Android Application Design

Software System Design

Zhu Hongjun

Session 3: Prototype Design

- Prototype Design Process
- Tools
- UI Design Patterns
- Android UI Components
- Conclusions

Prototype Design Process

- A prototype is an early sample, model, or release of a product built to test a concept or process or to act as a thing to be replicated or learned from
- A prototype is designed to test and trial a new design to enhance precision by system analysts and users
  - http://en.wikipedia.org/wiki/Prototype

Prototype Design Process

- A prototype typically simulates only a few aspects of, and may be completely different from, the final product
- Software prototyping
  - the activity of creating prototypes of software applications, i.e., incomplete versions of the software program being developed

Prototype Design Process

- Benefits
  - get valuable feedback from the users early
  - compare if the software made matches the software specification
  - insight into the accuracy of initial project estimates and whether the deadlines and milestones proposed can be successfully met

Prototype Design Process

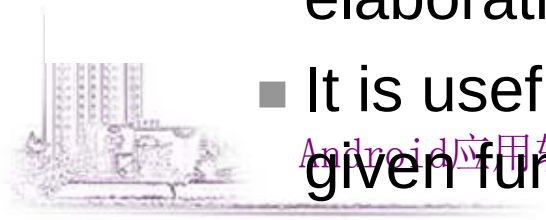
- Disadvantages
  - Insufficient analysis
  - User confusion of prototype and finished system
  - Developer misunderstanding of user objectives
  - Developer attachment to prototype
  - Excessive development time of prototype
  - Expense of implementing prototyping

Prototype Design Process

- Prototyping process
  - Indentify basic requirements
  - Develop initial prototype
    - the initial prototype is developed that includes only user interfaces
  - Review
    - provide feedback on additions or changes
  - Revise and enhance prototype

Prototype Design Process

- Dimensions of prototype
  - Horizontal prototype
    - A common term for a user interface prototype is the **horizontal prototype**
    - It provides a broad view of an entire system or subsystem
  - Vertical prototype
    - A **vertical prototype** is a more complete elaboration of a single subsystem or function
    - It is useful for obtaining detailed requirements for a given function

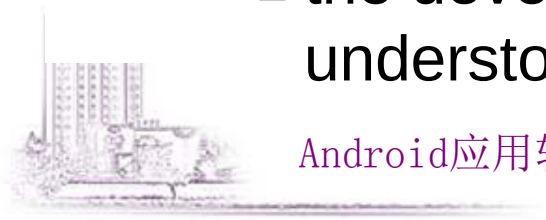


Prototype Design Process

- Types of prototyping
  - Throwaway prototyping
    - also called close-ended prototyping
    - in this approach the prototype is constructed with the idea that it will be discarded
      - Write preliminary requirements
      - Design the prototype
      - User experiences/uses the prototype, specifies new requirements
      - Repeat if necessary
      - Write the final requirements

Prototype Design Process

- Types of prototyping (cont.)
  - Evolutionary prototyping
    - also known as breadboard prototyping
    - evolutionary prototype, when built, forms the heart of the new system
    - when developing a system using Evolutionary Prototyping, the system is continually refined and rebuilt
    - the developer does not implement poorly understood features



Prototype Design Process

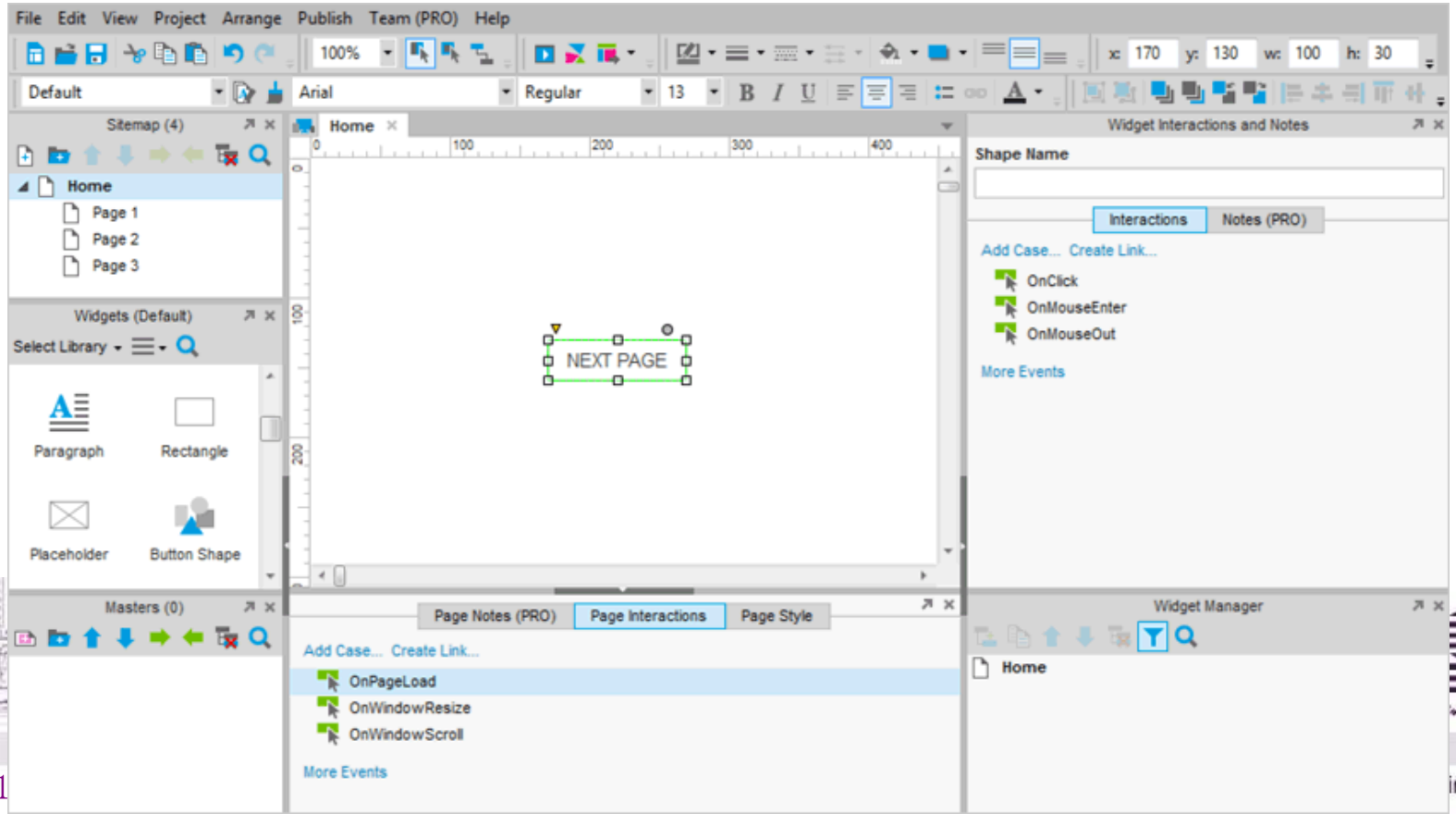
- Types of prototyping (cont.)
  - Incremental prototyping
    - the final product is built as separate prototypes
    - at the end the separate prototypes are merged in an overall design
  - Extreme prototyping
    - it breaks down web development into three phases, each one based on the preceding one
      - static prototype phase
      - UI programmed phase
      - Service implemented phase

Tools

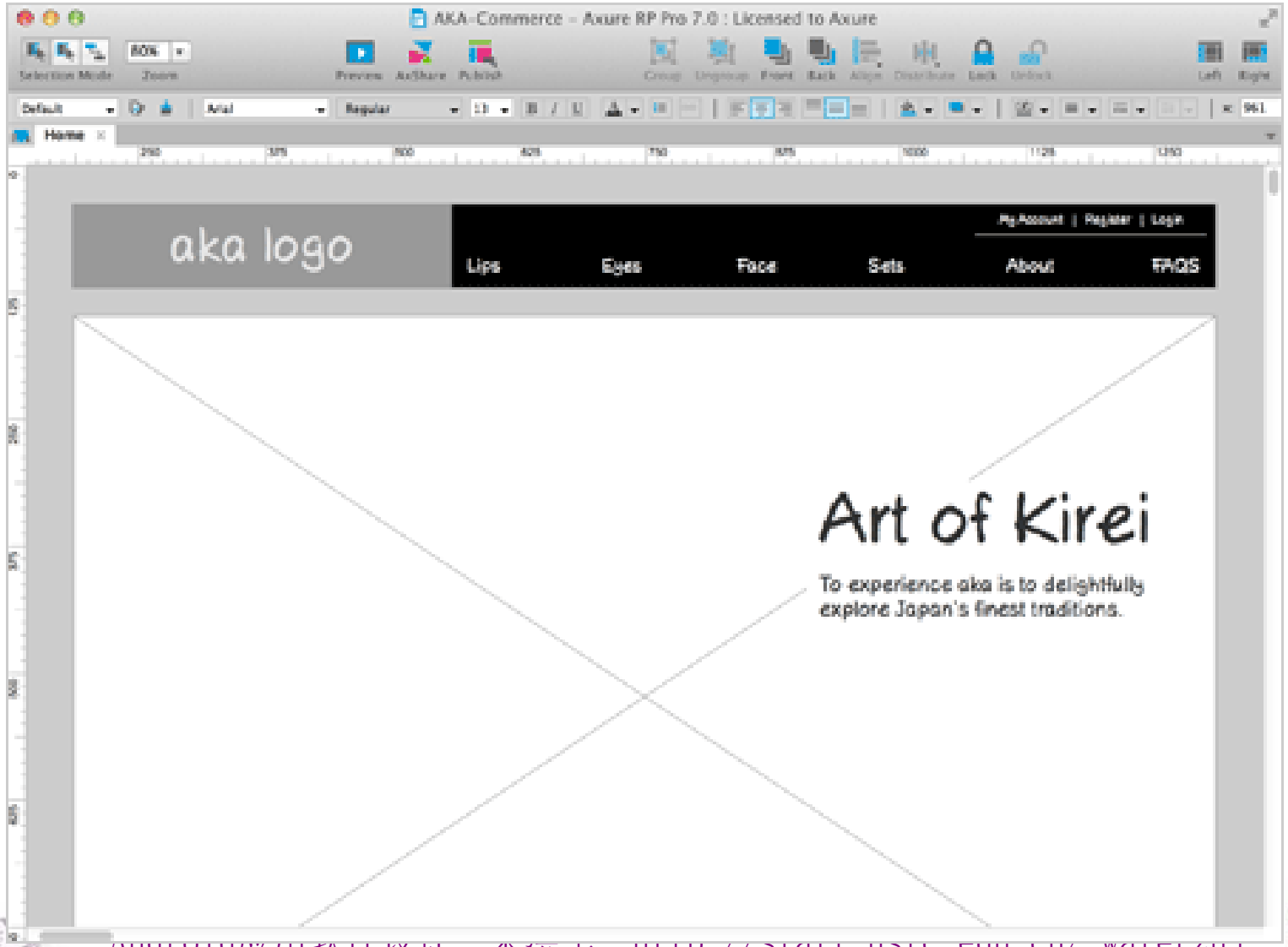
- Programming languages
  - VB, LabView, FoxPro, etc.
- CASE tools
  - Axure, Balsamiq Mockups, etc.
- Object-Oriented tools
  - LYMB, etc.
- Spreadsheet

Tools

- Axure RP
  - It is a commercial product

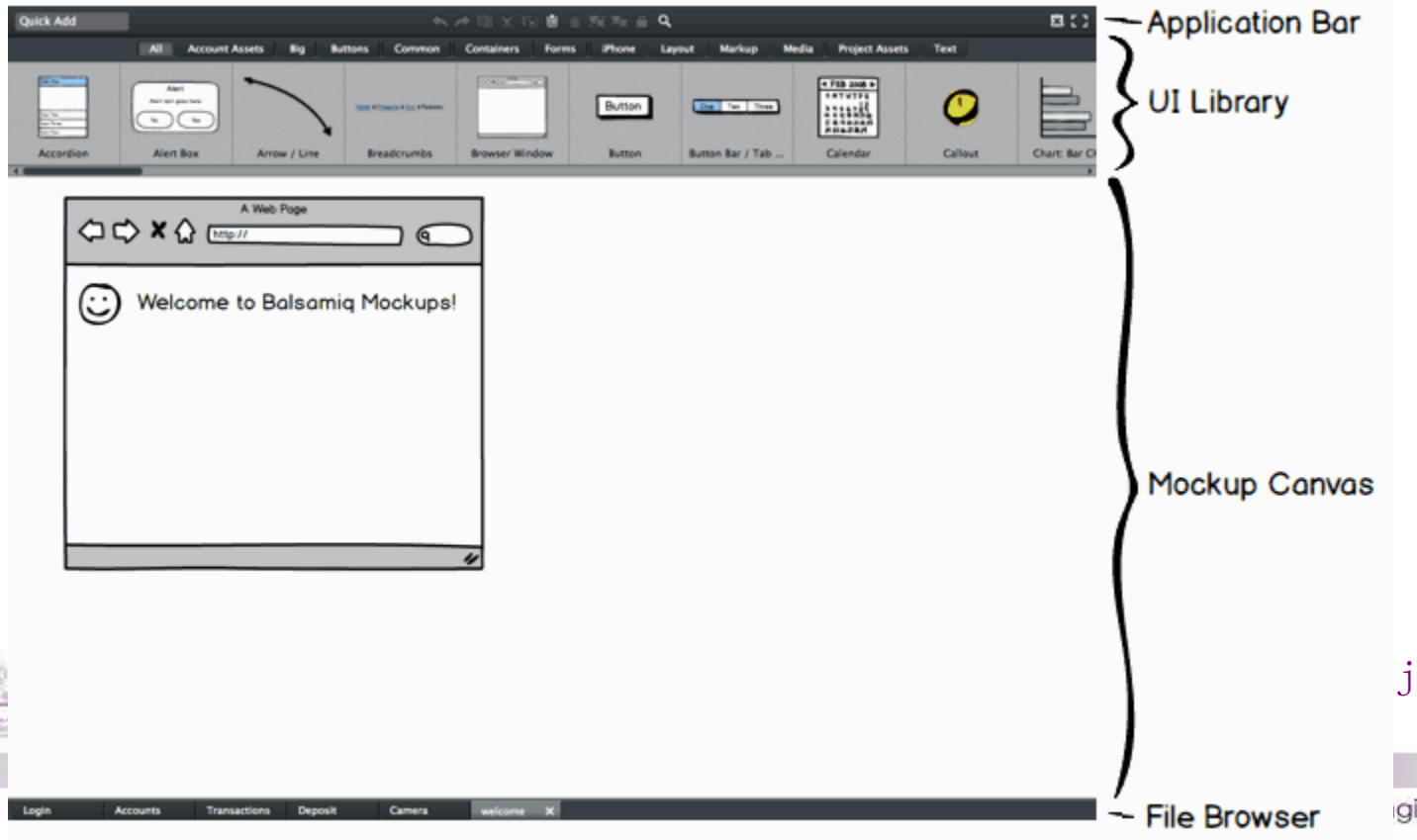


Prototype demo built by axure



Tools

- Balsamiq mockups
  - Another commercial product



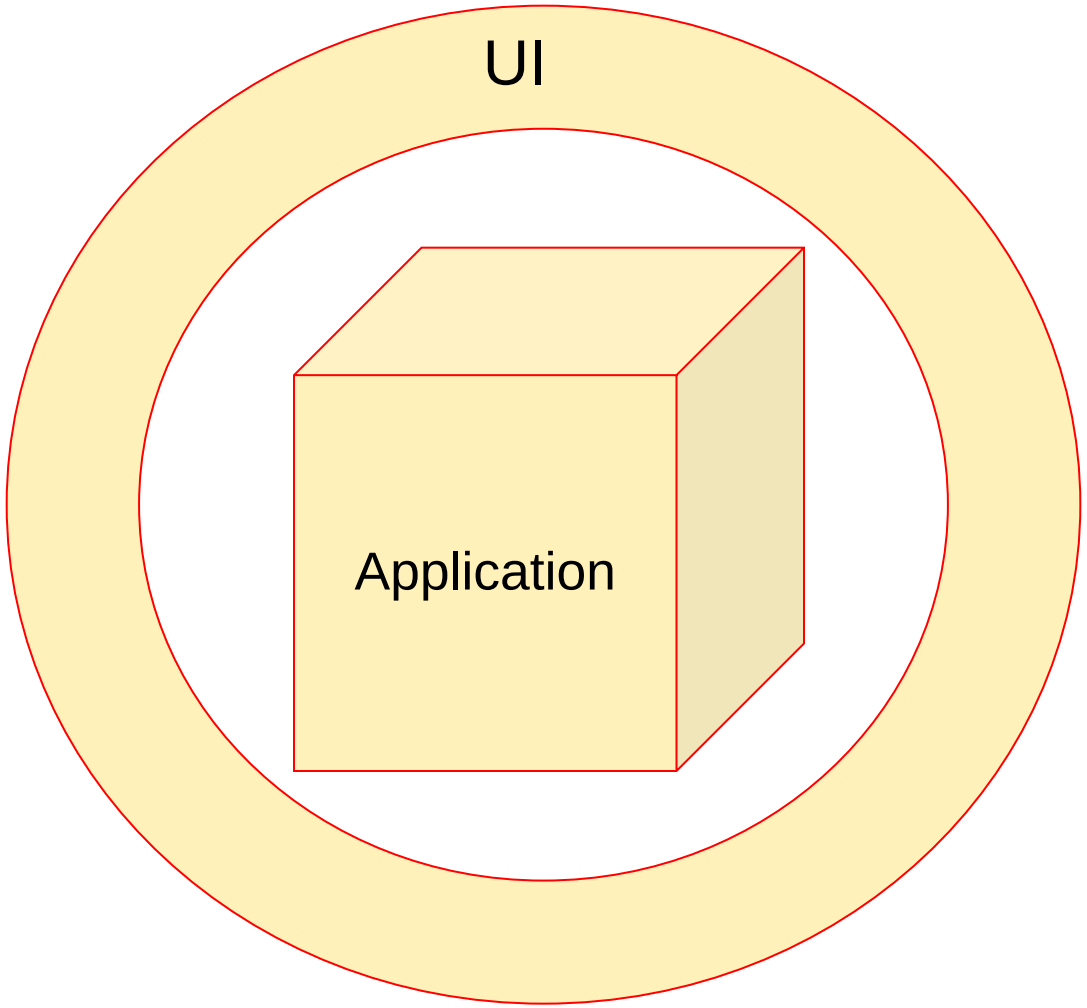
Prototype demo built by balsamiq

Prototype demo built by balsamiq

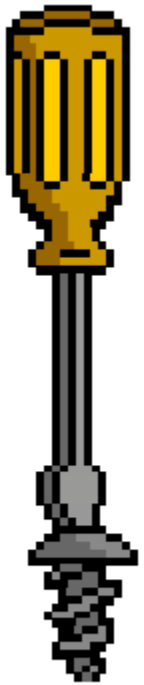
UI Design Patterns

- User interface
  - The part of an application (or tool) that is in contact with the user(s)
  - The complete experience that is detectable by the user while using the application (or tool)
  - The user's interactions with the application
  - Any interaction (direct or indirect) that a user has with the application

User，UI，Application



UI demos



UI

Application

Application

UI

UI

User

UI Design Patterns

- User mental model
  - How the user thinks the application will respond when manipulated through the UI
  - A mental model is based on belief, not facts
  - Individual users each have their own mental model
  - Mental models are influx exactly rather than fixed in an external medium

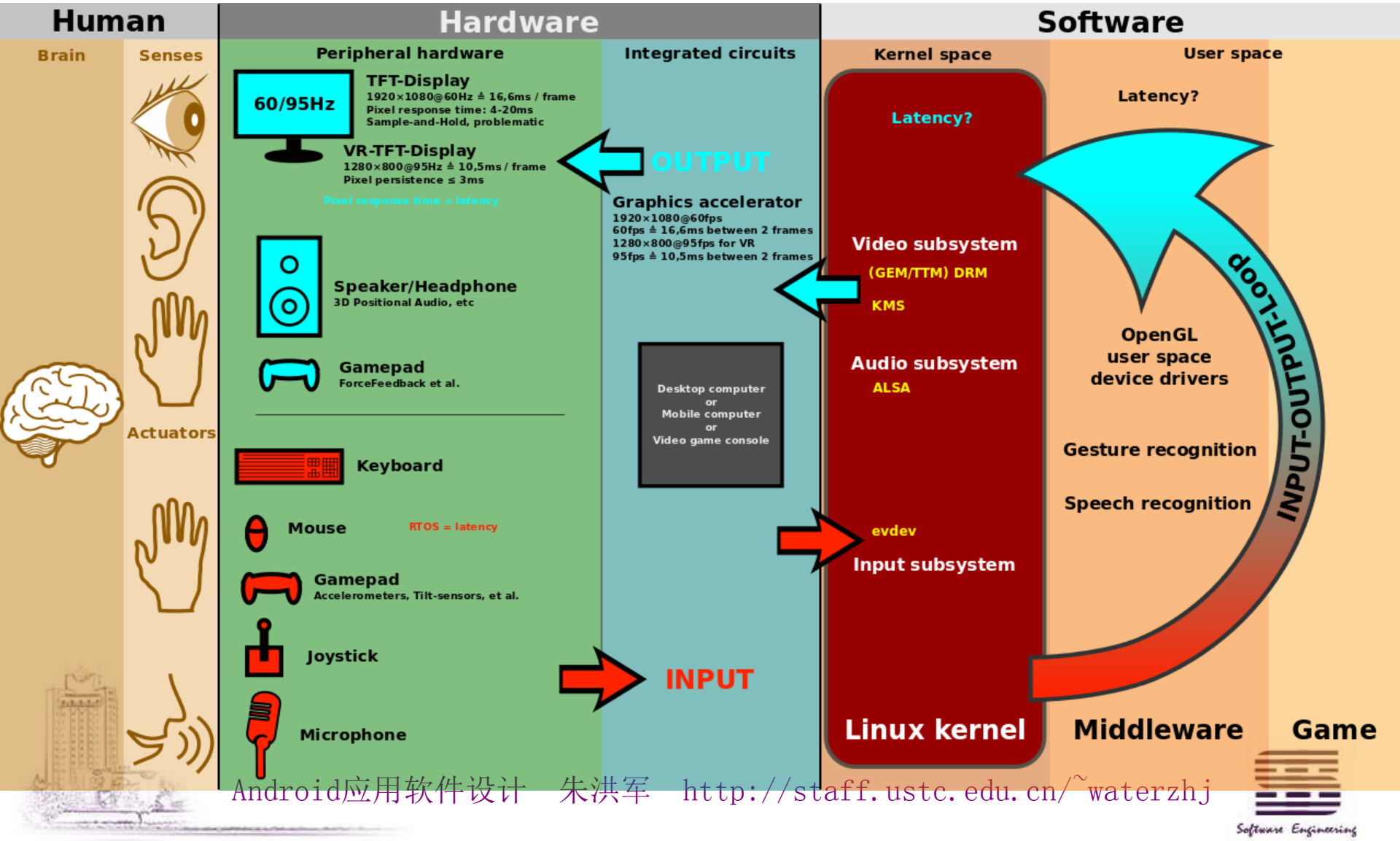
UI Design Patterns

- User mental model (cont.)
  - characteristics
    - founded on unquantifiable, impugnable, obscure, or incomplete facts
    - flexible
    - an information filter
    - limited
    - dependent on sources of information

UI Design Patterns

- Application models
  - user mental model
  - implementation model
    - The underlying code and how the application really works
    - May not completely agree with the users mental model

Input-Output loop between human and machine



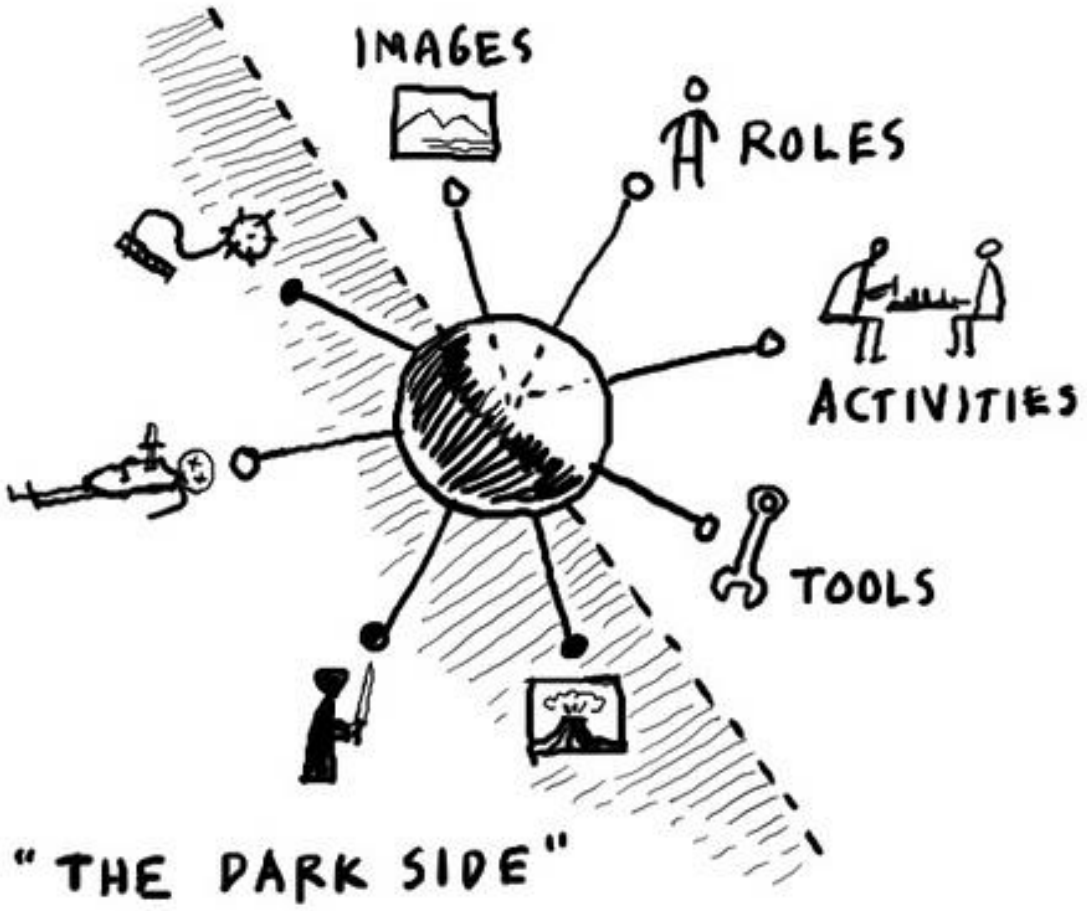
UI Design Patterns

- UI features
  - Input
    - device
    - language
    - conversation from user to ui
  - Output
    - device
    - language
    - conversation from ui to user

