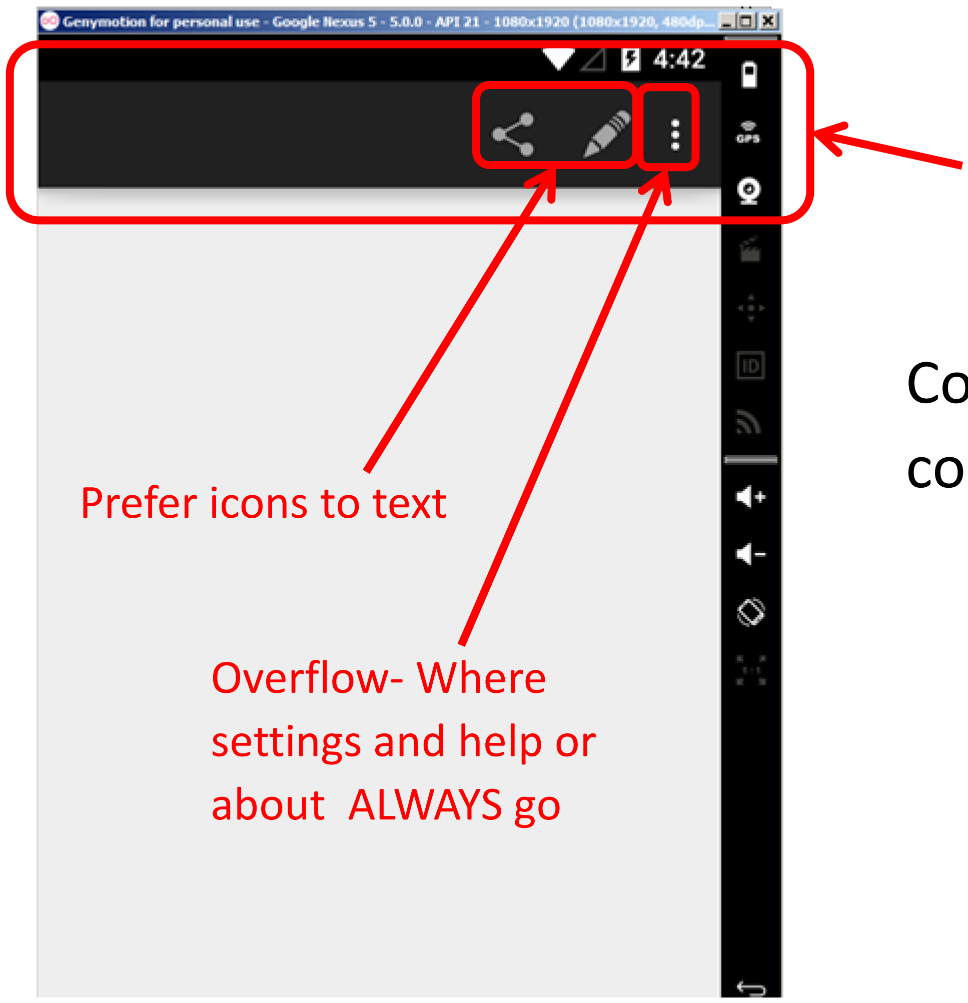
# CS 475/575

ToolBar (or App bar, or ActionBar)

Dialogs

Snackbar

# ToolBar


Genymotion for personal use - Google Nexus 5 - 5.0.0 - API 21 - 1080x1920 (1080x1920, 480dp...
4:42

Located here usually, but you can put it anywhere

Prefer icons to text

Overflow- Where settings and help or about ALWAYS go

Code that gives user ability to configure application options.

