Android App Dev Fundamentals

ZHENG Lingxiang

lxzheng@xmu.edu.cn

Agenda

- Android Program Anatomy by Hello World
- Some fundamental concepts in the Android system

Getting Started with Android Studio

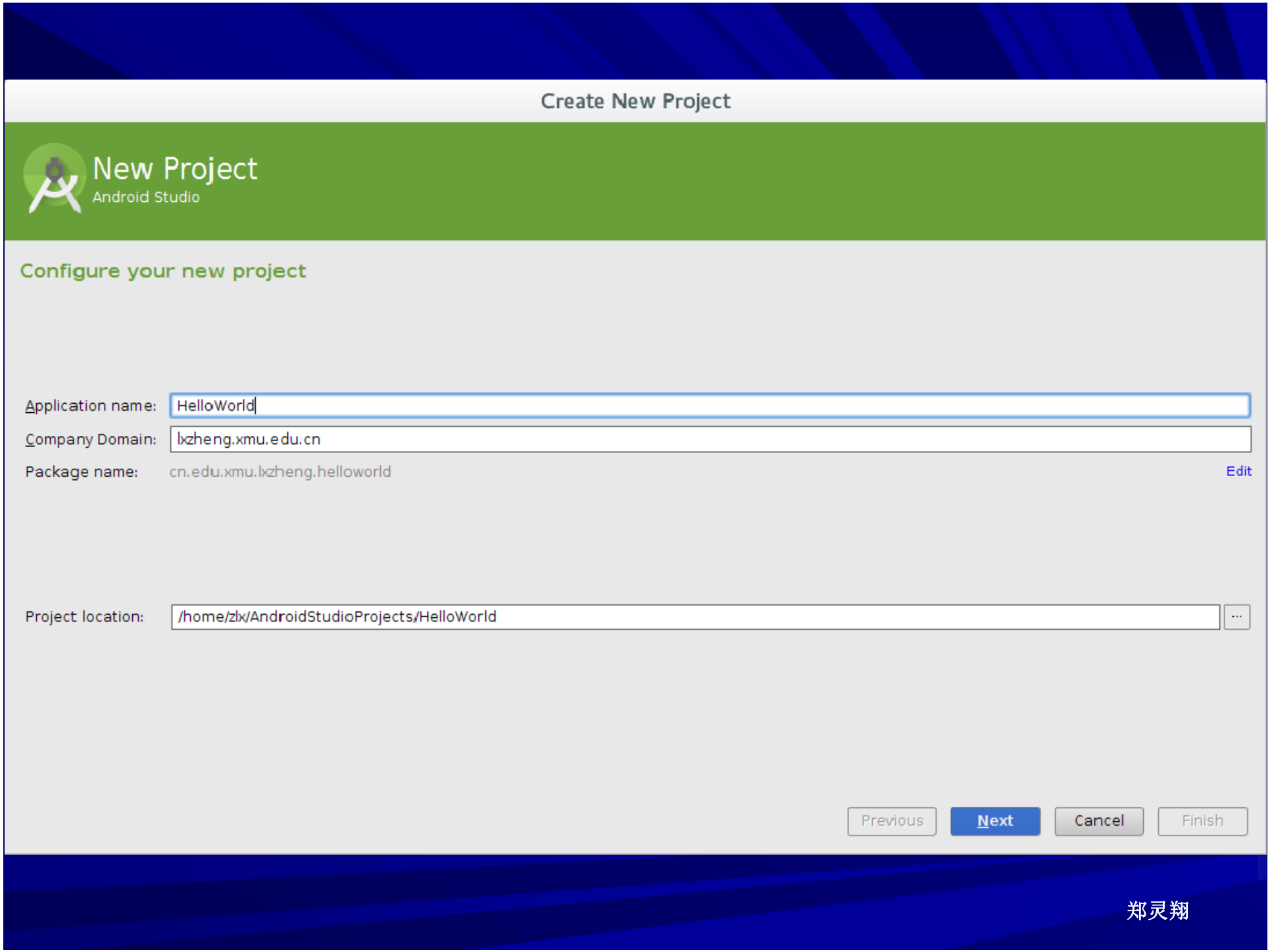
- a new Android development environment based on IntelliJ IDEA
- Similar to Eclipse with the ADT Plugin
- provides integrated Android developer tools for development and debugging

