Preferences

Marco Ronchetti
Università degli Studi di Trento

SharedPreferences

SharedPreferences allows to save and retrieve persistent key-value pairs of primitive data types. This data will persist across user sessions (even if your application is killed).

getSharedPreferences(String name, int mode)

A method of Contex

- Uses multiple preferences files identified by name, which you specify with the first parameter.

getPreferences()

A method of Activity

- Use this if you need only one preferences file for your Activity. This simply calls the underlying getSharedPreferences(String, int) method by passing in this activity's class name as the preferences name

SharedPreferences methods

boolean contains(String key)

Checks whether the preferences contains a preference.

T getT(String key, T defValue)

Value returned
If key does not exist

Retrieve a T value from the preferences where T={int, float, boolean, long, String, Set<String>}.

SharedPreferences.Editor edit()

All changes you make in an editor are batched, and not copied back to the original SharedPreferences until you call commit() or apply()

SharedPreferences.Editor methods

Void apply(), boolean commit()

Commit your preferences changes back

Editor putT(String key)

Stores a T value in the preferences where T={int, float, boolean, long, String, Set<String>}.

Editor remove(String key)

Mark in the editor that a preference value should be removed

Editor clear ()

Mark in the editor that all preference values should be removed

User Preferences

Shared preferences are not strictly for saving "user preferences," such as what ringtone a user has chosen.

For creating user preferences for your application, you should use PreferenceActivity, which provides an Activity framework for you to create user preferences, which will be automatically persisted (using shared preferences).

It is based on Fragments

Threads

Marco Ronchetti
Università degli Studi di Trento

Threads

When an application is launched, the system creates a thread of execution for the application, called "main" or "UI thread"

This thread dispatches events to the user interface widgets, and draws (uses the android.widget and android.view packages).

Unlike Java AWT/Swing, separate threads are NOT created automatically.

Methods that respond to system callbacks (such as onKeyDown() to report user actions or a lifecycle callback method) always run in the UI thread.

If everything is happening in the UI thread, performing long operations such as network access or database queries will block the whole UI. When the thread is blocked, no events can be dispatched, including drawing events. From the user's perspective, the application appears to hang.

If the UI thread is blocked for more than 5 sec the user is presented with the"ANR - application not responding" dialog.

the Andoid UI toolkit is not thread-safe !

Consequence:

you must not manipulate your UI from a worker thread—all manipulation to the user interface must be done within the UI thread.

You MUST respect these rules:

- Do not block the UI thread
- Do not access the Android UI toolkit from outside the UI thread

An example from android developers

```
public void onClick(View v) {
        Bitmap b = loadImageFromNetwork(
                        "http://example.com/image.png");
        myImageView.setImageBitmap(b);
}
```

WRONG!
Potentially
Slow
Operation!

```
public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            Bitmap b = loadImageFromNetwork(
                "http://example.com/image.png");
            myImageView.setImageBitmap(b);
        })
            .start();
}
```

WRONG!
A non UI thread
accesses the UI!

Still not the solution…

```
public void onClick(View v) {
  Bitmap b;
      new Thread(new Runnable() {
          public void run() {
              b = loadImageFromNetwork(
                  "http://example.com/image.png");
          })
      .start();
      myImageView.setImageBitmap(b);
}
```

WRONG!
This does not wait for the thread to finish!

The solution

public boolean post (Runnable action)

- Causes the Runnable to be sent to the UI thread and to be run therein. It is invoked on a View from outside of the UI thread.

public boolean postDelayed (Runnable action, long delayMillis)

```
public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            Bitmap b = loadImageFromNetwork(
                "http://example.com/image.png");
            myImageView.post(
                new Runnable() {
                    public void run() {
                        mImageView.setImageBitmap(bitmap);
                    }
                }
            })
            .start();
}
```

OK! This code will be run in the UI thread

Java reminder: varargs

void f(String pattern, Object... arguments);

The three periods after the final parameter's type indicate that the final argument may be passed

- as an array *or*
- as a sequence of arguments.

Varargs can be used *only* in the final argument position.

```
Object a, b, c, d[10];
…
f("hello",d);
f("hello",a,b,c);
```

Varargs example

```
public class Test {
    public static void main(String args[]){ new Test(); }

    Test(){
        String k[]={"uno","due","tre"};
        f("hello",k);
        f("hello",“alpha“,“beta“);
        // f("hello“,“alpha“,“beta“,k); THIS DOES NOT WORK!
    }

    void f(String s, String... d){
        System.out.println(d.length);
        for (String k:d) {
            System.out.println(k);
        }
    }

}
```

AsyncTask<Params,Progress,Result>

Creates a new asynchronous task. The constructor must be invoked on the UI thread.