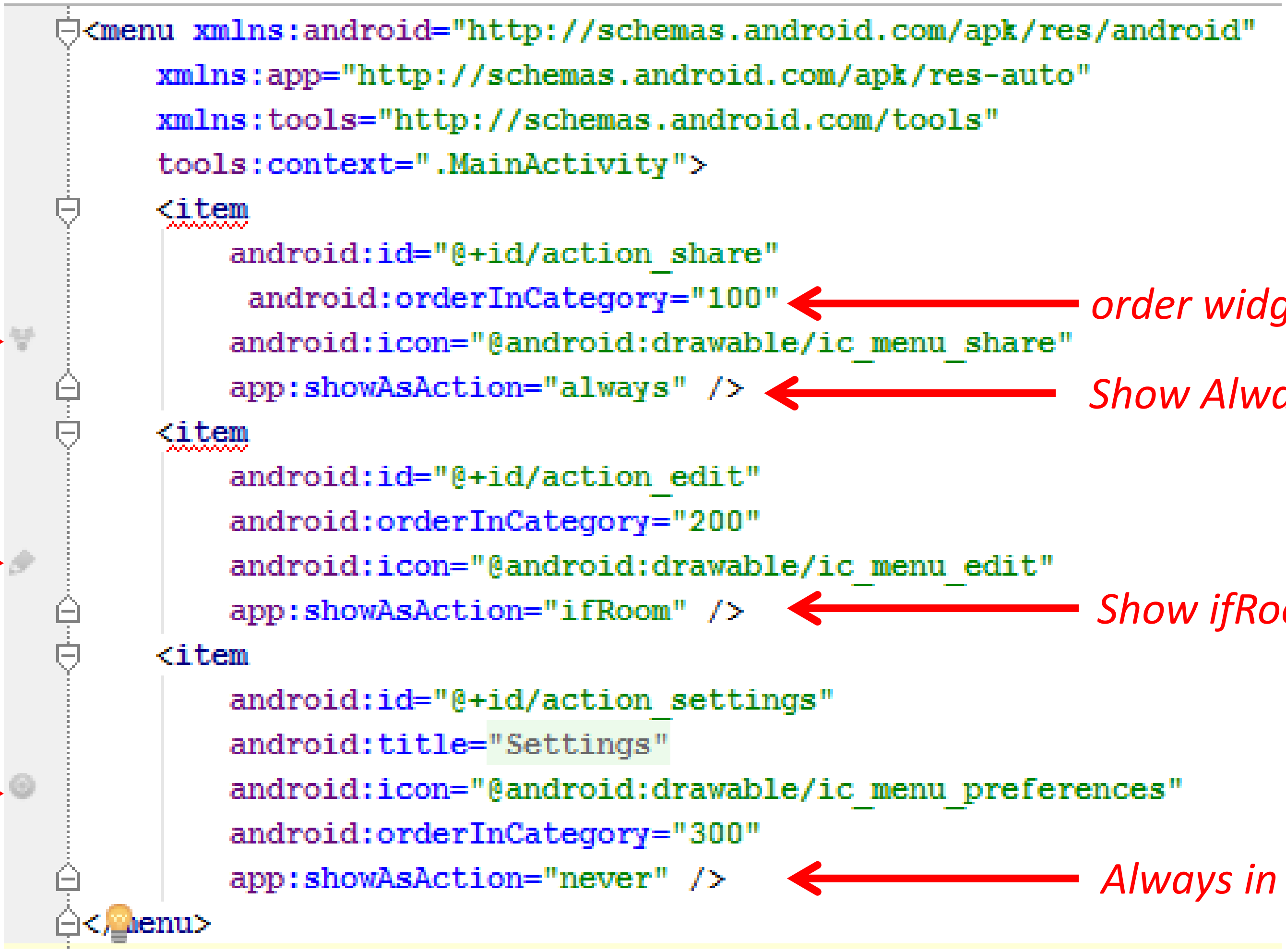
## ToolBar– What APIs?

