
public static interface
SeekBar.OnSeekBarChangeListener



Public Methods	
abstract void	onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) Notification that the progress level has changed.
abstract void	onStartTrackingTouch(SeekBar seekBar) Notification that the user has started a touch gesture.
abstract void	onStopTrackingTouch(SeekBar seekBar) Notification that the user has finished a touch gesture.

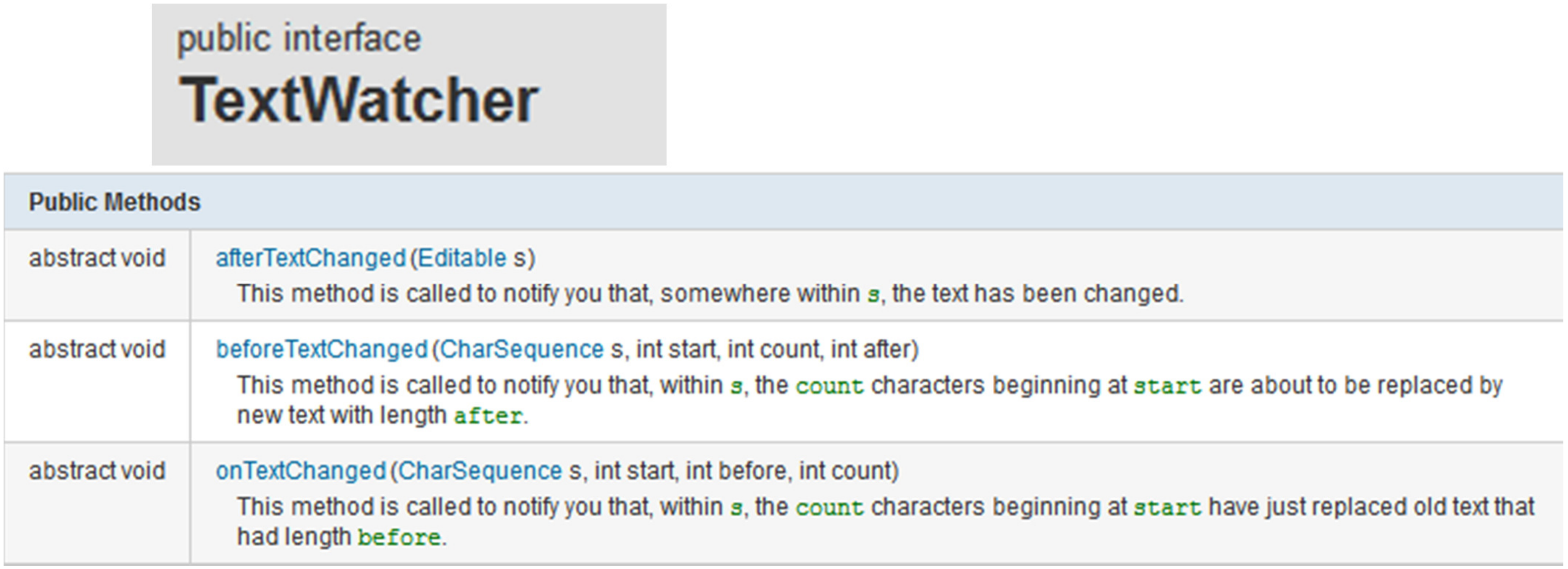
Create an Anonymous Inner Class

- Class notified when seek bar changed and program updates custom tip and total amount
- must register with the seekBar instance variable in onCreate.



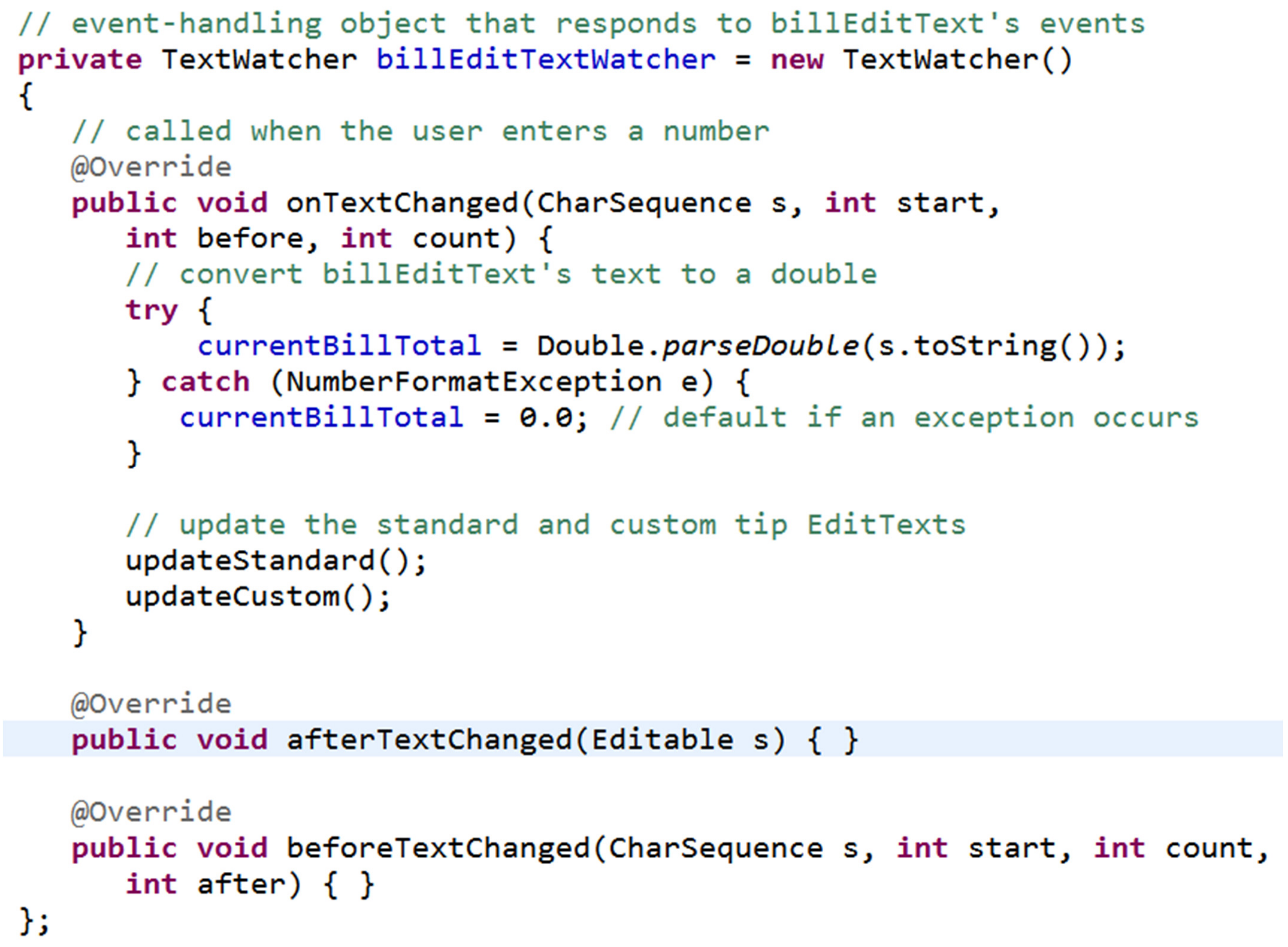
```
// called when the user changes the position of SeekBar
private OnSeekBarChangeListener customSeekBarListener =
   new OnSeekBarChangeListener()
{
   // update tipPercents[CUSTOM_INDEX], then call updateCustom
   @Override
   public void onProgressChanged(SeekBar seekBar, int progress,
      boolean fromUser)
   {
      // sets tipPercents[CUSTOM_INDEX] to position of the Seek
      tipPercents[CUSTOM_INDEX] = seekBar.getProgress();
      updateCustom(); // update EditTexts for custom tip and to
   } |
```