UI Design Patterns

- UI features (cont.)
  - user experience
    - Information Architecture
      - Mapping from the UI to the functionality and information provided by the application logic
    - Possibly idioms and/or metaphors
    - Feedback mechanisms
    - Navigation structure
    - Labels
    - Menus
    - Content and/or functionality

METAPHOR

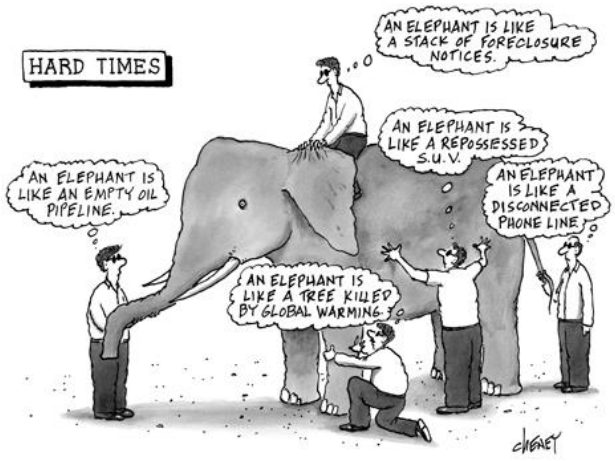


UI Design Patterns

- UI design
  - Creating a User Interface that matches the users needs by providing a UI that allows the user to achieve their goals as efficiently as possible
    - Intuitiveness / predictability
    - Consistency
    - Ease of learning
    - Obvious
    - Aesthetically pleasing
    - Support the appropriate level of user and their specific needs expert vs. novice, occasional vs. frequent users
    - Power
    - Safety

UI Design Patterns

- Principles of UI design
  - structure principle
  - simplicity principle
  - visibility principle
  - feedback principle
  - tolerance principle
  - reuse principle

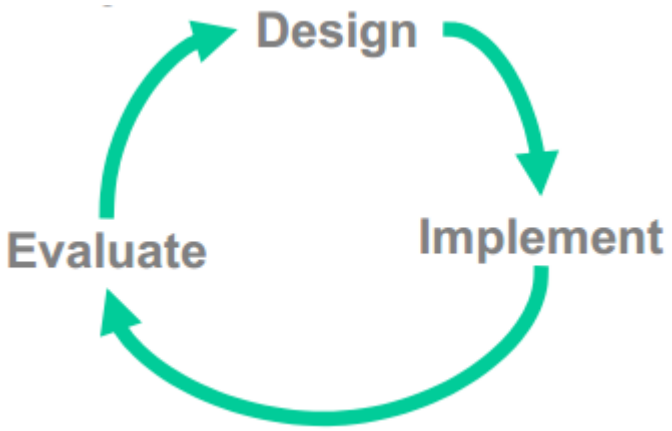


UI Design Patterns

- Two laws
  - A system shall not harm your work or, through inactivity, allow your work to come to harm
  - A system shall not waste your time or require you to do more work than is strictly necessary

UI Design Patterns

- UI design process
  - Functionality requirements gathering
  - User and task analysis
  - Information architecture
  - Prototyping
  - Usability inspection
  - Usability testing
  - Graphic interface design

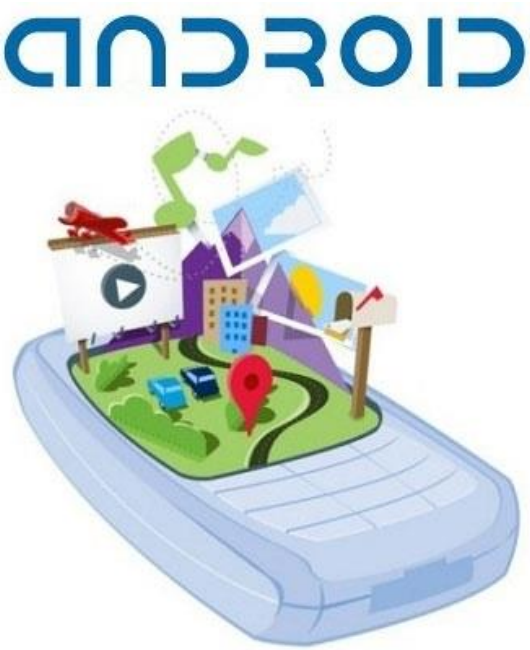


UI Design Patterns

- Mobile design
  - Observe Human-Mobile Interaction in the Real World
  - Your Prototyping Methods Must Allow for Variety in Form Factors
  - Your User Testing Must Allow People to Explore the Natural Range of Motion, Voice, and Multitouch
  - Touch Interfaces Embody Simplicity and Sophistication
  - Delight Is Mandatory
  - Tell a Complete Story—Design for Cross-Channel Experiences

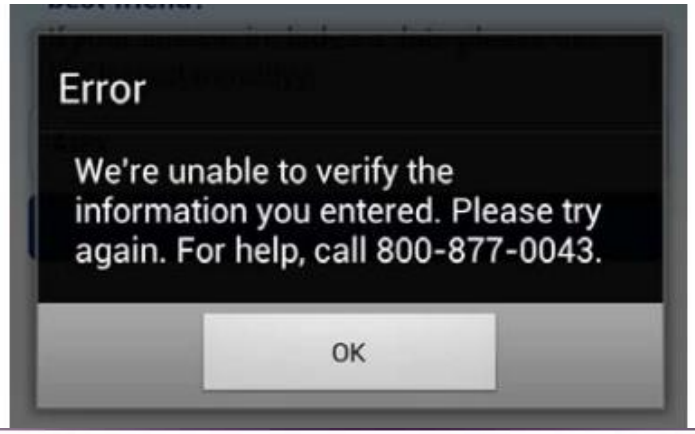
UI Design Patterns

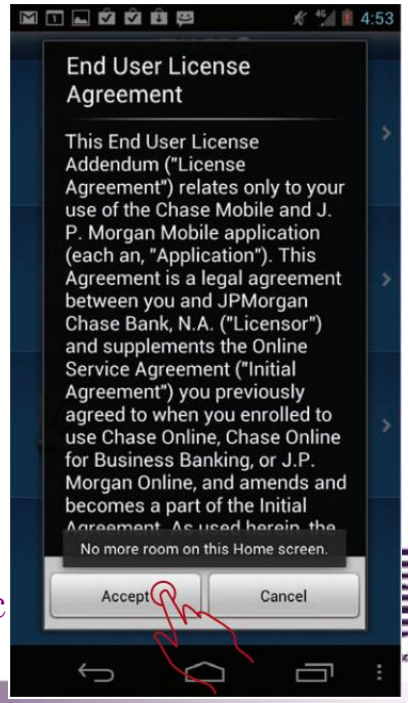
- Android features
  - Welcome to flatland
  - Tap anywhere
  - Right-size for every device
  - Mobile space, unbound
  - Think globally, act locally



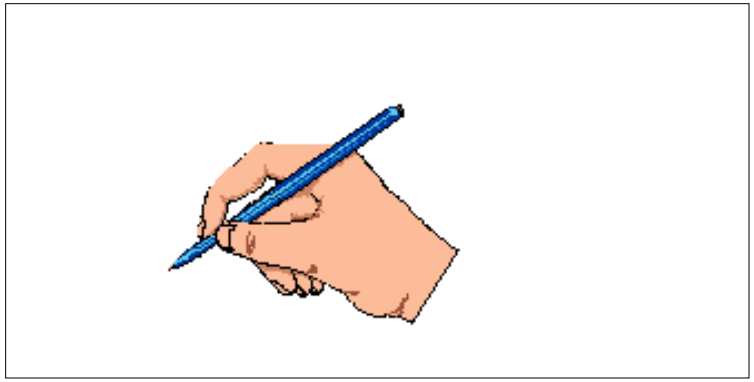
UI Design Patterns

- Android design patterns and antipatterns
  - Welcome experience
    - Antipattern: end user license agreements
    - Antipattern: sign up/sign in
    - Antipattern: contact us impediments
    - Pattern: welcome animation



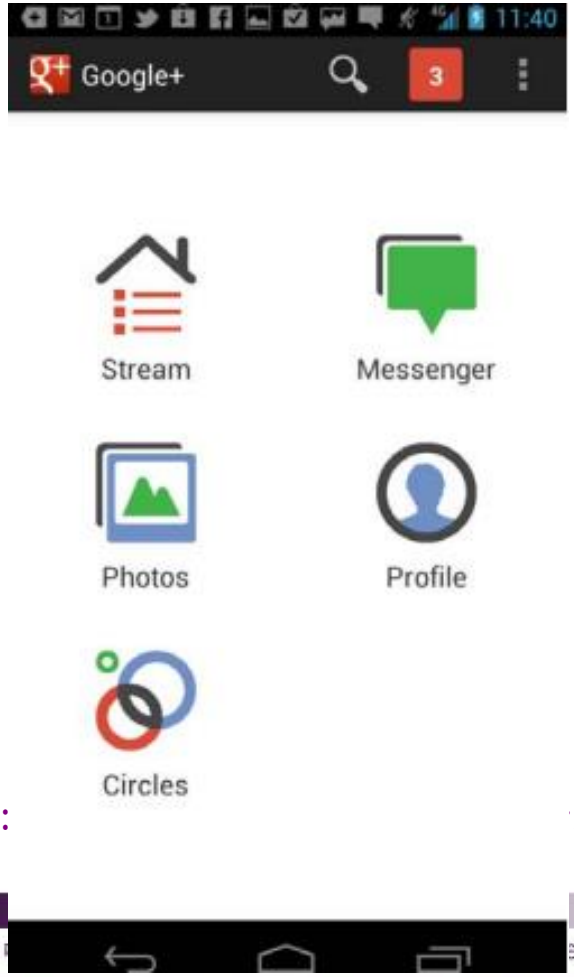


Welcome animation



UI Design Patterns

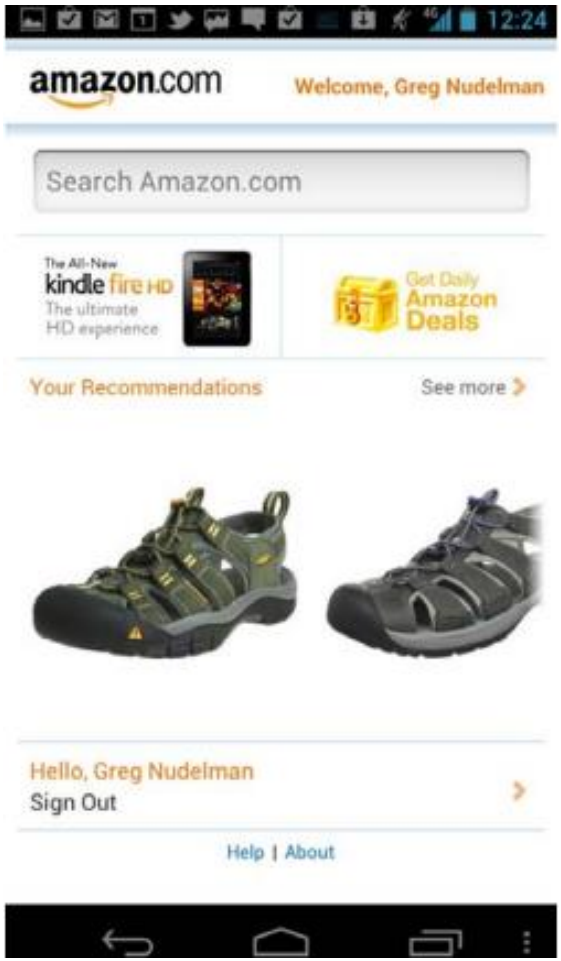
- Android design patterns and antipatterns (cont.)
  - Home screen
    - Pattern: list of links
    - Pattern: dashboard
    - Pattern: updates
    - Pattern: browse
    - Pattern: map
    - Pattern: history



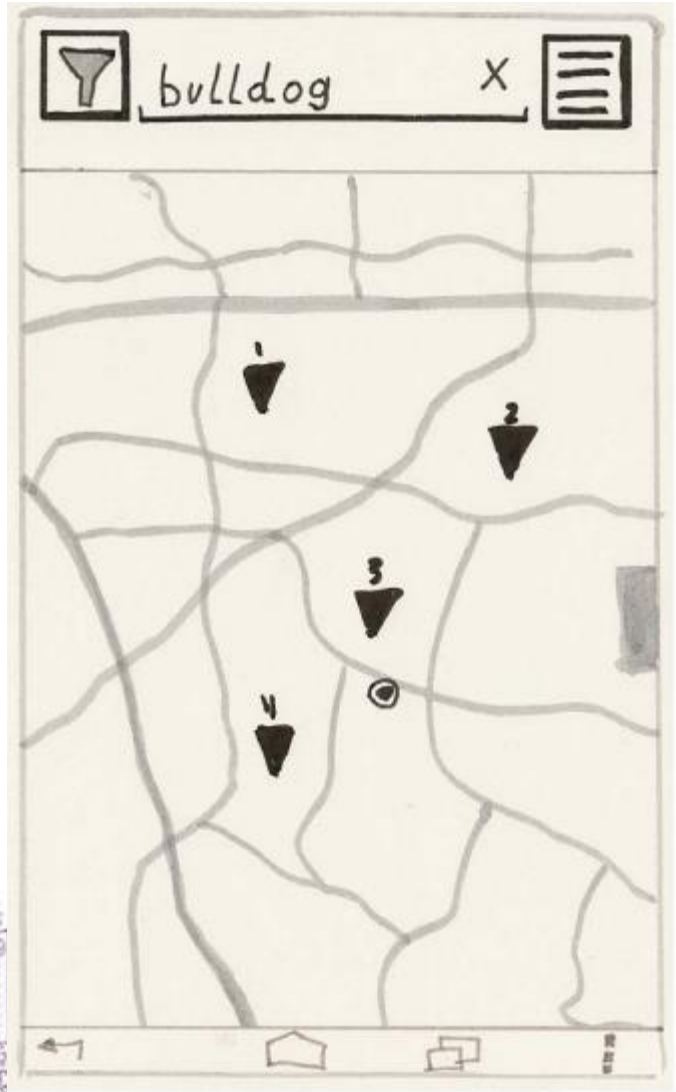
Home screen


1:58
mint.com
Overview
ACCOUNTS
$52,369 Cash
-$13,082 Credit Debt
BUDGET JULY
TODAY
CASH FLOW JULY
-$3,351
$12,751 earned
$16,102 spent
ALERTS
9
Overbudget
ADVICE
You're paying 20% or more for ...
Updated 07/27/2011 at 1:58 PM


12:40
Status
Photo
Check In
If you're in NY, you should definitely check it out... open until 10/28/12
http://whitney.org/Exhibitions/FirefliesOnTheWater — at Whitney Museum of American Art.


12:24
amazon.com
Welcome, Greg Nudelman
Search Amazon.com
The All-New kindle fire HD
The ultimate HD experience
Get Daily Amazon Deals
Your Recommendations
See more
Hello, Greg Nudelman
Sign Out
Help | About

Home screen

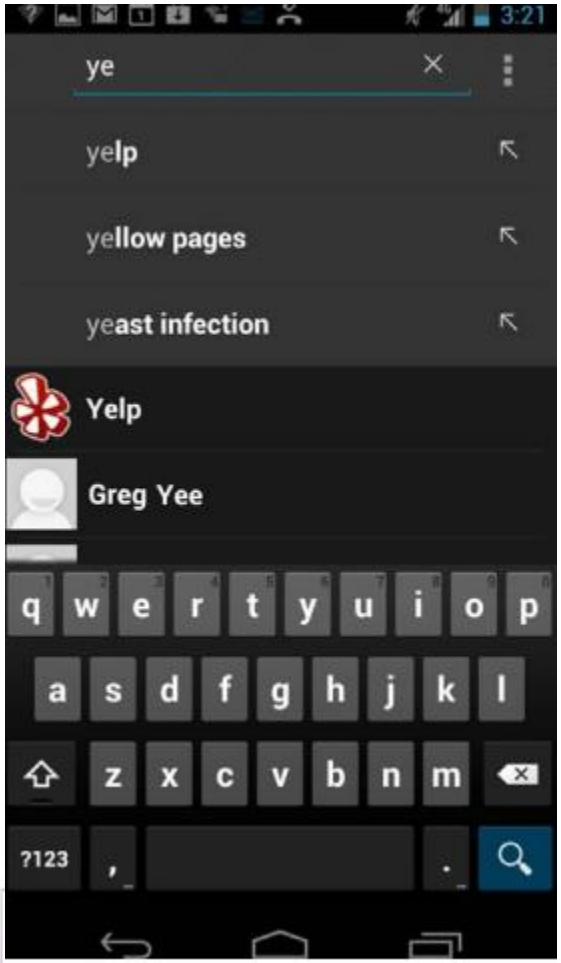


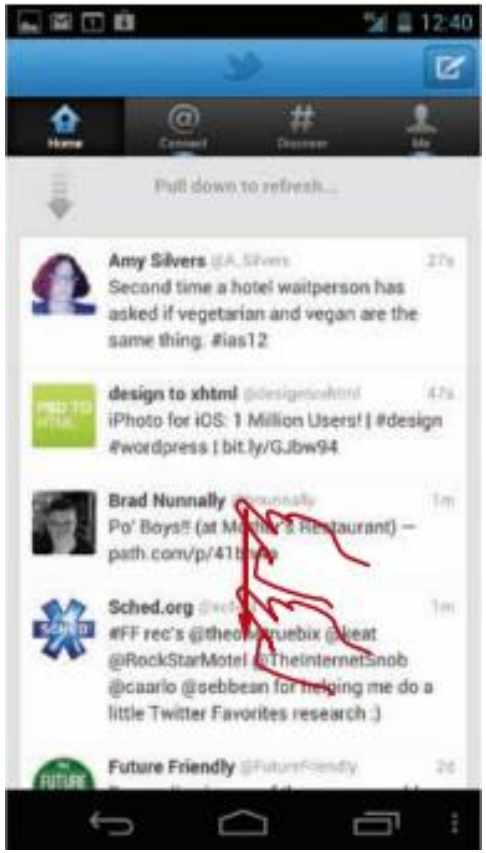
洪军 http:/

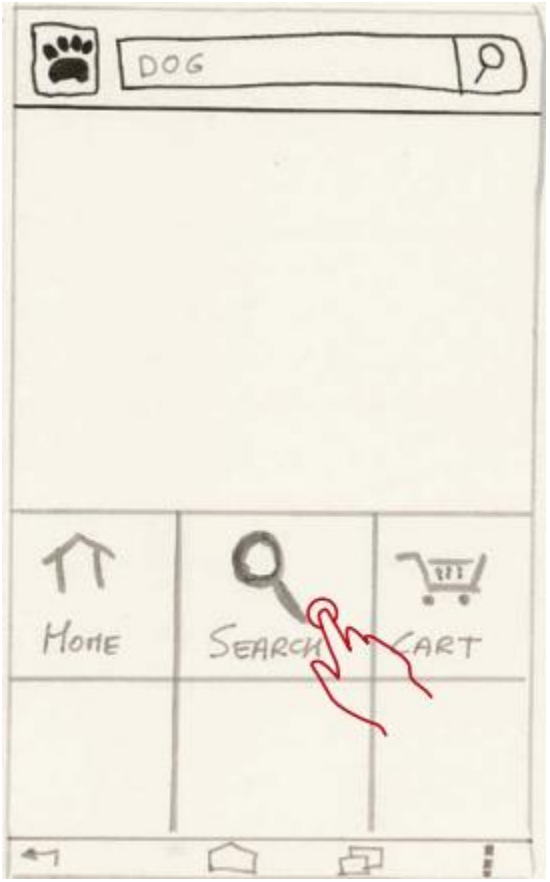
UI Design Patterns

- Android design patterns and antipatterns (cont.)
  - Search
    - Pattern: voice search
    - Pattern: Auto-complete and Auto-suggest
    - Pattern: pull to refresh
    - Pattern: search from menu
    - Pattern: search from Action bar
    - Antipattern: separate search and refinement

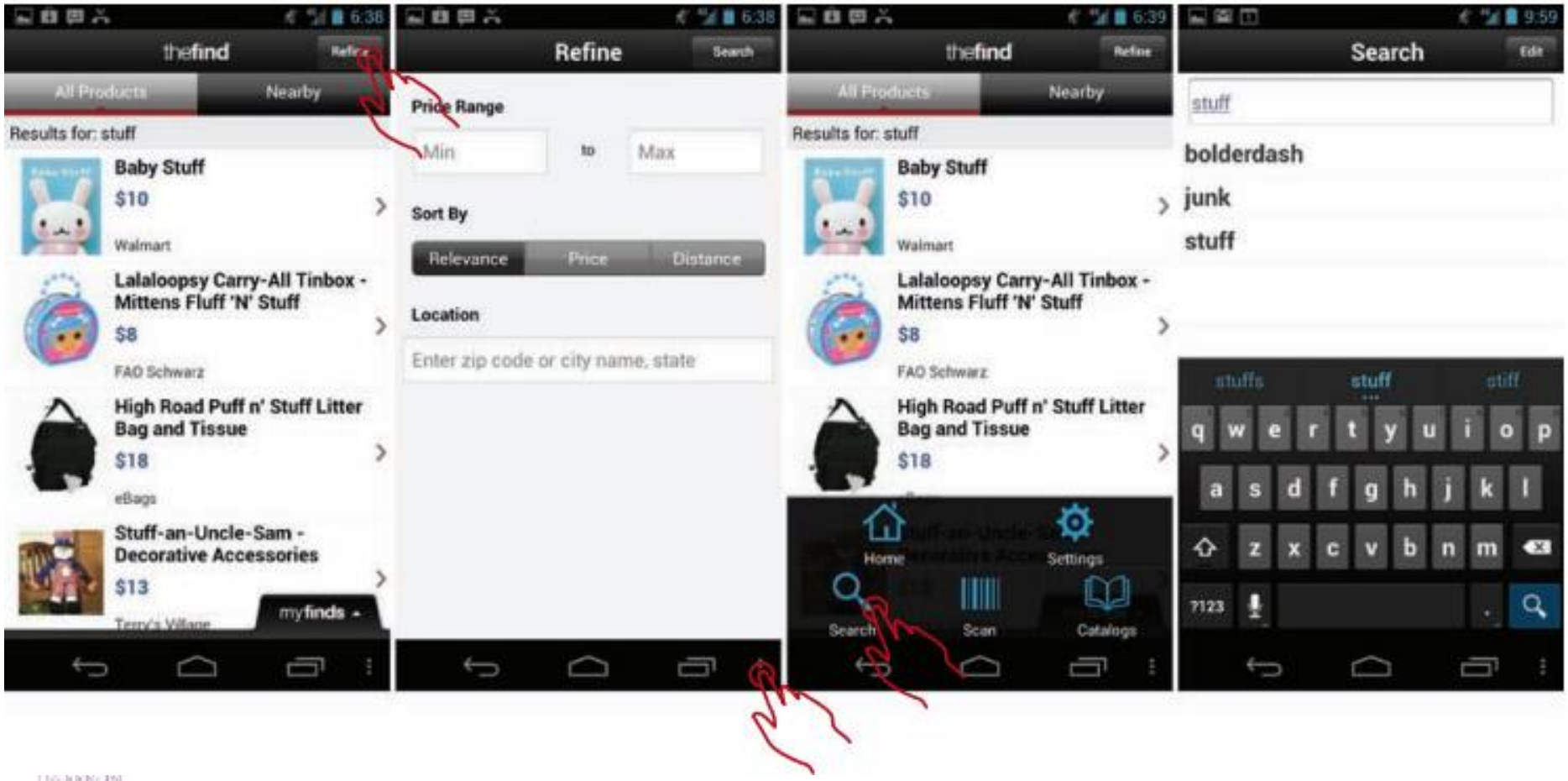
Search


3:21
ye
yelp
yellow pages
yeast infection
Yelp
Greg Yee


12:40
Pull down to refresh...
Amy Silvers
Second time a hotel waitperson has asked if vegetarian and vegan are the same thing. #ias12
design to xhtml
iPhoto for iOS: 1 Million Users! | #design #wordpress | bit.ly/GJbw94
Brad Nunnally
Sched.org
Future Friendly

