Android Applications

Dominic Duggan
Stevens Institute of Technology

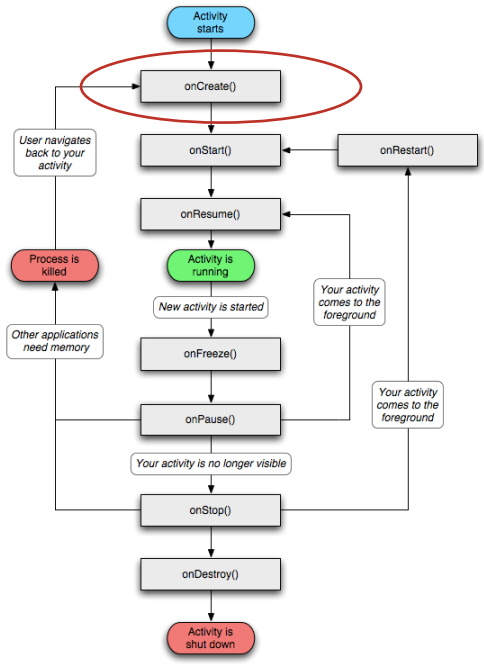
ANDROID APPLICATIONS

Android Basics

- (Mostly) Three Categories of Applications
  - Foreground activities
    - Suspended when not visible
    - E.g. games, mashups
  - Background services
    - E.g. call screening, SMS auto-responders
  - Intermittent activity
    - E.g. media player

3

Activity Life Cycle



4

Building Blocks for Applications

- Activities: presentation layer
- Services
- Broadcast Receivers
- Content Providers
- Intents
- Notifications

5

Building Blocks for Applications

- Activities: presentation layer
  - UI for one focused endeavor
  - Visual content via views
  - Activities invoke other activities
- Services
- Broadcast Receivers
- Content Providers
- Intents
- Notifications

6

Building Blocks for Applications

- Activities: presentation layer
- Services
  - Background services
  - RPC communication
  - Run in the *main* thread
- Broadcast Receivers
- Content Providers
- Intents
- Notifications

7

Building Blocks for Applications

- Activities: presentation layer
- Services
- Broadcast Receivers
  - React to broadcast messages
  - Publish-subscribe (intents)
- Content Providers
- Intents
- Notifications

8

Building Blocks for Applications

- Activities: presentation layer
- Services
- Broadcast Receivers
- Content Providers
  - Make data available
  - Content Resolver: start process
- Intents
- Notifications

9

Building Blocks for Applications

- Activities: presentation layer
- Services
- Broadcast Receivers
- Content Providers
- Intents
  - Asynchronous messages
  - Intent filters
- Notifications

10

Building Blocks for Applications

- Activities: presentation layer
- Services
- Broadcast Receivers
- Content Providers
- Intents
- Notifications
  - Dialogues and modal messages

11

APPLICATION MANIFEST

12

Application Manifest

- Stored in root of project hierarchy
  - AndroidManifest.xml
  - Define structure and metadata of application
  - Nodes for each of the components

```
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="edu.stevens.cs522.hello">
…
</manifest>
```

13

Application Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.stevens.cs522.hello"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="10" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="edu.stevens.cs522.hello.HelloActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

14

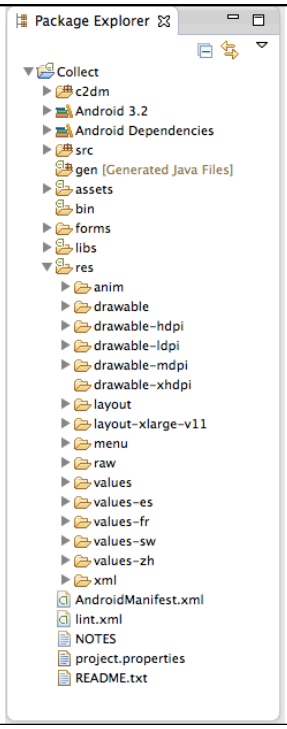
Application Manifest

- Manifest node tags
  - Application: container for…
    - Activity: specify intent filter
    - Service
    - Provider
    - Receiver: global broadcast receiver
  - Uses-permission:
    - Must be granted during installation
  - Permission
    - Declare to restrict access to components in app
  - instrumentation

15

Resources

- External resources
  - Values
    - Strings, colors, dimensions, string or integer arrays
  - Styles and themes
    - Colors and fonts
  - Drawables
    - Bitmaps and (stretchable PNG) images
  - Layouts
    - UI specified statically in XML
    - Android best practice



Using Resources

- In code:
  - Using the static R class
  - Static subclasses e.g. R.string, R.drawable
  - Reference to resource table e.g. R.layout.main
  - Dynamic lookup

```
Resources myResources = getResources();
CharSequence styledText = myResources.getText
  (R.string.stop_message);
Button b = (Button) findViewById(R.id.ok_button);
```

17

Using Resources

```
…<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Hello, World"
    />…
```

- In code:
  - Using the static R class
  - Static subclasses e.g. R.string, R.drawable
  - Reference to resource table e.g. R.layout.main
  - Dynamic lookup

```
Resources myResources = getResources();
CharSequence styledText = myResources.getText
  (R.string.stop_message);
Button b = (Button) findViewById(R.id.ok_button);
```

- In resources

18

Using Resources

```
…<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello_message"
    />…
```

- In code:
  - Using the static R class
  - Static subclasses e.g. R.string, R.drawable
  - Reference to resource table e.g. R.layout.main
  - Dynamic lookup

```
Resources myResources = getResources();
CharSequence styledText = myResources.getText
  (R.string.stop_message);
Button b = (Button) findViewById(R.id.ok_button);
```

- In resources

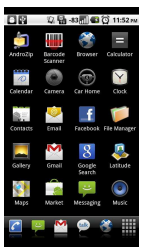
19

Activity Node in the Application Manifest

Entry point for a task

```
<activity android:label="@string/app_name"
          android:name=".MyActivity">
  <intent-filter>
    <action
        android:name="android.intent.action.MAIN"/>
    <category
      android:name="android.intent.category.LAUNCHER"/>
  </intent-filter>
</activity>
```

Icon in the application launcher

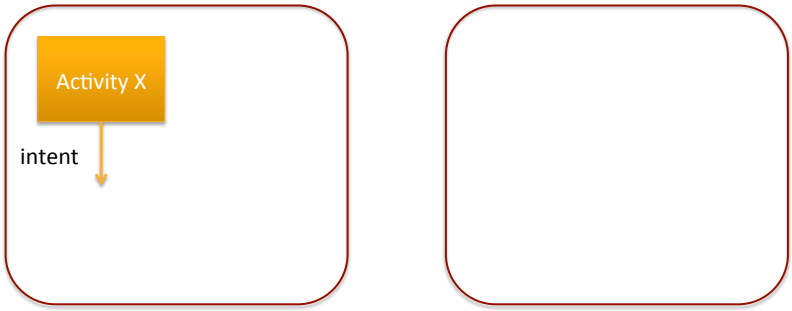


INTENTS

21

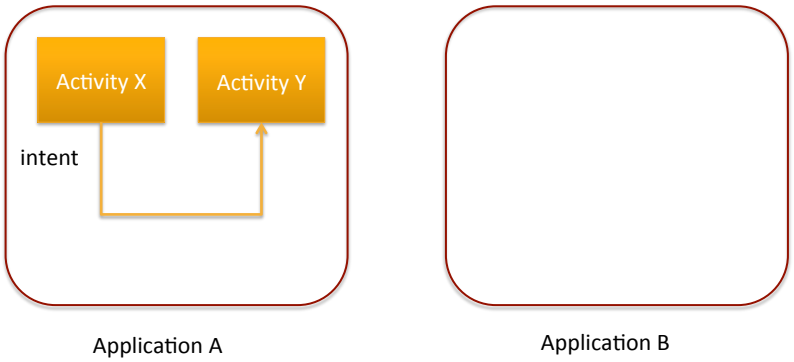
Intents

- Intents: Asynchronous messages to launch new activities / services, or send broadcasts