Functionality - Total EditText


public interface
TextWatcher

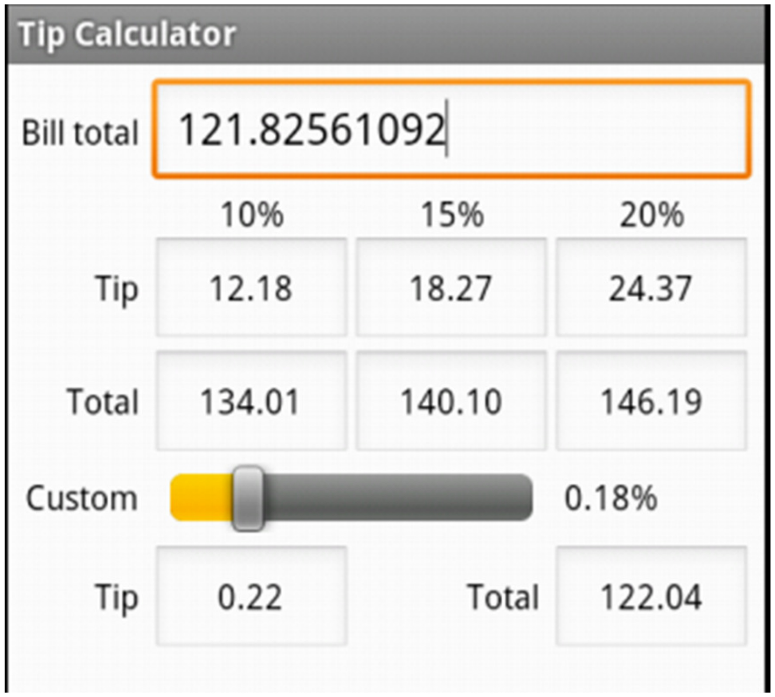
Public Methods	
abstract void	afterTextChanged(Editable s) This method is called to notify you that, somewhere within `s`, the text has been changed.
abstract void	beforeTextChanged(CharSequence s, int start, int count, int after) This method is called to notify you that, within `s`, the `count` characters beginning at `start` are about to be replaced by new text with length `after`.
abstract void	onTextChanged(CharSequence s, int start, int before, int count) This method is called to notify you that, within `s`, the `count` characters beginning at `start` have just replaced old text that had length `before`.

- Another anonymous inner class
- implement onTextChanged to converts to double and call update methods
- register with EditText for total in onCreate()!


```java
// event-handling object that responds to billEditText's events
private TextWatcher billEditTextWatcher = new TextWatcher()
{
   // called when the user enters a number
   @Override
   public void onTextChanged(CharSequence s, int start,
      int before, int count) {
      // convert billEditText's text to a double
      try {
          currentBillTotal = Double.parseDouble(s.toString());
      } catch (NumberFormatException e) {
         currentBillTotal = 0.0; // default if an exception occurs
      }

      // update the standard and custom tip EditTexts
      updateStandard();
      updateCustom();
   }

   @Override
   public void afterTextChanged(Editable s) { }

   @Override
   public void beforeTextChanged(CharSequence s, int start, int count,
      int after) { }
};
```

Constraining Input

- EditText from tip calculator
- input was numberDecimal
- Use InputFilter to constrain input
- Several built in filters such as AllCaps and LengthFilter



Custom Input Filter

- InputFilter has one method:

```
public CharSequence filter(
CharSequence source, int start,
int end, Spanned dest,
int dstart, int dend) {
```

- replace dstart to dend in dest with new text, start to end of source

```java
billEditText = (EditText) findViewById(R.id.billEditText);
billEditText.addTextChangedListener(billEditTextWatcher);
billEditText.setFilters(new InputFilter[]{new DecimalInputFilter(2)});
```

```java
@Override
public CharSequence filter(CharSequence source, int start, int end,
        Spanned dest, int dstart, int dend) {

    String destAsString = dest.toString();
    int dotPos = destAsString.indexOf('.');
    if(dotPos >= 0) {
        // has a decimal, so check number of digits after decimal
        String decimals = destAsString.substring(dotPos + 1);
        // if already max number of digits after decimal and input
        // is after decimal then don't allow
        if(decimals.length() >= NUM_DECIMALS && dstart > dotPos)
            return "";
    }

    // accept original replacement
    return null;
  }
```

DIALOGS

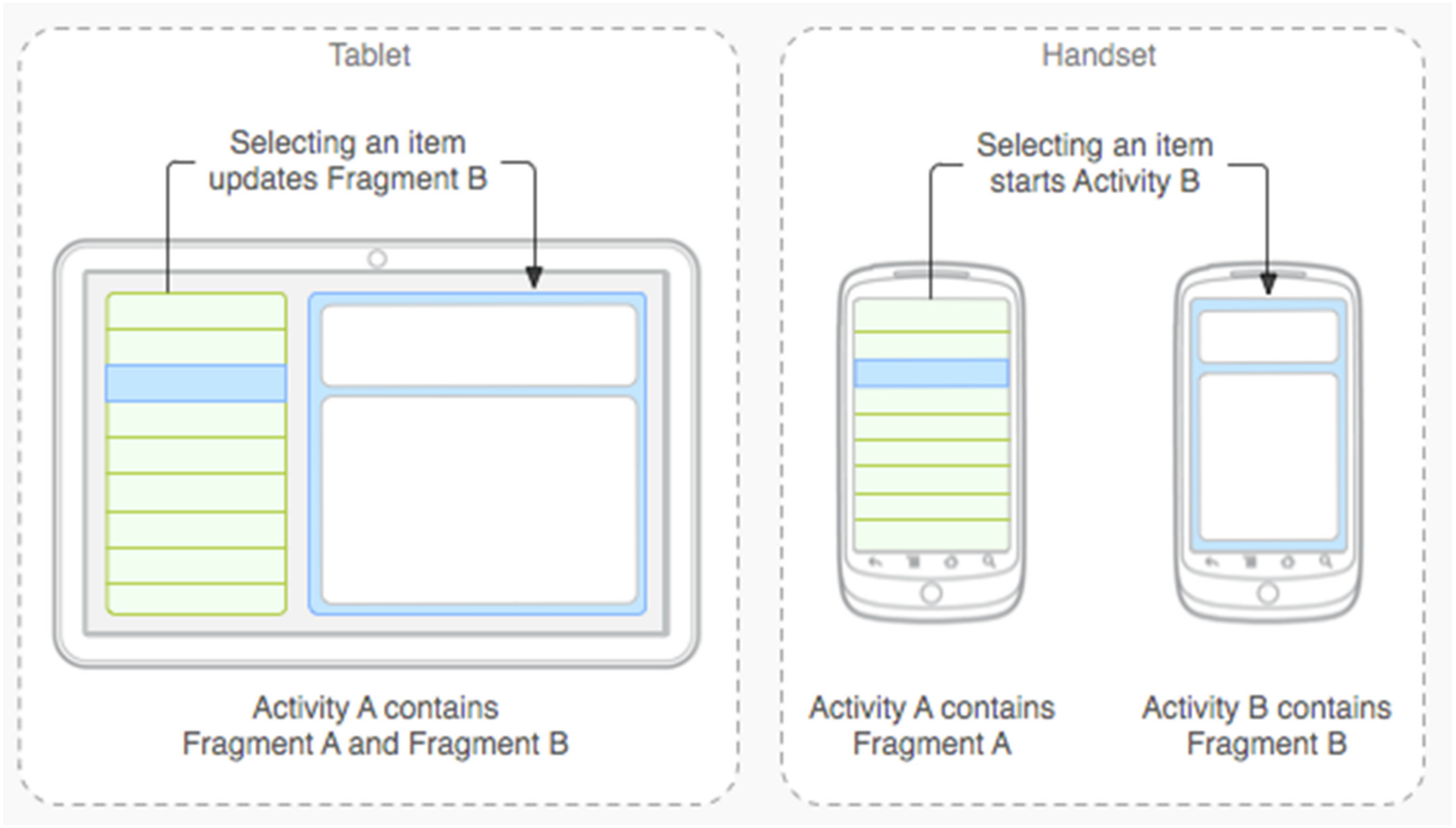
Dialogs - Old Way

- Dialogs from tutorials were cut and paste
- Implementing Dialogs demonstrates evolution of Android SDK
- legacy approach has Activity manage its own Dialogs
- created, initialized, updated, and destroyed using Activity class call back methods

Dialogs - New Way

- Android evolving from smartphone OS to smart device OS
- API level 11 (Android 3.0, the tablet release) introduced *Fragments*
- A fragment represents a behavior or a portion of a UI in an Activity
  - like a sub activity
- multiple fragments combined in multi-pane UI
- reuse fragments in multiple activities