AsyncTask must be subclassed, and instantiated in the UI thread.

Methods to be overridden:

method	where	when
void onPreExecute()	UI Thread	before
Result doInBackground(Params...)	Separate new thread	during
void onProgressUpdate(Progress...)	UI Thread	
void onPostExecute(Result)	UI Thread	after

The more elegant solution

```
public void onClick(View v) {
  new DownloadImageTask().execute("http://example.com/image.png");
}
```

```
private class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
    protected Bitmap doInBackground(String... urls) {
        return loadImageFromNetwork(urls[0]);
    }
    protected void onPostExecute(Bitmap result) {
        mImageView.setImageBitmap(result);
    }
}
```

Using Progress

```
package it.unitn.science.latemar;
import ...
```

```
public class AsyncDemoActivity extends ListActivity {
 private static final String[] item{"uno","due","tre","quattro",
          "cinque","sei", "sette","otto","nove",
          "dieci","undici","dodici",};

@Override
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  ListView listView = getListView();

  setListAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1,
                new ArrayList<String>()));

  new AddStringTask().execute();
 }
```

Adapted from the source code of
http://commonsware.com/Android/

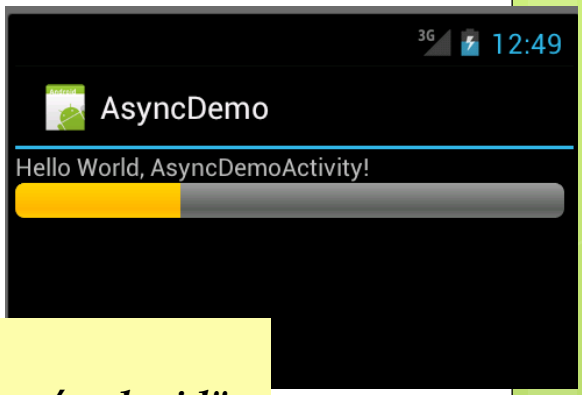
Using Progress

This is an inner class!

```
class AddStringTask extends AsyncTask<Void, String, Void> {
  @Override
  protected Void doInBackground(Void... unused) {
    for (String item : items) {
      publishProgress(item);
      SystemClock.sleep(1000);
    }
    return(null);
  }
  @SuppressWarnings("unchecked")
  @Override
  protected void onProgressUpdate(String... item) {
    ((ArrayAdapter<String>)getListAdapter()).add(item[0]);
  }
```

```
@Override
  protected void onPostExecute(Void unused) {
    Toast
      .makeText(AsyncDemoActivity.this,
                "Done!", Toast.LENGTH_SHORT)
      .show();
  }
}
```

Using the ProgressBar




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/hello" />
   <ProgressBar
      android:id="@+id/pb1"
      android:max="10"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      style="@android:style/Widget.ProgressBar.Horizontal"
      android:layout_marginRight="5dp" />
</LinearLayout>
```

