

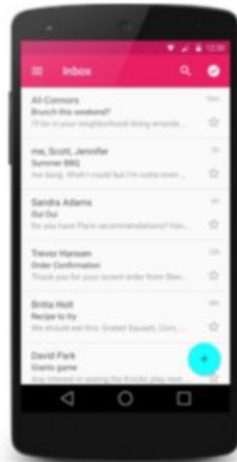
# ListView



Class 4

# Listview

*An ordered collection of selectable choices*



- key attributes in XML:

`android:clickable="bool"`

set to false to disable the list

`android:id="@+id/theID"`

unique ID for use in Java code

`android:entries="@array/array"`

set of options to appear in the list  
(must match an array in `strings.xml`)

# Static Lists

- **static list:** Content is fixed and known before the app runs.
  - Declare the list elements in the **strings.xml** resource file.

```
<!-- res/values/strings.xml -->
<resources>
    <string-array name="oses">
        <item>Android</item>
        <item>iPhone</item>
        ...
        <item>Max OS X</item>
    </string-array>
</resources>

<!-- res/layout/activity_main.xml -->
<ListView ... android:id="@+id/mylist"
    android:entries="@array/oses" />
```

Android

iPhone

WindowsMobile

Blackberry

WebOS

Ubuntu

Windows7

Max OS X

# String resources

- Declare constant strings and arrays in **res/values/strings.xml**:

```
1 <resources>
2     <string name="name">value</string>
3     <string name="name">value</string>
4
5     <string-array name="arrayname">
6         <item>value</item>
7         <item>value</item>    <!-- must escape ' as \' in values -->
8         ...
9         <item>value</item>
10    </string-array>
11 </resources>
```

- Refer to them in Java code:

- as a resource: `R.string.name`, `R.array.name`
- as a string or array: `getResources().getString(R.string.name)`,  
`getResources().getStringArray(R.array.name)`

# Dynamic Lists

- **dynamic list:** Content is read or generated as the program runs.
  - Comes from a data file, or from the internet, etc.
  - Must be set in the Java code.
  - Suppose we have the following file and want to make a list from it:

```
// res/raw/oses.txt  
Android  
iPhone  
...  
Max OS X
```

Android

iPhone

WindowsMobile

Blackberry

WebOS

Ubuntu

Windows7

Max OS X

# List Adapters

- **adapter**: Helps turn list data into list view items.
  - common adapters: ArrayAdapter, CursorAdapter

- Syntax for creating an adapter:

```
ArrayAdapter<String> name =  
    new ArrayAdapter<String>(activity, layout, array);
```

- the ***activity*** is usually this
  - the default ***layout*** for lists is `android.R.layout.simple_list_item_1`
  - get the ***array*** by reading your file or data source of choice  
(it can be an array like `String[]`, or a list like `ArrayList<String>`)
- Once you have an adapter, you can attach it to your list by calling the `setAdapter` method of the `ListView` object in the Java code.

# List Adapter Example

```
ArrayList<String> myArray = ...;  // load data from file

ArrayAdapter<String> adapter =
    new ArrayAdapter<String>(
        this,
        android.R.layout.simple_list_item_1,
        myArray);

ListView list = (ListView) findViewById(R.id.mylist);
list.setAdapter(myAdapter);
```

# Handling List Events

- Unfortunately lists don't use a simple `onClick` event.
  - Several fancier GUI widgets use other kinds of events.
  - The event listeners must be attached in the Java code, not in the XML.
  - Understanding how to attach these event listeners requires the use of Java **anonymous inner classes**.
- **anonymous inner class**: A shorthand syntax for declaring a small class without giving it an explicit name.
  - The class can be made to extend a given super class or implement a given interface.
  - Typically the class is declared and a single object of it is constructed and used all at once.

Android

iPhone

WindowsMobile

Blackberry

WebOS

Ubuntu

Windows7

Max OS X



# Attaching Event Listener in Java

```
<!-- activity_main.xml -->
<Button ... android:onClick="mybuttonOnClick" />
<Button ... android:id="@+id/mybutton" />

// MainActivity.java
public void mybuttonOnClick() { ... }
Button button = (Button) findViewById(R.id.mybutton);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // code to run when the button gets clicked
    }
});

// this was the required style for event listeners
// in older versions of Android :-/
```

# List Events

- List views respond to the following events:
  - **setOnItemClickListener**(AdapterView.OnItemClickListener)  
Listener for when an item in the list has been clicked.
  - **setOnItemLongClickListener**(AdapterView.OnItemLongClickListener)  
Listener for when an item in the list has been clicked and held.
  - **setOnItemSelectedListener**(AdapterView.OnItemSelectedListener)  
Listener for when an item in the list has been selected.
- Others:
  - onDrag, onFocusChanged, onHover, onKey, onScroll, onTouch, ...

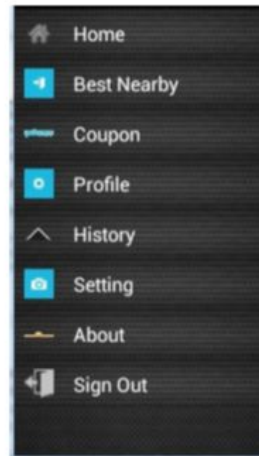


# List Event Listener Example

```
ListView list = (ListView) findViewById(R.id.id);
list.setOnItemClickListener(
    new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> list,
                                View row,
                                int index,
                                long rowID) {
            // code to run when user clicks that item
            ...
        }
    }
);
```

# Custom List Layouts

- If you want your list to look different than the default appearance (of just a text string for each line), you must:
  - Write a short **layout XML file** describing the layout for each row.
  - Write a **subclass of ArrayAdapter** that overrides the **getView** method to describe what view must be returned for each row.



# Custom List Layout XML

```
<!-- res/layout/mylistlayout.xml -->
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout ... android:orientation="horizontal">
    <ImageView ... android:id="@+id/list_row_image"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/smiley" />

    <TextView ... android:id="@+id/list_row_text"
        android:textStyle="bold"
        android:textSize="22dp"
        android:text=""
        android:background="#336699" />
</LinearLayout>
```

# Custom List Layout in Java

```
// MyAdapter.java
public class MyAdapter extends ArrayAdapter<String> {
    private int layoutResourceId;
    private List<String> data;

    public MyAdapter(Context context, int layoutId, List<String> list) {
        super(context, layoutResourceId, data);
        layoutResourceId = layoutId;
        data = list;
    }

    @Override
    public View getView(int index, View row, ViewGroup parent) {
        row = getLayoutInflater().inflate(layoutResourceId, parent, false);
        TextView text = (TextView) row.findViewById(R.id.list_row_text);
        text.setText(data.get(index));
        return row;
    }
}
```