Fragments


Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B

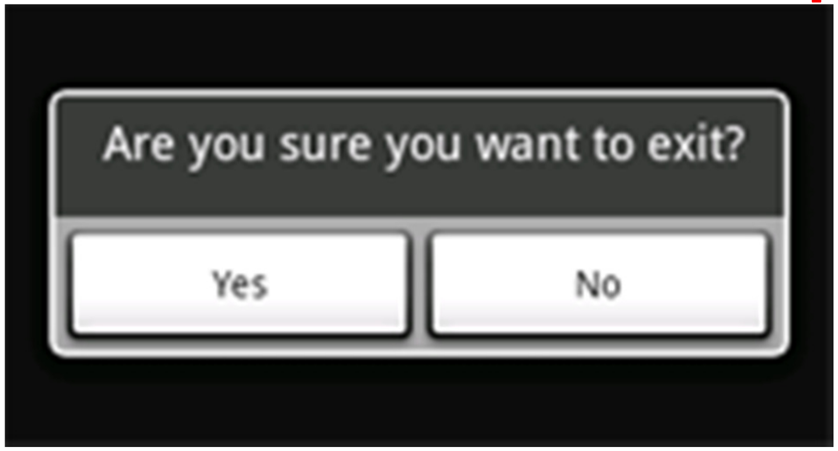
Dialogs as Fragments

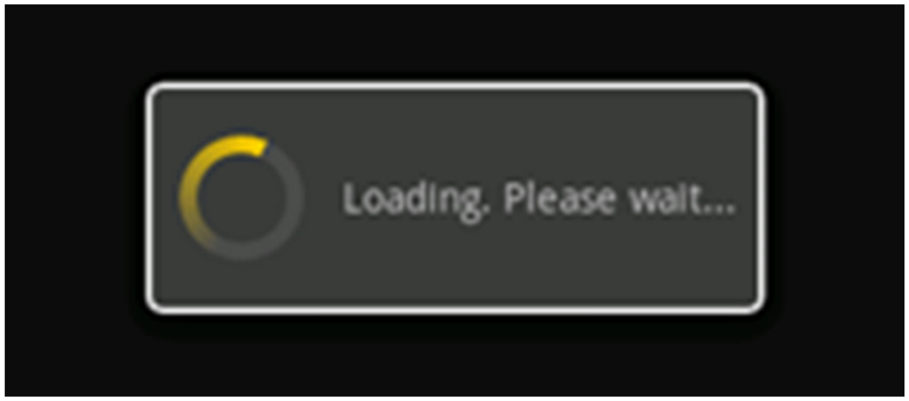
- Dialogs are special type of Fragment
- managed by the FragmentManager class
- still part of an activity, but lifecycle not managed by the Activity
  - life cycle issues of Dialogs as Fragments will be more difficult to deal with
  - must save state and restore instance

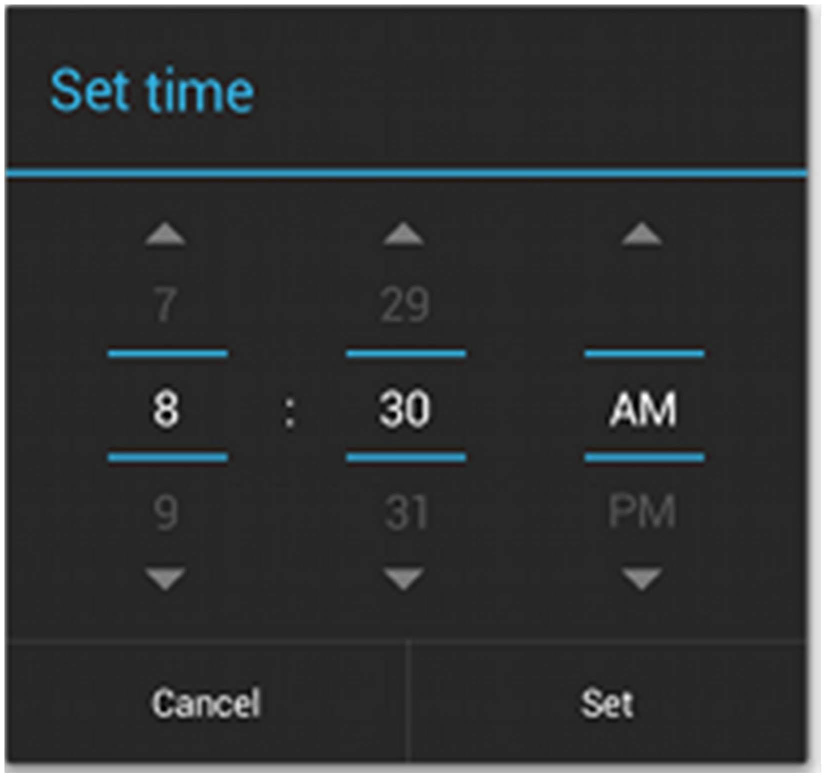
Types of Dialogs

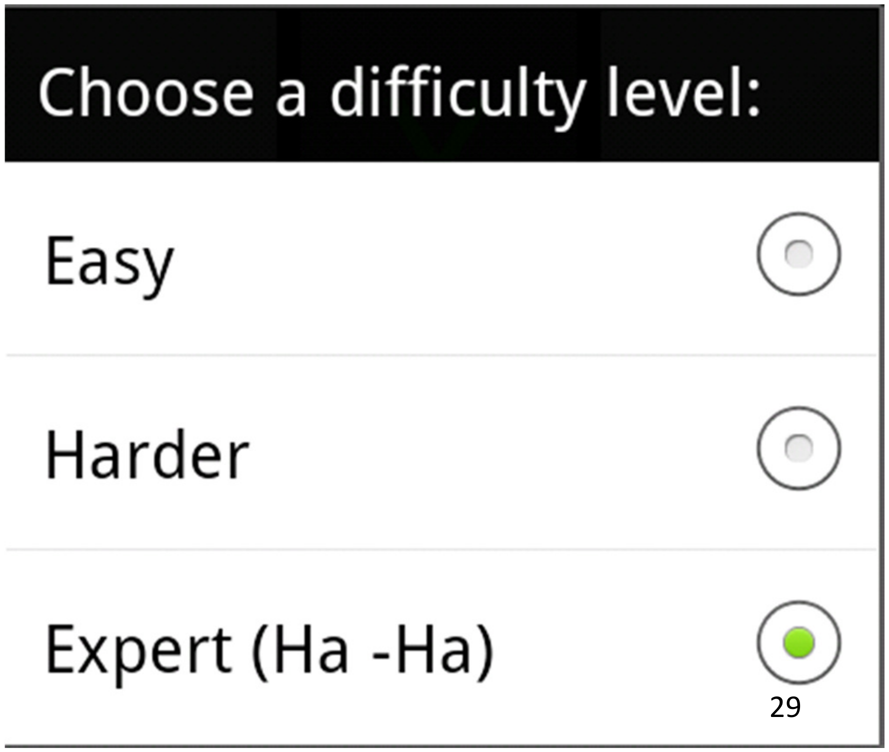
- Used to organize information and react to user events without creating a whole new activity
- Old Dialogs:
  - Dialog, AlertDialog, DatePickerDialog, TimePickerDialog, ProgressDialog
- New Dialogs:
  - DialogFragment

Sample Dialogs


Are you sure you want to exit?





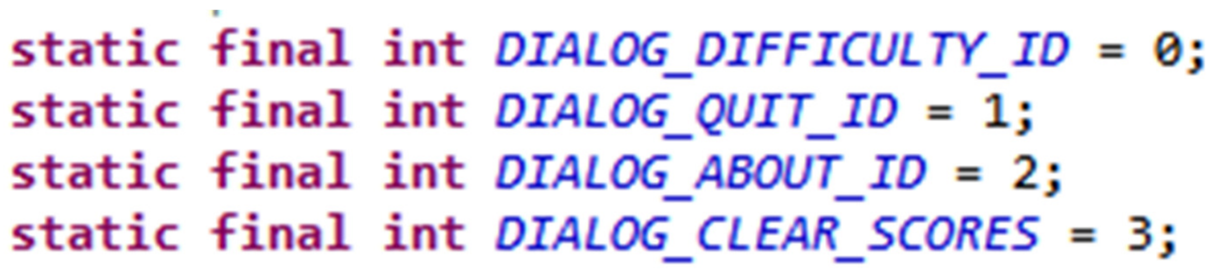

Choose a difficulty level:

Legacy Approach

- Dialog defined in Activity it is used
- Activity maintains a pool of Dialogs
- showDialog() method displays Dialog
- dismissDialog() method used to stop showing a Dialog
  - in tutorial, when we have difficulty
- removeDialog removes from pool