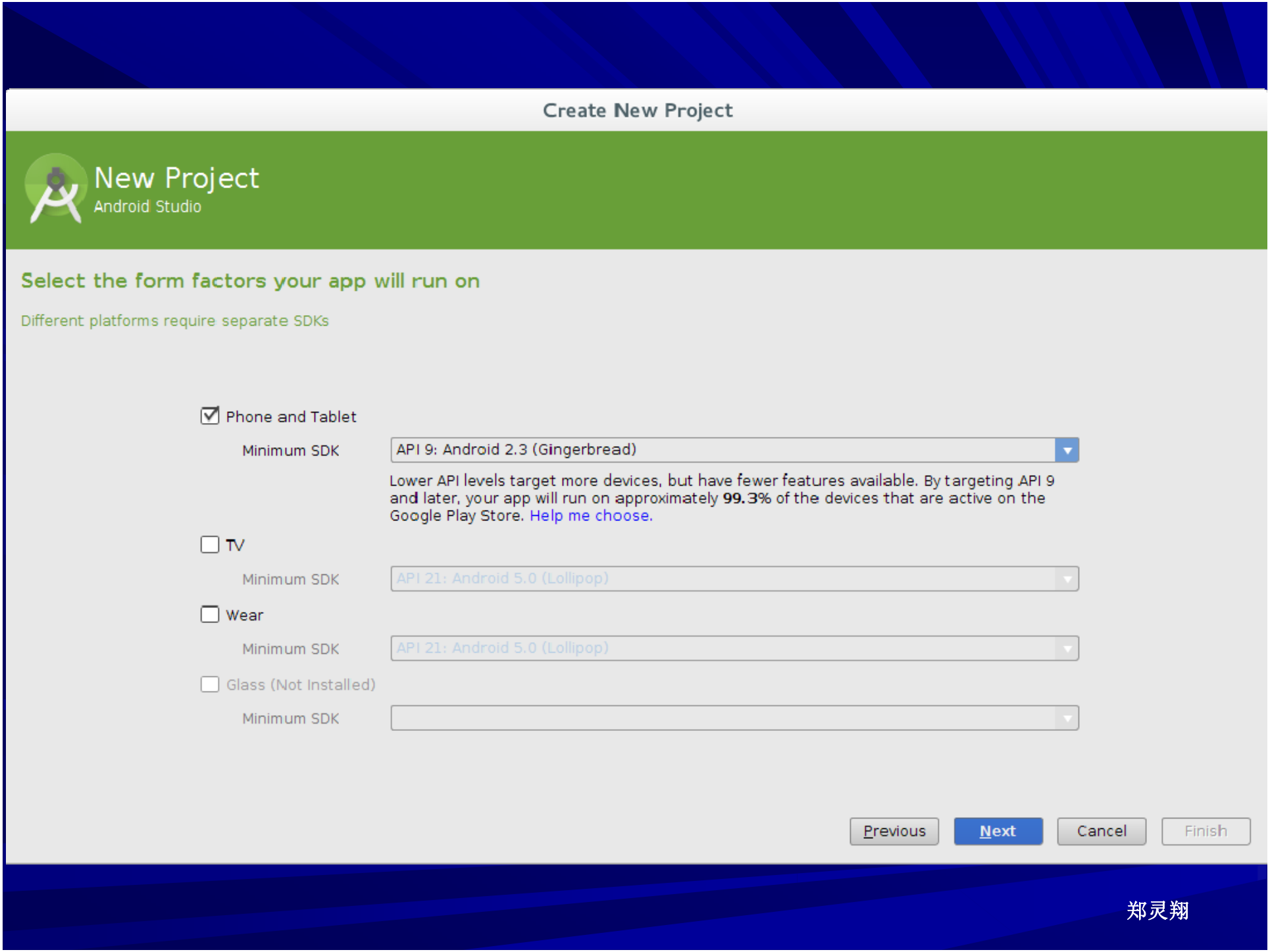
Android Studio offers

- Gradle-based build support.
- Android-specific refactoring and quick fixes.
- Lint tools to catch performance, usability, version compatibility and other problems.
- ProGuard and app-signing capabilities.
- Template-based wizards to create common Android designs and components.
- A rich layout editor that allows you to drag-and-drop UI components, preview layouts on multiple screen configurations, and much more.