ToolBar is available from API 7 (2.1, Eclair) on

Replaces ActionBar (changes to API caused ActionBar to have platform dependent behavior)

# Tool Bar – Step 1

## Add/Edit menu XML resource (in res\menu)



```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity">
    <item
        android:id="@+id/action_share"
         android:orderInCategory="100"
        android:icon="@android:drawable/ic_menu_share"
        app:showAsAction="always" />
    <item
        android:id="@+id/action_edit"
        android:orderInCategory="200"
        android:icon="@android:drawable/ic_menu_edit"
        app:showAsAction="ifRoom" />
    <item
        android:id="@+id/action_settings"
        android:title="Settings"
        android:icon="@android:drawable/ic_menu_preferences"
        android:orderInCategory="300"
        app:showAsAction="never" />
</menu>
```

*order widgets appear in tool bar*

*Show Always*

*Show ifRoom*

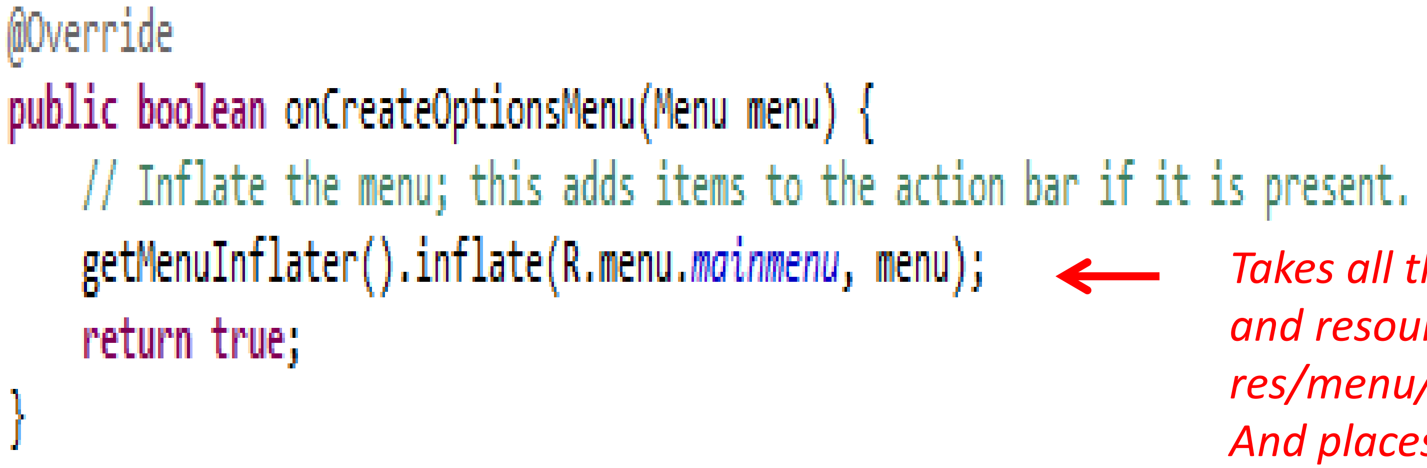
*Always in overflow*

## Overflow Menu

- <u>Always</u> have settings there
- Should also have help or about
  - Want visibility to be user controlled
  - Can use an activity
  - Can use a dialog (coming in a few minutes)

## Tool Bar – Step 2

- In main activity override onCreateOptionsMenu (done by AS if you start with Basic activity, but you can still use a blank one with a little more work)
- This is called once it creates your menu and adds it to toolbar unless you call invalidateOptionsMenu() to have it redone