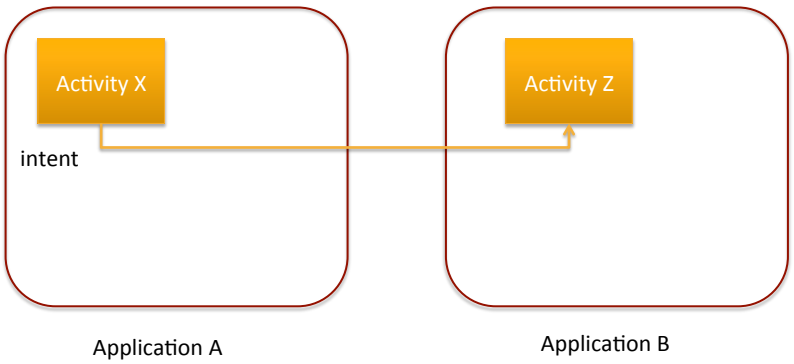
Intents

- Intents: Asynchronous messages to launch new activities / services, or send broadcasts



Intents

- Intents: Asynchronous messages to launch new activities / services, or send broadcasts



Intents

- Explicit intents: specify the activity to be started

```
Intent intent = new Intent(MyActivity.this,
                           MyOtherActivity.class);
startActivity(intent);
```

25

Intents

- Explicit intents: specify the activity to be started

```
Intent intent = new Intent(MyActivity.this,
                           MyOtherActivity.class);
startActivity(intent);
```

- Implicit intents: specify data and action, and system resolves the activity

```
Intent intent = new Intent(Intent.ACTION_DIAL,
                     Uri.parse("tel:555-2368"));
startActivity(intent);
```

26

Information in Intents (1)

- Component name: for handler of intent
  - For explicit intents

Information in Intents (1)

- Component name: for handler of intent
  - For explicit intents
- Action
  - String naming action to be performed
  - E.g. ACTION_CALL, ACTION_EDIT, ACTION_SYNC
  - Akin to a method name
- Data
  - URI, and opt MIME type, of data to be acted on
  - E.g. ACTION_CALL with tel: URI
  - E.g. ACTION_EDIT with URI of document
  - E.g. ACTION_VIEW with content://contacts/people/17

Information in Intents (2)

- Category:
  - The kind of component that should handle the intent
  - E.g. CATEGORY_BROWSABLE
  - E.g. CATEGORY_GADGET
  - E.g. CATEGORY_HOME
  - E.g. CATEGORY_LAUNCHER
- Extras
  - Key-value pairs
- Flags

Intent Filters (1)

- Intent filters register activities, services and broadcast receivers
  - `<intent-filter>` element in component's manifest node
  - Child elements for action, data, category

Intent Filters (2)

- Intent filters register activities, services and broadcast receivers
  - `<intent-filter>` element in component's manifest node
  - Child elements for action, data, category

```
<intent-filter . . . >
   <action
   android:name="com.example.project.SHOW_CURRENT" />
   <action
   android:name="com.example.project.SHOW_RECENT" />
   <action
   android:name="com.example.project.SHOW_PENDING" />
      . . .
</intent-filter>
```

31

Intent Filters (3)

- Intent filters register activities, services and broadcast receivers
  - `<intent-filter>` element in component's manifest node
  - Child elements for action, data, category

```
<intent-filter ... >
  <data android:mimeType="audio/mpeg" ... />

  <data android:mimeType="video/mpeg"
   android:scheme="http"
   host="www.example.com"
   port="200" path="/folder/subfolder" ... />
   ...
</intent-filter>
```

`http://www.example.com:200/folder/subfolder`

32

Intent Filters (3)

- Intent filters register activities, services and broadcast receivers
  - `<intent-filter>` element in component's manifest node
  - Child elements for action, data, category

```
<intent-filter ...>
  <category
    android:name="android.intent.category.DEFAULT" />
  <category
    android:name="android.intent.category.BROWSABLE" />
  ...
</intent-filter>
```

Every category in the intent object must match a category in the filter

33

Intent Resolution

- Android lists available intent filters
- Matching intent filters:
  - Intent filters must match action and category
    - Match all categories
  - Match intent data URI to intent filter data tag
    - Match scheme, host/authority, path or mime type, where specified

34

Resolving action and data

- Activity started because it matched intent
- Must learn action and data

```
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  setContentView (R.layout.main);

  Intent intent = getIntent();
  String action = intent.getAction();
  Uri data = intent.getData();
}
```

35

Subactivities with Results

- Parent Activity: Launching a subactivity:

```
Uri data = …;
Intent intent = new Intent(Intent.ACTION_PICK, data);
// Result returned in onActivityResult
startActivityForResult(intent, PICK_SUBACTIVITY);
```

Start a subactivity for a dialog (there are lighter weight alternatives)…

Request code

36

Subactivities with Results

- Child Activity: Returning Results (`onCreate` method):

```
Button okButton = (Button) findViewById(R.id.ok_button);

ButtonListener okListener = new View.OnClickListener() {
   public void onClick(View view) {
      Uri data = …;
      Intent result = new Intent(null, data);
      result.putExtra(IS_INPUT_CORRECT, inputCorrect);
      setResult(RESULT_OK, result);
      finish();
   }
}

okButton.setOnClickListener(okListener);
```

Subactivity registers listener for clicking OK…

37

Subactivities with Results

- Child Activity: Returning Results (`onCreate` method):

```
Button cancelButton =
   (Button) findViewById(R.id.cancel_button);

ButtonListener cancelListener = new View.OnClickListener
  () {
      public void onClick(View view) {
         setResult(RESULT_CANCELED, null);
         finish();
      }
   }

cancelButton.setOnClickListener(cancelListener);
```

…and registers listener for clicking CANCEL…

38

Subactivities with Results

- Parent Activity: Handling Results:

```
public void onActivityResult(int requestCode,
                             int resultCode,
                             Intent data) {
   super.onActivityResult(requestCode, resultCode, data);

   switch(requestCode) {
      case (PICK_SUBACTIVITY) :
         if (resultCode == Activity.RESULT_OK) {
            // TODO Handle user clicked OK.
         }
         break;
   }
}
```

39

S

```
Uri data = …;
Intent intent = new Intent(Intent.ACTION_PICK, data);
// Result returned in onActivityResult
startActivityForResult(intent, PICK_SUBACTIVITY);
```

- Handling Results:

```
public void onActivityResult(int requestCode,
                             int resultCode,
                             Intent data) {
   super.onActivityResult(requestCode, resultCode, data);

   switch(requestCode) {
      case (PICK_SUBACTIVITY) :
         if (resultCode == Activity.RESULT_OK) {
            // TODO Handle user clicked OK.
         }
         break;
   }
}
```

40

TASKS AND ACTIVITIES

41

Tasks and Activities

Task: stack of activities

42

Tasks and Activities

An activity uses an intent to activate another activity

43

Tasks and Activities

The user may choose to start another task from the home screen.