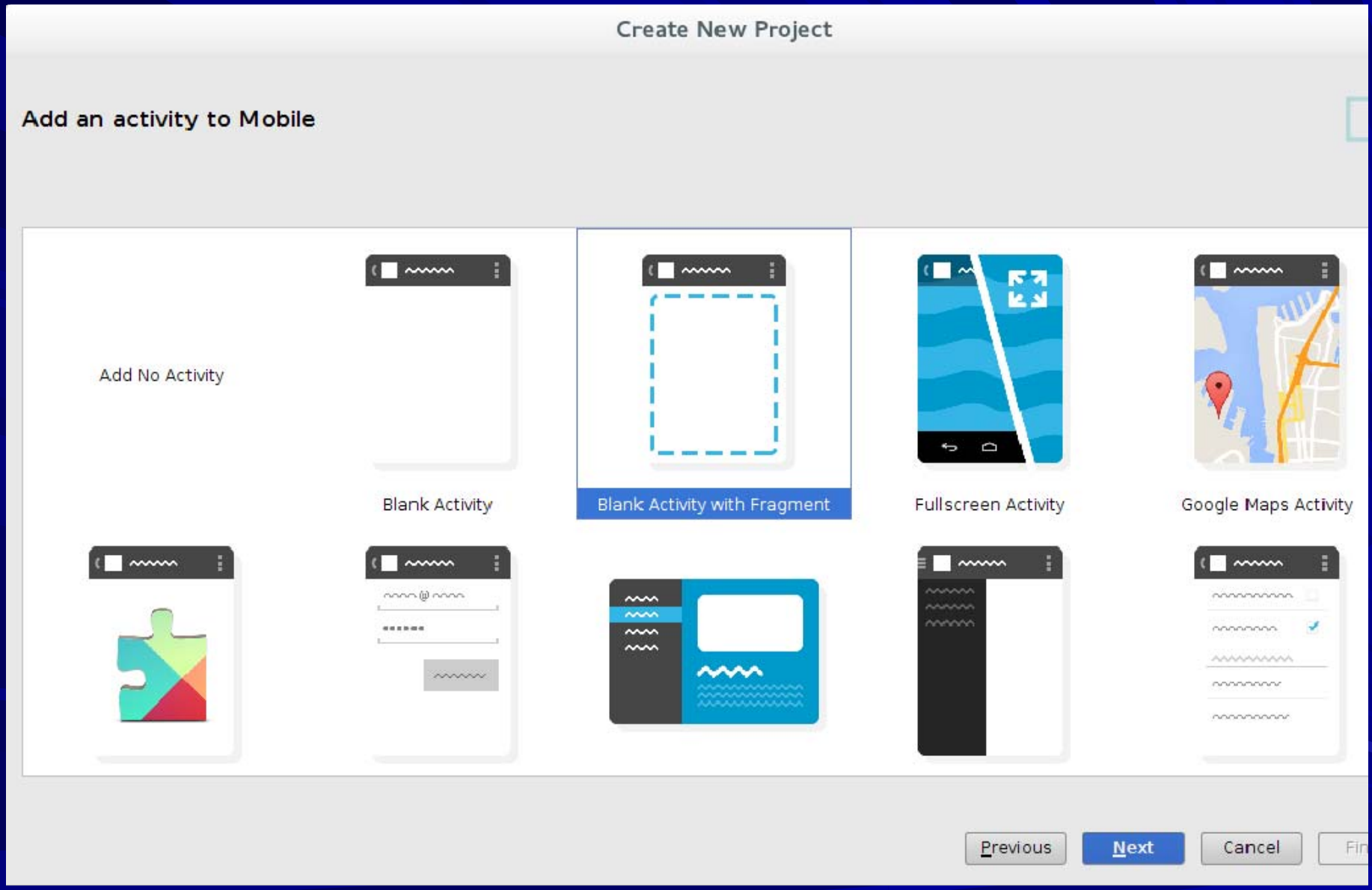
HelloWorld Example

- A Step by Step Guilding


Create New Project
New Project
Android Studio
Configure your new project
Application name:
HelloWorld
Company Domain:
lxzheng.xmu.edu.cn
Package name:
cn.edu.xmu.lxzheng.helloworld
Edit
Project location:
/home/zlx/AndroidStudioProjects/HelloWorld
Previous
Next
Cancel
Finish
郑灵翔