Legacy Approach - Steps

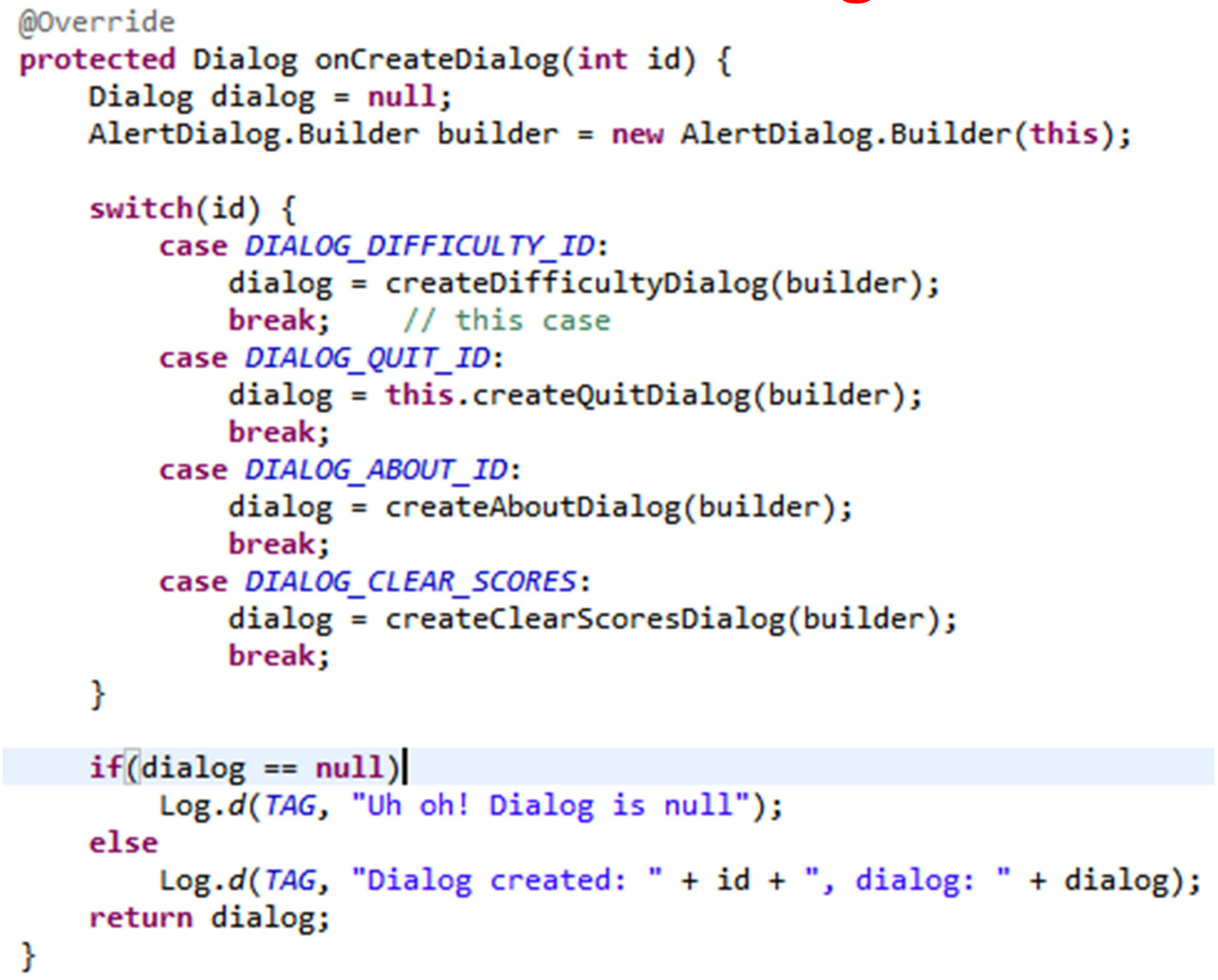
- Define unique indentifier for the Dialog in Activity (constants)



```
static final int DIALOG_DIFFICULTY_ID = 0;
static final int DIALOG_QUIT_ID = 1;
static final int DIALOG_ABOUT_ID = 2;
static final int DIALOG_CLEAR_SCORES = 3;
```

- implement onCreateDialog method, returns Dialog of appropriate type

onCreateDialog



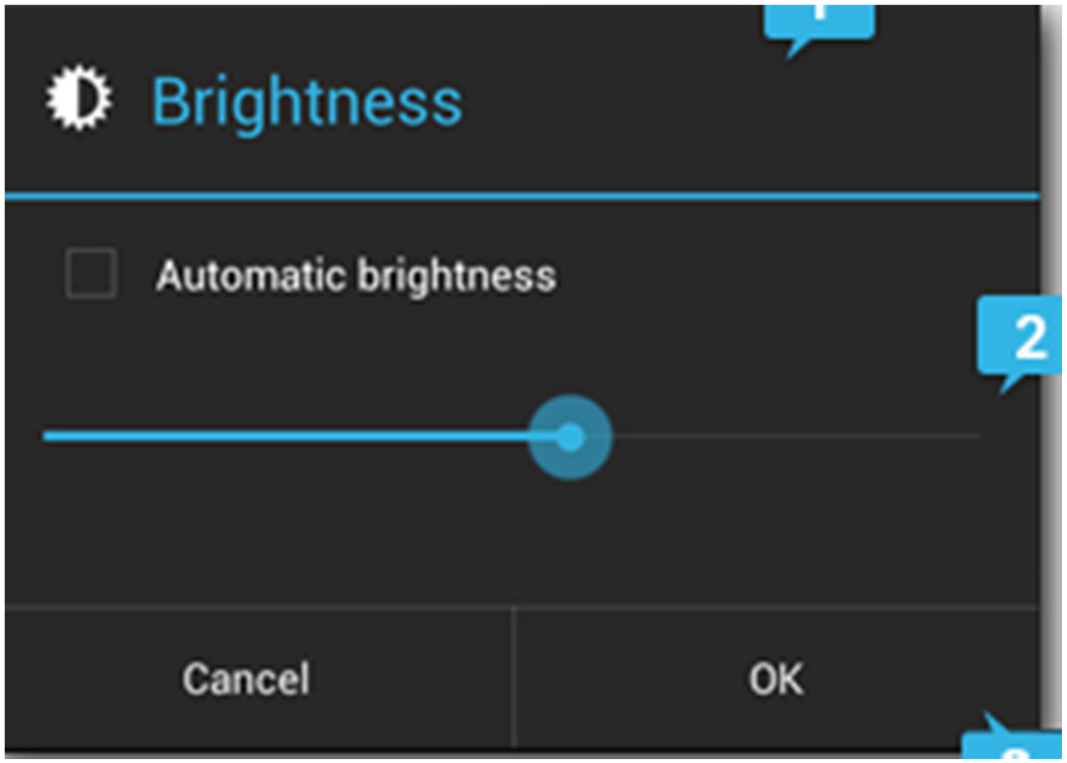
```java
@Override
protected Dialog onCreateDialog(int id) {
    Dialog dialog = null;
    AlertDialog.Builder builder = new AlertDialog.Builder(this);

    switch(id) {
        case DIALOG_DIFFICULTY_ID:
            dialog = createDifficultyDialog(builder);
            break;    // this case
        case DIALOG_QUIT_ID:
            dialog = this.createQuitDialog(builder);
            break;
        case DIALOG_ABOUT_ID:
            dialog = createAboutDialog(builder);
            break;
        case DIALOG_CLEAR_SCORES:
            dialog = createClearScoresDialog(builder);
            break;
    }

    if(dialog == null)
        Log.d(TAG, "Uh oh! Dialog is null");
    else
        Log.d(TAG, "Dialog created: " + id + ", dialog: " + dialog);
    return dialog;
}
```

Dialog Steps - Legacy Approach

- implement onPrepareDialog() if necessary
  - if necessary to update dialog each time it is displayed
  - for example, a time picker, update with the current time
- launch dialog with showDialog()
  - in tutorials done when a menu or action bar menu item selected
  - could launch Dialogs for other reasons