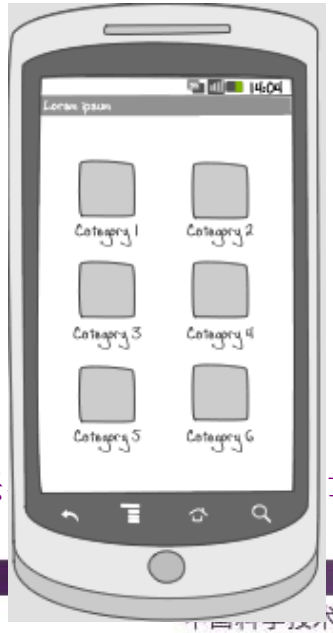

DOG
HOME
SEARCH
CART

Antipattern search

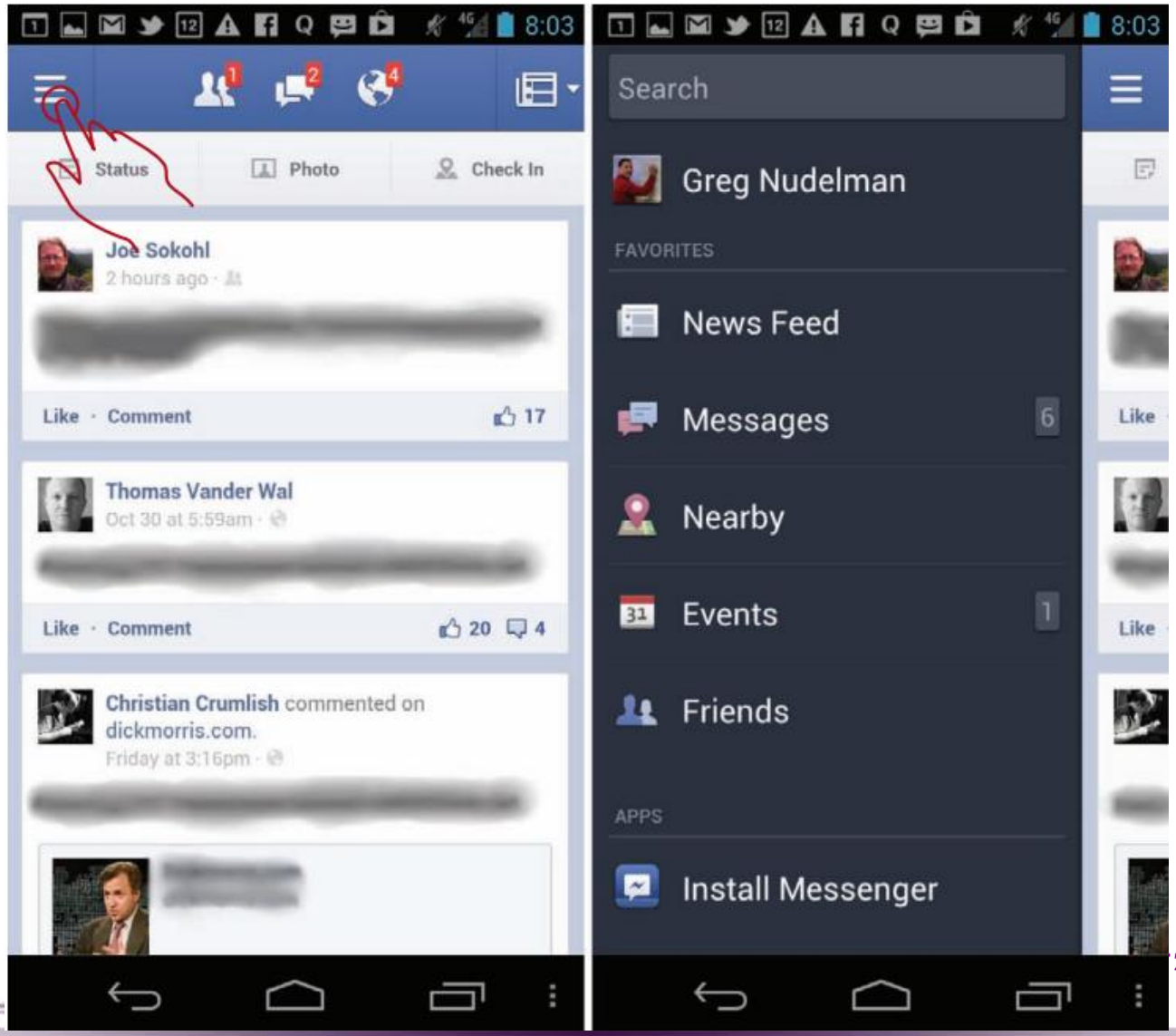


UI Design Patterns

- Android design patterns and antipatterns (cont.)
  - Navigation
    - Pattern: action bar
    - Pattern: launch board
    - Pattern: swiss-Army-knife
    - Pattern: drill-down

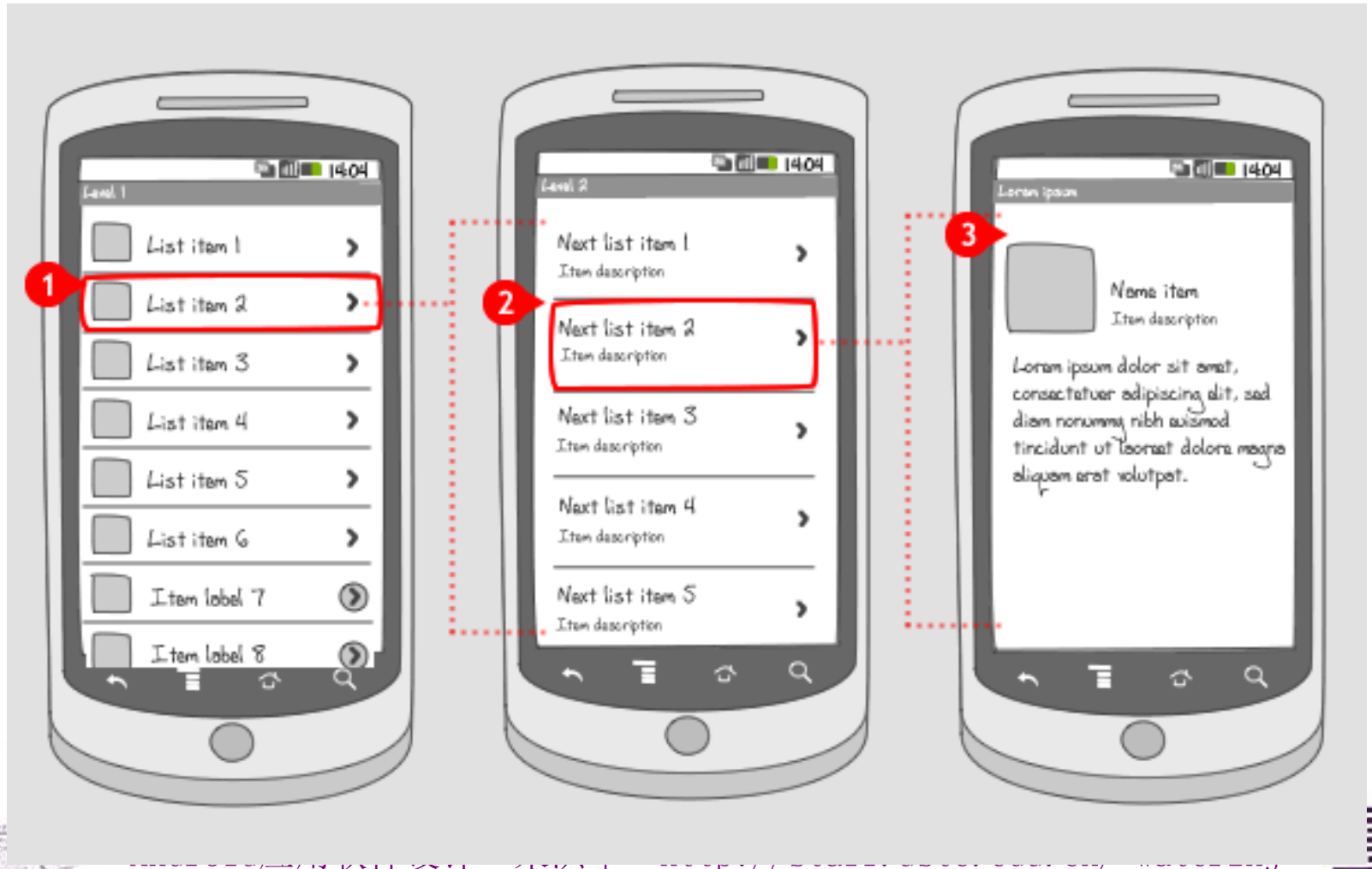


Swiss-Army-knife navigation



zhj

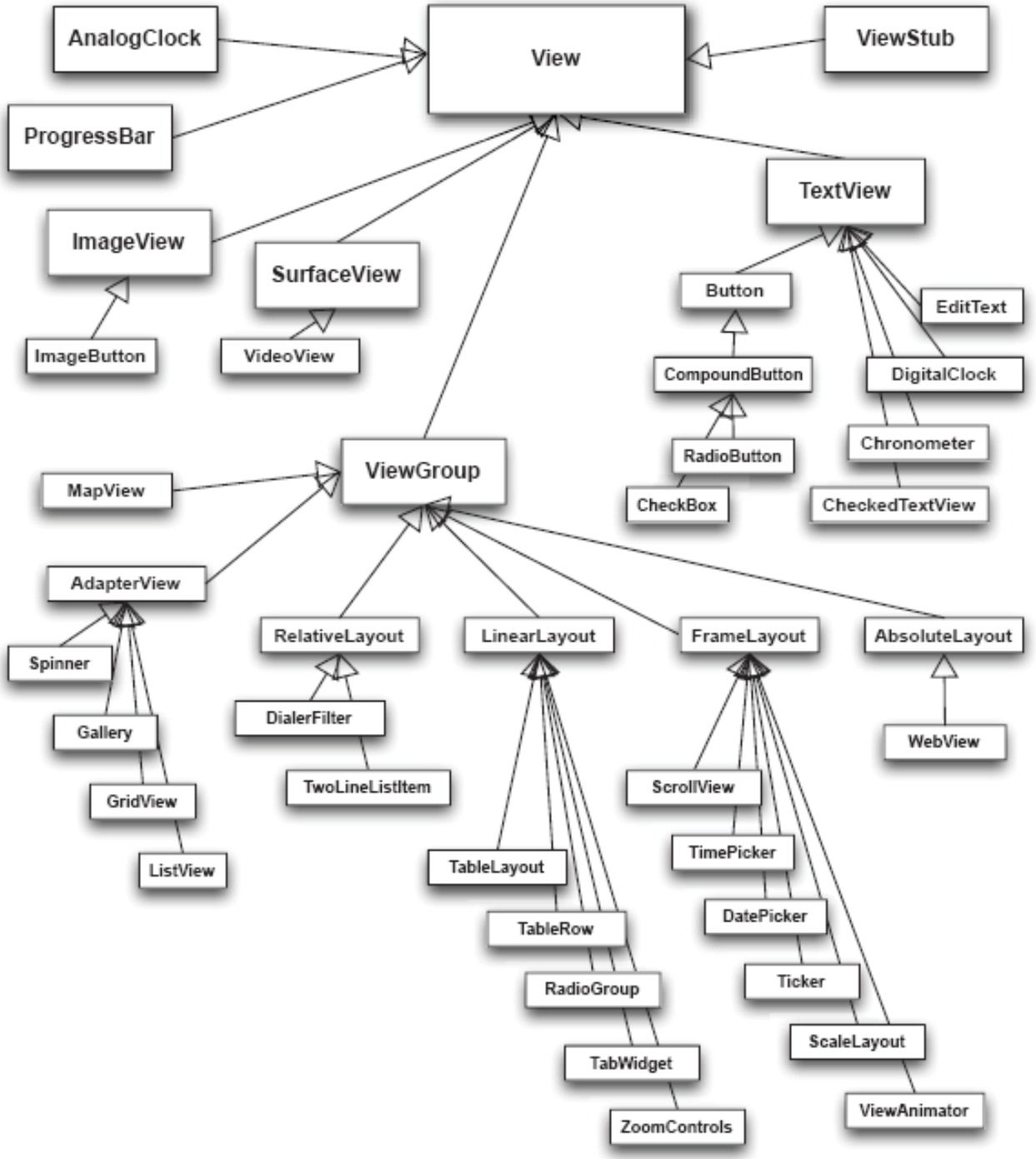
Drill-down navigation



Android UI Components

- Android UI Framework
  - Common view components
    - Button/EditText, etc.
  - Menu
    - Options menu/Context menu/Submenu/Action Bar/Tool Bar/App Bar
  - ListView
  - TabWidget
  - Dialog
    - AlertDialog/ProgressDialog
  - Custom UI component
  - Notify user
    - Toast
    - status bar notification

The class diagram of Android view API



Android UI Components

- Activity and intent
  - Creating an Activity
    - Extending Activity class
    - Declaring it in the manifest file (AndroidManifest.xml) for your project

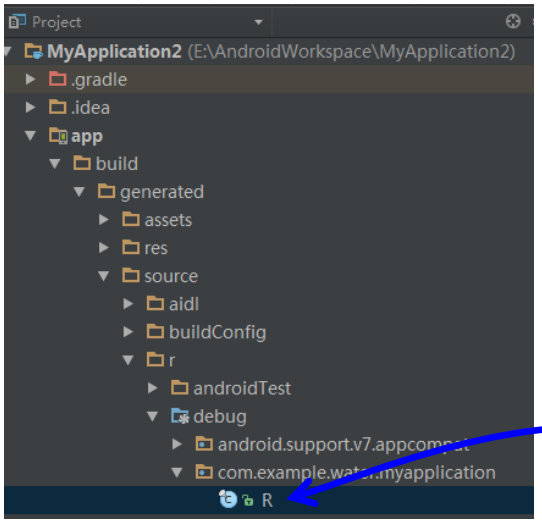
```
public class FirstAndroidActivity extends Activity
```

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".FirstAndroidActivity"
              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>
```

Android UI Components

- Activity and intent (cont.)
  - Creating an Activity (cont.)
    - Two important methods
      - onCreate()
        - Calling this method, when the system creates Activity
      - onPause()
        - When user leaves the current Activity, system calls this method firstly
        - If you wanna store states of application, you should do it in this method

onCreate method



annotation：standing for overriding the method of super class

Key-value data structure：the value is null if this Activity is newly created. Or, the value is used to recover the states of Activity, which is closed for memory reason.

```
@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
```

Call onCreate() method of super class：it is necessary. Or, the system will throw exception

Setting view of screen：need a parameter of view resource or view

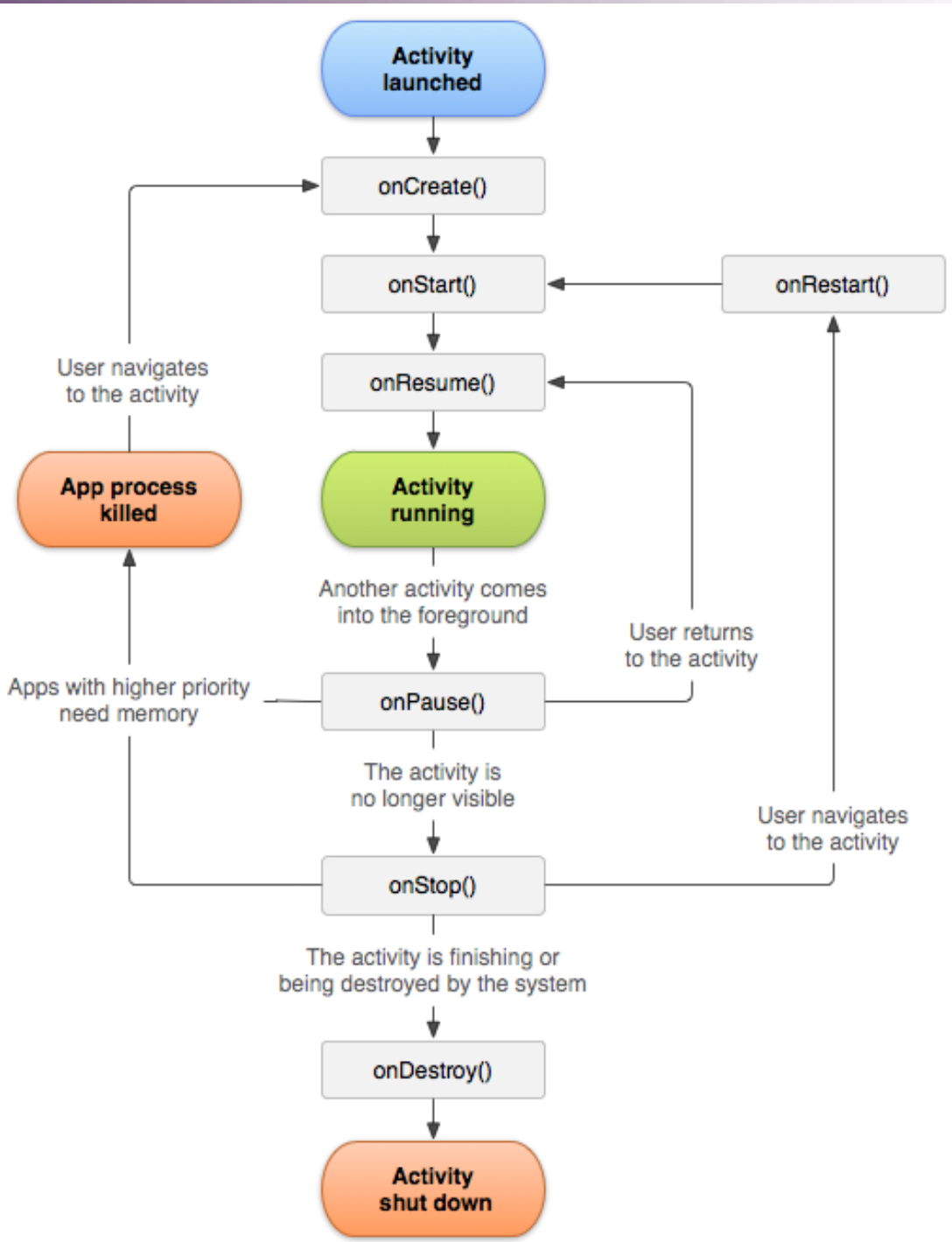
View resource ID：generated in R.java by system automatically and used to link the xml resource of view

Android UI Components

- Activity and intent (cont.)
  - Manage the activity lifecycle

```
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }
    @Override
    protected void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }
    @Override
    protected void onResume() {
        super.onResume();
        // The activity has become visible (it is now "resumed").
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Another activity is taking focus (this activity is about to be "paused").
    }
    @Override
    protected void onStop() {
        super.onStop();
        // The activity is no longer visible (it is now "stopped")
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // The activity is about to be destroyed.
    }
}
```

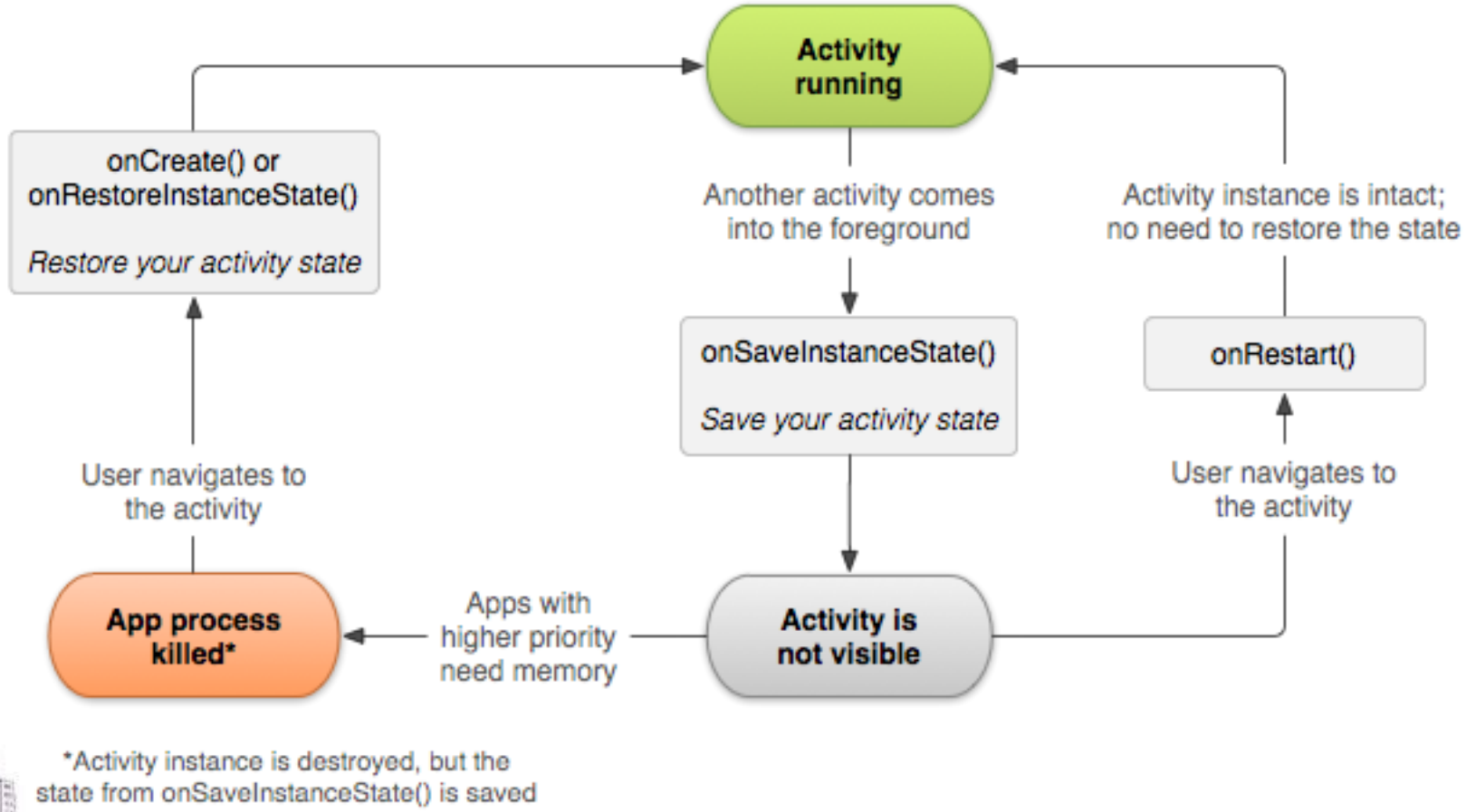
Lifecycle of Activity



Android UI Components

- Activity and intent
  - Save & restore activity state
    - onSaveInstanceState()
      - Called to retrieve per-instance state from an activity before being killed so that the state can be restored in onCreate() or onRestoreInstanceState()
    - onCreate () or onRestoreInstanceState()
      - Restore acitivity state from a previous state saved by onSaveInstanceState()

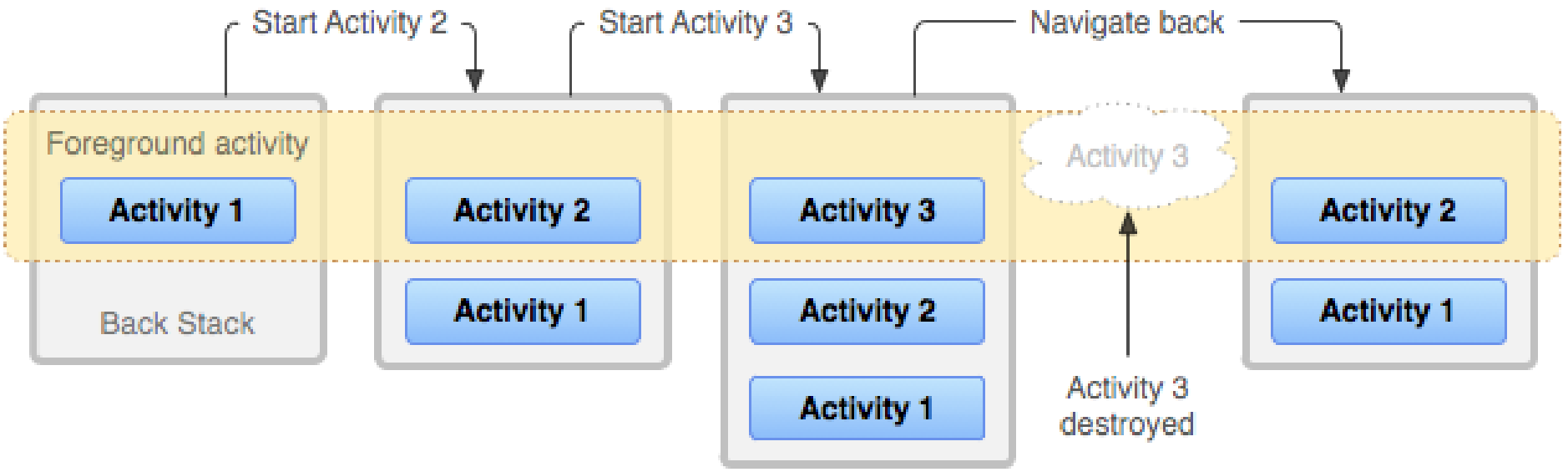
Save activity instance state



Android UI Components

- Activity and intent (cont.)
  - Tasks and Back Stack
    - A task is a collection of activities that users interact with when performing a certain job
    - The device Home screen is the starting place for most tasks
    - The activities are arranged in a stack (the "back stack"), in the order in which each activity is opened
    - The "main" activity for that application opens as the root activity in the stack

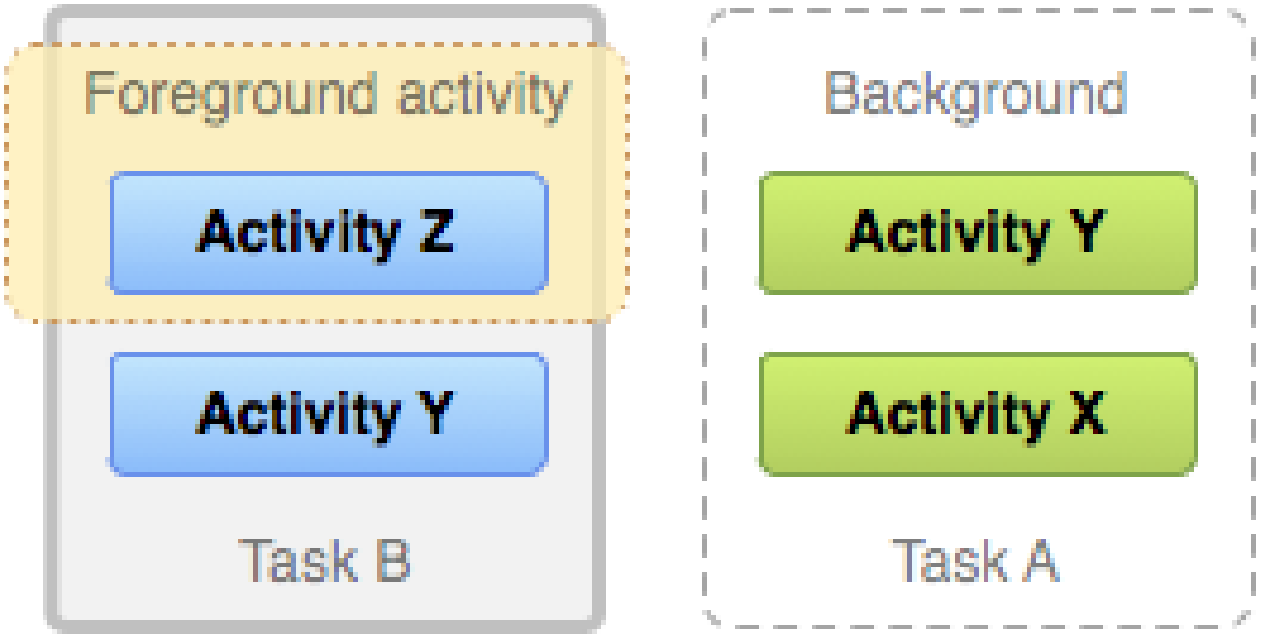
A representation of how each new activity in a task adds an item to the back stack. When the user presses the *Back* button, the current activity is destroyed and the previous activity resumes



Android UI Components

- Activity and intent (cont.)
  - Tasks and Back Stack
    - A task is a cohesive unit that can move to the "background" when users begin a new task or go to the Home screen, via the Home button
    - When all activities are removed from the stack, the task no longer exists
    - A task can then return to the "foreground" so users can pick up where they left off

Two tasks: Task B receives user interaction in the foreground, while Task A is in the background, waiting to be resumed



Android UI Components

- Activity and intent (cont.)
  - Activity launch modes
    - standard
      - Multiple instances of the activity class can be instantiated and multiple instances can be added to the same task or different tasks
    - singleTop
      - if an instance of activity already exists at the top of the current task and system routes intent to this activity, no new instance will be created

Android UI Components

- Activity and intent (cont.)
  - Activity launch modes (cont.)
    - singleTask
      - A new task will always be created and a new instance will be pushed to the task as the root one
    - singleInstance
      - The activity with such launch mode is always in a single activity instance task

Activity launch modes

Use Cases	Launch Mode	Multiple Instances?	Comments
Normal launches for most activities	`"standard"`	Yes	Default. The system always creates a new instance of the activity in the target task and routes the intent to it.
	`"singleTop"`	Conditionally	If an instance of the activity already exists at the top of the target task, the system routes the intent to that instance through a call to its `onNewIntent()` method, rather than creating a new instance of the activity.
Specialized launches *(not recommended for general use)*	`"singleTask"`	No	The system creates the activity at the root of a new task and routes the intent to it. However, if an instance of the activity already exists, the system routes the intent to existing instance through a call to its `onNewIntent()` method, rather than creating a new one.
	`"singleInstance"`	No	Same as `"singleTask"`, except that the system doesn't launch any other activities into the task holding the instance. The activity is always the single and only member of its task.

Android UI Components

- Activity and intent (cont.)
  - Intent
    - is a facility for late run-time binding between components in the same or different applications
    - is a passive data structure holding an abstract description of an operation to be performed
    - It contains
      - component name, action, data, category, extras, flags

Intent