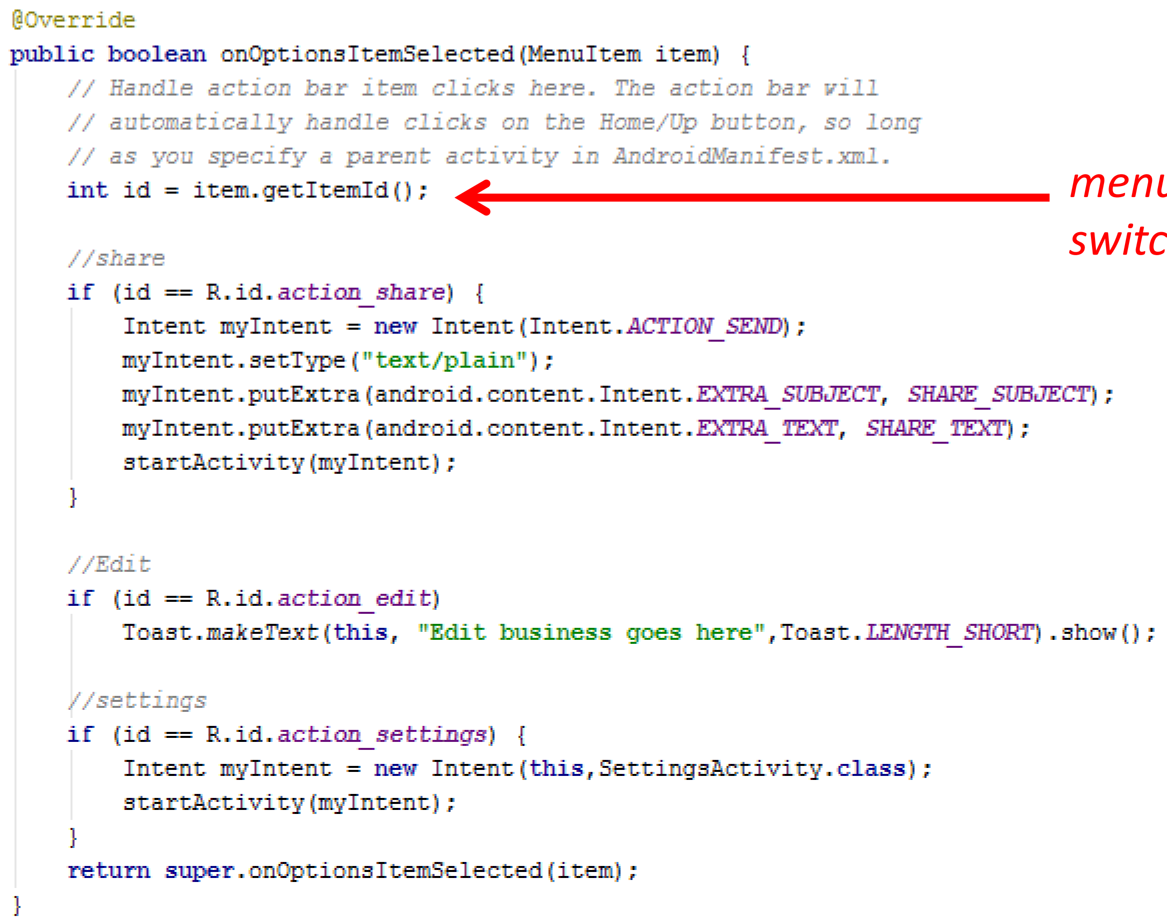


```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.mainmenu, menu);
    return true;
}
```

← *Takes all the XML items and resources in res/menu/mainmenu.xml And places them in menu*

# Tool Bar – Step 3

In main activity fill in onOptionsItemSelected to respond to menu or action items