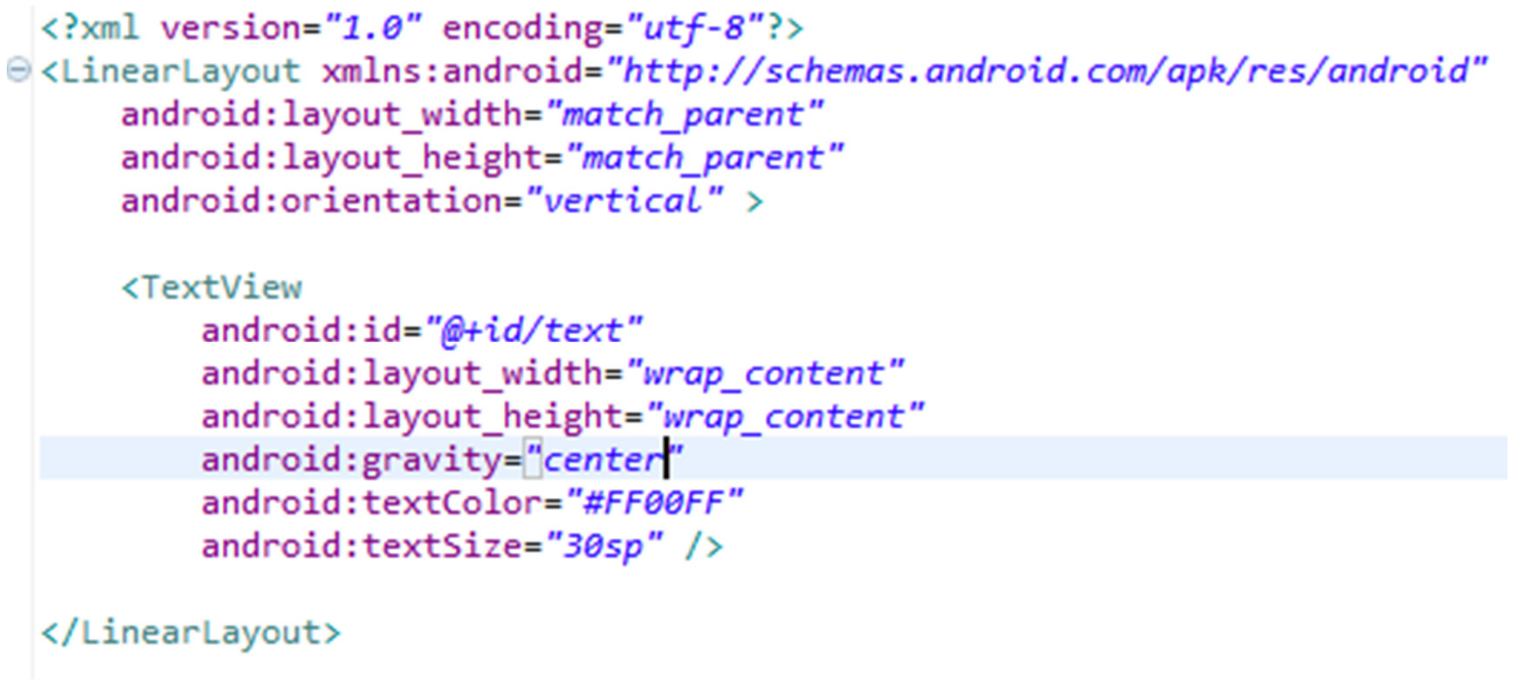
Alert Dialogs

- Most common type
- Title, Content Area, Action buttons (up to 3)
- Content area could be message, list, seekbar, etc.
- set positive, set negative, set neutral



Custom Dialogs

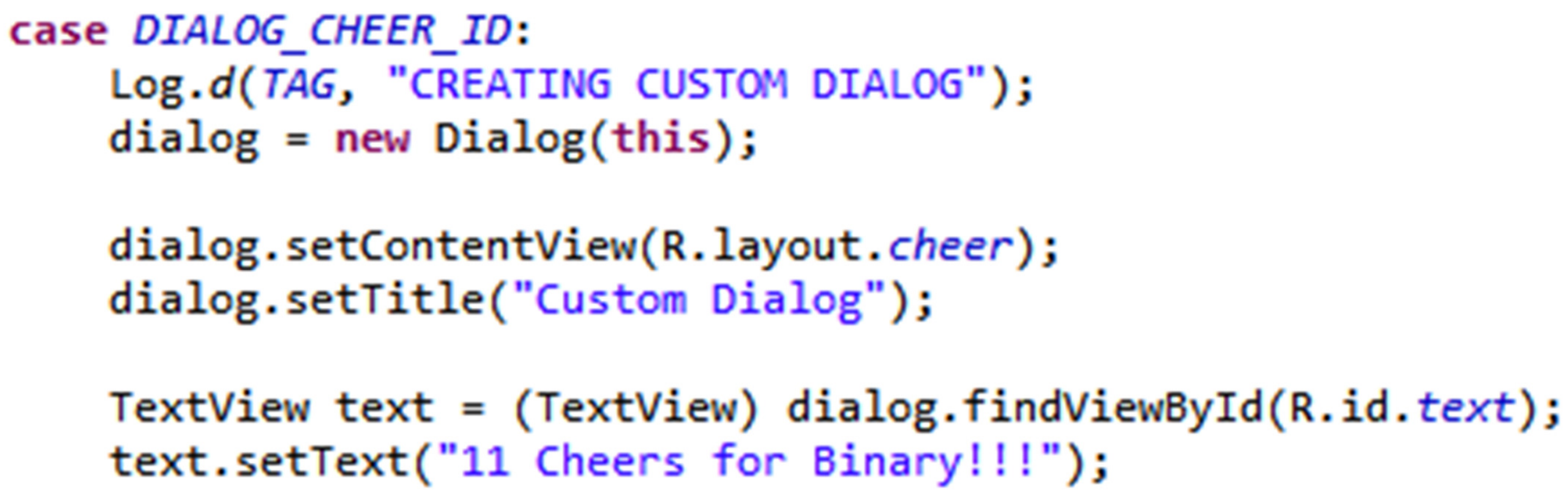
- AlertDialog very flexible, but you can create CustomDialogs
- Create a layout



```
<?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="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textColor="#FF00FF"
        android:textSize="30sp" />

</LinearLayout>
```

Custom Dialogs

- from onCreateDialog

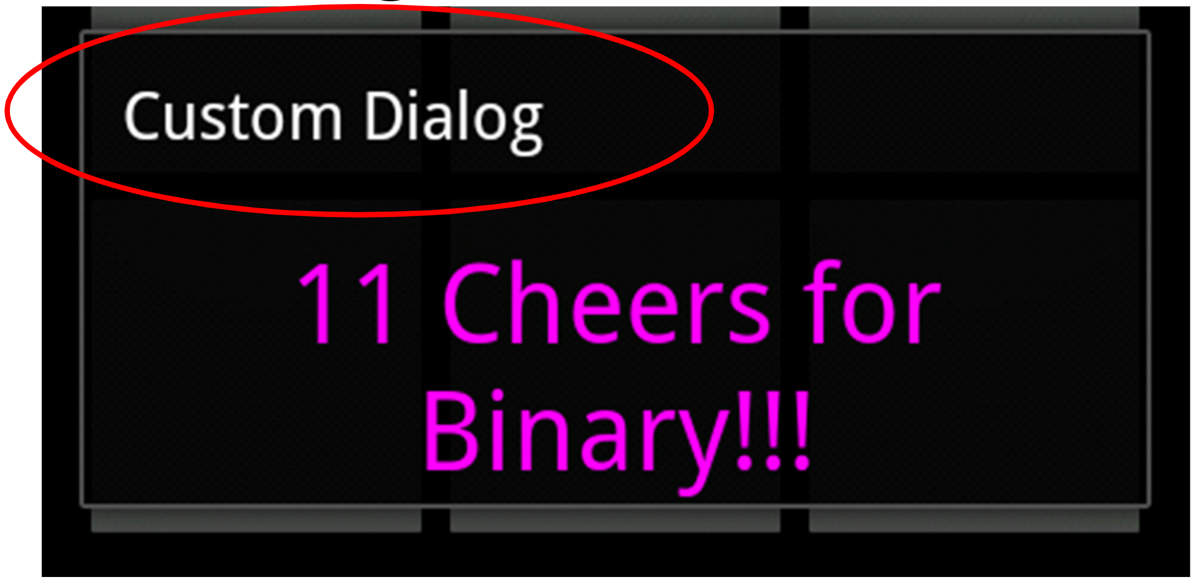


```
case DIALOG_CHEER_ID:
    Log.d(TAG, "CREATING CUSTOM DIALOG");
    dialog = new Dialog(this);

    dialog.setContentView(R.layout.cheer);
    dialog.setTitle("Custom Dialog");

    TextView text = (TextView) dialog.findViewById(R.id.text);
    text.setText("11 Cheers for Binary!!!");
```