44

Tasks and Activities

Go back to previous task via the home screen

45

Tasks and Activities

BACK: go to the previous activity in *this* task

Almost all of this can be reprogrammed using intents and affinities.

46

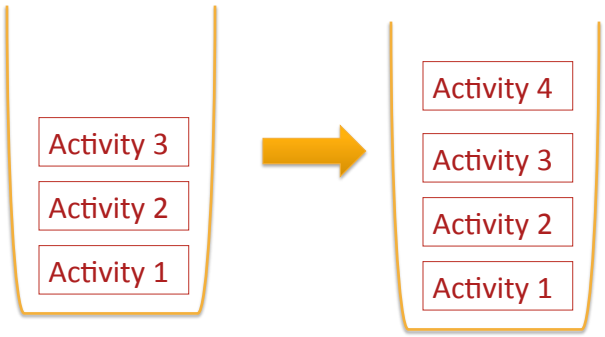
Affinities

- Afinity between activities in an application
- `taskAffinity` attribute of <activity>
  - element to override default
- May be shared with activities in other tasks

47

Affinities

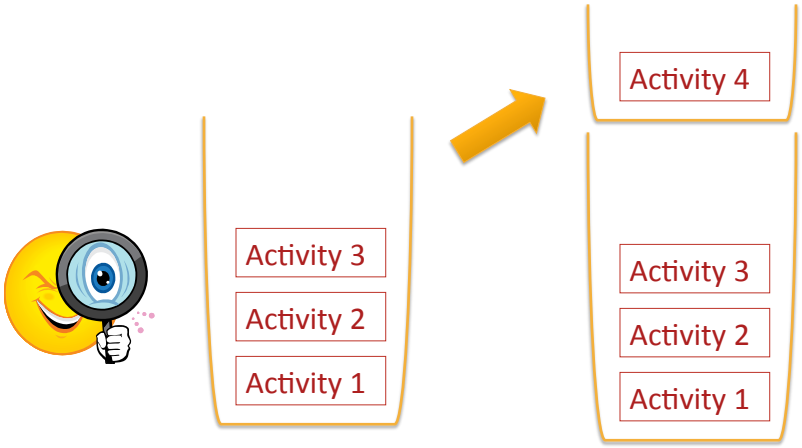
Default behavior



48

Affinities

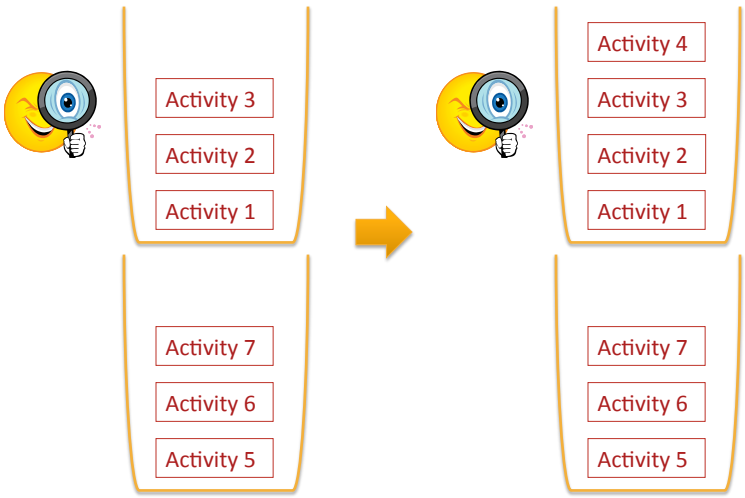
FLAG_ACTIVITY_NEW_TASK flag
in the intent



49

Affinities

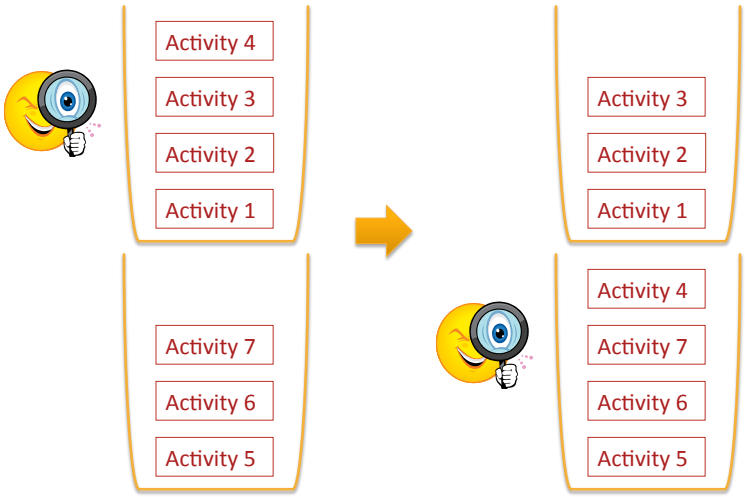
allowTaskReparenting attribute in
activity element



50

Affinities

allowTaskReparenting attribute in activity element



51

Processes and Threads

- Component elements (<activity>, <service>, <receiver>, <provider>) can specify with process attribute where they should run
  - Each component may run in its own process
  - Some components may share a process
  - Components of different applications may run in same process

52

Launch Modes

Specified in <activity> element's launchMode attribute

	Which task activity starts in (default)	Multiple instances of activity	Other activities in its task	New instance for new intent
standard (default)	Originating task	Yes	Yes	Yes
singleTop	Originating task	Yes	Yes	Re-used if on top of stack
singleTask	New task	No	Yes (but it is always root activity)	No, intent dropped if not on top
singleInstance	New task	No	No	No, only activity in task

53

Processes and Threads

- Each component runs in the main thread
- Free up resources: which process to terminate?