```
public class AsyncDemoActivity2
              extends Activity {
  ProgressBar pb;
  @Override
  public void onCreate(Bundle state) {
    super.onCreate(state);
    setContentView(R.layout.main);
    pb=(ProgressBar) findViewById(R.id.pb1);
    new AddStringTask().execute();
  }
```

Using the ProgressBar

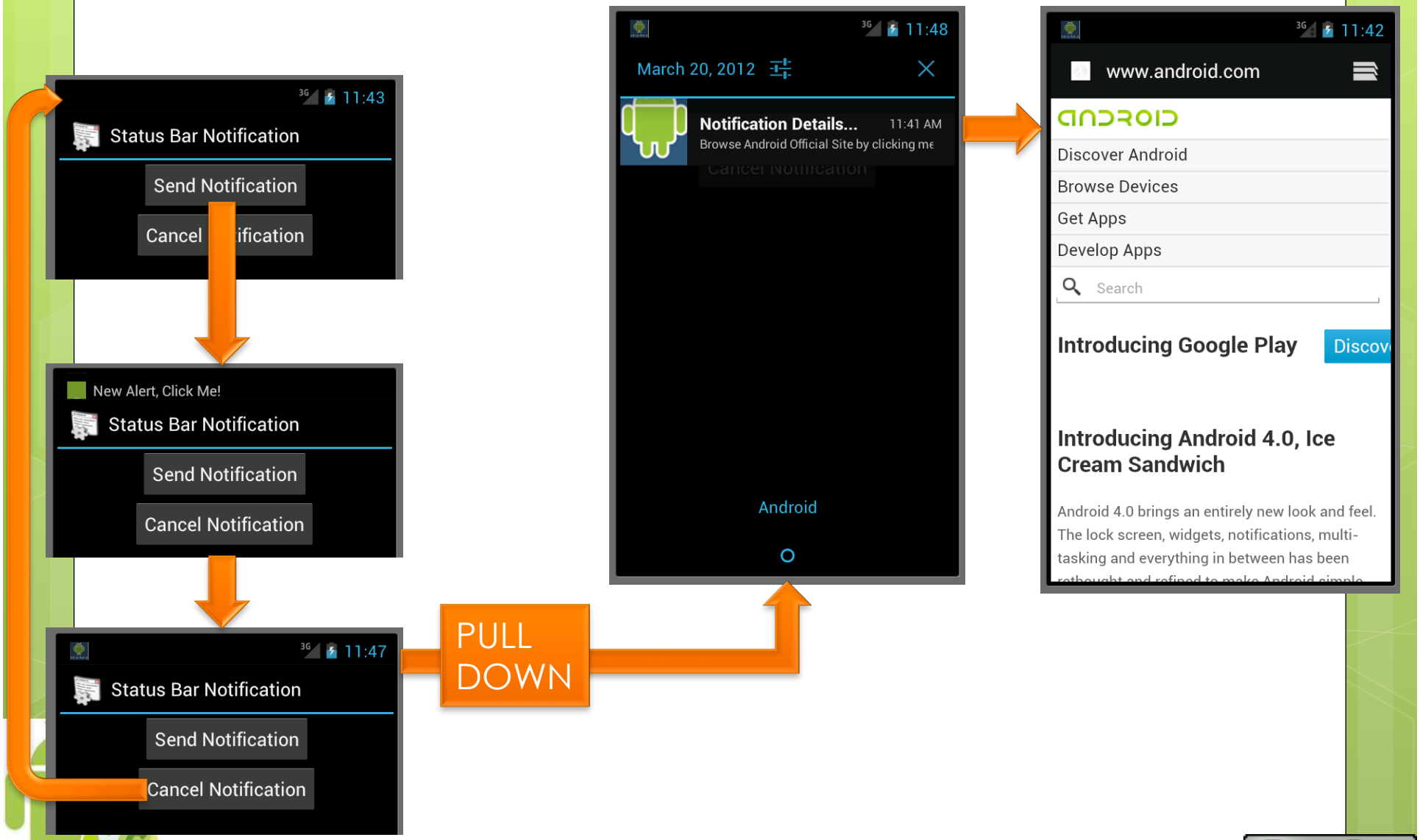
```
class AddStringTask extends AsyncTask<Void, Integer, Void> {
@Override
protected void doInBackground(Void... unused) {
  int item=0;
  while (item<10 ){
    publishProgress(++item);
    SystemClock.sleep(1000);
  }
}
@Override
protected void onProgressUpdate(Integer... item) {
  pb.setProgress(item[0]);
}
}
```

20

Notification

Marco Ronchetti
Università degli Studi di Trento

Notification Bar



SimpleNotification

```
public class SimpleNotification extends Activity {
   private NotificationManager nm;
   private int SIMPLE_NOTIFICATION_ID;
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      nm = (NotificationManager)getSystemService(NOTIFICATION_SERVICE);
      final Notification notifyDetails = new Notification(
            R.drawable.android,"New Alert, Click Me!",
            System.currentTimeMillis());
      Button cancel = (Button)findViewById(R.id.cancelButton);
      cancel.setOnClickListener(new OnClickListener() {
         public void onClick(View v) {
            nm.cancel(SIMPLE_NOTIFICATION_ID);
         }});}
```

Adapted from http://saigeethamn.blogspot.it

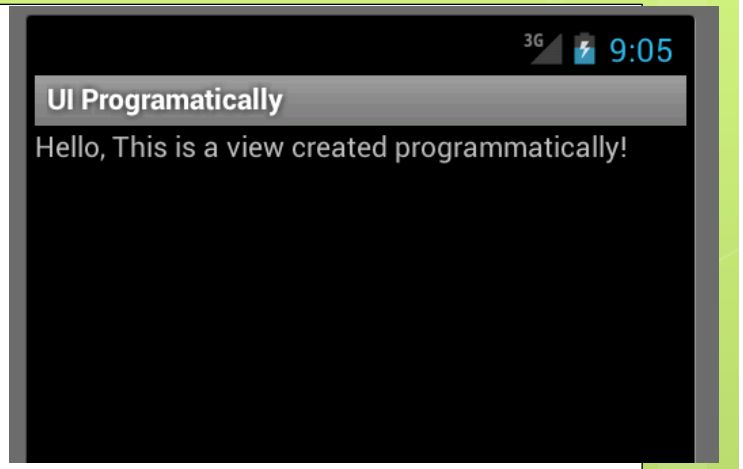
SimpleNotification – part 2