Custom Dialog

- Simple dialogs are dismissed with the back button

dialog title

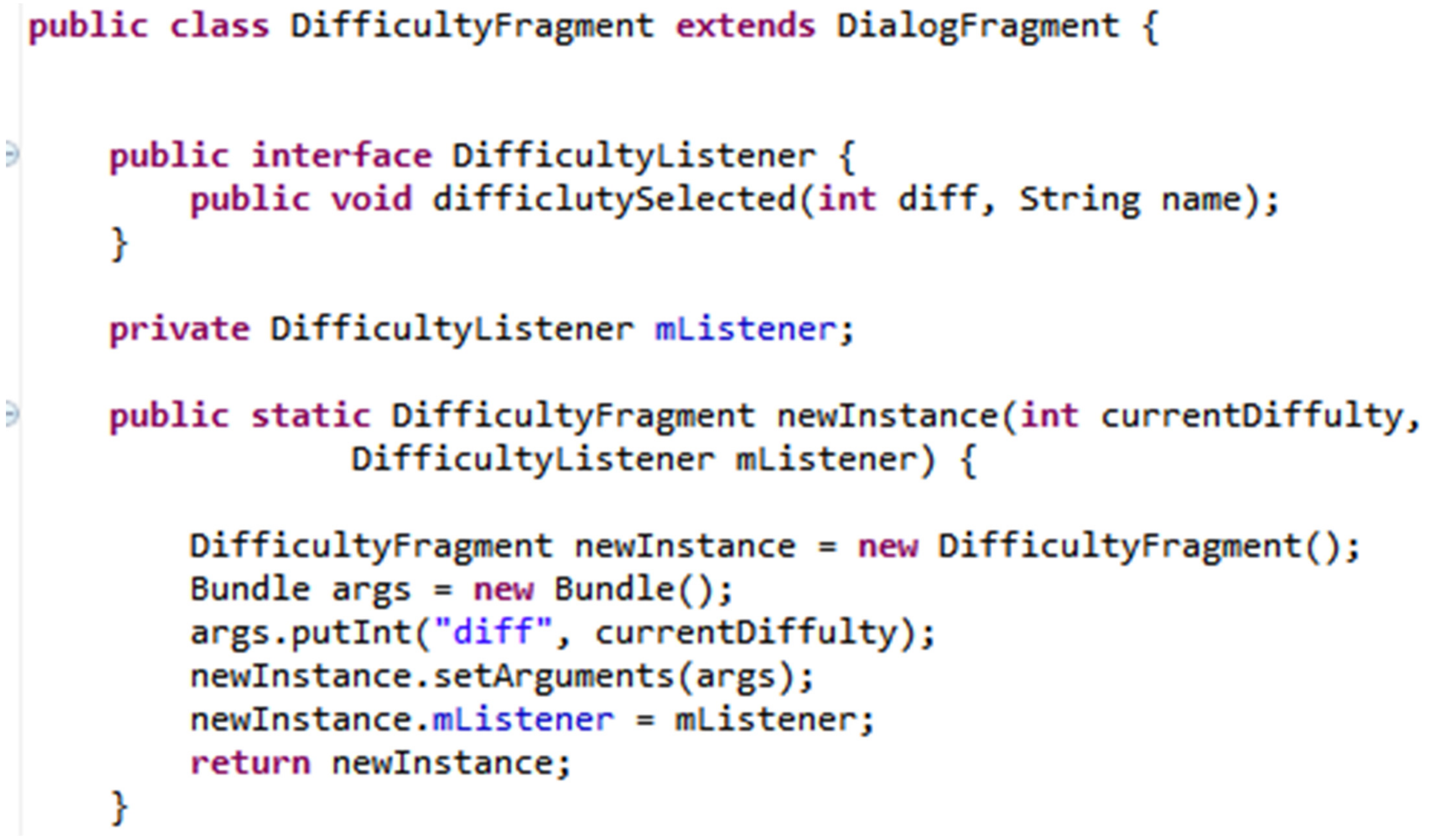

Custom Dialog

11 Cheers for Binary!!!

Dialogs - Fragment Method

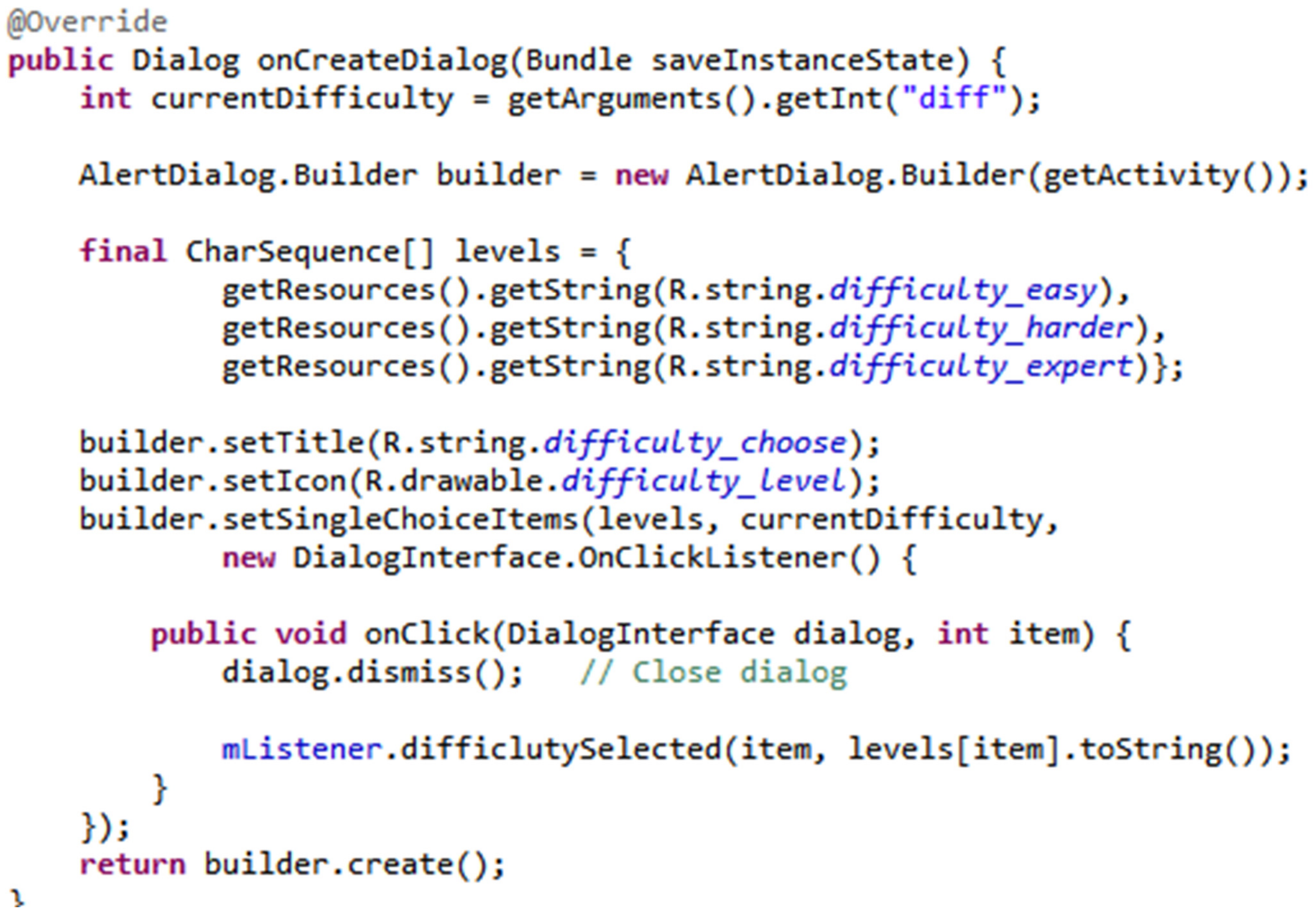
- Decouple Dialogs from the Activity
  - good SE approach?
  - TicTacToe UI is almost 500 lines long!
- Implement a class that is a subclass of DialogFragment
  - DifficultyFragment
  - Send info to newInstance method (current difficulty, listener for updates)
  - onCreateDialog now in DifficultyFragment

DifficultyFragment



```
public class DifficultyFragment extends DialogFragment {


    public interface DifficultyListener {
        public void difficlutySelected(int diff, String name);
    }

    private DifficultyListener mListener;

    public static DifficultyFragment newInstance(int currentDiffulty,
                DifficultyListener mListener) {

        DifficultyFragment newInstance = new DifficultyFragment();
        Bundle args = new Bundle();
        args.putInt("diff", currentDiffulty);
        newInstance.setArguments(args);
        newInstance.mListener = mListener;
        return newInstance;
    }
```