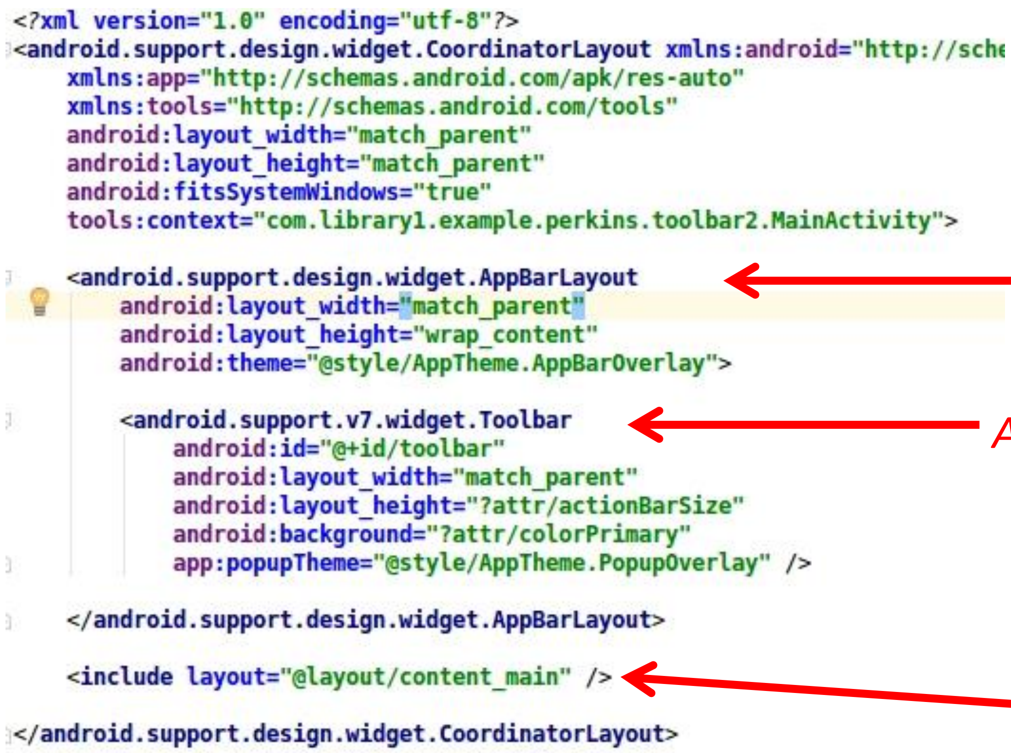


```java
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //share
    if (id == R.id.action_share) {
        Intent myIntent = new Intent(Intent.ACTION_SEND);
        myIntent.setType("text/plain");
        myIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, SHARE_SUBJECT);
        myIntent.putExtra(android.content.Intent.EXTRA_TEXT, SHARE_TEXT);
        startActivity(myIntent);
    }

    //Edit
    if (id == R.id.action_edit)
        Toast.makeText(this, "Edit business goes here",Toast.LENGTH_SHORT).show();

    //settings
    if (id == R.id.action_settings) {
        Intent myIntent = new Intent(this,SettingsActivity.class);
        startActivity(myIntent);