Create New Project
New Project
Android Studio
Select the form factors your app will run on
Different platforms require separate SDKs
Phone and Tablet
Minimum SDK
API 9: Android 2.3 (Gingerbread)
Lower API levels target more devices, but have fewer features available. By targeting API 9 and later, your app will run on approximately 99.3% of the devices that are active on the Google Play Store. Help me choose.
TV
Minimum SDK
API 21: Android 5.0 (Lollipop)
Wear
Minimum SDK
API 21: Android 5.0 (Lollipop)
Glass (Not Installed)
Minimum SDK
Previous
Next
Cancel
Finish

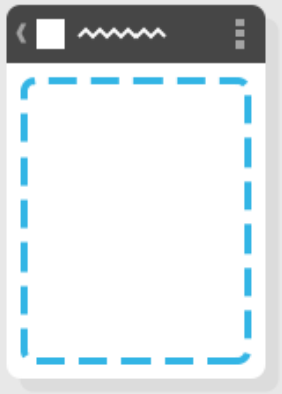
郑灵翔


Create New Project
Add an activity to Mobile
Add No Activity
Blank Activity
Blank Activity with Fragment
Fullscreen Activity
Google Maps Activity
Previous
Next
Cancel

郑灵翔

Create New Project

Choose options for your new file



Blank Activity with Fragment

Creates a new blank activity, with an action bar and a contained Fragment.

Activity Name:	MainActivity
Layout Name:	activity_main
Fragment Layout Name:	fragment_main
Title:	MainActivity
Menu Resource Name:	menu_main

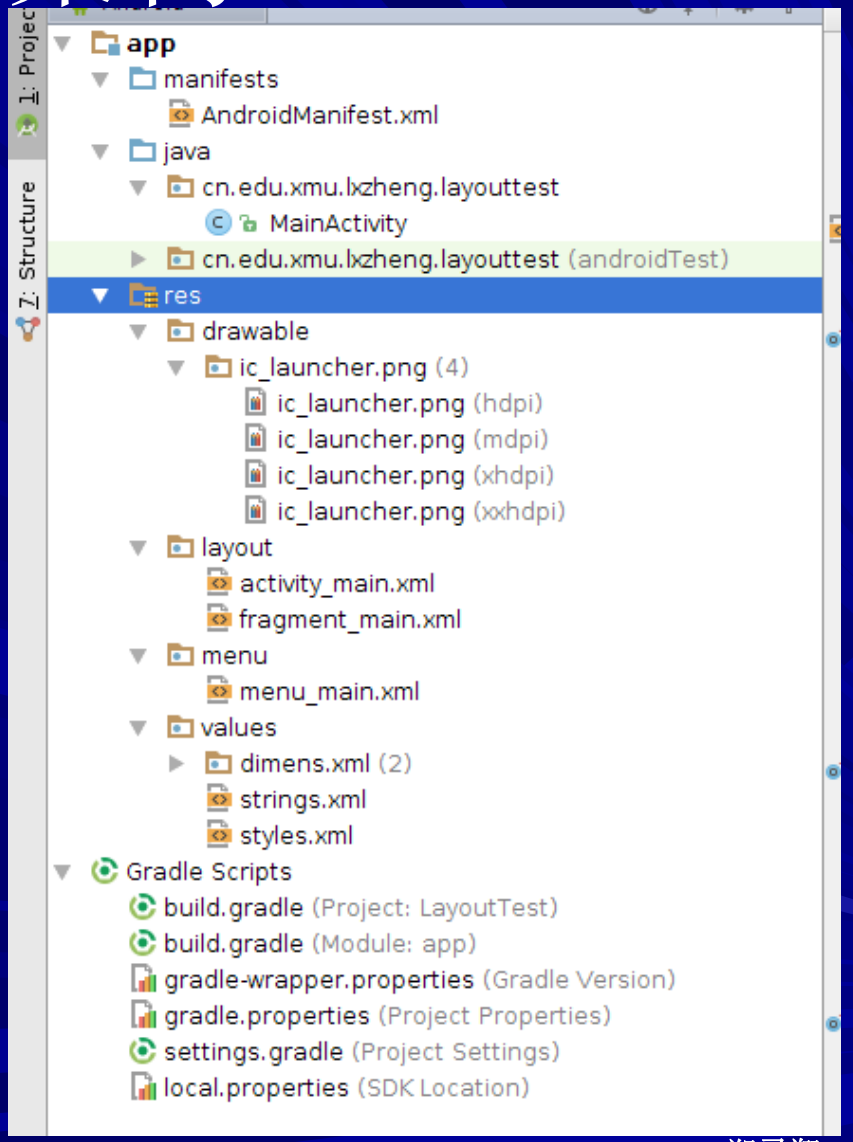
The name of the activity class to create