54

ACTIVITY LIFE CYCLE

55

Activity States

- Active
- Paused
- Stopped
- Inactive

Visible and running

Visible but some other activity is running

No longer visible

Not yet launched or just killed

56

Application Priority

Active Process

Visible Process

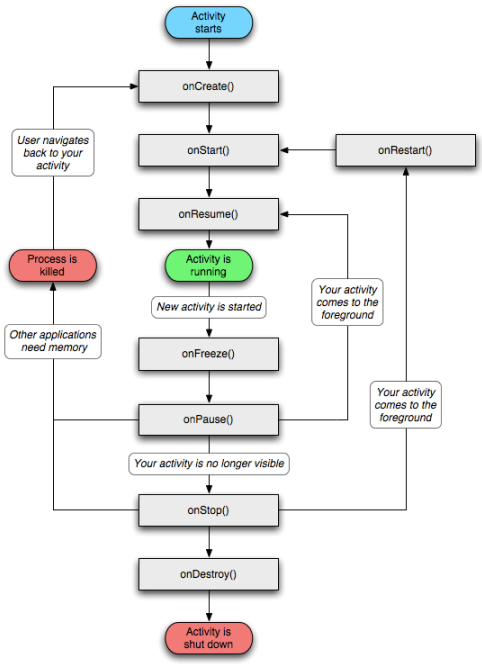
Started Service Process

Background Process

Empty Process

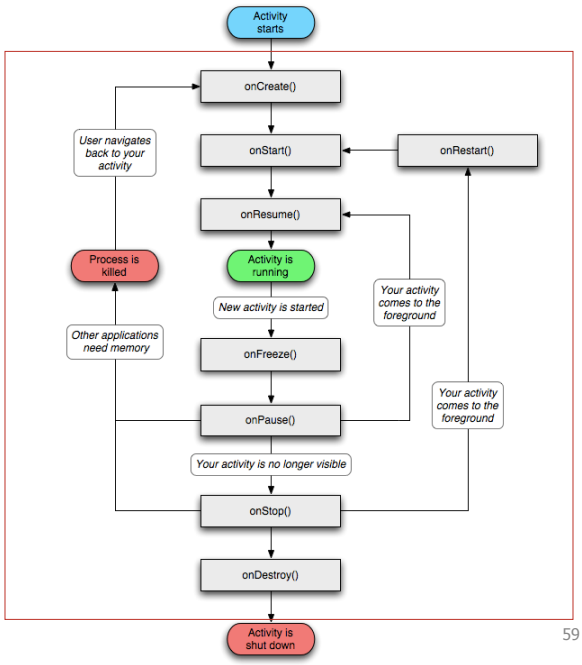
57

Activity Life Cycle



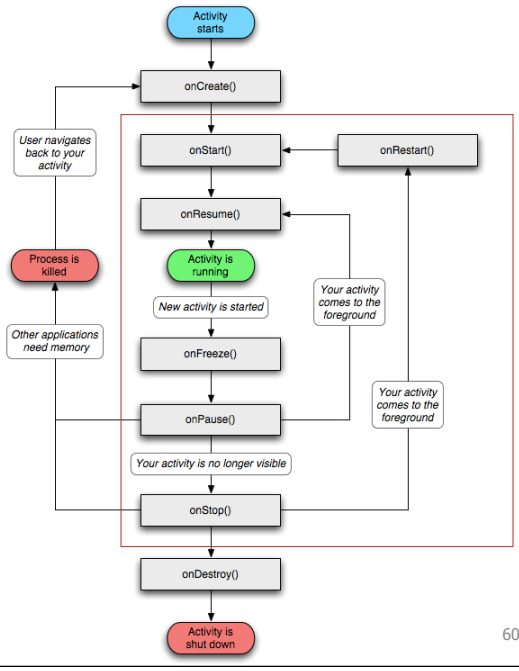
58

Full lifetime
- From first call to onCreate…
- …to final call to onDestroy
- Avoid creation of short-term objects, create them once in onCreate



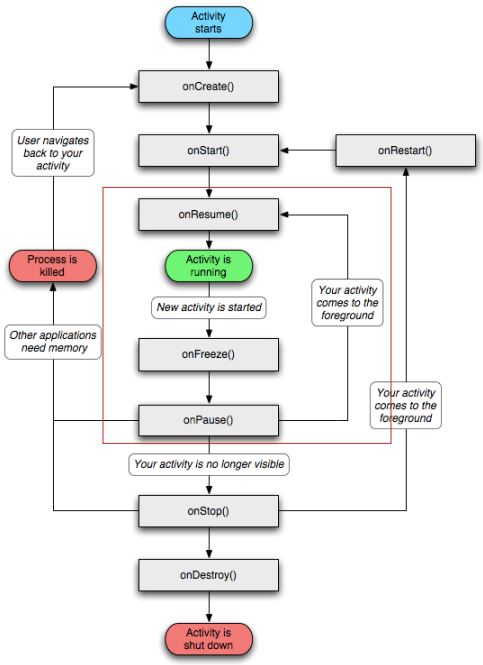
59

Visible lifetime
- From call to onStart to onStop
- Visible but does not have the focus



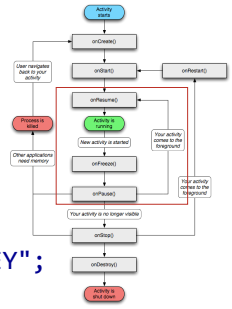
60

Active lifetime
- From call to onResume to onPause
- onSaveInstanceState called just before onPause, to save UI state, to be reloaded (just after onCreate) by call to onRestoreInstanceState



61

Application UI State

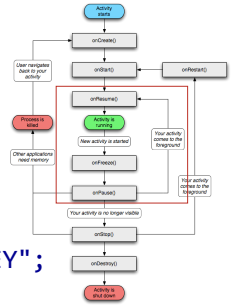


- Save UI state while activity is not active

```
private static final String
    TEXTVIEW_STATE_KEY = "TEXTVIEW_STATE_KEY";

@Override
public void onSaveInstanceState(Bundle state) {
  // Retrieve the View
  TextView myTextView =
       (TextView)findViewById(R.id.myTextView);
  // Save its state
  state.putString(TEXTVIEW_STATE_KEY,
                  myTextView.getText().toString());
  super.onSaveInstanceState(state);
}
```

62

Application UI State



- Save UI state while activity is not active

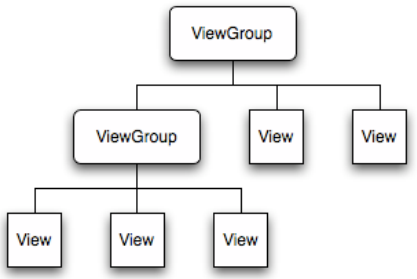
```
private static final String
    TEXTVIEW_STATE_KEY = "TEXTVIEW_STATE_KEY";

@Override
public void onRestoreInstanceState(Bundle state) {
  super.onRestoreInstanceState(state);
  // Todo Restore instance state values
}
```

63

USER INTERFACES

64

View Hierarchy



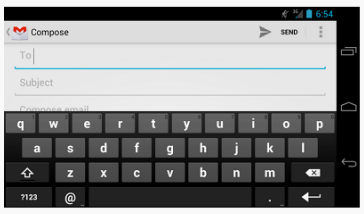
- View groups define hierarchies of views.
- The layout on the screen is described in an XML document (elements are views and view groups).
- Input controls are views that support interaction e.g. buttons, text entry, etc.

65

Views

- TextView
- EditText
- Spinner
- Button
- CheckBox
- RadioButton
- Etc…

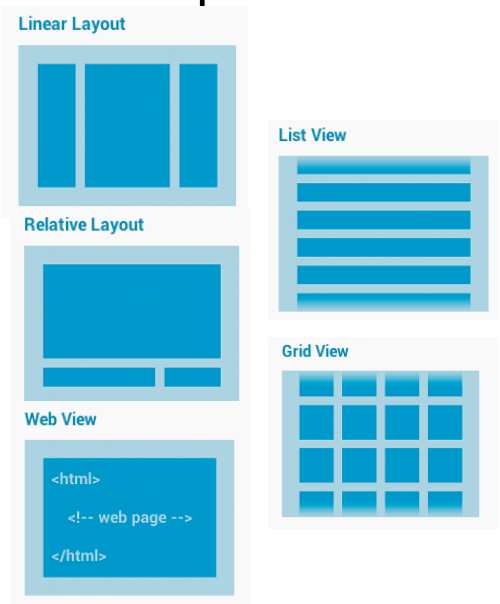
```
<EditText
    android:id="@+id/email_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/email_hint"
    android:inputType="textEmailAddress" />
```