```
Intent intent = new Intent();//create a new intent message
Bundle info =new Bundle();//create a new key-value variable
info.putCharSequence("key1", "testValue");

intent.setClass(this, SecondAndroidActivity.class);//set component name
intent.putExtras(info);// set extra infomation
intent.setFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);//set flag

startActivity(intent);//start a new activity by intent
```

```
Intent intent = new Intent();//create a new intent message

intent.setAction(Intent.ACTION_VIEW);//set action for intent
intent.setData(Uri.parse("http://www.ustc.edu.cn"));//set data for intent
intent.addCategory(Intent.CATEGORY_BROWSABLE);//set category for intent

startActivity(intent);//start a new activity by intent
```

Android UI Components

- Activity and intent (cont.)
  - Intent (cont.)
    - *Explicit* Intent
      - Designating the target component by its name
      - Being used for application-internal messages
    - *Implicit* Intent
      - Using Intent filters to find target component
      - Intent filters contains：action、data、category
      - Flags and extras are not used as filter condition
      - Being used to activate components in other applications

Android UI Components

- Activity and intent (cont.)
  - Intent (cont.)
    - Intent filter
      - To inform the system which implicit intents they can handle
      - activities, services, and broadcast receivers can have one or more intent filters
      - A filter has fields that parallel the action, data, and category fields of an Intent object. An implicit intent is tested against the filter in all three areas

Android UI Components

- Activity and intent (cont.)
  - Intent filter
    - is invalid when use Explicit Intent to activate the target component
    - “android.intent.category.DEFAULT” must be included in defined intent filter, EXCEPT
      - the pair of “android.intent.category.LAUNCHER” and “android.intent.action.MAIN”, which stands for entrance point when user launches the app on apps launch board

Intent filter

```
Intent intent = new Intent();
intent.setAction("water.action.FIRSTVIEW");//设置intent的Action
intent.addCategory("water.category.FIRSTCATEGORY");//设置intent的category
intent.setDataAndType(Uri.parse("http://www.testAndroid.com"),
        "text/html");//设置intent的data和type
startActivity(intent);//启动intent
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="water.activity" android:versionCode="1" android:versionName="1.0">
    <uses-sdk android:minSdkVersion="9" />
    <uses-permission android:name="android.permission.CALL_PHONE"></uses-permission>

    <permission android:protectionLevel="normal" android:name="water.permission.VIEW"/><!--添加自定义权限 -->

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

        <activity android:name="SecondAndroidActivity"
            android:permission="water.permission.VIEW">
            <intent-filter> <!--定义intent filter -->
                <action android:name="water.action.FIRSTVIEW" /><!--添加自定义action -->
                <action android:name="water.action.SECONDVIEW" /><!--添加自定义action -->
                <category android:name="water.category.FIRSTCATEGORY" /><!--添加自定义category -->
                <category android:name="android.intent.category.DEFAULT"></category><!--添加DEFAULT category -->
                <data android:mimeType="text/html" android:scheme="http"
                    android:host="www.testAndroid.com" android:port="8888" /><!--添加自定义data -->
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Software Engineering

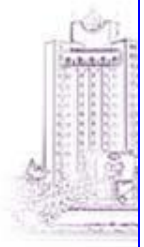
Android UI Components

- Activity and intent (cont.)
  - Pass data between activities
    - startActivityForResult() -> finish() -> onActivityResult()
      - data passed from activities started to original activities
      - requestCode: If the value >=0, the system calls onActivityResult() to handle result. Or, the system doesn’t call onActivityResult() method
    - startActivity() -> getIntent()
      - data passed from original activities to activities started

startActivity() -> getIntent()

```
Intent intent=new Intent();
intent.setClass(this, FinishActivity.class);
intent.putExtra("data", "data bundled");
startActivity(intent);
```

```
public class FinishActivity extends Activity implements OnClickListener{

    Button finishButton;

    private String dataBundled=null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_finish);

        Intent data=getIntent();
        dataBundled=data.getStringExtra("data");

        finishButton=(Button)findViewById(R.id.finish);
        finishButton.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        ((Button)v).setText(dataBundled);
    }
}
```



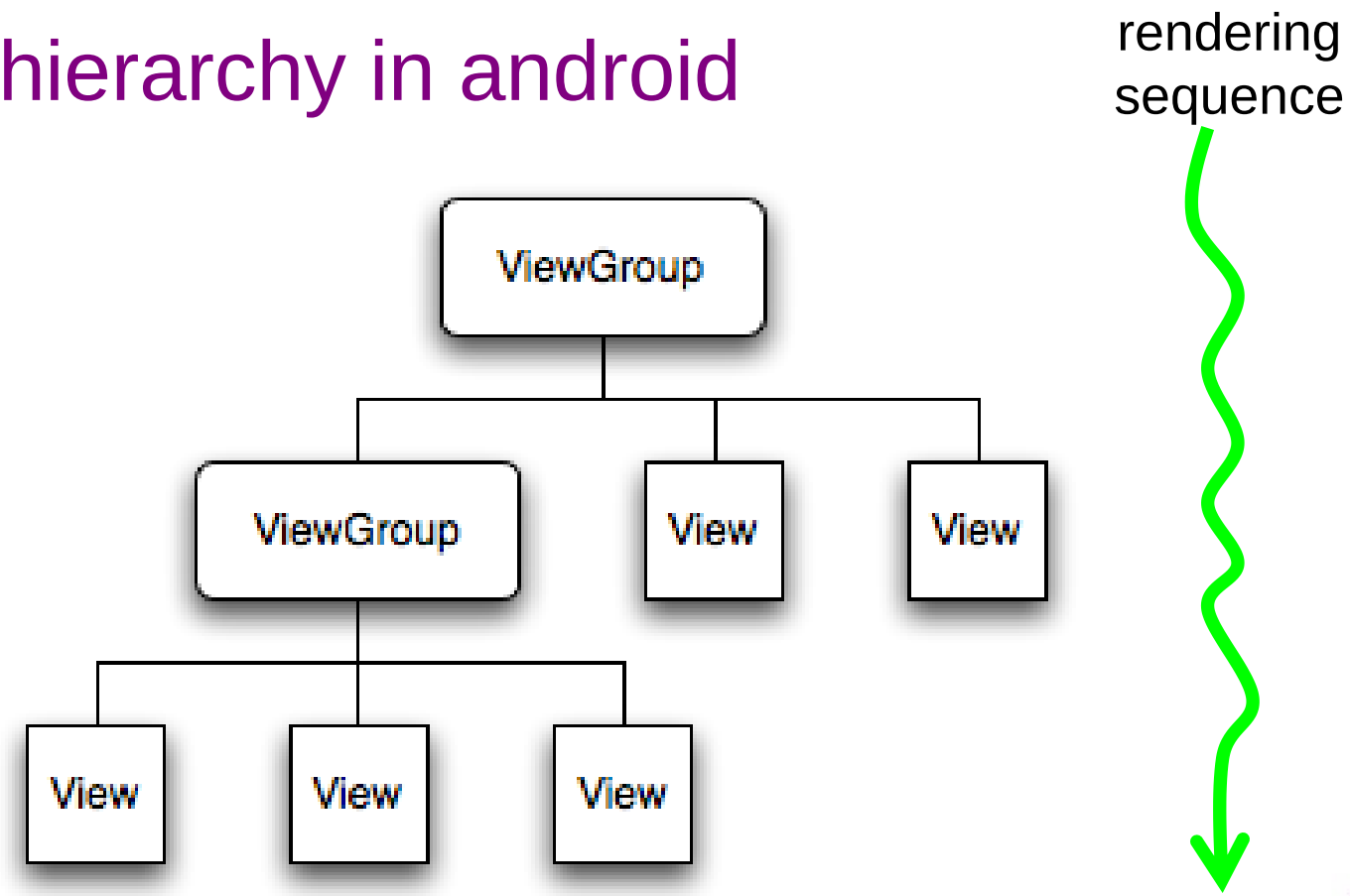
startActivityForResult() -> finish() -> onActivityResult()

```java
public class MainActivity extends Activity implements OnCl
    Button nextButton;
    private int mainActivityRequest = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        nextButton = (Button) findViewById(R.id.next);
        nextButton.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {

        Intent intent = new Intent();
        intent.setClass(this, FinishActivity.class);
        intent.putExtra("data", "data passed");
        startActivityForResult(intent, mainActivityRequest);

    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        // TODO Auto-generated method stub
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == mainActivityRequest) {

            String returnResult = data.getStringExtra("returnData");

            nextButton.setText(returnResult);

        }
    }
}
```

```java
public class FinishActivity extends Activity implements OnClickListener{

    Button finishButton;
    private int finishResult=5;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_finish);
        finishButton=(Button)findViewById(R.id.finish);
        finishButton.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {

        Intent intent=new Intent();
        intent.putExtra("returnData", "resultData");
        setResult(finishResult, intent);
        finish();

    }
}
```

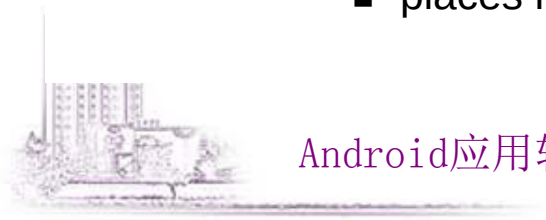
Android UI Components

- View hierarchy in android



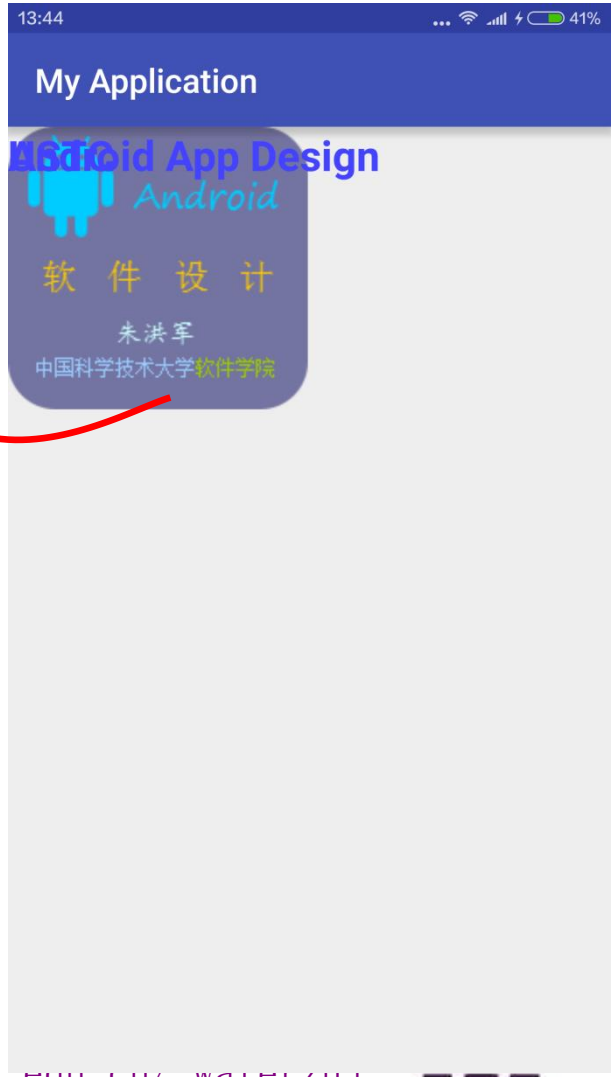
Android UI Components

- Android UI Layout
  - Being defined in xml file. Each XML node is a view or ViewGroup
  - Common Layout Objects
    - FrameLayout/CardView
      - It's basically a blank space on your screen that you can later fill with a single object. Subsequent child views will simply be drawn over previous ones
    - LinearLayout
      - aligning all children in a single direction — vertically or horizontally
    - TableLayout
      - positioning its children into rows and columns
    - RelativeLayout/ConstraintLayout (compatible with api 9 or higher)
      - leting child views specify their position relative to the parent view or to each other (specified by ID)
    - AbsoluteLayout (Deprecated)
      - positioning its children by their coordinates (x, y)
    - GridLayout (From api level 14)
      - places its children in a rectangular grid



FrameLayout

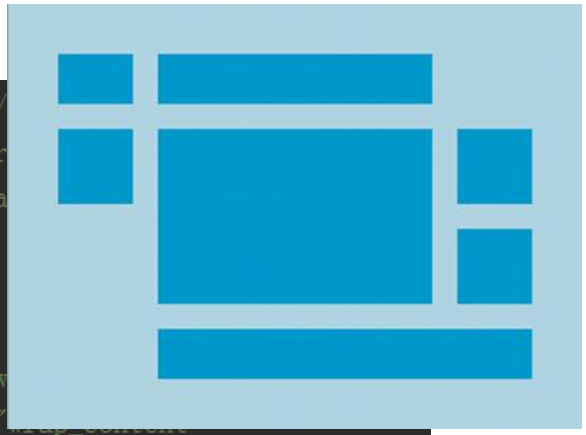
```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/logo"
        android:contentDescription="@string/app_name"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="USTC"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Android App Design"/>
</FrameLayout>
```



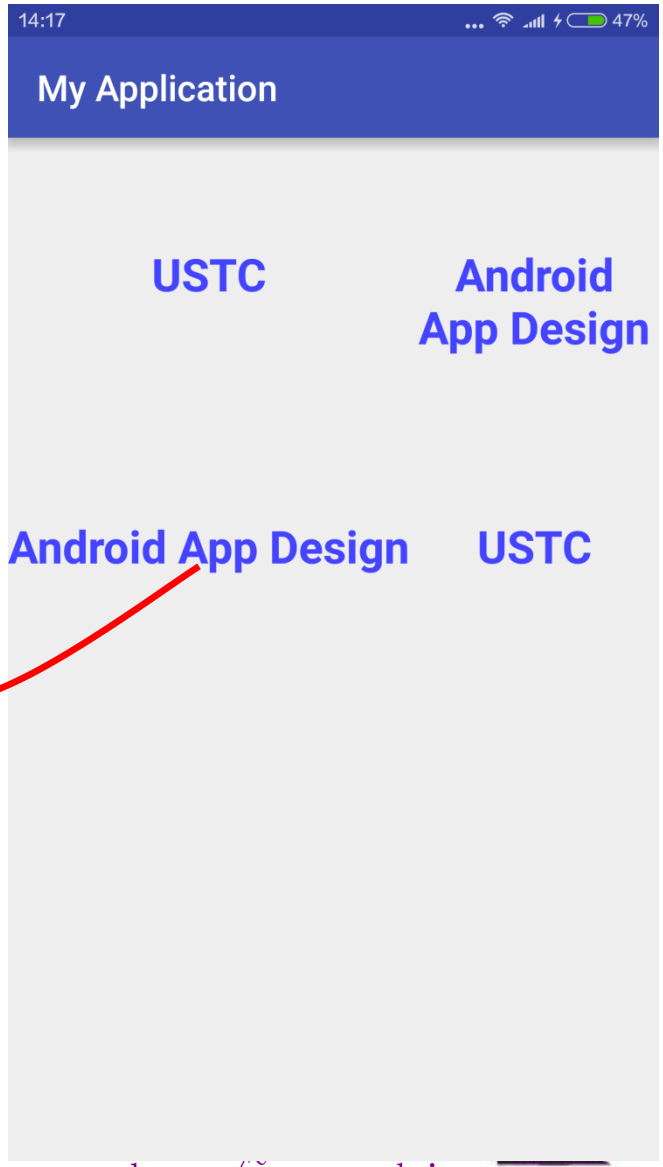
LinearLayout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/logo"
        android:contentDescription="My Application"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/ustc"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Android App Design"/>
</LinearLayout>
```

TableLayout



```
<TableLayout xmlns:android="http://
    android:layout_width="match_par
    android:layout_height="match_pa
    android:shrinkColumns="1">
    <TableRow>
        <TextView
            android:layout_width="w
            android:layout_height="
            android:text="USTC" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/androidapp" />
    </TableRow>
    <TableRow>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_conten
            android:text="Android App Design" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="USTC" />
    </TableRow>
</TableLayout>
```

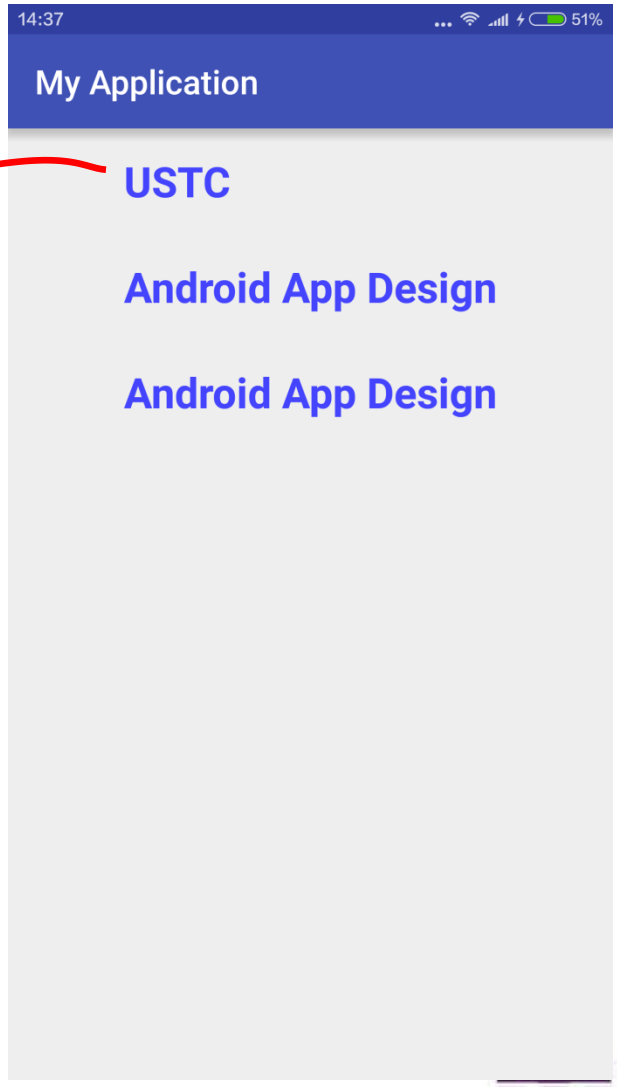


//staff.ustc.edu.cn/~waterzhj

Software Engineering

RelativeLayout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/r
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal">

    <TextView
        android:id="@+id/text1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/ustc" />

    <TextView
        android:id="@+id/text2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/text1"
        android:text="@string/androidapp" />

    <TextView
        android:id="@+id/text3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/text2"
        android:text="Android App Design" />

</RelativeLayout>
```



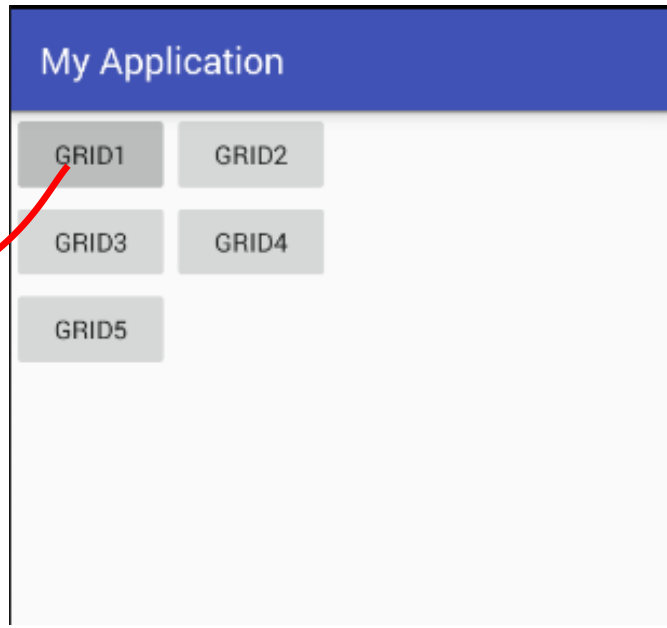
://staff.ustc.edu.cn/~waterzhj

Software Engineering

GridLayout

GridLayout

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="2" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="grid1"></Button>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="grid2"></Button>
    <Button
```



Android UI Components