    }
    return super.onOptionsItemSelected(item);
}
```

*menu item selected (can do switch)*

# Tool Bar – Step 4

If not already done for you then in your Main activities layout



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://sche
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="com.library1.example.perkins.toolbar2.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

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

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main" />

</android.support.design.widget.CoordinatorLayout>
```

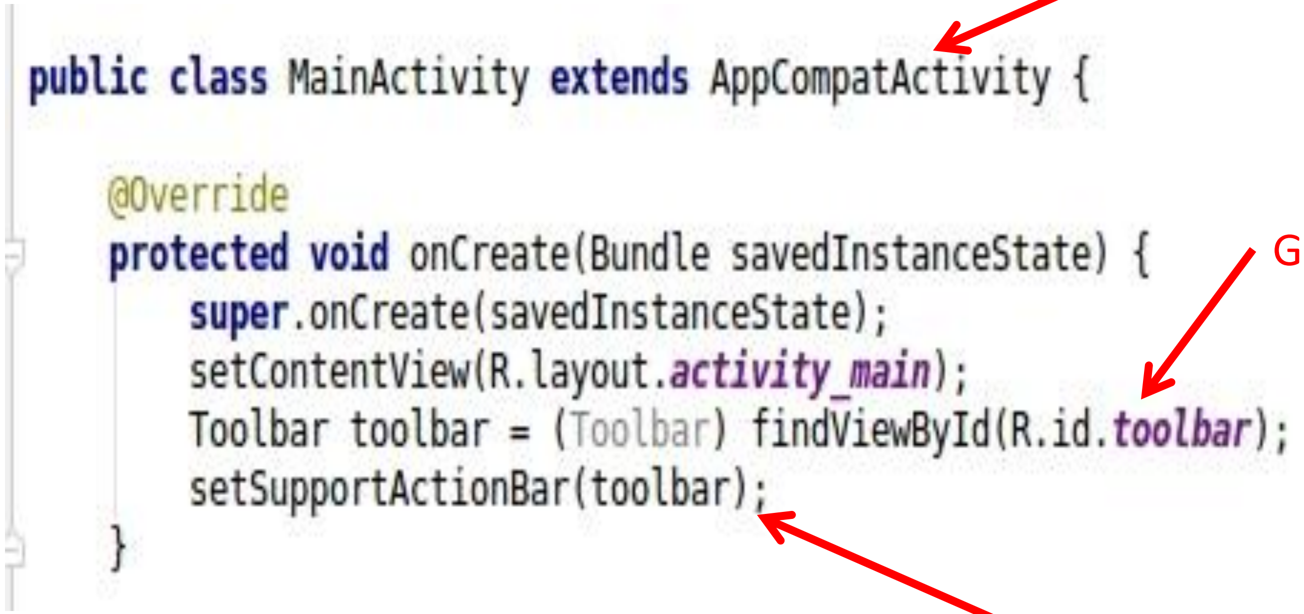
*Appbar layout*

*Appbar*

*Contains the widgets for this layout*

# Tool Bar – Step 5

In your MainActivity