```
Button start = (Button)findViewById(R.id.notifyButton);
start.setOnClickListener(new OnClickListener() {
   public void onClick(View v) {
      Context context = getApplicationContext();
      CharSequence contentTitle = "Notification Details...";
      CharSequence contentText = "Browse Android Site by clicking me";
      Intent notifyIntent = new Intent
         (android.content.Intent.ACTION_VIEW,
         Uri.parse("http://www.android.com"));
      PendingIntent intent =
      PendingIntent.getActivity(SimpleNotification.this, 0, notifyIntent,
           android.content.Intent.FLAG_ACTIVITY_NEW_TASK);
      notifyDetails.setLatestEventInfo(context, contentTitle,
          contentText, intent);
      nm.notify(SIMPLE_NOTIFICATION_ID, notifyDetails);
      }
});
}}
```

Basic UI elements: Defining Activity UI in the code

Marco Ronchetti
Università degli Studi di Trento

UI Programmatically




```
public class UIThroughCode extends Activity {
  LinearLayout lLayout;
  TextView tView;
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      lLayout = new LinearLayout(this);
      lLayout.setOrientation(LinearLayout.VERTICAL);
      lLayout.setLayoutParams(new LayoutParams(LayoutParams.MATCH_PARENT,
                              LayoutParams.MATCH_PARENT));
      tView = new TextView(this);
      tView.setText("Hello, This is a view created programmatically! ")");
      tView.setLayoutParams(new LayoutParams(LayoutParams.MATCH_PARENT,
                            LayoutParams.WRAP_CONTENT));
      lLayout.addView(tView);
      setContentView(lLayout);
   }
}
```

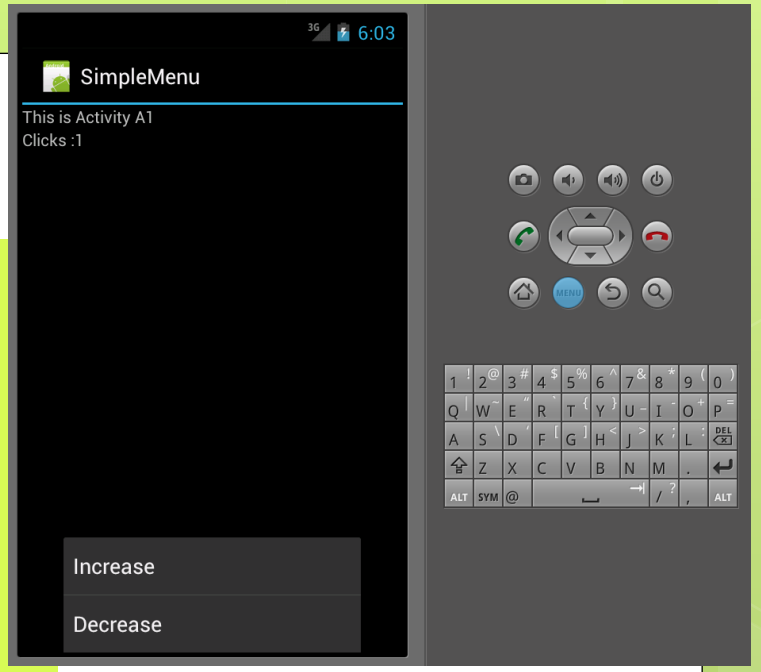
From http://saigeethamn.blogspot.it

Basic UI elements: Menus, a deeper insight

Marco Ronchetti
Università degli Studi di Trento

OptionMenu