Previous | Next | Cancel | Finish

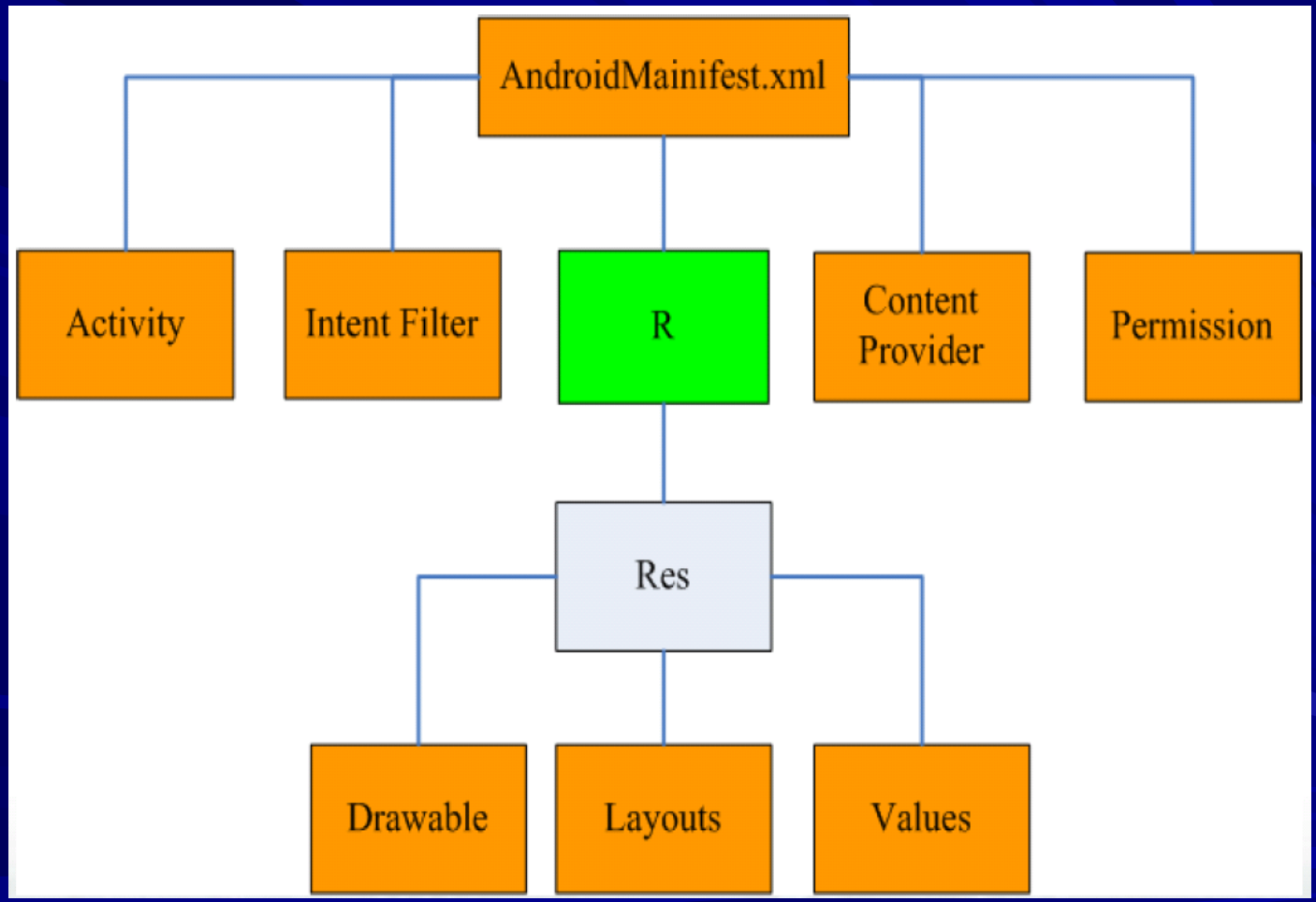
郑灵翔

Android程序项目架构

- maifests
  - AndroidManifest.xml
- java
  - MainActivity.java
  - Android Test
- res——>R.java
  - drawable
  - layout
    - activity_main.xml
  - values
    - strings.xml
- Gradle Scripts
  - build.gradle