```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
    }
```

derive from AppCompatActivity

Get ref to toolbar

Set it as your apps toobar

# Dialogs

- Get input from user or display data
- Has focus until the user closes it
- Dialog is base class
  - AlertDialog
  - ProgressDialog
  - DatePickerDialog
  - TimePickerDialog
- Can also subclass to make your own custom dialog
- Use Builder Pattern

## Builder Pattern

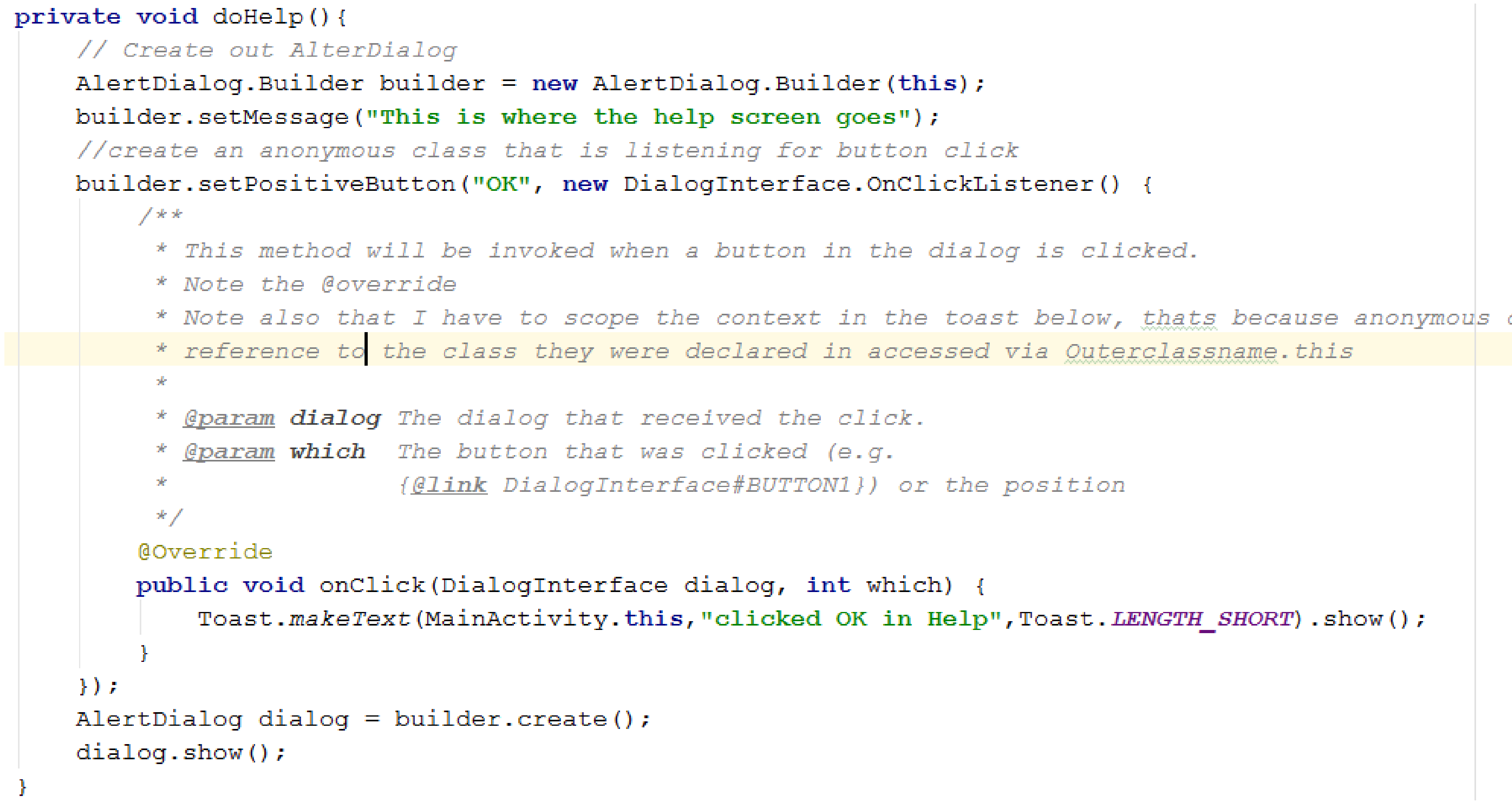
- Objects sometimes have many optional fields
- Multiple Constructors? Works but is hard to read, easy reverse params if types are the same leading to subtle bugs.  Scaling?  What if 20 params?
- One Constructor for required fields and then setters? What if a setter throws? Cannot enforce consistency.
- Best – Use a builder – Build an object with all required data, use to construct final object
- See 6_BuilderPatternDemo Project

## Dialogs (AlertDialog)

Lets create a dialog that responds to ‘about’