```
public final class R {
    public static final class id {
        public static final int test_image=0x7f070003;
    }
```

- Android UI Component Properties
  - ID（Optional）
    - A integer value, being used to identify different view components
    - Defining ID
      - demo：
      ```
      android:id="@+id/test_image"
      ```
    - Using existing ID
      - demo：
      ```
      android:id="@android:id/empty"
      ```
    - Using ID to link view resouce to a component
      - demo：
      ```
      Button myButton = (Button) findViewById(R.id.my_button);
      ```

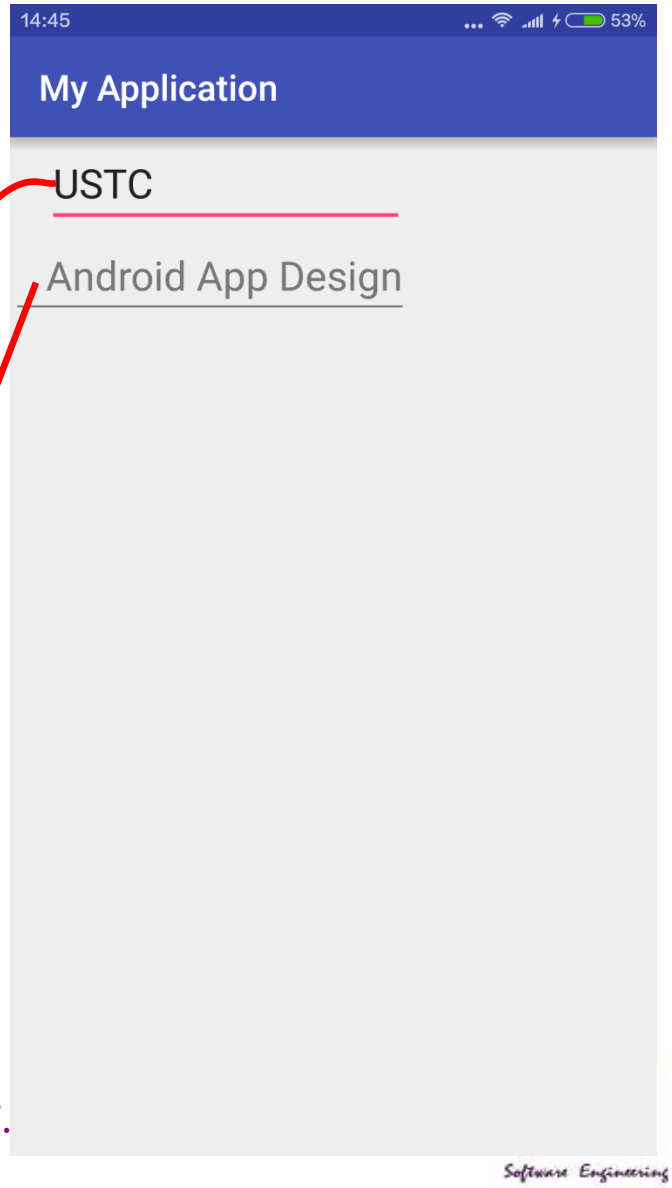
"+" mark means when the ID does not exist, the system will create it

Android UI Components

- Android UI Component Properties (cont.)
  - Size（Necessary）
    - layout_width
      - The values can be {fill_parent|*match_parent*, wrap_content, custom size}
    - layout_height：
      - The values can be {fill_parent|*match_parent*, wrap_content, custom size}
  - Paddings（Optional）
    - Using pixels to pad View to get margin effect
  - Margins（Optional）
    - Using pixels to pad View Groups to get margin effect

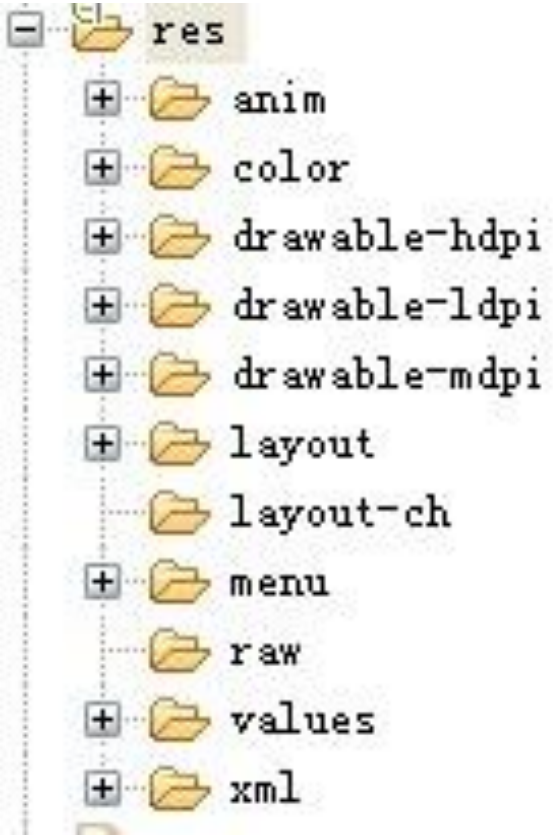
Setting UI component properties

```
<RelativeLayout xmlns:android="http://schemas.an
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="left">

    <EditText
        android:id="@+id/text1"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:text="USTC" />

    <EditText
        android:id="@+id/text2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/text1"
        android:hint="@string/androidapp"
        android:paddingLeft="20dp" />
</RelativeLayout>
```

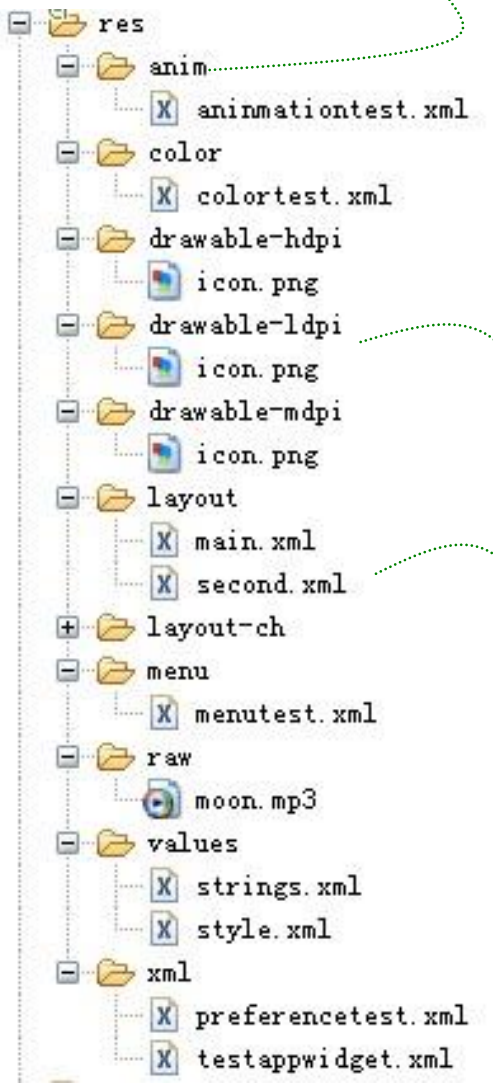


Android UI Components

- Using Android UI Resources
  - Resource Types
    - anim/
      - Ainimation resource（XML file）
    - color/
      - Color resource（XML file）
    - drawable/
      - Graphic resource（png/jpg file）
    - layout/
      - Layout resource（XML file）
    - menu/
      - Menu resource（XML file）
    - raw/
      - Raw file resource（all uncompressed files）
    - values/
      - Value type resource（XML file）
    - xml/
      - Xml resource（XML file）



Resource file and R.java

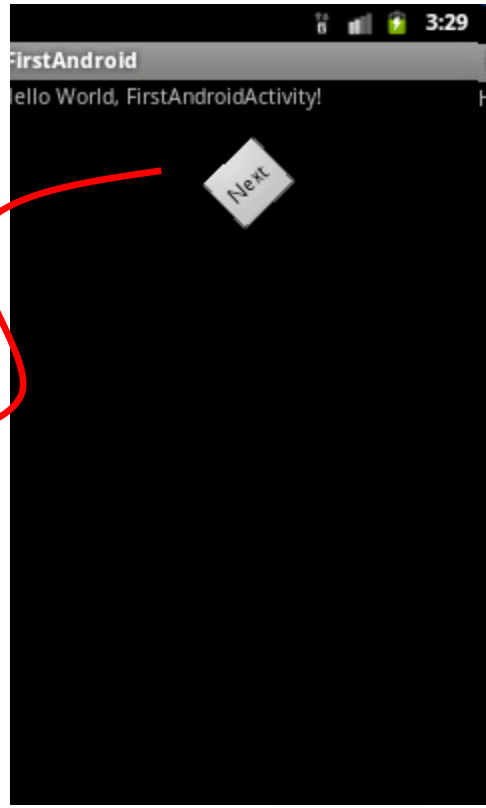



```
public final class R {
    public static final class anim {
        public static final int aninmationtest=0x7f040000;
    }
    public static final class array {
        public static final int colors=0x7f0d0000;
    }
    public static final class bool {
        public static final int text_enabled=0x7f0a0000;
    }
    public static final class color {
        public static final int color_black=0x7f090001;
        public static final int colortest=0x7f090002;
        public static final int view_background=0x7f090000
    }
    public static final class dimen {
        public static final int location_x1=0x7f0e0000;
        public static final int location_x2=0x7f0e0001;
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int button_finish=0x7f0b0000;
        public static final int finish_button=0x7f0b0004;
    }
    public static final class integer {
        public static final int max_time=0x7f0c0000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
        public static final int second=0x7f030001;
        public static final int testframelayout=0x7f030002
    }
    public static final class menu {
        public static final int menutest=0x7f0f0000;
    }
    public static final class raw {
        public static final int moon=0x7f060000;
    }
    public static final class string {
        public static final int app_name=0x7f070001;
```

Using animation resource

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <rotate android:duration="5000" android:fromDegrees="0"
        android:toDegrees="360" android:interpolator="@android:anim/linear_interpolator"
        android:repeatCount="3"></rotate>
</set>
```

```
Animation anim=AnimationUtils.loadAnimation(this, R.anim.aninmationtest);
((Button)findViewById(R.id.next_button)).startAnimation(anim);
```




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:layout_width="fill_parent" android:id="@+id/test_text1"
        android:layout_height
    <Button android:layout_wi
        android:layout_height
        android:onClick="onCl
        android:layout_gravit
</LinearLayout>
```

When user clicks the button, it will rotate from 0 to 360 degree. It rotates 3 times and each cycle lasts 5 seconds. The frames change in constant speed

Using color resource

```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="#FF00FF" android:state_focused="true"/>
    <item android:color="#00FF00"/>
</selector>
```

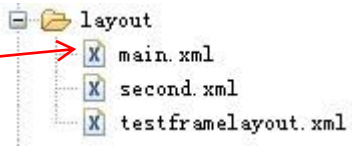
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <EditText android:layout_width="fill_parent" android:id="@+id/test_text1"
        android:layout_height="wrap_content" android:layout_marginTop="50dp"
        android:text="@string/hello" android:textColor="@color/colortest" />

    <EditText android:layout_width="fill_parent" android:id="@+id/test_text2"
        android:layout_height="wrap_content" android:hint="@string/hint_str"
        android:textColor="@color/colortest" />

    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:id="@+id/next_button"
        android:onClick="onClickNextButton" android:text="@string/next"
        android:layout_gravity="center_horizontal" android:layout_marginTop="50dp" />

</LinearLayout>
```

When EditText is on focus, its color turns to #FF00FF from #00FF00

Using other resources

```
<ImageView android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:src="@drawable/icon"
    android:layout_x="@dimen/location_x1" android:layout_y="@dimen/location_y1" />
```


```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}
```




```
if (mp == null) {
    mp = MediaPlayer.create(this, R.raw.moon);
}
```

Android UI Components

- Using Android UI Resources（cont.）
  - Value type resources
    - **String**：being used to define constant string values
    - **Style**：being used to define styles of components
    - **Dimension**：being used to define dimensions of components
    - **Color**：being used to define color of components
    - **Bool**：being used to define boolean values
    - **ID**：being used to define identifier values of components
    - **Integer**：being used to define integer values
    - **Array**：being used to define constant array values
      - Integer array
      - Other types array

Defining and using String and Style resource

```xml
    <string name="next">Next</string>
    <string name="finish">Finish</string>
    <string name="test_str">Test FrameLayout!</string>
    <string name="test_str2">USTC~</string>
    <string name="hint_str">Please input your words here!</string>
```

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

    <EditText android:layout_width="wrap_content" style="@style/style1_text"
        android:id="@+id/test_text1" android:layout_height="wrap_content"
        android:layout_marginLeft="30dp" android:layout_marginTop="15dp"/>

    <EditText android:layout_width="wrap_content"
        android:layout_height="wrap_content" style="@style/style2_text"
        android:id="@+id/test_text2" android:text="@string/test_str2"
        android:layout_below="@id/test_text1" android:paddingLeft="30dp" />

</RelativeLayout>
```

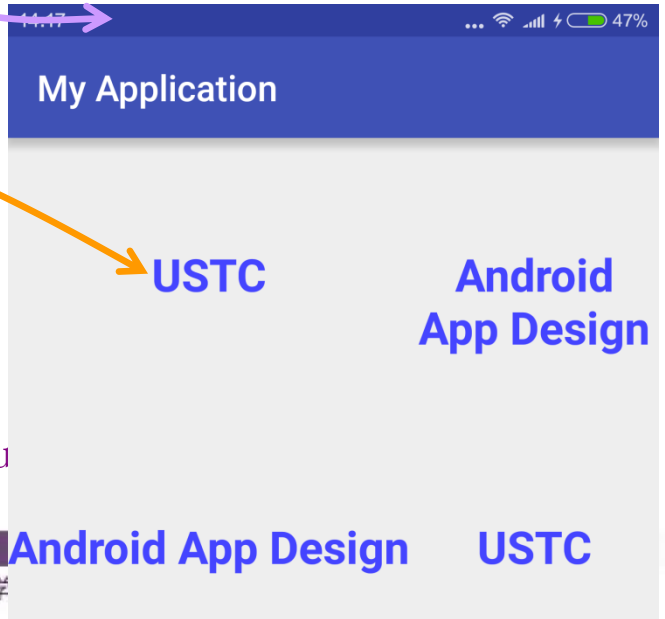
```xml
<style name="style1_text">
    <item name="android:textColor">#FF0000</item>
    <item name="android:textStyle">italic</item>
    <item name="android:textSize">8pt</item>
</style>
<style name="style2_text">
    <item name="android:textColor">#000000</item>
    <item name="android:textStyle">bold</item>
    <item name="android:textSize">8pt</item>
</style>
```

Style and theme

```xml
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

```xml
<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
    <item name="android:textViewStyle">@style/text_style</item>
</style>
<style name="text_style">
    <item name="android:textSize">25dp</item>
    <item name="android:textColor">#4444FF</item>
    <item name="android:textStyle">bold</item>
    <item name="android:gravity">center</item>
    <item name="android:singleLine">false</item>
    <item name="android:lines">2</item>
    <item name="android:layout_centerInParent">true</item>
</style>
```

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.water.myapplication">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```



Android UI Components

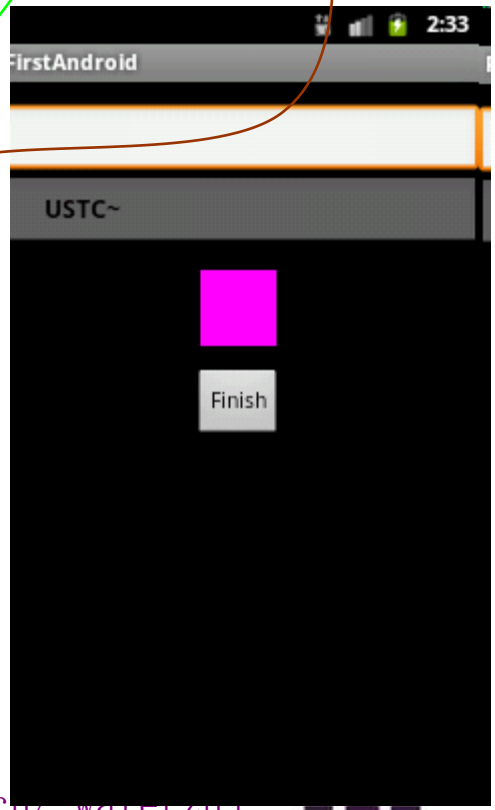
- Using Android UI Resources（cont.）
  - Value type resources（cont.）
    - Dimension
      - **dp**：Density-independent Pixels - an abstract unit that is based on the physical density of the screen
      - **sp**：Scale-independent Pixels - this is like the dp unit, but it is also scaled by the user's font size preference
      - **pt**：Points - 1/72 of an inch based on the physical size of the screen
      - **px**：Pixels - corresponds to actual pixels on the screen
      - **mm**：Millimeters - based on the physical size of the screen
      - **in**：Inches - based on the physical size of the screen

```
<dimen name="textview_height">25dp</dimen>
<dimen name="textview_width">150dp</dimen>
<dimen name="ball_radius">30dp</dimen>
<dimen name="font_size">16sp</dimen>
```

Other value type resources

```
<EditText android:layout_width="fill_parent"
    android:layout_height="wrap_content" style="@style/style2_text"
    android:id="@+id/test_text2" android:text="@string/test_str2"
    android:paddingLeft="30dp" android:enabled="@bool/text_enabled" />

<View style="@style/view_style" android:background="@color/view_background"
    android:layout_marginTop="15dp" android:layout_gravity="center" />

<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:id="@id/button_finish"
    android:text="@string/finish" android:layout_gravity="center"
    android:layout_marginTop="15dp" />
```

```
<color name="view_background">#ff00ff</color>
<bool name="text_enabled">true</bool>
<item type="id" name="button_finish"></item>
<integer name="max_time">5000</integer>
<array name="colors">
    <item>#FFFF0000</item>
    <item>#FF00FF00</item>
    <item>#FF0000FF</item>
</array>
```



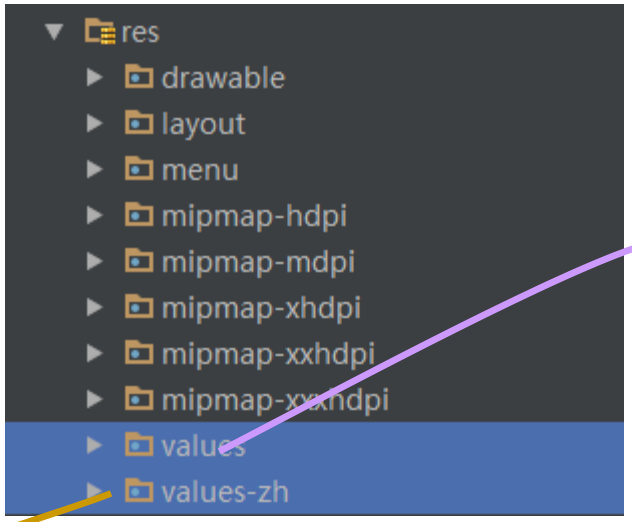

```
Resources r=getResources();
TypedArray ta=r.obtainTypedArray(R.array.colors);
int color=ta.getColor(1,0);
```

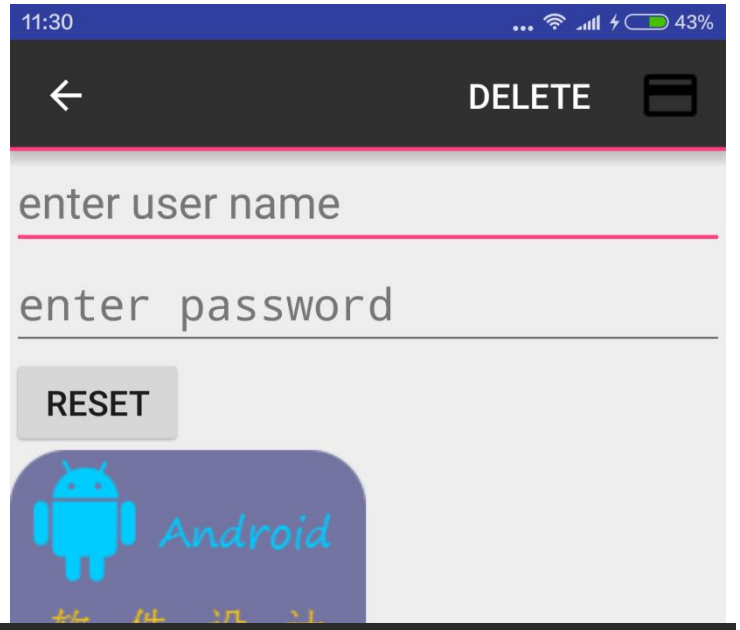
```
<rotate android:duration="@integer/max_time"
    android:fromDegrees="0" android:toDegrees="360"
    android:interpolator="@android:anim/linear_interpolator"
    android:repeatCount="3"></rotate>
```

Android UI Components

- Localization
- use the Android resource framework to separate the localized aspects of your application as much as possible from the core Java functionality
  - res/values is requried as default resources directory
  - alternative resources for different languages
    - res/values-languageCode-languageRegion

Localization






```
<resources>
    <string name="app_name">My Application</string>
    <string name="user_name">enter user name</string
    <string name="password">enter password</string>
    <string name="reset">Reset</string>
```

```
<resources>
    <string name="app_name">我的应用</string>
    <string name="user_name">输入用户名</string
    <string name="password">输入密码</string>
```

/staff.ustc.edu.cn/~waterzhj

Android UI Components

- Supporting Different Screens
- size
  - small, normal, large, xlarge
- density
  - low (ldpi), medium (mdpi), high (hdpi), extra high (xhdpi)
- different layouts
  - res/layout-size
- different bitmaps
  - res/drawable-density

```
res/
    layout/                 # default (portrait)
        main.xml
    layout-land/            # landscape
        main.xml
    layout-large/           # large (portrait)
```

```
res/
    drawable-xhdpi/
        awesomeimage.png
    drawable-hdpi/
        awesomeimage.png
    drawable-mdpi/
```

Android UI Components

- Common view components
  - Button
    - Graphic user interface, being used to accept user's click operations
    - Subclasses：RadioButton/CheckBox, etc.
  - TextView
    - Graphic user interface, being used to accept user's input
    - Subclasses：EditText/CheckedTextView, etc.

Android UI Components

- Common view components（cont.）
  - ImageView
    - Being used to display image resources
    - Subclasses：ImageButton, etc.
  - ProgressBar
    - Being used to display progress information
  - AnalogClock
    - Being used to construct a clock
  - Etc.