66

View Groups

- LinearLayout
- AbsoluteLayout
- TableLayout
- RelativeLayout
- FrameLayout
- WebView
- ScrollView
- ListView
- GridView



67

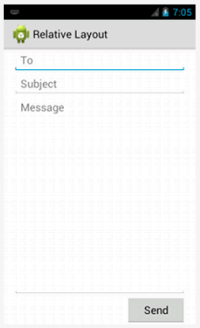
Attributes

- `layout_width, layout_height`
  - `fill_parent`
  - `wrap_content`
- `layout_marginTop, layout_marginBottom`
- `layout_marginLeft, layout_marginRight`
- `layout_gravity`
- `layout_weight`
- `layout_x`
- `layout_y`

For LinearLayout or TableLayout

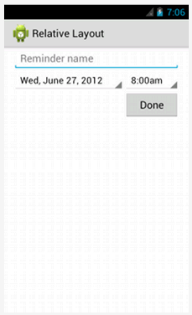
Linear Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```



Relative Layout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.co
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```



Event Listener API

- `onClick()`
  - from `View.OnClickListener`
- `onLongClick()`
  - from `View.OnLongClickListener`
- `onFocusChange()`
  - from `View.onFocusChange`
- `onKey()`
  - from `View.OnKeyListener`
  - *Hardware key*
- `onTouch()`
  - from `View.OnTouchListener`
- `onCreateContextMenu()`
  - from View.OnCreateContextMenuListener

Example: Button

```
private OnClickListener btnListener =
   new OnClickListener() {
      public void onClick(View v) {
        // do something when the button is clicked
      }
   };

protected void onCreate(Bundle savedValues) {
    ...
    // Capture button from layout
    Button okButton = (Button)findViewById(R.id.ok_button);
    // Register the onClick listener
    okButton.setOnClickListener(btnListener);
    ...
}
```

Example: Button

```
public class ExampleActivity extends Activity
                              implements OnClickListener {
   protected void onCreate(Bundle savedValues) {
      ...
      Button okButton =
                (Button)findViewById(R.id.ok_button);
      okButton.setOnClickListener(this);
   }

   // Implement the OnClickListener callback
   public void onClick(View v) {
      // do something when the button is clicked
   }
   ...
}
```

Example: Edit Text

```
myEditText = (EditText)findViewById(R.id.myEditText);

// Assign a listener to the DPad button
OnKeyListener textListener = new OnKeyListener() {
    public boolean onKey(View v, int keyCode, KeyEvent event) {
        if (event.getAction() == KeyEvent.ACTION_DOWN)
            if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
               ... myEditText.getText().toString() ...
               myEditText.setText("");
               return true;
            }
        return false;
    }
}

myEditText.setOnKeyListener(textListener);
```

Example: Edit Text

```
myEditText = (EditText)findViewById(R.id.myEditText);

// Process input when textbox loses input focus
OnFocusChangeListener textListener = new OnFocusChangeListener() {
    private boolean hadFocus = false;
    public void onFocusChange(View v, boolean hasFocus) {
        if (hasFocus) {
           hadFocus = true;
        } else {
           ... myEditText.getText().toString() ...
           myEditText.setText("");
           hadFocus = false;
        }
}
}

myEditText.setOnFocusChangeListener(textListener);
```

In button click listener:
`button.requestFocusFromTouch()`

Event Handlers

- Defined for View API
- Modify for customized view

- `onKeyDown (int, KeyEvent)`
- `onKeyUp (int, KeyEvent)`
- `onTrackballEvent (MotionEvent)`
- `onTouchEvent (MotionEvent)`
- `onFocusChanged (boolean, int, Rect)`

Creating New Views

- Modify an existing view
  - Override event handlers e.g. onDraw(), onMeasure(), …
- Compound Control: Combine controls
  - Example: dropdown box combining TextView and Button
  - Best approach: Extend Layout class
- Custom View: Create an entirely new control
- Custom key press handling: e.g. for games

77

ADAPTERS

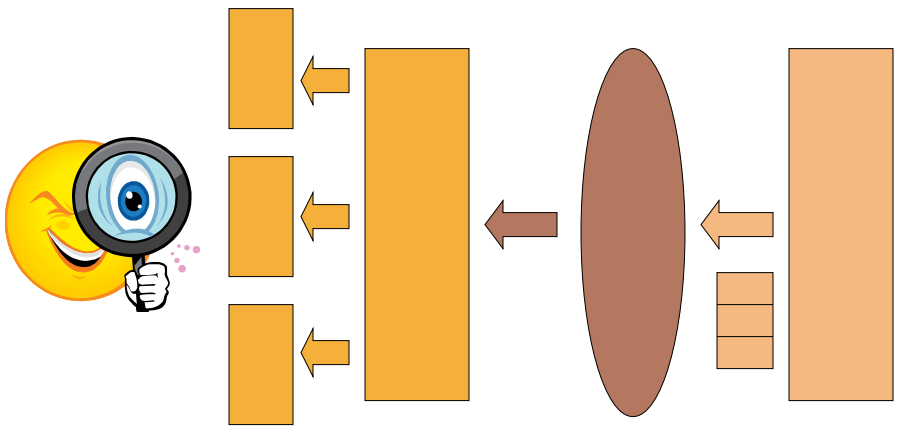
78

Adapters

- Bind data to user-interface views
- Create child views to represent each item
- Provide access to the underlying data
- Useful pre-defined adapter classes:
  - ArrayAdapter
    - Bind array of data to a view
  - SimpleCursorAdapter
    - Bind result of database query to a view

79

Adapters

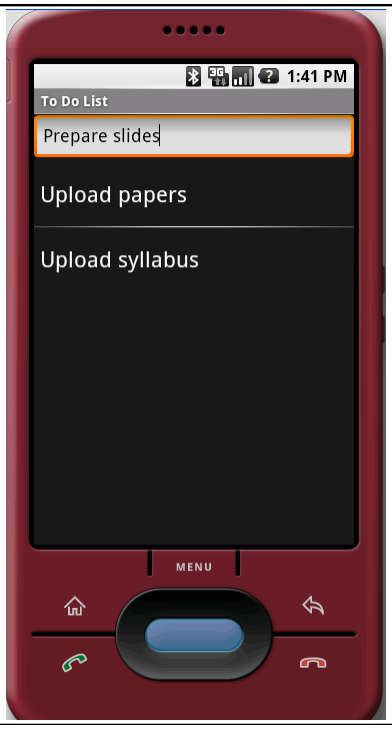


80

Example: To Do List

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <EditText
        android:id="@+id/myEditText"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="New To Do Item"
   />
   <ListView
        android:id="@+id/myListView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
   />
</LinearLayout>
```

81



82

Example (cont'd)

```
myListView = (ListView)findViewById(R.id.myListView);

todoItems = new ArrayList<String>();

aa = new ArrayAdapter<String>(getApplicationContext(),
                              R.layout.todolist_item,
                              todoItems);

myListView.setAdapter(aa);

...

registerForContextMenu(myListView);
```

```
<ListView
    android:id="@+id/myListView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
/>
```

83

```
myEditText = (EditText)findViewById(R.id.myEditText);