```
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    switch (id){
        case R.id.about:
            doHelp();
            return true;
    }

    //all else fails let super handle it
    return super.onOptionsItemSelected(item);
}
```

# Dialogs (AlertDialog) note the builder pattern



```
private void doHelp(){
    // Create out AlterDialog
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("This is where the help screen goes");
    //create an anonymous class that is listening for button click
    builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        /**
         * This method will be invoked when a button in the dialog is clicked.
         * Note the @override
         * Note also that I have to scope the context in the toast below, thats because anonymous
         * reference to the class they were declared in accessed via Outerclassname.this
         *
         * @param dialog The dialog that received the click.
         * @param which  The button that was clicked (e.g.
         *               {@link DialogInterface#BUTTON1}) or the position
         */
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(MainActivity.this,"clicked OK in Help",Toast.LENGTH_SHORT).show();
        }
    });
    AlertDialog dialog = builder.create();
    dialog.show();
}
```

# Snackbar

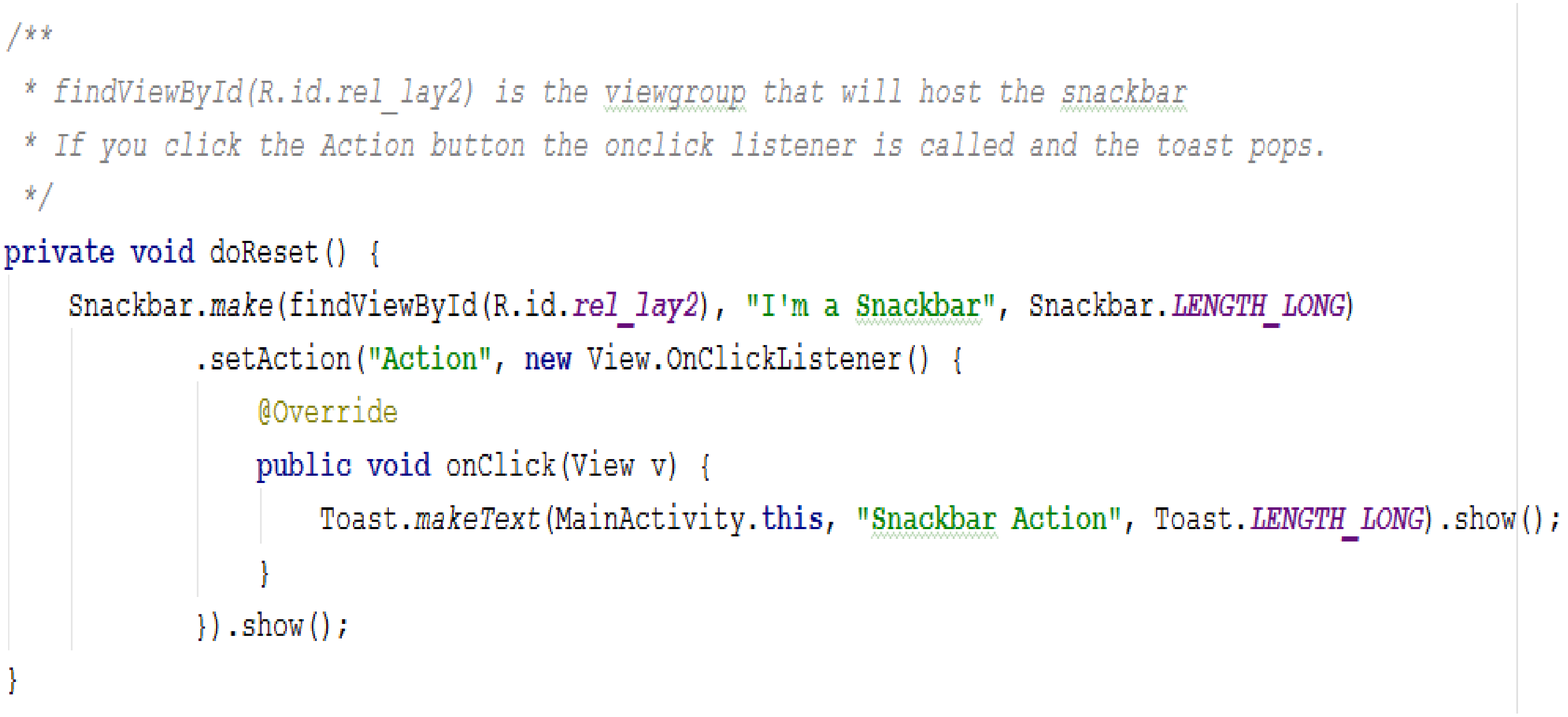
- Toast alternative
- Shown at the bottom of the screen
- Contain text with an optional single action.
- Automatically time out after the given time by animating off the screen.
- Can also swipe them away

## Snackbar

- Lets use a snackbar for reset

```
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    switch (id){

         case R.id.reset:
             doReset();
             return true;
     }

     //all else fails let super handle it
     return super.onOptionsItemSelected(item);
 }
```

# Snackbar



```java
/**
 * findViewById(R.id.rel_lay2) is the viewgroup that will host the snackbar
 * If you click the Action button the onclick listener is called and the toast pops.
 */
private void doReset() {
    Snackbar.make(findViewById(R.id.rel_lay2), "I'm a Snackbar", Snackbar.LENGTH_LONG)
            .setAction("Action", new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Toast.makeText(MainActivity.this, "Snackbar Action", Toast.LENGTH_LONG).show();
                }
            }).show();
}
```

## Summary

- Toolbar
- Dialogs  (Builder Pattern)
- Snackbar