Android UI Components

- Button
  - Button is always defined in XML file, and handles events in code
  - Two ways for event handling
    - Using OnClickListener interface in source code
    - Using android:onClick properties in xml file
      - Its value is the name of event recall method
      - The recall method needs a view as input parameter

```
public void onClickNextButton(View v) {
    }
```

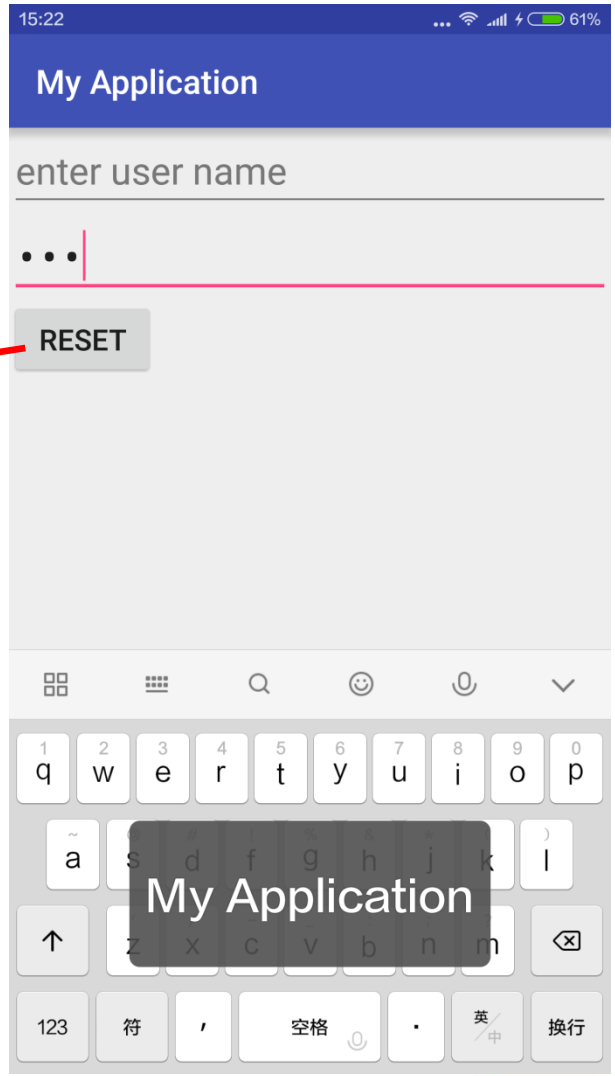
```
final Button button = (Button) findViewById(R.id.button_id);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Perform action on click
    }
});
```

```
<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:id="@+id/next_button"
    android:onClick="onClickNextButton" android:text="@string/next"/>
```

onClick property

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/text2"
    android:onClick="clickButton"
    android:text="@string/reset" />
```

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_layout);
    }

    public void clickButton(View v) {
        Toast.makeText(this, R.string.app_name, Toast.LENGTH_LONG).show();
    }

}
```



OnClickListener

```xml
<Button android:id="@+id/reset"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/text2"
    android:text="Reset" />
```

```java
public class MainActivity extends AppCompatActivity implements View.OnClickListener{

    private Button reset;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_layout);
        reset=(Button)findViewById(R.id.reset);
        reset.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        Toast.makeText(this,R.string.app_name,Toast.LENGTH_LONG).show();
    }

}
```

hj

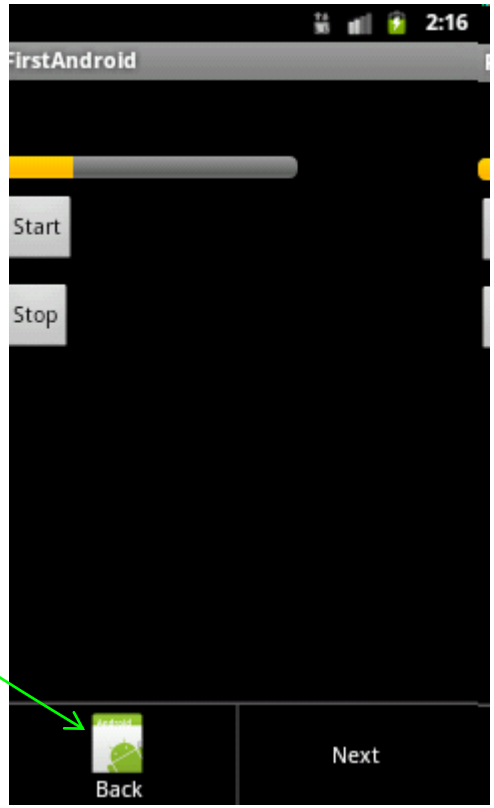
Android UI Components

- Menu
  - The ways to create menu
    - Using XML resource file
    - Using source code
  - OptionsMenu（2.3.X or lower）
    - The primary collection of menu items for an activity, which appears when the user touches the MENU button
    - Two important methods
      - onCreateOptionsMenu(Menu menu)
        - Being used to create OptionsMenu
        - Using MenuInflater to inflate Menu
      - onOptionsItemSelected(MenuItem item)
        - Being used to handle menu selected event
        - Using item.getItemId() to identify id of menuItem

Using xml resource file to create OptionsMenu （2.3.X or lower）

```xml
?xml version="1.0" encoding="utf-8"?>
menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_back" android:title="@string/menu_back_str"
        android:icon="@drawable/icon" />
    <item android:id="@+id/menu_next" android:title="@string/menu_next_str" />
/menu>
```

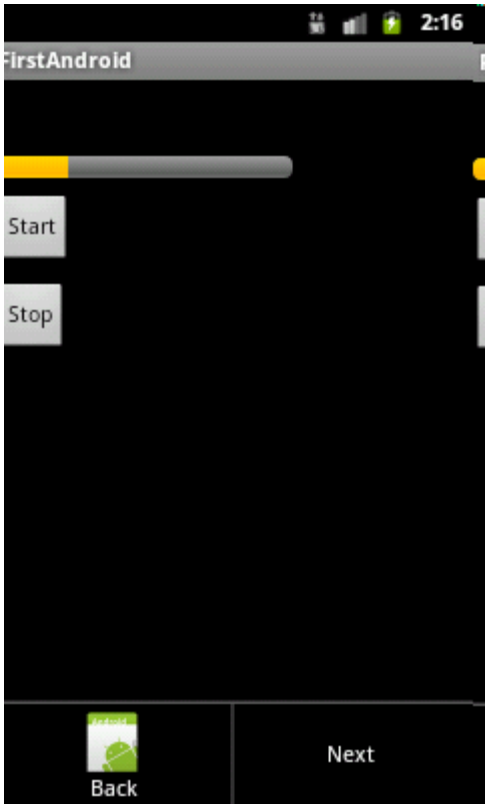
```java
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    switch (item.getItemId()) {
    case R.id.menu_back: {
        Toast.makeText(this, item.getTitle(), 2000).show();
        break;
    }
    case R.id.menu_next: {
        Toast.makeText(this, item.getTitle(), 2000).show();
        break;
    }
    default: {
        Toast.makeText(this, "NULL", 2000).show();
        break;
    }
    }
    return super.onOptionsItemSelected(item);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater mf = getMenuInflater();
    mf.inflate(R.menu.menutest, menu);
    return true;
}
```



Create OptionsMenu in source code （2.3.X or lower）

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    switch (item.getItemId()) {
    case R.id.menu_back: {
        Toast.makeText(this, item.getTitle(), 2000).show();
        break;
    }
    case R.id.menu_next: {
        Toast.makeText(this, item.getTitle(), 2000).show();
        break;
    }
    default: {
        Toast.makeText(this, "NULL", 2000).show();
        break;
    }
    }
    return super.onOptionsItemSelected(item);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuItem mi1 = menu.add(0, R.id.menu_back, 0, R.string.menu_back_str);
    mi1.setIcon(R.drawable.icon);
    menu.add(0, R.id.menu_next, 1, R.string.menu_next_str);
    return true;
}
```

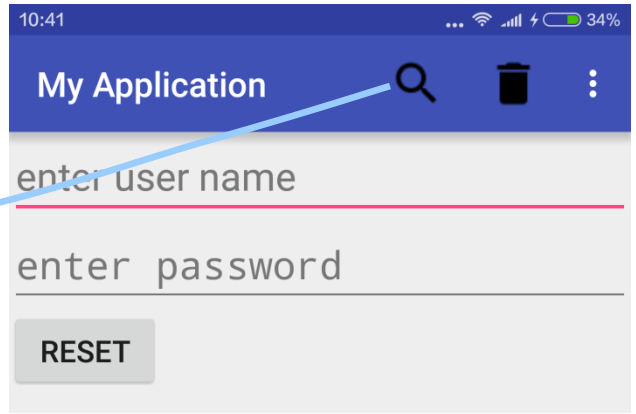
Deprecated for it is difficult to reuse

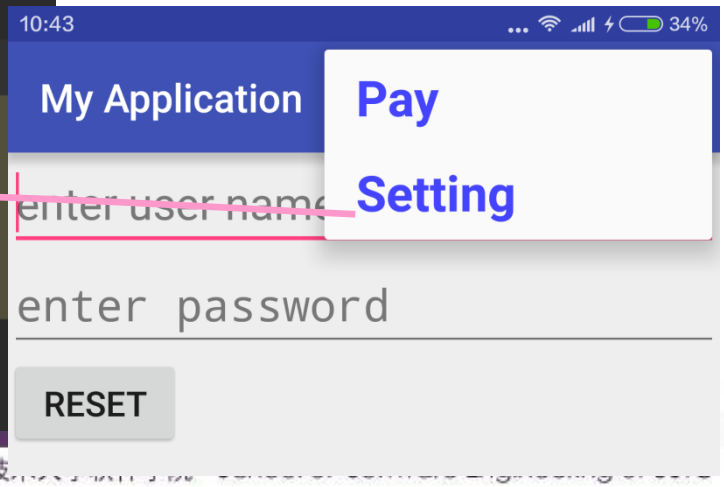


Android UI Components

- OptionsMenu/App bar (3.0 or higher)

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/search"
        android:icon="@drawable/search"
        android:title="@string/search"
        app:showAsAction="always"></item>
    <item...>
    <item...>
    <item
        android:id="@+id/setting"
        android:icon="@drawable/setting"
        android:title="@string/setting"
        app:showAsAction="never"></item>
</menu>
```





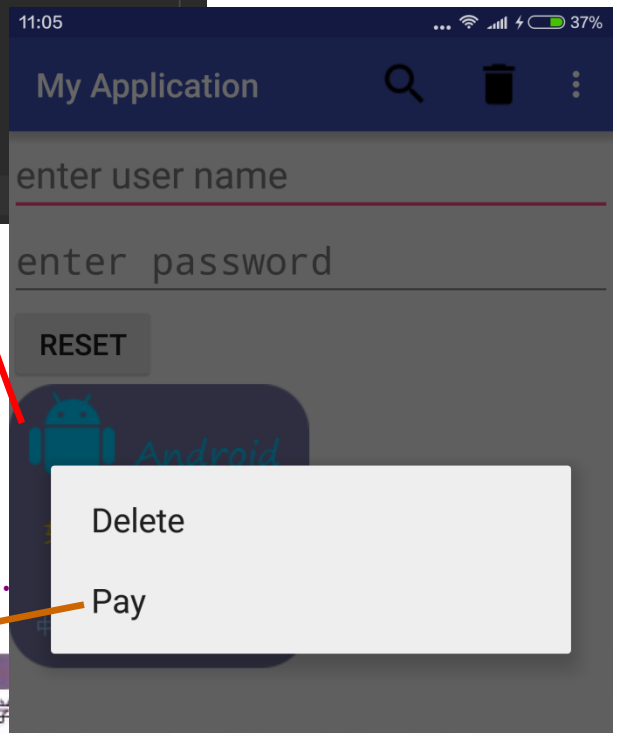
Android UI Components

- ContextMenu
  - A floating list of menu items that appears when the user touches and holds a view that's registered to provide a context menu
  - The important methods
    - onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo)
      - Being used to create ContextMenu
      - When the user performs a "long press" (press and hold) on an item, the system will call this method
    - onContextItemSelected(MenuItem item)
      - Being used to handle ContextMenu selected event
    - registerForContextMenu(View view)
      - Being used to register ContextMenu to a view

Create context menu

```java
public class MainActivity extends AppCompatActivity {
    ImageView imageView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_layout);
        imageView = (ImageView) findViewById(R.id.logo);
        registerForContextMenu(imageView);
    }

    @Override
    public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);
        getMenuInflater().inflate(R.menu.context_menu, menu);
    }

    @Override
    public boolean onContextItemSelected(MenuItem item) {...}
```

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/delete"
        android:title="@string/delete"></item>
    <item
        android:id="@+id/pay" android:icon="@drawable/pay"
        android:title="@string/pay"></item>
</menu>
```



Android UI Components

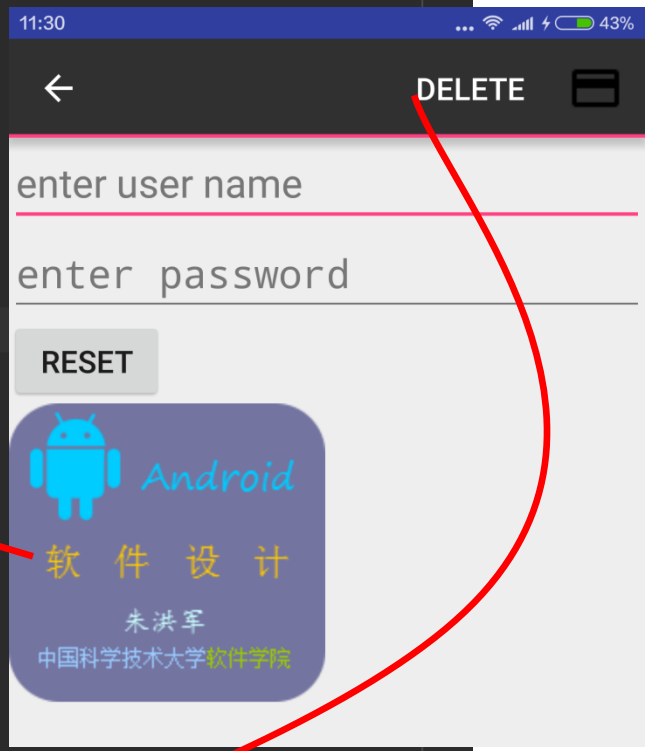
- Using the contextual action mode
  - invoke the contextual action mode upon one of two events (or both)
    - performs a long-click on the view
    - selects a checkbox or similar UI component within the view
  - enable the mode
    - Implement the ActionMode.Callback interface
    - Call startActionMode() when you want to show the action bar
      - such as when the user long-clicks the view

Cotextual action mode

```
ends AppCompatActivity implements View.OnLongClickListener{

    ActionMode actionMode=null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_layout);
        imageView = (ImageView) findViewById(R.id.logo);
        imageView.setOnLongClickListener(this);
    }
    @Override
    public boolean onLongClick(View v) {
        if(actionMode!=null){
            return false;
        }
        actionMode=this.startActionMode(actionModeCallBack);
        return true;
    }
    private android.view.ActionMode.Callback actionModeCallBack=new android.view.ActionMode.Callback(){
        @Override
        public boolean onCreateActionMode(ActionMode mode, Menu menu) {...}
```

```
) { return false; }

item) {...}

=null; }
```

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/delete"
        android:title="@string/delete"></item>
    <item
        android:id="@+id/pay" android:icon="@drawable/pay"
        android:title="@string/pay"></item>
</menu>
```



Android UI Components

- SubMenu
  - A submenu is a menu that the user can open by selecting an item in another menu
  - When the user selects an item from a submenu, the parent menu's respective on-item-selected callback method receives the event
- The features of menu
  - Menu group
    - A menu group is a collection of menu items that share certain traits
  - Checkable menu items
    - You can define the checkable behavior for individual menu items using the android:checkable attribute in the <item> element, or for an entire group with the android:checkableBehavior attribute in the <group> element

Using SubMenu demo

```
MenuInflater mi = getMenuInflater();
mi.inflate(R.menu.menutest, menu);
```

Using XML resource

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_back" android:title="@string/start">
        <menu>
            <item android:id="@+id/zoom_in" android:title="@string/zoomin" />
            <item android:id="@+id/zoom_out" android:title="@string/zoomout" />
        </menu>
    </item>
    <item android:id="@+id/menu_next" android:title="@string/finish" />
</menu>
```

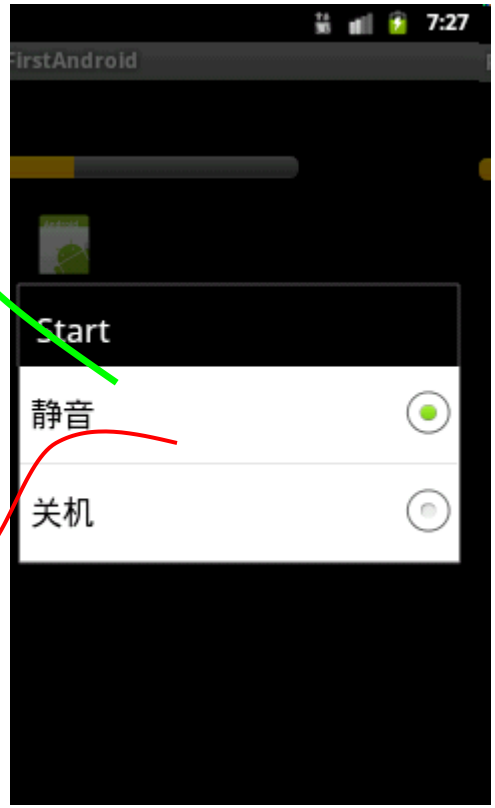
Creating SubMenu in source code

```
SubMenu sm = menu.addSubMenu(R.string.start);
sm.add(R.string.zoomin);
sm.add(R.string.zoomout);

menu.add(R.string.finish);
```

Using menu group demo

```
if(item.isChecked()){
    item.setChecked(false);
}else{
    item.setChecked(true);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_back" android:title="@string/start">
        <menu>
            <group android:id="@+id/menu_group1" android:checkableBehavior="single">
                <item android:id="@+id/zoom_in" android:title="@string/silent"
                    android:checked="true" />
                <item android:id="@+id/zoom_out" android:title="@string/close" />
            </group>
        </menu>
    </item>
    <item android:id="@+id/menu_next" android:title="@string/finish" />
</menu>
```

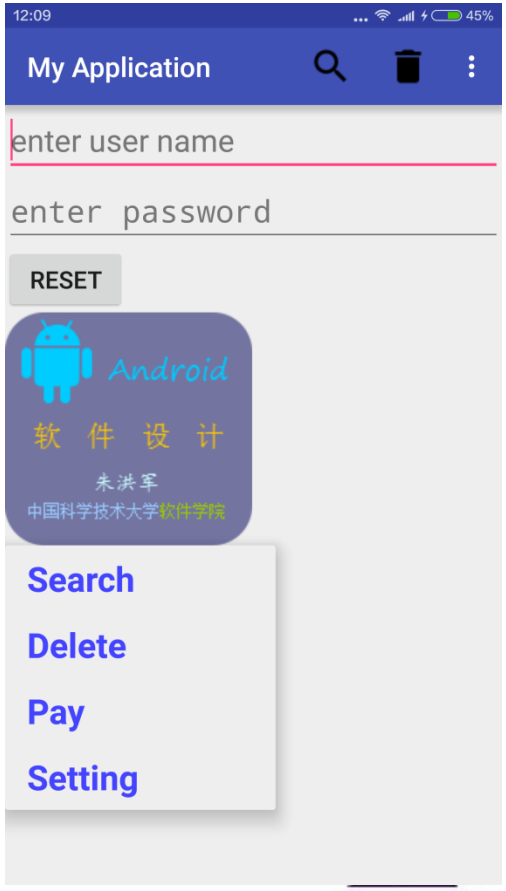


Android UI Components

- Popup Menu
  - A PopupMenu is a modal menu anchored to a View
  - appears below the anchor view if there is room, or above the view otherwise

```
<ImageView
    android:id="@+id/logo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/reset"
    android:onClick="onClickImage"
    android:src="@drawable/logo" />
```

```
public void onClickImage(View v) {
    PopupMenu popupMenu = new PopupMenu(this, v);
    popupMenu.getMenuInflater().inflate(R.menu.main_menu, popupMenu.getMenu());
    popupMenu.show();
}
```



u.cn/~waterzhj

Android UI Components

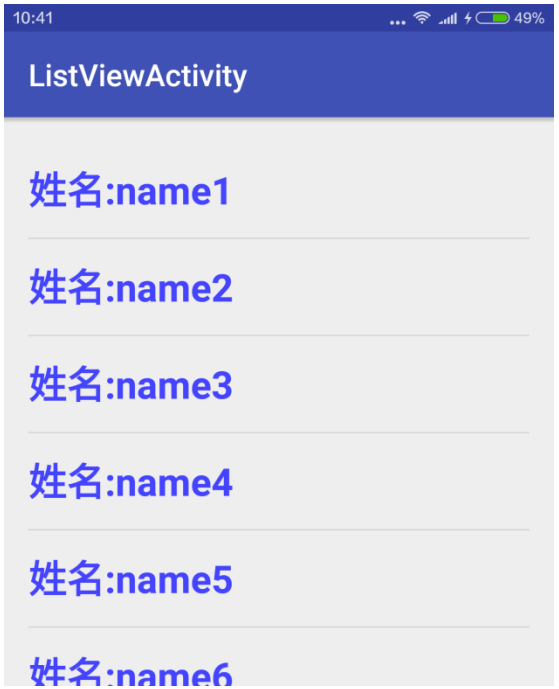
- ListView
  - A view that shows items in a vertically scrolling list
  - Using adapter to associate data with view
    - **ArrayAdapter**：By default this class expects that the provided resource id references a single TextView
    - **SimpleAdapter**：An easy adapter to map static data to views defined in an XML file
    - **CursorAdapter**：An easy adapter to map columns from a cursor to TextViews or ImageViews defined in an XML file
    - **BaseAdapter**：Common base class of common implementation for an Adapter is used to custom an adapter

Android UI Components

- Using ListActivity
  - There must be a ListView which id is defined as *@id/android:list* in Layout.xml file
  - Using onListItemClick(ListView l, View v, **int** position, **long** id) to handle item selected event
- Two ways to create a listview
  - In XML resource file
  - In source code
- Handling item selected event
  - OnItemClickListener，**onItemClick** (AdapterView<?> parent, View view, int position, long id)

Using ArrayAdapter to construct ListView

```java
public class ListViewActivity extends AppCompatActivity {
    ListView nameList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.content_list_view);

        nameList = (ListView) findViewById(R.id.name_list);
        ArrayAdapter<String> namesAdapter = new ArrayAdapter<String>(this,
                R.layout.list_layout, R.id.name, getArray());
        nameList.setAdapter(namesAdapter);
    }

    public List<String> getArray() {...}

}
```




```xml
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">
    <TableRow>
        <TextView android:text="@string/name"/>
        <TextView android:id="@+id/name"/>
    </TableRow>
</TableLayout>
```

```xml
<LinearLayout ......>

    <ListView
        android:id="@+id/name_list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"></ListView>
</LinearLayout>
```

Using SimpleAdapter to construct ListView

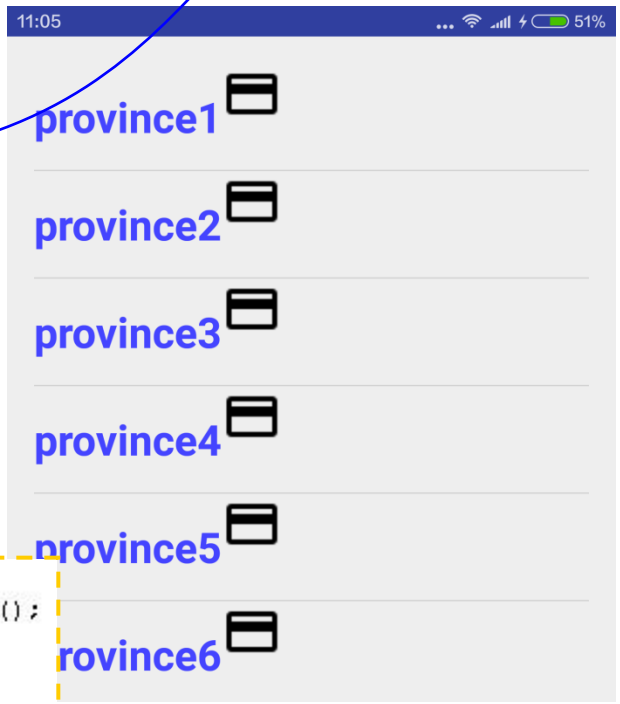
```
<string-array>
    <item>province3</item>
    <item>province4</item>
    ......
</string-array>
```

```
...s.android.com/apk/res/android"
...ndroid:layout_height="match_parent">
    <TableRow>
        <TextView android:id="@+id/list_text"/>
        <ImageView android:id="@+id/img" style="@style/image_style"/>
    </TableRow>
</TableLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ListView android:id="@+id/list1" android:layout_width="fill_parent"
        android:layout_height="wrap_content"></ListView>
</LinearLayout>
```

```
ListView lv = (ListView) findViewById(R.id.list1);
SimpleAdapter aa = new SimpleAdapter(this, initData(),
        R.layout.listlayout,
        new String[] { "text_view", "image_view" }, new int[] {
                R.id.list_text, R.id.img });
lv.setAdapter(aa);
```

```
public List<HashMap<String, Object>> initData() {
    List<HashMap<String, Object>> data = new ArrayList<HashMap<String, Object>>();
    TypedArray ta = getResources().obtainTypedArray(R.array.province_array);
    HashMap<String, Object> hm = null;
    for (int i = 0; i < ta.length(); i++) {
        hm = new HashMap<String, Object>();
        hm.put("text_view", ta.getString(i));
        hm.put("image_view", R.drawable.icon);
        data.add(hm);
    }
    return data;
}
```



Using SimpleCursorAdapter to construct ListView

```
public class ListViewActivity extends AppCompatActivity
        implements android.app.LoaderManager.LoaderCallbacks<Cursor> {

    ListView nameList;
    SimpleCursorAdapter namesAdapter;
```

```
ProgressBar progressBar = new ProgressBar(this);
progressBar.setIndeterminate(true);

nameList = (ListView) findViewById(R.id.name_list);
nameList.setEmptyView(progressBar);
ViewGroup root = (ViewGroup) findViewById(android.R.id.content);
root.addView(progressBar);

namesAdapter = new SimpleCursorAdapter(this, R.layout.list_
        new String[]{ContactsContract.Data.DISPLAY_NAME}, ne
nameList.setAdapter(namesAdapter);
getLoaderManager().initLoader(0, null, ListViewActivity.this
```

```
<ListView
    android:id="@+id/name_list"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"></ListView>
```


```
@Override
public android.content.Loader<Cursor> onCreateLoader(int id, Bundle args) {
    return new android.content.CursorLoader(this, ContactsContract.Data.CONTENT_URI,
            new String[]{ContactsContract.Data._ID,
                    ContactsContract.Data.DISPLAY_NAME}, null, null, null);
}

@Override
public void onLoadFinished(android.content.Loader<Cursor> loader, Cursor data) {
    namesAdapter.swapCursor(data);
}

@Override
public void onLoaderReset(android.content.Loader<Cursor> loader) {
    namesAdapter.swapCursor(null);
}
```

Custom BaseAdapter

```
listData = getListData();
CustomListAdapter customListAdapter = new CustomListAdapter(this, LayoutInflater.from(this));
nameList.setAdapter(customListAdapter);
```