// Assign the KeyListener to the DPad button to add new items
OnKeyListener textListener = new OnKeyListener() {
    public boolean onKey(View v, int keyCode, KeyEvent event) {
        if (event.getAction() == KeyEvent.ACTION_DOWN)
            if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
                // Add a new todo item and clear the input box.
                todoItems.add(0, myEditText.getText().toString());
                myEditText.setText("");
                aa.notifyDataSetChanged();
                cancelAdd();
                return true;
            }
        return false;
    }
}

myEditText.setOnKeyListener(textListener);

registerForContextMenu(myListView);
```

```
<EditText
    android:id="@+id/myEditText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="New To Do Item"
/>
```

84

DEFINING NEW VIEWS

Example: Modified To-Do List

- Modify its "theme" to use the default Android theme with a specific font size
- Modify the text view for items to add a paper background

86

Example: Specialized Theme

- In manifest file:

```
<application android:icon="@drawable/icon">
  <activity
      android:name=".ToDoList"
      android:label="@string/app_name"
      android:theme="@style/ToDoTheme"> …
   </activity>
</application>
```

- In style resources file:

```
<style name="ToDoTheme"
       parent="@android:style/Theme.Black">
  <item name="android:textSize">12sp</item>
</style>
```

87

Example: Specialize the Text View

```
public class ToDoList extends Activity {

   public void onCreate(Bundle icicle) {
      super.onCreate(icicle);
      setContentView(R.layout.main);

      ListView myListView = (ListView)findViewById(R.id.myListView);

      final ArrayList<String> todoItems = new ArrayList<String>();
      int resID = R.layout.todolist_item;
      final ArrayAdapter<String>aa =
           new ArrayAdapter<String>(this, resID, todoItems);
      myListView.setAdapter(aa);

   }

}
```

```
public class TodoListItemView
     extends TextView {
   ...
}
```

```
<com.paad.todolist.TodoListItemView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:padding="10dp"
  android:scrollbars="vertical"
  android:textColor="@color/notepad_text"
  android:fadingEdge="vertical"
/>
```

88

Example: Specialize the Text View

```
public class TodoListItemView extends TextView {

   // Initialize the View by creating the
   // paint objects needed for customizing onDraw

   private void init () {
      // Create the paint brushes we will use in the onDraw method.
      Resources myResources = getResources();
      marginPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
      marginPaint.setColor(
      myResources.getColor(R.color.notepad_margin));
      linePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
      linePaint.setColor(myResources.getColor(R.color.notepad_lines));

      // Get the paper background color and the margin width.
      paperColor = myResources.getColor(R.color.notepad_paper);
      margin = myResources.getDimension(R.dimen.notepad_margin);
   }

}
```

89

Example: Specialize the Text View

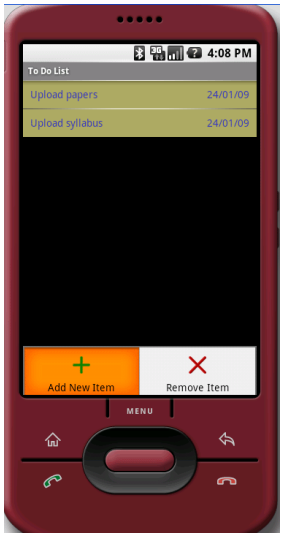
```
public class TodoListItemView extends TextView {

   public void onDraw (Canvas canvas) {
      canvas.drawColor(paperColor);

      // Draw ruled lines
      canvas.drawLine(0, 0, getMeasuredHeight(), 0, linePaint);
      canvas.drawLine(0, getMeasuredHeight(), getMeasuredWidth(),
                    getMeasuredHeight(), linePaint);

      // Draw margin
      canvas.drawLine(margin, 0, margin, getMeasuredHeight(), marginPaint);

      // Move the text across from the margin
      canvas.save();
      canvas.translate(margin, 0);

      // Use the TextView to render the text.
      super.onDraw(canvas);
      canvas.restore();
  }
}
```

90

91

Example: Task with Date

Modify the to-do list so that items have date as well as text



92

Example: Task with Date

- ToDo item:

```
public class ToDoItem {
   public String getTask() {
      return task;
   }
   String task;
   public Date getCreated() {
       return created;
   }
   Date created;
}
```

93

```
myListView = (ListView)findViewById(R.id.myListView);
myEditText = (EditText)findViewById(R.id.myEditText);
todoItems = new ArrayList<String>();
aa = new ArrayAdapter<String>(getApplicationContext(),
                              R.layout.todolist_item,
                              todoItems);
myListView.setAdapter(aa);

// Assign the KeyListener to the DPad button to add new items
myEditText.setOnKeyListener(new OnKeyListener() {
   public boolean onKey(View v, int keyCode, KeyEvent event) {
      if (event.getAction() == KeyEvent.ACTION_DOWN)
         if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
            // Add a new todo item and clear the input box.
            todoItems.add(0, myEditText.getText().toString());
            myEditText.setText("");
            aa.notifyDataSetChanged();
            cancelAdd();
            return true;
        }
        return false;
    }
});

registerForContextMenu(myListView);
```

94

```
myListView = (ListView)findViewById(R.id.myListView);
myEditText = (EditText)findViewById(R.id.myEditText);
todoItems = new ArrayList<ToDoItem>();
aa = new ArrayAdapter<ToDoItem>(getApplicationContext(),
                                R.layout.todolist_item,
                                todoItems);
myListView.setAdapter(aa);

// Assign the KeyListener to the DPad button to add new items
myEditText.setOnKeyListener(new OnKeyListener() {
   public boolean onKey(View v, int keyCode, KeyEvent event) {
      if (event.getAction() == KeyEvent.ACTION_DOWN)
         if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
            // Add a new todo item and clear the input box.
            ToDoItem item =
               new ToDoItem(myEditText.getText().toString())
            todoItems.add(0,item);
            myEditText.setText("");
            aa.notifyDataSetChanged();
            cancelAdd();
            return true;
         }
         return false;
    }
});

registerForContextMenu(myListView);
```

95

```
public class ToDoItemAdapter extends ArrayAdapter<ToDoItem> {
   public View getView(int position,
                       View convertView,
                       ViewGroup parent) {
      ToDoItem item = getItem(position);
      String taskString = item.getTask();
      Date createdDate = item.getCreated();
      SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy");
      String dateString = sdf.format(createdDate);

      … todoView = (LinearLayout) convertView;

      TextView dateView =
           (TextView)todoView.findViewById(R.id.rowDate);
      TextView taskView =
           (TextView)todoView.findViewById(R.id.row);
      dateView.setText(dateString);
      taskView.setText(taskString);

      return todoView;
   }
}
```

Extract task and date at current position…

```
public class ToDoItemAdapter extends ArrayAdapter<ToDoItem> {
   public View getView(int position,
                       View convertView,
                       ViewGroup parent) {
      ToDoItem item = getItem(position);
      String taskString = item.getTask();
      Date createdDate = item.getCreated();
      SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy");
      String dateString = sdf.format(createdDate);

      … todoView = (LinearLayout) convertView;