```
public class A1 extends Activity {
            ...
   public boolean onCreateOptionsMenu(Menu menu){
       super.onCreateOptionsMenu(menu);
       int base=Menu.FIRST;
       MenuItem item1=menu.add(base,1,1,"Increase");
       MenuItem item2=menu.add(base,2,2,"Decrease");
       return true;
   }
   public boolean onOptionsItemSelected(MenuItem item) {
            ...
    }
```



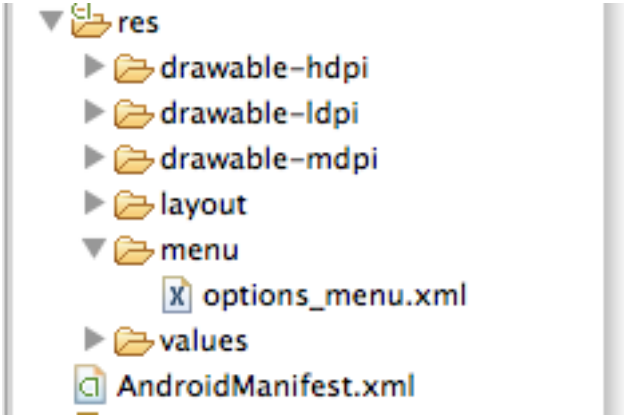
Menu is created
In code

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

Menu is created
From XML

option_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
        <item android:id="@+id/increase"
                android:title="@string/increase" />
        <item android:id="@+id/decrease"
            android:title="@string/decrease" />
</menu>
```



Dynamically changing menu

Menu onPrepareOptionsMenu() (Activity class).

you get the Menu object as it currently exists, and you can modify it (add, remove, or disable items).

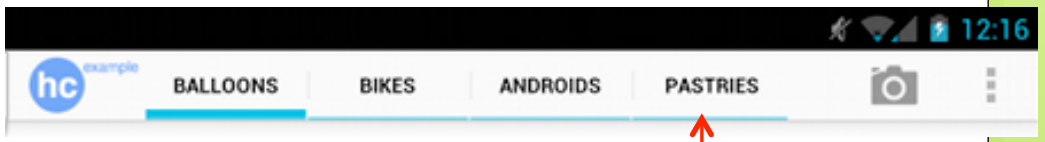
Android < 3.0:

- the system calls onPrepareOptionsMenu() each time the user opens the options menu.

Android >= 3.0

- When you want to perform a menu update, you must call invalidateOptionsMenu() to request that the system calls onPrepareOptionsMenu().

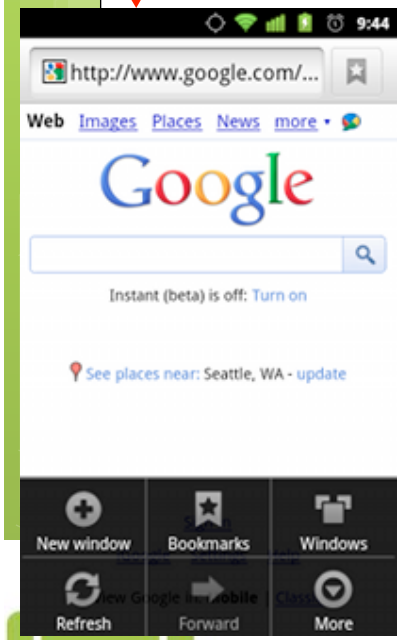
Menu types

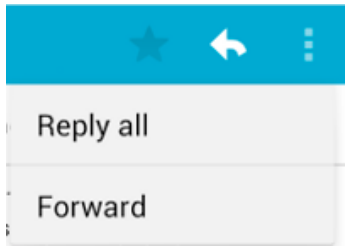


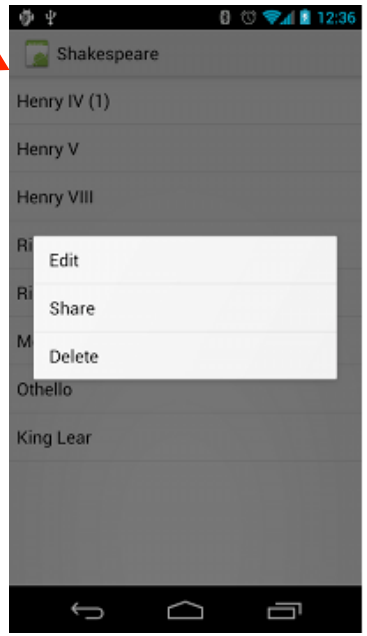
Traditional menus are awkward on a small screen.

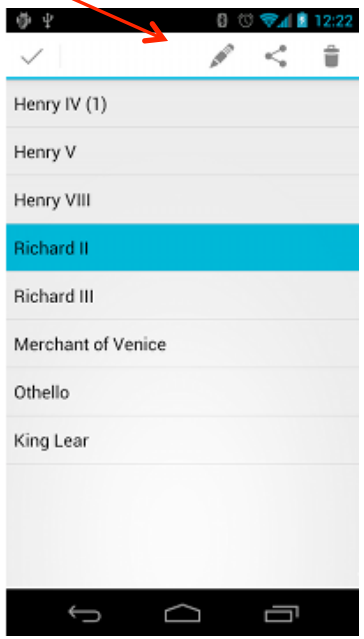
⇒ Three stages menus:

- Option Menu (< 3.0) / Action Bar (>=3.0)
- Context Menu (< 3.0) / Contextual Action mode(>=3.0)
- Popup Menu

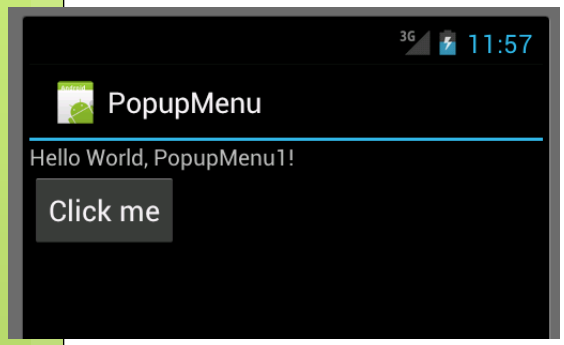


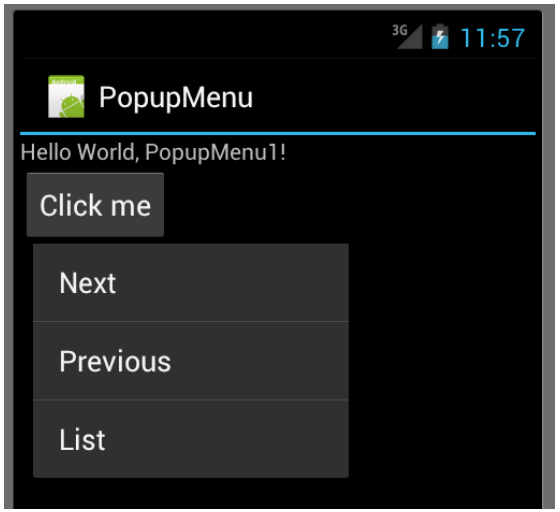


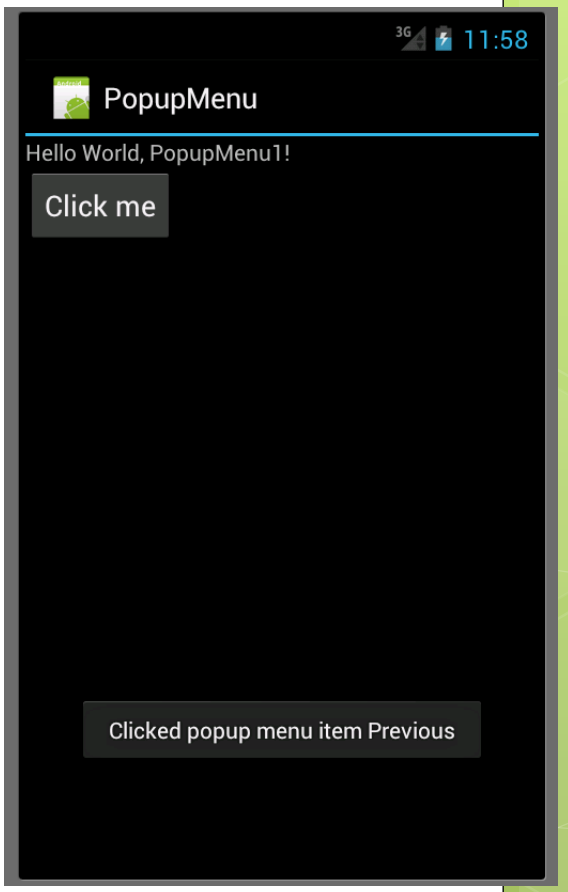




PopupMenu







Popup Menu

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/
android">
        <item android:id="@+id/next"
                android:title="@string/next" />
        <item android:id="@+id/previous"
            android:title="@string/previous" />
        <item android:id="@+id/list"
            android:title="@string/list" />
</menu>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=
   "http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
   <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/hello" />
   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:onClick="showPopup"
      android:text="Click me" />
</LinearLayout>
```

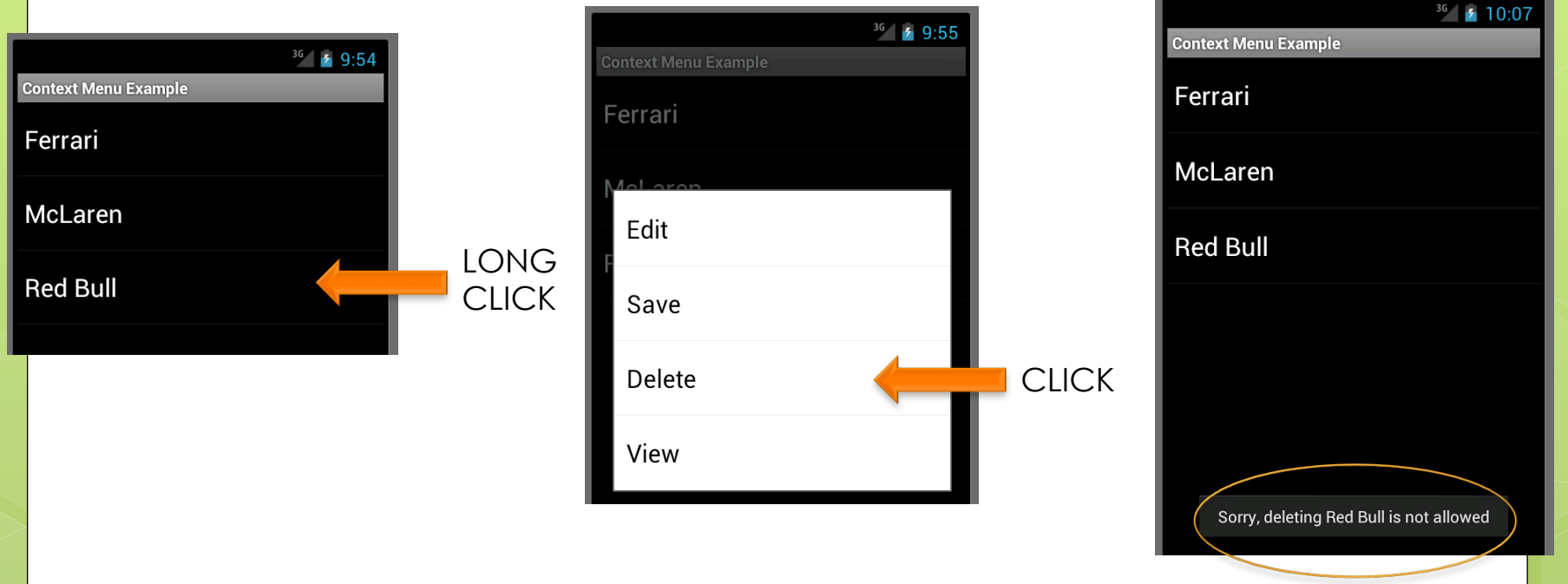