App Framwork



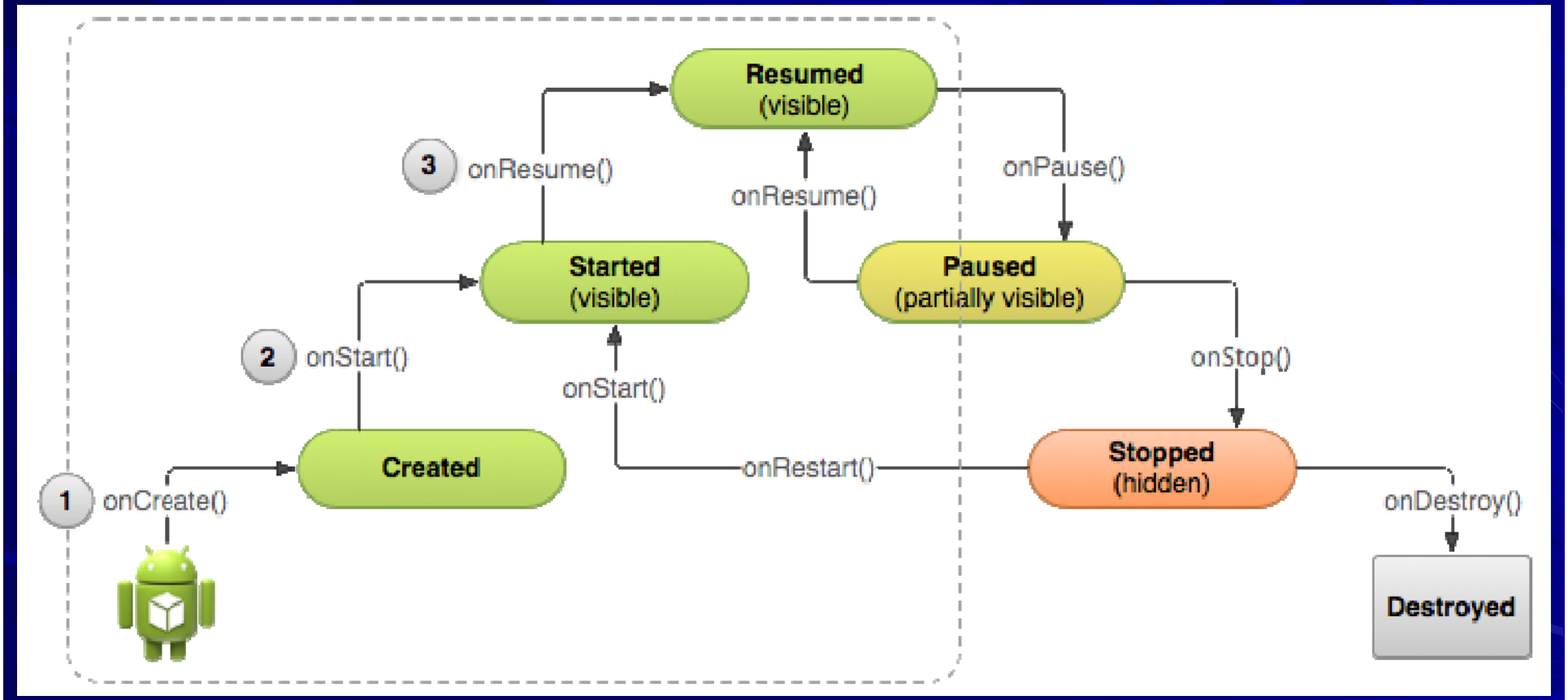
Basic components of the application

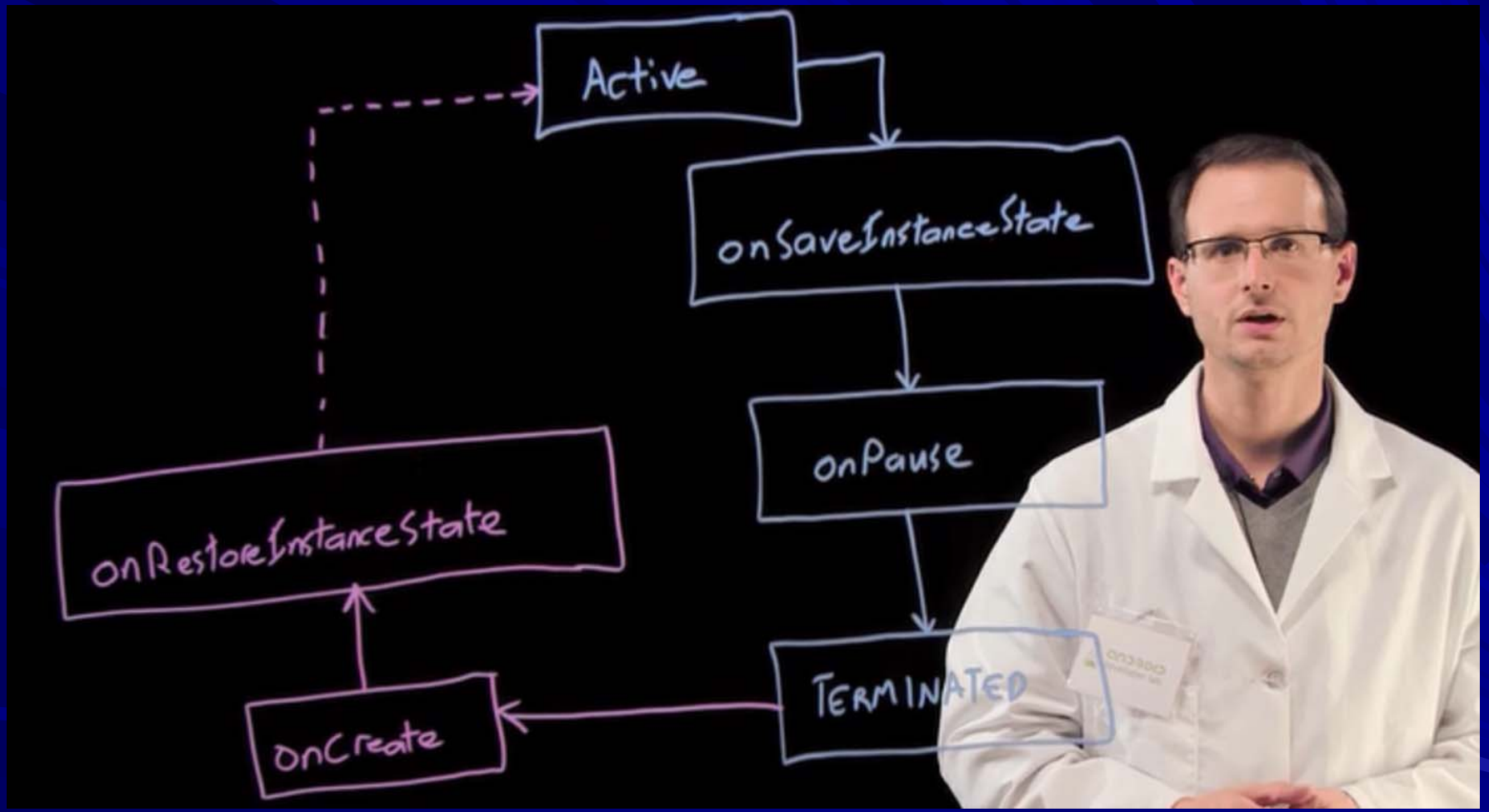
Activities	UI component typically corresponding to one screen.
BroadcastReceivers	Respond to broadcast Intents.
Services	Faceless tasks that run in the background.
ContentProviders	Enable applications to share data.