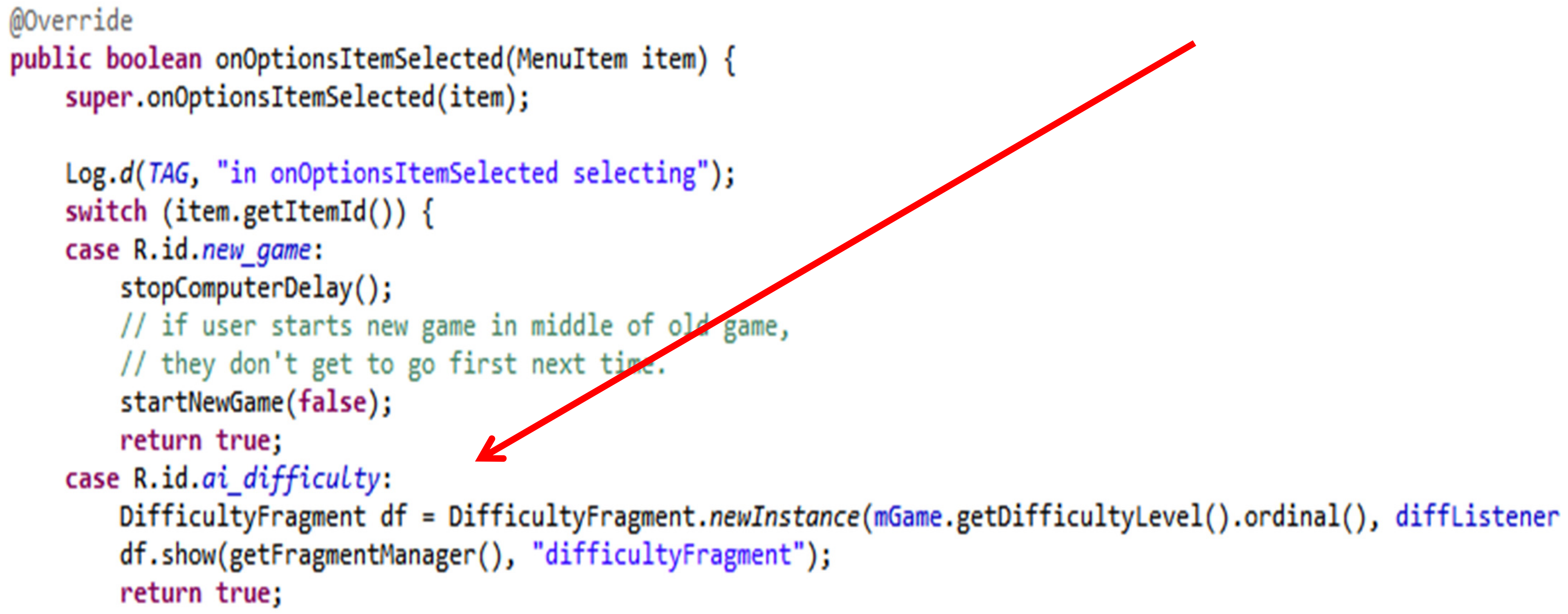
DifficultyFragment - onCreateDialog



```java
@Override
public Dialog onCreateDialog(Bundle saveInstanceState) {
    int currentDifficulty = getArguments().getInt("diff");

    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

    final CharSequence[] levels = {
            getResources().getString(R.string.difficulty_easy),
            getResources().getString(R.string.difficulty_harder),
            getResources().getString(R.string.difficulty_expert)};

    builder.setTitle(R.string.difficulty_choose);
    builder.setIcon(R.drawable.difficulty_level);
    builder.setSingleChoiceItems(levels, currentDifficulty,
            new DialogInterface.OnClickListener() {

        public void onClick(DialogInterface dialog, int item) {
            dialog.dismiss();   // Close dialog

            mListener.difficlutySelected(item, levels[item].toString());
        }
    });
    return builder.create();
}
```

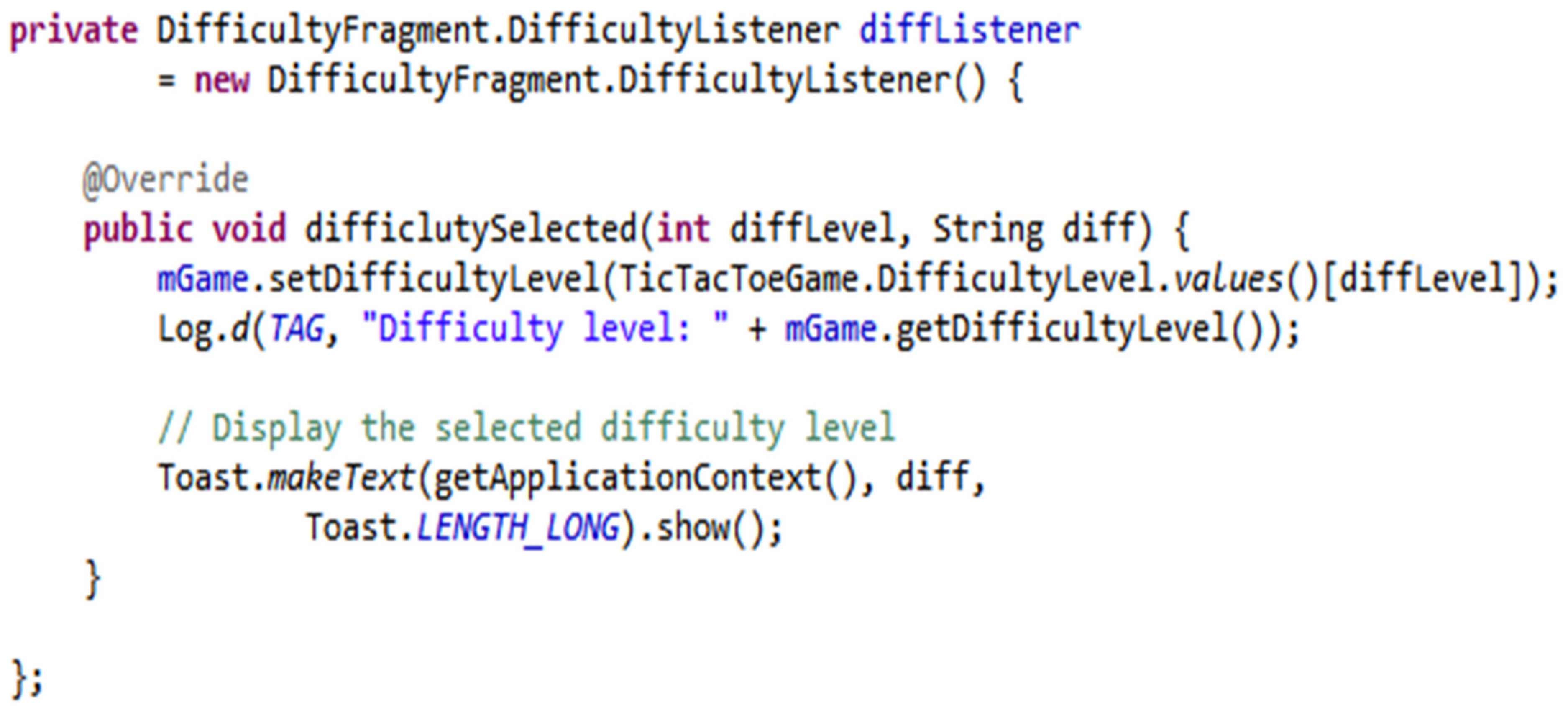
Using DifficultyFragment

- In AndroidTicTacToe create a listener to pass to the newInstance method
- create and show Dialog as part of onOptionsItemSelected()



```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    super.onOptionsItemSelected(item);

    Log.d(TAG, "in onOptionsItemSelected selecting");
    switch (item.getItemId()) {
    case R.id.new_game:
        stopComputerDelay();
        // if user starts new game in middle of old game,
        // they don't get to go first next time.
        startNewGame(false);
        return true;
    case R.id.ai_difficulty:
        DifficultyFragment df = DifficultyFragment.newInstance(mGame.getDifficultyLevel().ordinal(), diffListener
        df.show(getFragmentManager(), "difficultyFragment");
        return true;
```