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="hello">Hello World, PopupMenu1!</string>
   <string name="app_name">PopupMenu</string>
   <string name="next">Next</string>
   <string name="previous">Previous</string>
   <string name="list">List</string>
</resources>
```

Popup Menu

```
public class PopupMenu1 extends Activity {
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
```

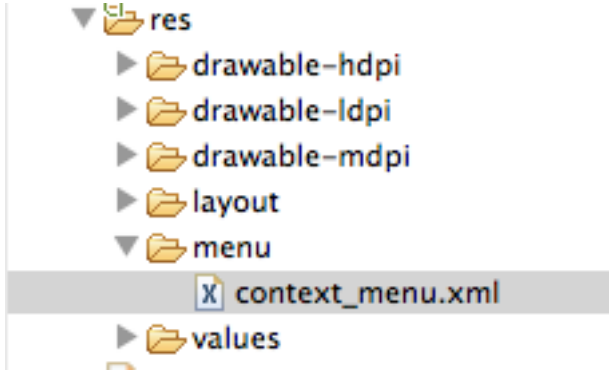
```
public void showPopup(View button) {
        PopupMenu popup = new PopupMenu(this, button);
        popup.getMenuInflater().inflate(R.menu.popup, popup.getMenu());
        popup.setOnMenuItemClickListener(new
              PopupMenu.OnMenuItemClickListener() {
           public boolean onMenuItemClick(MenuItem item) {
              Toast.makeText(PopupMenu1.this, "Clicked popup menu item " +
                   item.getTitle(),  Toast.LENGTH_LONG).show();
              return true;
           }
        });
        popup.show();
    }
}
```