```
final class ViewHolder {...}

class CustomListAdapter extends BaseAdapter {
    Context con;
    LayoutInflater li;
    CustomListAdapter(Context context, LayoutInflater layoutInflater) {...}
    @Override
    public int getCount() { return listData.size(); }
    @Override
    public Object getItem(int position) { return listData.get(position); }
    @Override
    public long getItemId(int position) { return position; }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        final int pos = position;
        ViewHolder viewHolder;
        if (convertView == null) {
            viewHolder = new ViewHolder();
            convertView = li.inflate(R.layout.list_layout, null);
            viewHolder.textView = (TextView) convertView.findViewById(R.id.list_text);
            viewHolder.imageView = (ImageView) convertView.findViewById(R.id.img);
            convertView.setTag(viewHolder);
        } else {
            viewHolder = (ViewHolder) convertView.getTag();
        }
        viewHolder.textView.setText((String) listData.get(pos).get("text_value"));
        viewHolder.imageView.setImageResource((int) listData.get(pos).get("img_value"));
        return convertView; }
```

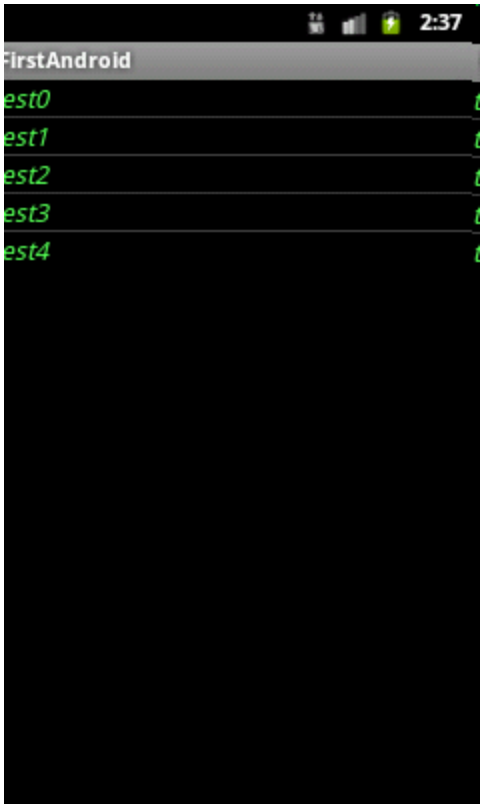
```
public class ListViewActivity extends AppCompatActivity {
    ListView nameList;
    List<HashMap<String, Object>> listData;
```

```
<ListView
    android:id="@+id/name_list"
    android:layout_width="match_parent"
    android:layout_height="match_parent"></ListView>
```

Using ListActivity

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

```java
public class SecondAndroidActivity extends ListActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.second);
        SimpleAdapter sa=new SimpleAdapter(this, initList(),
                R.layout.testframelayout, new String[] { "test_text1" },
                new int[] { R.id.test_text1 });
        setListAdapter(sa);
    }
    public ArrayList<HashMap<String, Object>> initList() {
        ArrayList<HashMap<String, Object>> al = new ArrayList<HashMap<String, Object>>();
        HashMap<String, Object> hm = null;
        for (int i = 0; i < 5; i++) {
            hm = new HashMap<String, Object>();
            hm.put("test_text1", "test" + i);
            al.add(hm);
        }
        return al;
    }
}
```




```xml
<?xml version="1.0" encoding="utf-8"?>
<TextView android:layout_width="fill_parent" style="@style/style1_text"
    xmlns:android="http://schemas.android.com/apk/res/android" android:id="@+id/test_text1"
    android:layout_height="wrap_content" android:layout_marginTop="15dp" />
```

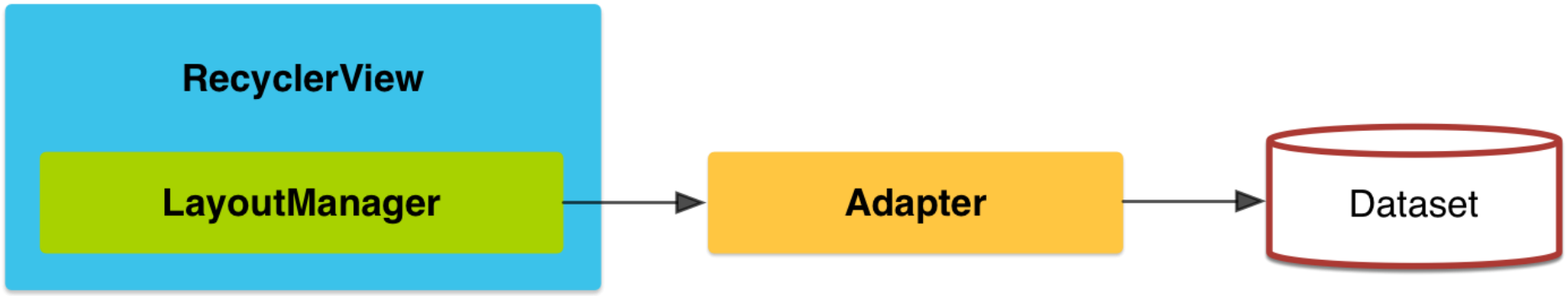
ListView Events Handling

```
public class ListViewActivity extends AppCompatActivity implements AdapterView.OnItemClickListener{
    ListView nameList;
    List<HashMap<String, Object>> listData;
    nameList.setOnItemClickListener(this);
```

```
@Override
public void onItemClick(AdapterView<?> listView, View itemView, int itemPosition, long itemId) {
    HashMap<String,Object> itemData=(HashMap<String,Object>)listView.getItemAtPosition(itemPosition);
    Toast.makeText(this,itemData.get("text_value").toString(),Toast.LENGTH_SHORT).show();
}

final class ViewHolder {...}

class CustomListAdapter extends BaseAdapter {
    Context con;
    LayoutInflater li;
    CustomListAdapter(Context context, LayoutInflater layoutInflater) {...}
    @Override
    public int getCount() { return listData.size(); }
    @Override
    public Object getItem(int position) {
        return listData.get(position);
    }
```

Android UI Components

- RecyclerView (From api 22.0)
  - A new ViewGroup that is prepared to render any adapter-based view in a similar way
  - It is supposed to be the successor of ListView and GridView
  - Working with
    - RecyclerView.Adapter
    - LayoutManager
    - ItemAnimator



RecyclerView Structure

RecyclerView

```
public class RecyclerViewScreen extends AppCompatActivity {
    private RecyclerView rv;
    private List<String> texts;
    @Override
    protected void onCreate(Bundle savedInstanceState) {...}

    private void initTexts(){...}

    public class CustomAdapter extends RecyclerView.Adapter<ViewCache>{...}

    public class ViewCache extends RecyclerView.ViewHolder{...}
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.androi
    xmlns:app="http://schemas.android.com/apk/res-au
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <android.support.v7.widget.RecyclerView
        android:id="@+id/recycler_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </android.support.v7.widget.RecyclerView>
</RelativeLayout>
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_recycler_view_screen);

    initTexts();
    rv=(RecyclerView)findViewById(R.id.recycler_list);
    rv.setAdapter(new CustomAdapter(this));
    LinearLayoutManager llm=new LinearLayoutManager(this);
    llm.setOrientation(LinearLayoutManager.HORIZONTAL);
    rv.setLayoutManager(llm);
}
```

RecyclerView.Adapter

```
public class CustomAdapter extends RecyclerView.Adapter<ViewCache>{
    private Context c;
    public CustomAdapter(Context con) { c=con; }
    @Override
    public ViewCache onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflater li=LayoutInflater.from(c);
        View itemView=li.inflate(R.layout.recycler_item,parent,false);
        ViewCache vc=new ViewCache(itemView);
        return vc;
    }
    @Override
    public void onBindViewHolder(ViewCache holder, final int position) {
            TextView tv=holder.tv;
            tv.setText(texts.get(position));
            Button delete=holder.b;
            delete.setOnClickListener((v) → {
                    Toast.makeText(c,texts.get(position),Toast.LENGTH_SHORT).show();
            });
    }
    @Override
    public int getItemCount() {
}
```


```
public class ViewCache extends RecyclerView.ViewHolder{
    public TextView tv;
    public Button b;
    public ViewCache(View itemView){
        super(itemView);
        tv=(TextView) itemView.findViewById(R.id.recycler_text);
        b=(Button)itemView.findViewById(R.id.delete);
    }
}
```

Android UI Components

- TabWidget
  - Displays a list of tab labels representing each page in the parent's tab collection
    - TabActivity
      - Being used to construct tabs effect screen, using fragment instead of it later Android3.0
    - TabHost
      - Container for a tabbed window view，containing one Label and one FrameLayout
    - TabSpec
      - A tab has a tab indicator, content, and a tag that is used to keep track of it
    - Event handling
      - Listener：OnTabChangeListener
      - Recall method：public void onTabChanged(String tag)

Using TabActivity

```
public class AndroidTabActivity extends TabActivity implements
        OnTabChangeListener {

    private String[] tab_str = new String[] { "tab1", "tab2", "tab3" };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tablayout);
        TabHost th = getTabHost();

        TabHost.TabSpec ts1, ts2, ts3;
        Resources r = getResources();
        ts1 = th.newTabSpec(tab_str[0]);
        ts1.setIndicator(r.getText(R.string.tab1));
        ts1.setContent(R.id.tab1);

        ts2 = th.newTabSpec(tab_str[1]);
        ts2.setIndicator(r.getText(R.string.tab2));
        ts2.setContent(R.id.tab2);

        ts3 = th.newTabSpec(tab_str[2]);
        ts3.setIndicator(r.getText(R.string.tab3));
        ts3.setContent(R.id.tab3);

        th.addTab(ts1);
        th.addTab(ts2);
        th.addTab(ts3);
        th.setOnTabChangedListener(this);

    }

    @Override
    public void onTabChanged(String tag) {
        Toast.makeText(this, tag, 2000).show();
    }
```


```
<?xml version="1.0" encoding="utf-8"?>
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout android:id="@+id/linearLayout1"
        android:layout_width="match_parent" android:layout_height="match_parent"
        android:orientation="vertical">
        <TabWidget android:layout_width="match_parent"
            android:layout_height="wrap_content" android:id="@android:id/tabs"></TabWidget>
        <FrameLayout android:layout_width="match_parent"
            android:layout_height="match_parent" android:id="@android:id/tabcontent">
            <LinearLayout android:layout_width="match_parent"
                android:layout_height="match_parent" android:id="@+id/tab1">
                <TextView android:layout_width="wrap_content"
                    android:layout_height="wrap_content" android:text="@string/tab1" />
            </LinearLayout>
            <LinearLayout android:layout_width="match_parent"
                android:layout_height="match_parent" android:id="@+id/tab2">
                <TextView android:layout_width="wrap_content"
                    android:layout_height="wrap_content" android:text="@string/tab2" />
            </LinearLayout>
            <LinearLayout android:layout_width="match_parent"
                android:layout_height="match_parent" android:id="@+id/tab3">
                <TextView android:layout_width="wrap_content"
                    android:layout_height="wrap_content" android:text="@string/tab3" />
            </LinearLayout>
        </FrameLayout>
    </LinearLayout>
</TabHost>
```

ViewPager + TabLayout

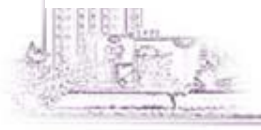
```
public class SectionsPagerAdapter extends FragmentPagerAdapter {...}

public static class PlaceholderFragment extends Fragment {...}
```

```
mSectionsPagerAdapter = new SectionsPagerAdapter(getSupportFragmentManager());

// Set up the ViewPager with the sections adapter.
mViewPager = (ViewPager) findViewById(R.id.container);
mViewPager.setAdapter(mSectionsPagerAdapter);

tabLayout=(TabLayout)findViewById(R.id.tab_layout);
tabLayout.setupWithViewPager(mViewPager);
```

```
<android.support.v4.view.ViewPager
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    app:layout_behavior="android.support.design.widget.AppBarLayout$ScrollingVie..." />

<android.support.design.widget.TabLayout
    android:id="@+id/tab_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingBottom="16dp"  app:tabGravity="fill"></android.support.design.widget.TabLayout>
```

Android UI Components

- Dialog
  - A dialog is usually a small window that appears in front of the current Activity
  - Types
    - AlertDialog
      - AlertDialog.Builder：being used to construct dialog
      - DialogInterface.OnClickListener：being used to handle dialog events
      - dismiss()/cancel()：closing a dialog
    - ProgressDialog
      - Being used to construct a progress dialog
    - DatePickerDialog
      - Being used to construct a date dialog
    - TimePickerDialog
      - Being used to construct a time dialog



Android UI Components

- onCreateDialog(int id)：this method is in the activity for constructing a dialog
- showDialog (int id)：this method is in the activity for showing a dialog
- Custom dialog
  - If you want a customized design for a dialog, you can create your own layout for the dialog window with layout and widget elements

Using AlertDialog before 3.0


```
@Override
protected Dialog onCreateDialog(int id) {
    Dialog dialog = null;
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    switch (id) {
    case ERROR_DIALOG: {
        builder.setTitle(R.string.error_title);
        builder.setMessage(R.string.error_message);
        builder.setPositiveButton(R.string.confirm,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });
        dialog = builder.create();
        break;
    }
    case WARNING_DIALOG: {
        builder.setTitle(R.string.warning_title);
        builder.setMessage(R.string.warning_message);
        builder.setPositiveButton(R.string.confirm,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });
        builder.setNegativeButton(R.string.cancel,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });
        dialog = builder.create();
        break;
    }
    }
    return dialog;
}
```

```
showDialog(ERROR_DIALOG);
```

Adding checkboxes and radio buttons to a dialog before 3.0

```
<string name="level1">最难</string>
<string name="level2">中等难度</string>
<string name="level3">容易</string>
```

```
items = new String[] { r.getString(R.string.level1),
        r.getString(R.string.level2), r.getString(R.string.level3) };
```

```
builder.setTitle(R.string.pick_item);
builder.setSingleChoiceItems(items, -1,
        new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });
dialog = builder.create();
```

The value -1 of Index shows no default selected item

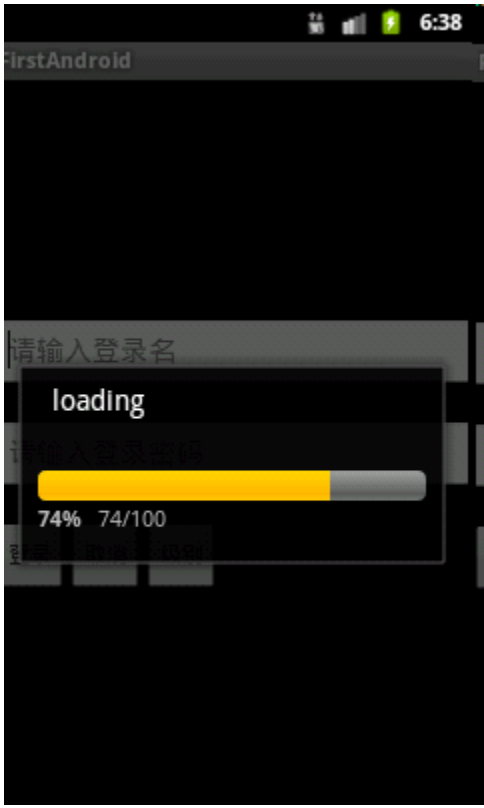
Using ProgressDialog before 3.0

```
showDialog(PROGRESS_DIALOG);
pd.setProgress(0);
PThread pt = new PThread(h);
pt.start();
```

```
final Handler h = new Handler()
```

```
class PThread extends Thread
```

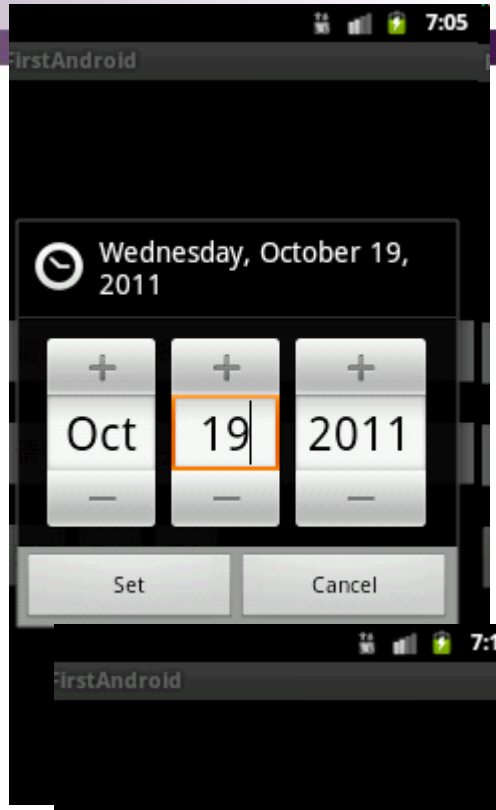
```
ProgressDialog pd;

@Override
protected Dialog onCreateDialog(int id) {
    Dialog dialog = null;
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    switch (id) {
    case PROGRESS_DIALOG: {
        pd = new ProgressDialog(this);
        pd.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
        pd.setMessage("loading");
        return pd;
    }
    default: {
        builder.setTitle(R.string.pick_item);
        builder.setSingleChoiceItems(items, -1,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });
        dialog = builder.create();
        break;
    }
    }
    return dialog;
}
```



Constructing date and time dialog before 3.0

```
int day, month, year, hour, minute;
```

```
Calendar c = Calendar.getInstance();

day = c.get(Calendar.DAY_OF_MONTH);
month = c.get(Calendar.MONTH);
year = c.get(Calendar.YEAR);
hour = c.get(Calendar.HOUR_OF_DAY);
minute = c.get(Calendar.MINUTE);
```



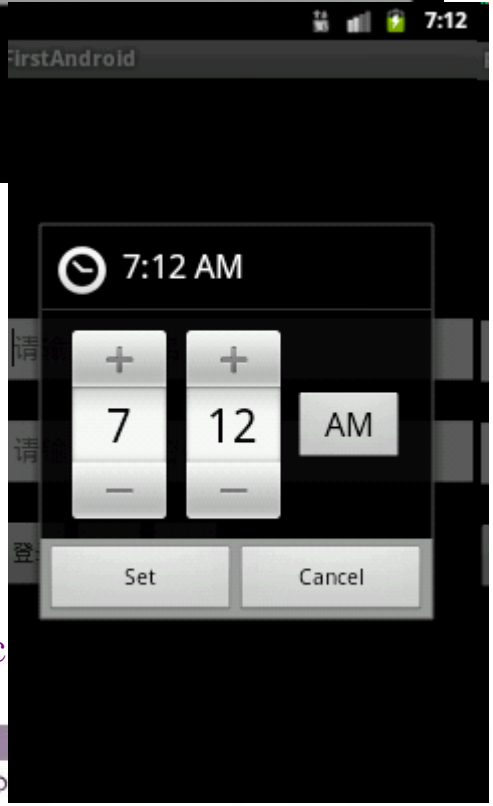

```
switch (id) {
case TIME_DIALOG: {
    return new TimePickerDialog(this, null, hour, minute, false);
}
case DATE_DIALOG: {
    return new DatePickerDialog(this, null, year, month, day);
}
```

```
showDialog(DATE_DIALOG);
```

```
showDialog(TIME_DIALOG);
```

Setting listener

This parameter means whether it shows time in 24 hours



Custom Dialog before 3.0

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="match_parent">
    <TextView android:layout_width="wrap_content" style="@style/style1_text"
        android:layout_height="wrap_content" android:id="@+id/list_text"
        android:layout_marginTop="5dp" />
    <ImageView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:id="@+id/img"
        android:layout_toRightOf="@id/list_text" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:id="@+id/confirm"
        android:text="@string/confirm" android:layout_below="@id/img" />
</RelativeLayout>
```

```java
case CUSTOM_DIALOG: {
    dialog = new Dialog(this);
    dialog.setContentView(R.layout.customdialog);
    dialog.setTitle(R.string.custom_dialog);
    TextView text = (TextView) dialog.findViewById(R.id.list_text);
    text.setText(R.string.custom_dialog);
    ImageView iv = (ImageView) dialog.findViewById(R.id.img);
    iv.setImageResource(R.drawable.icon);
    Button finishB = (Button) dialog.findViewById(R.id.confirm);
    finishB.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            dismissDialog(CUSTOM_DIALOG);
        }
    });
    break;
}
```

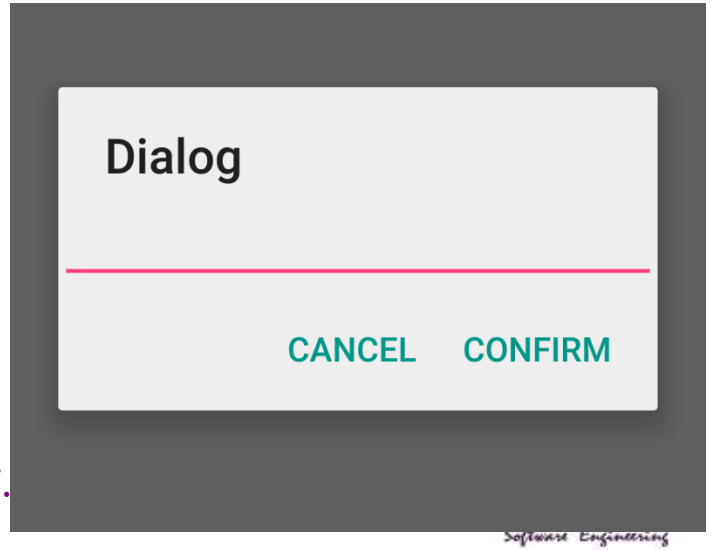
Android UI Components

- Since Android 3.0 (API level 11), Activity.onCreateDialog() is deprecated to use
- And, you should use a DialogFragment as a container for your dialog
- Using DialogFragment to manage the dialog ensures that it correctly handles lifecycle events

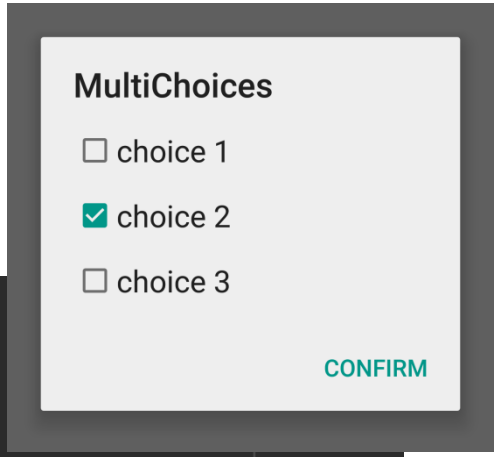
DialogFragment

```
public class DialogTestActivity extends FragmentActivity implements NormalDialog.NormalDialogListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {...}

    public void showDialog(View v) {
        NormalDialog normalDialog = new NormalDialog();
        normalDialog.show(getFragmentManager(), "dialog");
    }
    @Override
    public void onPositiveButtonClick(DialogFragment dialogFragment) {
        EditText passText=(EditText)dialogFragment.getDialog().findViewById(R.id.password_input);
        Toast.makeText(this, passText.getText(), Toast.LENGTH_SHORT).show();
        dialogFragment.dismiss();
    }
    @Override
    public void onNegtiveButtonClick(DialogFragment dialogFragment) {...}
}
```

```
public class NormalDialog extends

    NormalDialogListener listener
    @Override
    public Dialog onCreateDialog
        AlertDialog.Builder ab =
        LayoutInflater layoutInf
        ab.setTitle("Dialog");
        ab.setView(layoutInflater.inflate(R.layout.dialog_layout,null));
        ab.setPositiveButton("Confirm", (dialog, which) -> {
                listener.onPositiveButtonClick(NormalDialog.this);
        });
        ab.setNegativeButton("Cancel", (dialog, which) -> {
                listener.onNegtiveButtonClick(NormalDialog.this);
        });
        return ab.create();
    }
    @Override
    public void onAttach(Activity activity) {...}

    public interface NormalDialogListener {...}

}
```



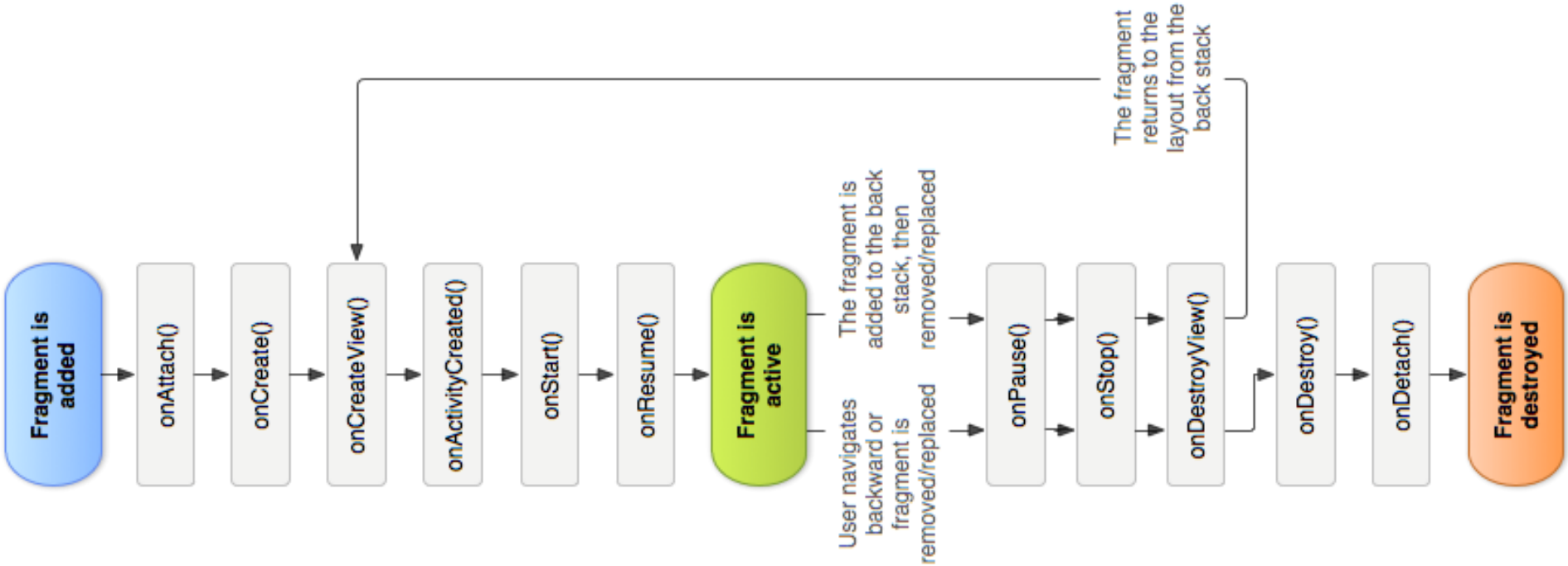
MultiChoices Dialog




```
public class MultiChoiceDialog extends DialogFragment{
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());
        builder.setTitle(R.string.mul_title);
        builder.setPositiveButton(R.string.confirm, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        });
        builder.setMultiChoiceItems(R.array.choices, null, new DialogInterface.OnMultiChoiceClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which, boolean isChecked) {
                Toast.makeText(getActivity(),which+" is checked:"+isChecked,Toast.LENGTH_SHORT).show();
            }
        });
        return builder.create();
    }
}
```