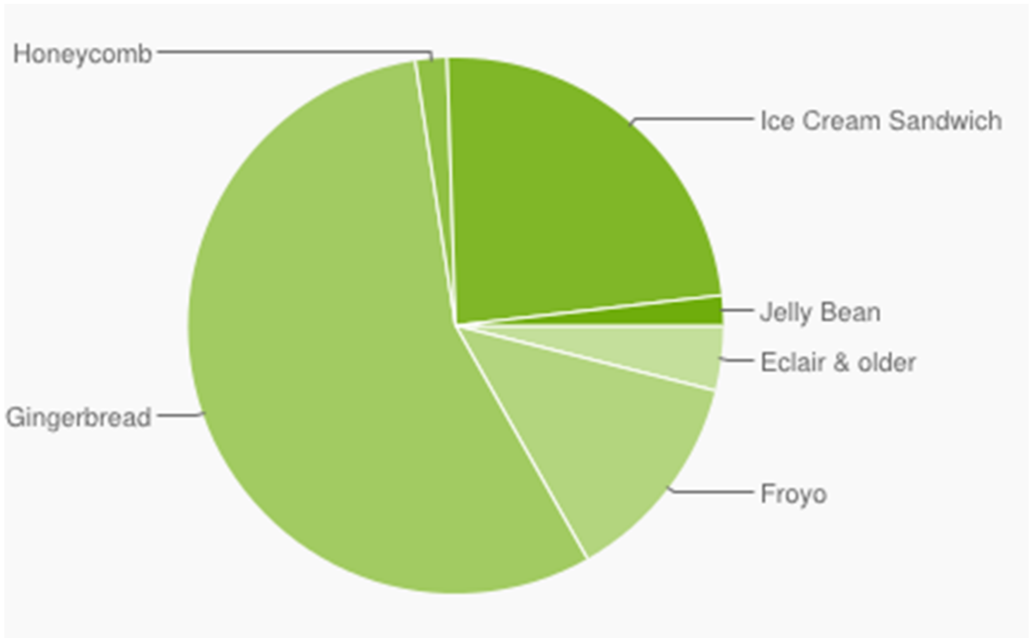
DifficultyListener



```
private DifficultyFragment.DifficultyListener diffListener
        = new DifficultyFragment.DifficultyListener() {

    @Override
    public void difficlutySelected(int diffLevel, String diff) {
        mGame.setDifficultyLevel(TicTacToeGame.DifficultyLevel.values()[diffLevel]);
        Log.d(TAG, "Difficulty level: " + mGame.getDifficultyLevel());

        // Display the selected difficulty level
        Toast.makeText(getApplicationContext(), diff,
                Toast.LENGTH_LONG).show();
    }

};
```

Using Fragments

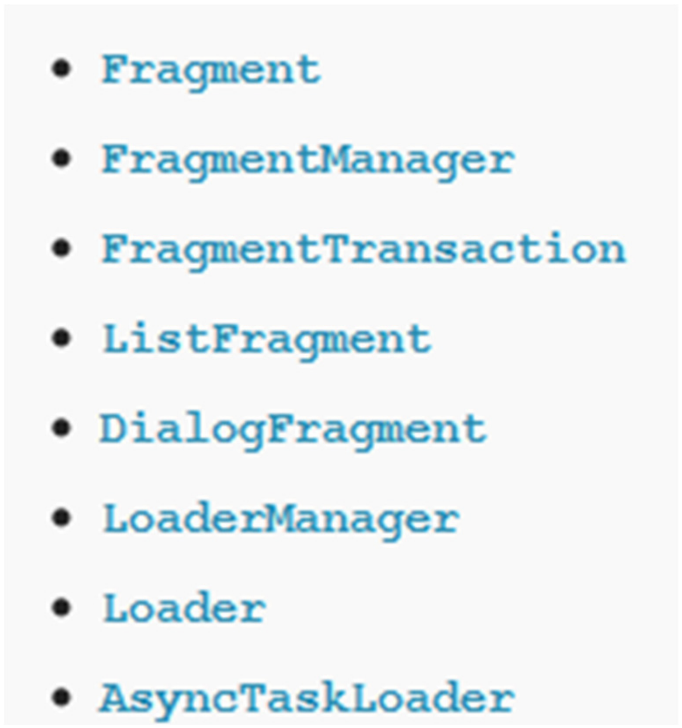
- Fragments added in API level 11, Android 3.0, the tablet release
- Developers behind Android think fragments are so important that can be used in pre API 11 builds using the *Android Support Library*



Froyo and Gingerbread pre API 11

Android Support Library (ASL)

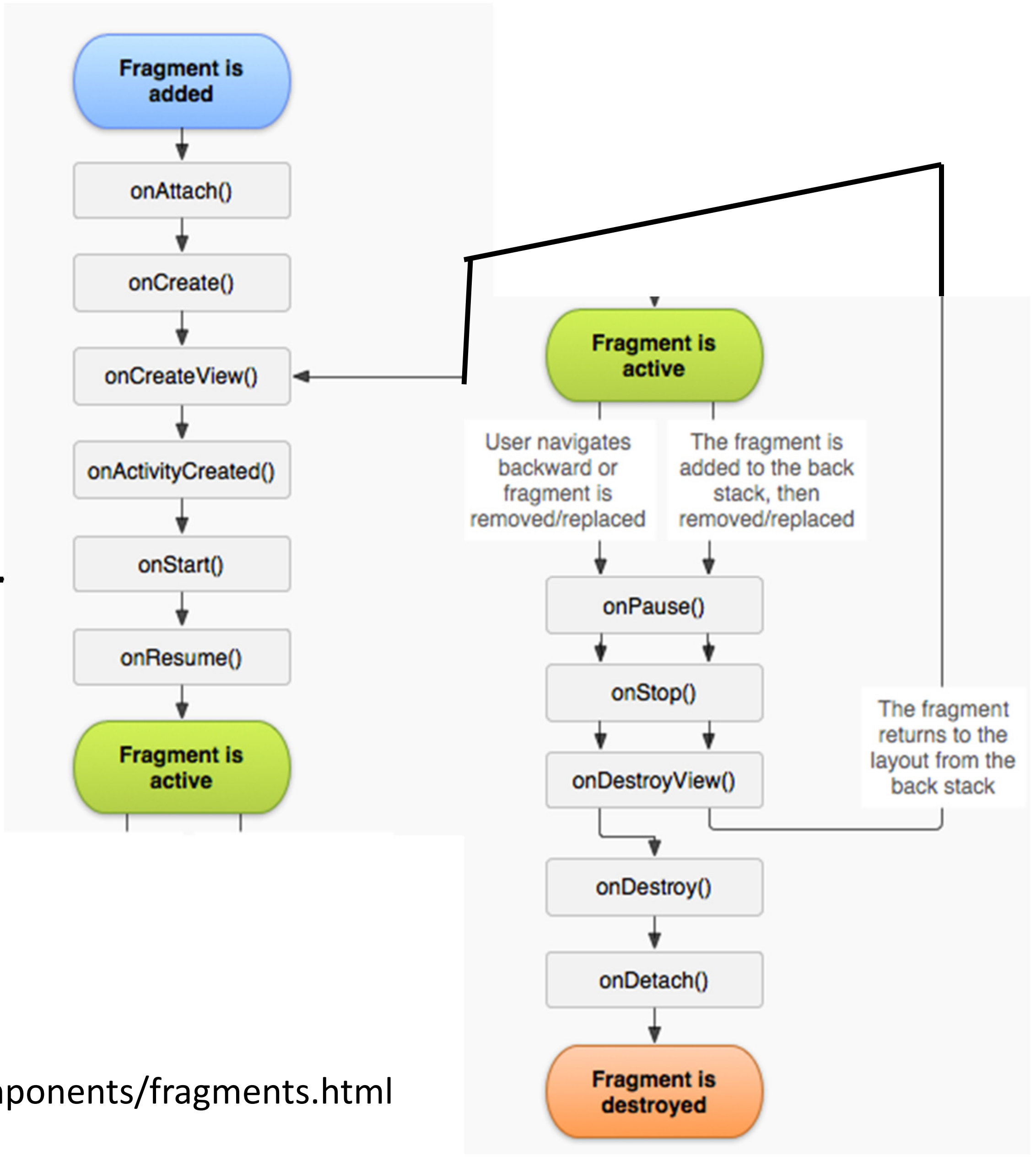
- add library to project and application
- android.support.v4.app.DialogFragment
  - for example
  - instead of android.app.DialogFragment
- ASL does not support action bar in earlier versions of API



- Fragment
- FragmentManager
- FragmentTransaction
- ListFragment
- DialogFragment
- LoaderManager
- Loader
- AsyncTaskLoader

Fragment Lifecycle

- Demo Tic Tac Toe with old style Dialog and Fragment Dialog
- Alter orientation - result?



http://developer.android.com/guide/components/fragments.html