ContextMenu



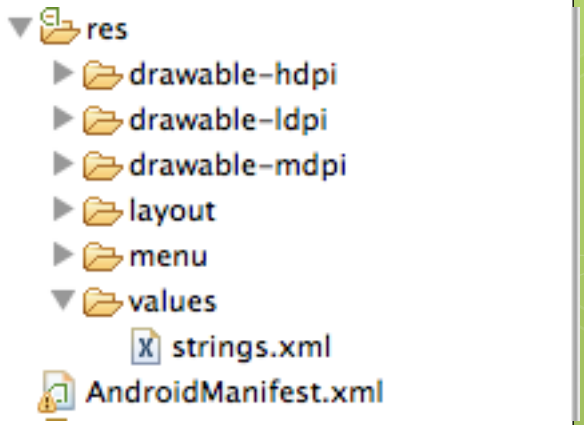
context_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu
 xmlns:android="http://schemas.android.com/apk/res/android">
        <item android:id="@+id/edit"
                android:title="@string/edit" />
        <item android:id="@+id/save"
            android:title="@string/save" />
        <item android:id="@+id/delete"
            android:title="@string/delete" />
        <item android:id="@+id/view"
            android:title="@string/view" />
</menu>
```



Strings

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="hello">ShowContextMenu</string>
   <string name="app_name">Context Menu Example</string>
   <string name="edit">Edit</string>
   <string name="save">Save</string>
   <string name="delete">Delete</string>
   <string name="view">View</string>
<string-array name="names">
            <item>Ferrari</item>
            <item>McLaren</item>
            <item>Red Bull</item>
</string-array>
</resources>
```



ContextMenu

```
public class ShowContextMenu extends ListActivity {
@Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setListAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1,
            getResources().getStringArray(R.array.names)));
      registerForContextMenu(getListView());
   }

 public void onCreateContextMenu(ContextMenu menu,
                                 View v, ContextMenuInfo menuInfo) {
            super.onCreateContextMenu(menu, v, menuInfo);
            MenuInflater inflater = getMenuInflater();
            inflater.inflate(R.menu.context_menu, menu);
   }
```

Responding to event

```
public boolean onContextItemSelected(MenuItem item) {
        AdapterContextMenuInfo info =
                (AdapterContextMenuInfo) item.getMenuInfo();
        String[] names = getResources().getStringArray(R.array.names);
        switch(item.getItemId()) {
        case R.id.delete:
            Toast.makeText(this,
                    Toast t=Toast.makeText(this, "Sorry, deleting " +
                    ((TextView)info.targetView).getText()
                    + " is not allowed", Toast.LENGTH_LONG).show();
            return true;
        default:
           Toast.makeText(this, "You have chosen the " + item.getTitle() +
                    " context menu option for " + names[(int)info.id],
                    Toast.LENGTH_SHORT).show();
           return true;
        }
    }
```

3.5 sec

2.0 sec

```
for (int i=0; i < 2; i++){ Toast.makeText(this, "blah", Toast.LENGTH_LONG).show(); }
```

A similar concept: QuickAction

displays contextual actions in a list view

Not (yet) a standard API!
For a hint in the implementation see
http://www.londatiga.net/it/how-to-create-quickaction-dialog-in-android/

Contextual action mode

Contextual action mode

```
public class ShowContextMenu extends ListActivity {
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setListAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1,
                  getResources().getStringArray(R.array.names)));
      //registerForContextMenu(getListView());
      ListView listView=getListView();
      final Context ctx=this;
      listView.setChoiceMod(ListView.CHOICE_MODE_MULTIPLE_MODAL);
      listView.setMultiChoiceModeListener(new MultiChoiceModeListener() {
            …
      });
   }
}
```

MultiChoiceModeListener

```
listView.setMultiChoiceModeListener(new MultiChoiceModeListener() {
    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
            MenuInflater inflater = getMenuInflater();
            inflater.inflate(R.menu.context_menu, menu);
            mode.setTitle("selection");
            return true;
    }
    public void onDestroyActionMode(ActionMode mode) {
        // Here you can make any necessary updates to the activity when
        // the CAB is removed. By default, selected items are deselected/unchecked
    }
    public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
        // Here you can perform updates to the CAB due to an invalidate() request
        return true;
    }
```

MultiChoiceModeListener

```
public void onItemCheckedStateChanged(ActionMode mode,
                    int position, long id, boolean checked) {
        // Here you can do something when items are selected/de-selected,
        // such as update the title in the CAB
        String[] names = getResources().getStringArray(R.array.names);
        if (checked) mode.setTitle(names[position]);
    }
```

MultiChoiceModeListener

```
public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
        switch(item.getItemId()) {
            case R.id.delete:
                Toast t=Toast.makeText(ctx, "Sorry, deleting " +
                        mode.getTitle()  + " is not allowed", Toast.LENGTH_LONG);
                t.show();
                mode.finish();
                return true;
            default:
                Toast.makeText(ctx, "You have chosen the " + item.getTitle() +
                        " context menu option for " + mode.getTitle(),
                        Toast.LENGTH_LONG).show();
                mode.finish();
                return true;
            }
}
```