      TextView dateView =
           (TextView)todoView.findViewById(R.id.rowDate);
      TextView taskView =
           (TextView)todoView.findViewById(R.id.row);
      dateView.setText(dateString);
      taskView.setText(taskString);

      return todoView;
   }
}
```

…and return a LinearLayout view that has text views for task and date, suitably initialized.

MENUS

98

Example: “Game” Menu

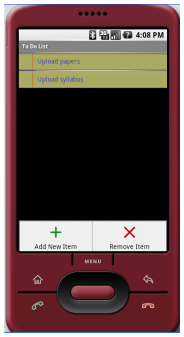
```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android=”...">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game"
          android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"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

Example: “File” Submenu

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android=”...">
    ...
    <item android:id="@+id/file"
          android:title="@string/file" >
        <!-- "file" submenu -->
        <menu>
            <item android:id="@+id/create_new"
                  android:title="@string/create_new" />
            <item android:id="@+id/open"
                  android:title="@string/open" />
        </menu>
    </item>
</menu>
```

Options Menu (Android 2.3-)

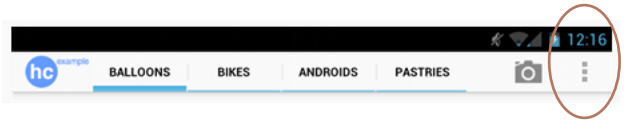
- Access via Menu button
  - *Deprecated*
- Icon menu
  - Up to 6 items
  - No check boxes or radio buttons
- Expanded menu
  - Pop-up menu of extra items
- Submenus



101

Options Menu: Android 3.0+

- Accessing menu:
  - Action bar
- Default:
  - All items under *action overflow*



- Promote menu items to action bar
  - `android:showAsAction="ifRoom"` in `<item>` element

102

Options Menu: Creation

- Inflating menu in activity:

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

  - Android 2.3-: Called when user first selects menu
  - Android 3.0+: Called when activity created
- Runtime addition: **`add()`**
- Changing menu items: **`onPrepareOptionsMenu()`**

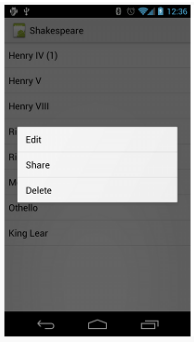
Options Menu: Handling Input

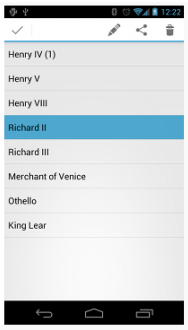
- Handling item selection:

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

Context Menu

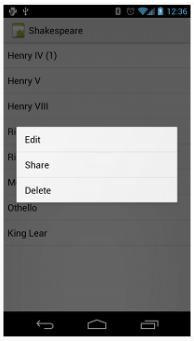
- Based on currently focused view
  - Like "right-click" (long click)
  - Typically for ListView or GridView item
- Floating Context Menu
  - Operate on one view at a time
- Contextual Action Bar
  - Operate on multiple views





Context Menu: Workflow

- Register context menu for view:
  `registerForContextMenu(View)`
  - ListView or GridView: menu for all items
- Define callback for menu creation:
  `onCreateContextMenu(ContextMenu, View, ContextMenuInfo)`
- Define response to user input:
  `onContextItemSelected(MenuItem)`



Context Menu: Creation

```
public class ToDoList extends Activity {

public void onCreate(Bundle icicle) {
   ...
   myListView = (ListView)findViewById(R.id.myListView);
   ...
   registerForContextMenu(myListView);
  }

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

107

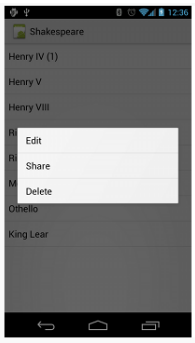
Context Menu: Selection

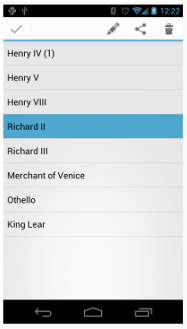
```
@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterContextMenuInfo info =
            (AdapterContextMenuInfo) item.getMenuInfo();
    switch (item.getItemId()) {
        case R.id.edit:
            editNote(info.id);
            return true;
        case R.id.delete:
            deleteNote(info.id);
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}
```

108

Contextual Action Mode

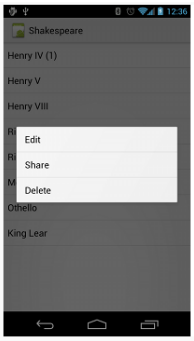
- Trigger:
  - Long click on view
  - User selects checkbox etc
- Contextual Action Bar
  - Visually overtakes action bar
- Exit CAM
  - BACK button
  - Select Done from action bar
  - Deselect all items





Contextual Action Mode: Single Item

- Implement `ActionMode.Callback`
  - `ActionMode` parameter
  - `setTitle()`, `setSubTitle()`, etc
- Call `startActionMode()` with callback
- Ex: in `View.onLongClickListener` handler



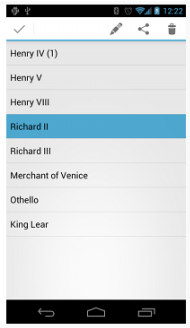
```
private ActionMode.Callback callback = new ActionMode.Callback() {

   public boolean onCreateActionMode(ActionMode mode, Menu menu) {
      MenuInflater inflater = mode.getMenuInflater();
      inflater.inflate(R.menu.context_menu, menu);
      return true;
   }

   public boolean onPrepareActionMode(ActionMode m, Menu u)
   { return false; }

   public boolean onActionItemClicked(ActionMode m, MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_foo:
                ... m.finish(); return true;
            default:
                return false;
        }
   }

   public void onDestroyActionMode(ActionMode mode)
   { actionMode = null; /* "actionMode" defined in parent class */ }
};
```

Contextual Action Mode: Single Item

```
View.OnLongClickListener longListener =
   new View.OnLongClickListener() {

      // Called when the user long-clicks on someView
      public boolean onLongClick(View view) {
         if (actionMode != null) {
            return false;
         }

         // Start the CAB using the callback defined above
         mActionMode =
            getActivity().startActionMode(callback);
         view.setSelected(true);
         return true;
      }
   };

view.setOnLongClickListener(longListener);
```

Contextual Action Mode: Multi Item

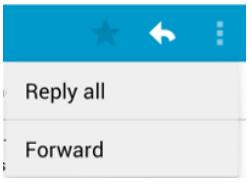
- Implement `AbsListView.MultiChoiceItemListener`
- Call `setChoiceMode()` with `CHOICE_MODE_MULTIPLE_MODAL`
- Call `setMultiChoiceModeListener()` with callback
- Call `setItemChecked()` to add item to selection




```
listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE_MODAL);
listView.setMultiChoiceModeListener(new MultiChoiceModeListener() {

   public void onItemCheckedStateChanged(ActionMode m, int pos,
                                         long id, boolean checked) {
      // Do something when items are selected/de-selected,
   }
   public boolean onActionItemClicked(ActionMode m, MenuItem item) {
      // See Single Item example
   }
   public boolean onCreateActionMode(ActionMode mode, Menu menu) {
      // See Single Item example
   }
   public void onDestroyActionMode(ActionMode mode) {
      // When CAB is removed. Default: items are deselected.
   }
   public boolean onPrepareActionMode(ActionMode m, Menu u)
   { return false; }
})
```

Popup Menu

- Modal menu anchored to a view
  - Overflow-style menu
  - Options for menu command
  - Drop-down (similar to Spinner)