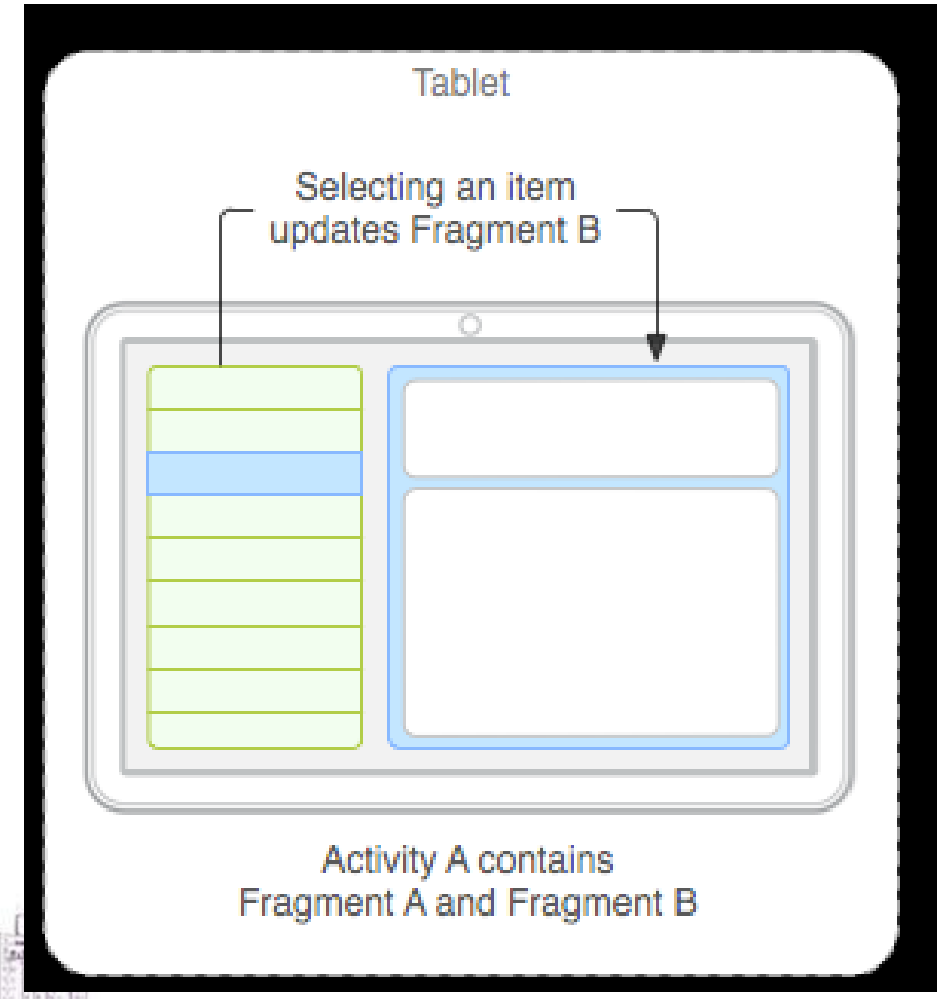
Android UI Components

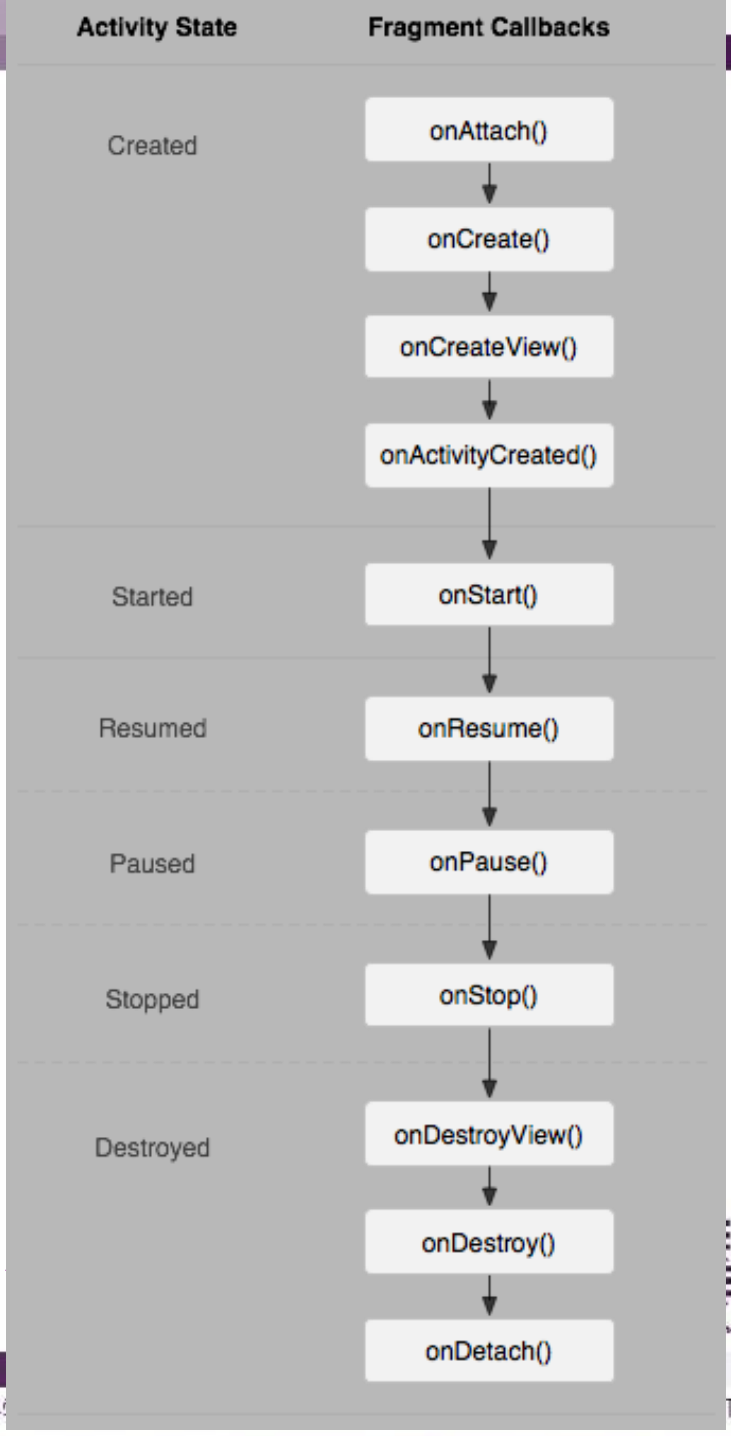
- New Features Added After Android 3.0
  - Fragment
    - A Fragment represents a behavior or a portion of user interface in an Activity
    - A fragment must always be embedded in an activity and the fragment's lifecycle is directly affected by the host activity's lifecycle
    - The lifecycle of fragment, as the following diagram



The lifecycle of a fragment (while its activity is running)

Fragment VS Activity

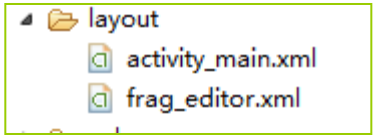




Android UI Components

- Fragment
  - FragmentActivity is base class for activities that want to use the support-based Fragment and Loader APIs
  - Fragment development needs support of pacakge “ android.support.v4.app”
  - If you got the exception like the following, you need to check your support package according two rules above
    - Caused by: android.view.InflateException: Binary XML file line #14: Error inflating class fragment

Fragment




```
public class MainActivity extends Activity implements OnCl
    private Button remove_show;
    private Fragment test_frag;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        remove_show = (Button) findViewById(R.id.remove_sh
        remove_show.setOnClickListener(this);
        test_frag=new EditorFragment();
    }

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        if (remove_show.getText().equals("remove")) {
            FragmentManager fm = getFragmentManager();
            FragmentTransaction fTransaction = fm.beginTransaction();
            fTransaction.remove(test_frag);
            fTransaction.commit();
            remove_show.setText("show");
        } else {
            FragmentManager fm = getFragmentManager();
            FragmentTransaction fTransaction = fm.beginTransaction();
            fTransaction.add(R.id.ui_container,test_frag);
            fTransaction.commit();
            remove_show.setText("remove");
        }

    }
}
```

```
public class EditorFragment extends Fragment implements OnClickListener {
    private Button testButton;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View rootView = inflater
                .inflate(R.layout.frag_editor, container, false);
        testButton = (Button) (rootView.findViewById(R.id.test_button));
        testButton.setOnClickListener(this);
        return rootView;
    }

    @Override
    public void onClick(View v) {
        Log.v("test_button", "test_button clicked");
    }

}
```

://staff.ustc.edu.cn/~waterzhj

Android UI Components

- Loaders
  - Loaders make it easy to asynchronously load data in an activity or fragment
  - The LoaderManager manages one or more Loader instances within an Activity or Fragment
  - There is only one LoaderManager per activity or fragment
  - **LoaderManager.LoaderCallbacks is** a callback interface for a client to interact with the LoaderManager

Loader

```
public static class CursorLoaderListFragment extends ListFragment
        implements OnQueryTextListener, LoaderManager.LoaderCallbacks<Cursor> {
```

```
getLoaderManager().initLoader(0, null, this);
```

```
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
    // This is called when a new Loader needs to be created.  This
    // sample only has one Loader, so we don't care about the ID.
    // First, pick the base URI to use depending on whether we are
    // currently filtering.
    Uri baseUri;
    if (mCurFilter != null) {
        baseUri = Uri.withAppendedPath(Contacts.CONTENT_FILTER_URI,
                Uri.encode(mCurFilter));
    } else {
        baseUri = Contacts.CONTENT_URI;
    }

    // Now create and return a CursorLoader that will take care of
    // creating a Cursor for the data being displayed.
    String select = "((" + Contacts.DISPLAY_NAME + " NOTNULL) AND ("
            + Contacts.HAS_PHONE_NUMBER + "=1) AND ("
            + Contacts.DISPLAY_NAME + " != '' ))";
    return new CursorLoader(getActivity(), baseUri,
            CONTACTS_SUMMARY_PROJECTION, select, null,
            Contacts.DISPLAY_NAME + " COLLATE LOCALIZED ASC");
}

public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
    // Swap the new cursor in.  (The framework will take care of closing the
    // old cursor once we return.)
    mAdapter.swapCursor(data);
}

public void onLoaderReset(Loader<Cursor> loader) {
    // This is called when the last Cursor provided to onLoadFinished()
    // above is about to be closed.  We need to make sure we are no
    // longer using it.
    mAdapter.swapCursor(null);
}
```

Android UI Components

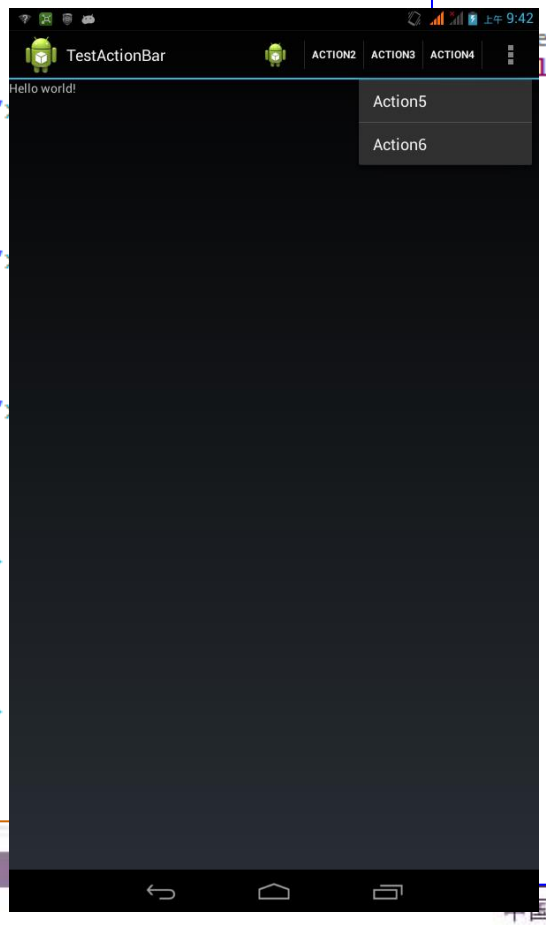
- Action Bar
  - The action bar is a window feature that identifies the application and user location, and provides user actions and navigation modes
  - It provides a dedicated space for identifying the application brand and user location
  - It provides consistent navigation and view refinement across different applications
  - Make key actions for the activity prominent and accessible to the user in a predictable way

The action bar is split into four different functional areas that apply to most apps

(1.App icon 2.View control 3. Action buttons 4. Action overflow)

ActionBar

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/item1"
        android:icon="@drawable/ic_launcher"
        android:title="Action1"
        android:showAsAction="always">
    </item>
    <item
        android:id="@+id/item2"
        android:title="Action2"
        android:showAsAction="ifRoom">
    </item>
    <item
        android:id="@+id/item3"
        android:title="Action3"
        android:showAsAction="ifRoom">
    </item>
    <item
        android:id="@+id/item4"
        android:title="Action4"
        android:showAsAction="ifRoom">
    </item>
    <item
        android:id="@+id/item5"
        android:title="Action5"
        android:showAsAction="never">
    </item>
    <item
        android:id="@+id/item6"
        android:title="Action6"
        android:showAsAction="never">
    </item>

</menu>
```




```java
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // TODO Auto-generated method stub
        getMenuInflater().inflate(R.menu.action_menu, menu);
        return true;
    }

    erride
    lic boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    switch (item.getItemId()) {
    case R.id.item1: {
        Log.v("action1", "action1");
        break;
    }
    case R.id.item2: {
        Log.v("action2", "action2");
        break;
    }
    case R.id.item3: {
        Log.v("action3", "action3");
        break;
    }
    case R.id.item4: {
        Log.v("action4", "action4");
        break;
    }
    default: {
        Log.v("other actions", "other actions!!");
    }
    }
    return true;
```

Android UI Components

- **Building Custom Components**
  - If none of the prebuilt widgets or layouts meets your needs, you can create your own View subclass
  - If you only need to make small adjustments to an existing widget or layout, you can simply subclass the widget or layout and override its methods

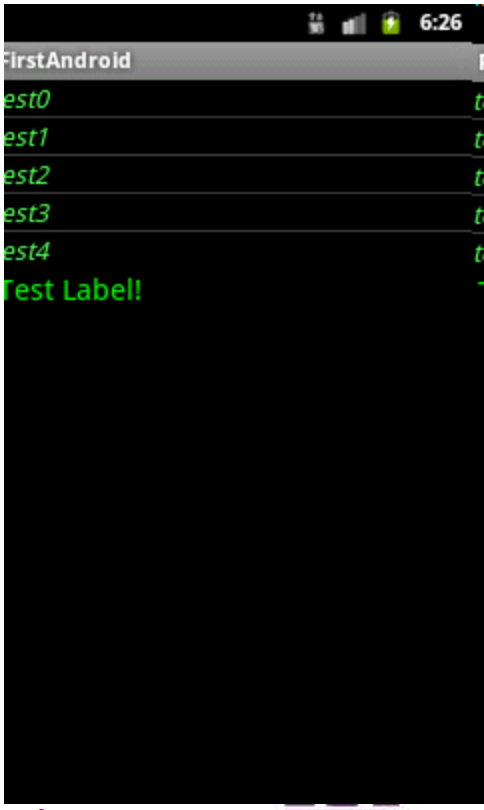
Android UI Components

- The most generic view you can extend is View, so you will usually start by extending this to create your new super component
  - override onMeasure() and onDraw()
  - onDraw()
    - delivers you a Canvas upon which you can implement anything you want, like 2D graphics, styled text, etc.
  - onMeasure()
    - is a critical piece of the rendering contract between your component and its container

Custom Components

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:label="http://schemas.android.com/apk/res/water.activity"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ListView android:id="@+id/list1" android:layout_width="fill_parent"
        android:layout_height="wrap_content"></ListView>
    <water.activity.LabelView
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        label:textValue="Test Label!" label:textSize="20dp"></water.activity.LabelView>
</LinearLayout>
```

```java
public class LabelView extends View
{
    public LabelView(Context context, AttributeSet attrs) {
        super(context, attrs);
        initLabel();

        TypedArray ta = context
                .obtainStyledAttributes(attrs, R.styleable.Label);
        String textValue = ta.getString(R.styleable.Label_textValue);
        if (textValue != null) {
            setTextValue(textValue);
        }
        setTextColor(ta.getColor(R.styleable.Label_textColor, 0xFF00FF00));
        int textSize = ta.getDimensionPixelSize(R.styleable.Label_textSize, 0);
        if (textSize > 0) {
            setTextSize(textSize);
        }
        ta.recycle();
    }
```



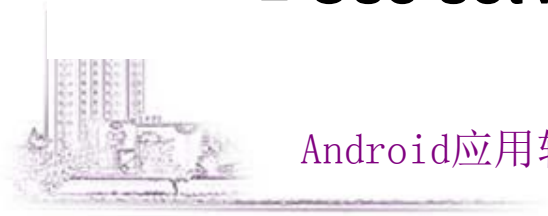

```xml
<declare-styleable name="Label">
    <attr name="textValue" format="string" />
    <attr name="textSize" format="dimension" />
    <attr name="textColor" format="color" />
</declare-styleable>
```

Android UI Components

- Notifying the User
  - Using Toast Notifications/SnackBar (From api 22.2.0)
    - A toast notification is a message that pops up on the surface of the window
    - The notification automatically fades in and out, and does not accept interaction events
  - Using Status Bar Notifications
    - A status bar notification adds an icon to the system's status bar and an expanded message in the "Notifications" window
    - You can also configure the notification to alert the user with a sound, a vibration, and flashing lights on the device

Android UI Components

- Toast Notifications
  - Instantiate a Toast object with one of the makeText() method
  - Display the toast notification with show()
  - Position the toast notification with the setGravity(int, int, int)
  - Create a customized layout for your toast notification
    - Use setView(View) method



Using Toast Notification and Customing a Toast View

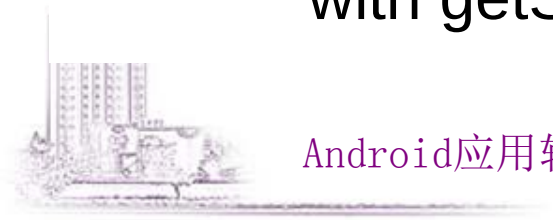
```
Toast t=Toast.makeText(this, R.string.error_message, 2000);
t.setGravity(Gravity.CENTER_HORIZONTAL|Gravity.CENTER_VERTICAL, 0, 0);
t.show();
```

```
LayoutInflater li = getLayoutInflater();
View toastView = li.inflate(R.layout.customtoast,
        (ViewGroup) findViewById(R.id.toast_layout));

TextView tv = (TextView) toastView.findViewById(R.id.list_text);
tv.setText(R.string.error_message);
ImageView iv = (ImageView) toastView.findViewById(R.id.img);
iv.setImageResource(R.drawable.icon);

Toast t = new Toast(this);
t.setDuration(2000);
t.setView(toastView);
t.show();
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:id="@+id/toast_layout"
    android:background="#777777">
    <TextView android:layout_width="wrap_content" style="@style/style1_text"
        android:layout_height="wrap_content" android:id="@+id/list_text"
        android:layout_marginTop="5dp" />
    <ImageView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:id="@+id/img" />
</LinearLayout>
```

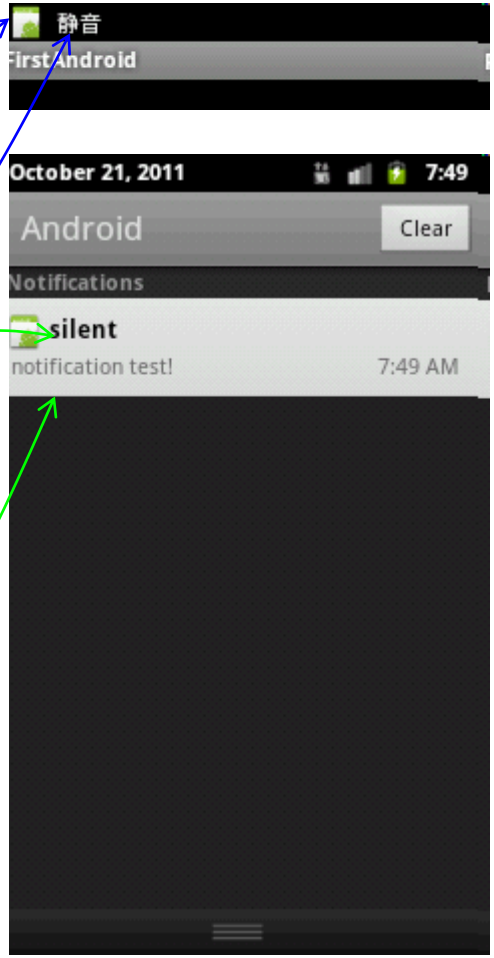
Android UI Components

- Status Bar Notifications
  - Notification
    - defines the properties of your status bar notification
  - NotificationManager
    - is an Android system service that executes and manages all Notifications
    - retrieve a reference to the NotificationManager with getSystemService()



Using Status Bar Notifications before 3.0

```
NotificationManager nm = (NotificationManager)
                         getSystemService(Context.NOTIFICATION_SERVICE);

Notification noti = new Notification(R.drawable.icon,
        getResources().getString(R.string.silent),
        System.currentTimeMillis());

Intent intent = new Intent(this, AndroidTabActivity.class);
PendingIntent pi = PendingIntent.getActivity(this, 0, intent, 0);
noti.setLatestEventInfo(this, "silent", "notification test!", pi);

nm.notify(1, noti);
```




```
notification.sound = Uri.parse("file:///sdcard/notification/ringer.mp3");
```

```
notification.defaults |= Notification.DEFAULT_VIBRATE;
```

You can add sound or vibrate to notification, but it needs corresponding permission

Using Notification.Builder after 3.0

```
Notification.Builder notificationBuilder = new Notification.Builder(this);
notificationBuilder.setContentTitle(getString(R.string.attention));
notificationBuilder.setContentText(getString(R.string.notification_message));
notificationBuilder.setSmallIcon(R.drawable.delete);

Intent pageOpen = new Intent(this, DialogTestActivity.class);
PendingIntent nextPage = PendingIntent.getActivity(this, REQUEST_CODE, pageOpen, PendingIntent.FLAG_UPDATE_CURRENT);

notificationBuilder.setContentIntent(nextPage);
notificationBuilder.setAutoCancel(true);

Notification notification = notificationBuilder.build();
notification.defaults = Notification.DEFAULT_VIBRATE;

NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify(NOTIFICATION_ID, notification);
```

Notification with progress

```
final Notification.Builder notificationBuilder = new Notification.Builder(this);
notificationBuilder.setContentTitle(getString(R.string.attention));
notificationBuilder.setContentText(getString(R.string.notification_message));
notificationBuilder.setSmallIcon(R.drawable.delete);

final NotificationManager notificationManager = (NotificationManager) getSystemService

(Thread) run() -> {
        for(int progress=0;progress<=100;progress+=3){
            notificationBuilder.setProgress(100,progress,false);
            notificationManager.notify(NOTIFICATION_ID,notificationBuilder.build());
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        notificationBuilder.setContentText(getString(R.string.completed));
        notificationBuilder.setProgress(0,0,false);
        notificationManager.notify(NOTIFICATION_ID,notificationBuilder.build());
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        notificationManager.cancel(NOTIFICATION_ID);
}.start();
```

Define Notification Layout


```
Notification.Builder notificationBuilder = new Notification.Builder(this);
notificationBuilder.setSmallIcon(R.mipmap.ic_launcher);

Intent serviceIntent = new Intent(this, MyService.class);
PendingIntent pendingIntent = PendingIntent.getService(this, REQUEST_CODE, serviceIntent, PendingIntent.FLAG_UPDATE_CURRENT);

RemoteViews remoteViews = new RemoteViews(getPackageName(), R.layout.notification_layout);
remoteViews.setOnClickPendingIntent(R.id.cancel, pendingIntent);
remoteViews.setTextColor(R.id.cancel,R.color.colorPrimaryDark);

Notification notification = notificationBuilder.build();
notification.defaults = Notification.DEFAULT_VIBRATE;
notification.contentView=remoteViews;

NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify(NOTIFICATION_ID, notification);
```

Conclusions

- Prototype Design Process
- Tools
- UI Design Patterns
- Android UI Components
- Conclusions