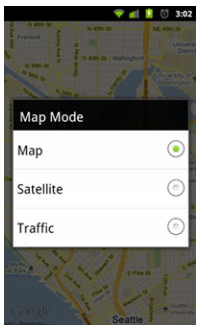
- Instantiate PopupMenu class
- Inflate with MenuInflater or PopupMenu.inflate()
- Show with PopupMenu.show()
- Input: PopupMenu.OnMenuItemClickListener
- Dismissal: PopupMenu.OnDismissListener

Popup Menu

```
public void showPopup(View v) {
   PopupMenu popup = new PopupMenu(this, v);

   // This activity implements OnMenuItemClickListener
   popup.setOnMenuItemClickListener(this);

   MenuInflater inflater = popup.getMenuInflater();
   inflater.inflate(R.menu.actions, popup.getMenu());
   // Android v11+
   // popup.inflate(R.menu.actions);

   popup.show();
}
```

Menu Groups and Checkable Menu Items

```
<menu xmlns:android=”...">
   <group android:checkableBehavior="single">
      <item android:id="@+id/red"
            android:title="@string/red" />
      <item android:id="@+id/blue"
            android:title="@string/blue" />
   </group>
</menu>
```



Checkable Behavior:

- `single`: radio button
- `all`: check box
- `none`

Checkable Menu Items

- Respond to selection of menu item:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.vibrate:
        case R.id.dont_vibrate:
            if (item.isChecked())
               item.setChecked(false);
            else
               item.setChecked(true);
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Intent Filters for Plugins (1)

- Idea: dynamically generate menus that describe actions that can be applied to data displayed on the screen
- Range of options can vary dynamically depending on intent filters of installed application base

Intent Filters for Plugins (2)

- Idea: dynamically generate menus that describe actions that can be applied to data displayed on the screen
- Range of options can vary dynamically depending on intent filters of installed application base
- Use `addIntentOptions` method of a menu object to add menu options at run-time
  - Categories ALTERNATIVE and SELECTED_ALTERNATIVE identify activities that can be presented to users in a menu of options

```
public boolean onCreateOptionsMenu(Menu menu){
    super.onCreateOptionsMenu(menu);

   // The offering app must include a category of
   // Intent.CATEGORY_ALTERNATIVE.
   Intent intent = new Intent(null, dataUri);
   intent.addCategory(Intent.CATEGORY_ALTERNATIVE);

   // Search and populate the menu with acceptable offering apps.
   menu.addIntentOptions(
       R.id.intent_group,  // Menu group for new items
       0,      // Unique item ID (none)
       0,      // Order for the items (none)
       this.getComponentName(),   // The current activity name
       null,   // Specific items to place first (none)
       intent, 0, null // Intent, flag, MenuItems (array)
   );
   return true;
}
```

Intent Filters for Plugins (3)

Intent Filter for TitleEditor activity (from Notepad example):

```
<intent-filter android:label="@string/resolve_title">
  <action android:name="com.android.notepad.action.EDIT_TITLE" />
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.ALTERNATIVE" />
  <category android:name="android.intent.category.SELECTED_ALTERNATIVE
  <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
</intent-filter>
```

This intent resolves to TitleEditor:

```
action: com.android.notepad.action.EDIT_TITLE
data: content://com.google.provider.NotePad/notes/ID
```

Asks the activity to display the title associated with note *ID*,
and allow the user to edit the title.

Dialogs

- Common UI Metaphor
- Three ways to implement:
  - Dialog-themed Activities
  - Derive from Dialog class
    - Example: AlertDialog
    - Constructed entirely within calling activity
    - No need to register in the manifest
  - Toasts
    - Non-modal transient message boxes

123

SOCKET PROGRAMMING

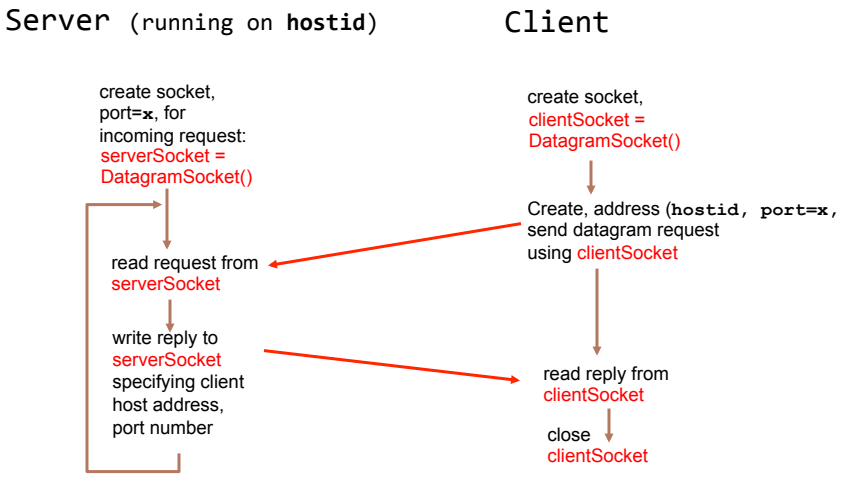
124

Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.stevens.cs522.chat.oneway"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WIFI" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".ChatServer"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

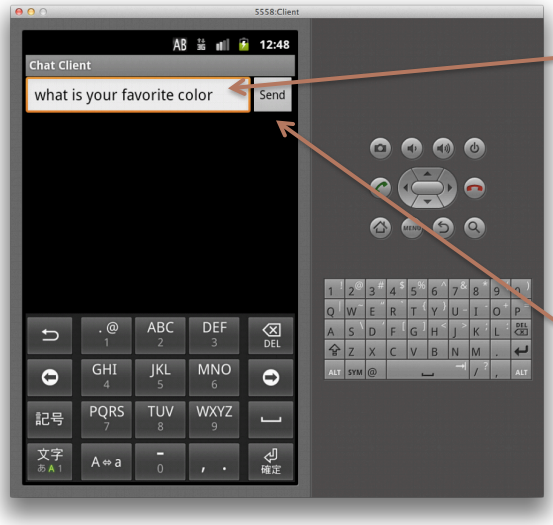
125

Client/server socket interaction: UDP



126

Socket Client

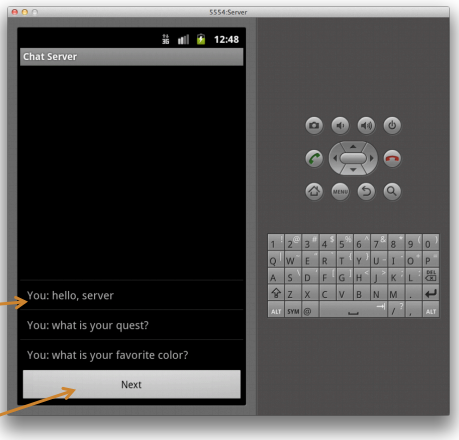


Text input from the user

Sends the input from the user to the UDP Server

Socket Server

- Create an application that acts as a UDP server to receive messages from the UDP client



List of messages received so far

Press to get the next message

128

Handling Exceptions

```
try {
  // Operations that may fail
  // and throw an exception;
} catch (Exception e) {
    Log.e(TAG,
     "Caught UDP Exception: "+e.getMessage());
    Toast.makeText(UDPServer.this,
     "UDP Error: "+ e.getMessage()),
     Toast.LENGTH_LONG).show();
}
```