Activities

- Typically correspond to one screen in a UI
- But, they can:
  - be faceless
  - be in a floating window
  - return a value

Activity Lifecycle

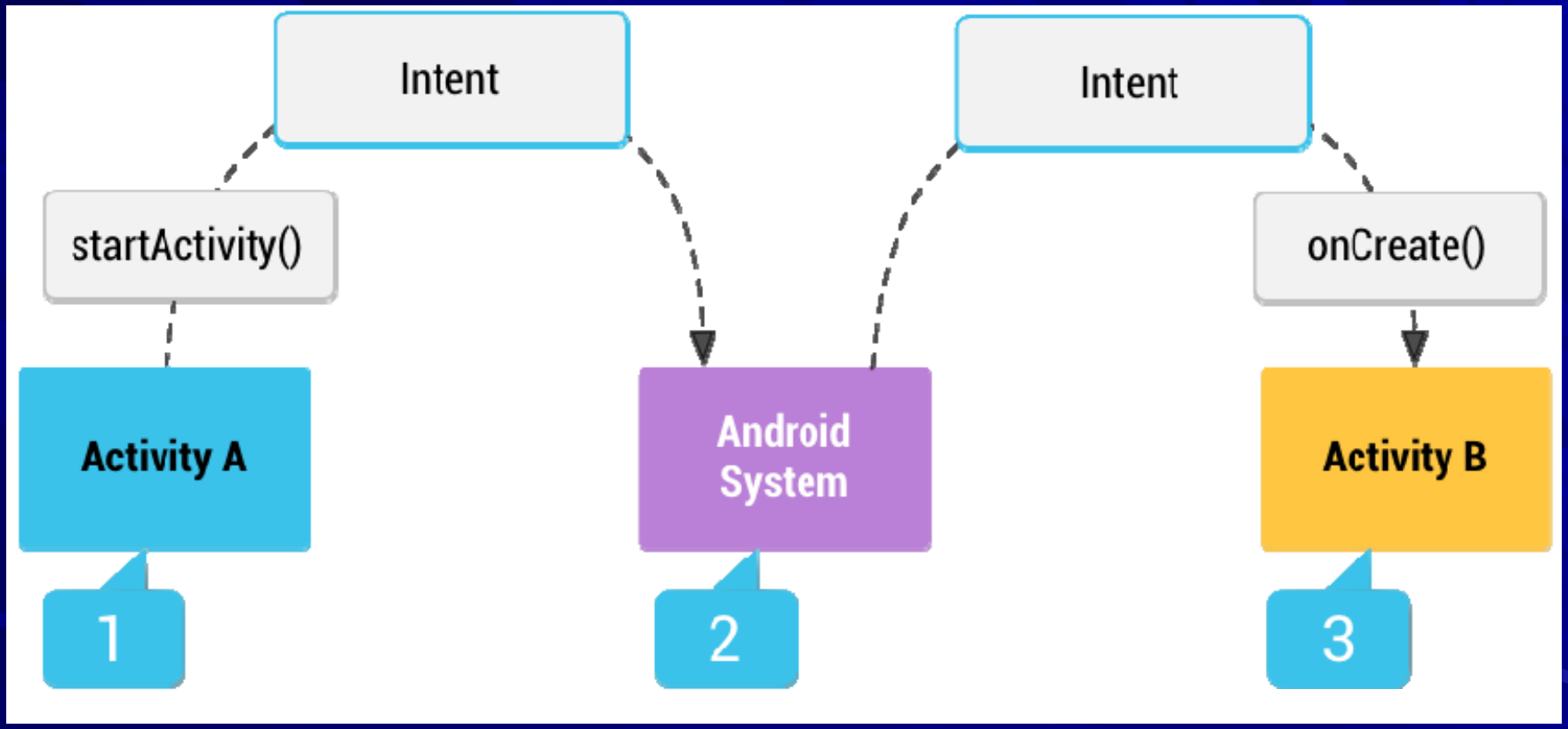



Active
onSaveInstanceState
onPause
TERMINATED
onCreate
onRestoreInstanceState

BroadcastReceivers

- Components designed to respond to broadcast Intents
- Think of them as a way to respond to external notifications or alarms
- Applications can invent and broadcast their own Intents as well

Intent


Intent
Intent
startActivity()
onCreate()
Activity A
Android System
Activity B
1
2
3

Intents

- Think of Intents as a verb and object; a description of what you want done
  - Examples: `VIEW`, `CALL`, `PLAY`, etc.
- System matches Intent with Activity that can best provide that service
- Activities and IntentReceivers describe what Intents they can service in their IntentFilters (via AndroidManifest.xml)

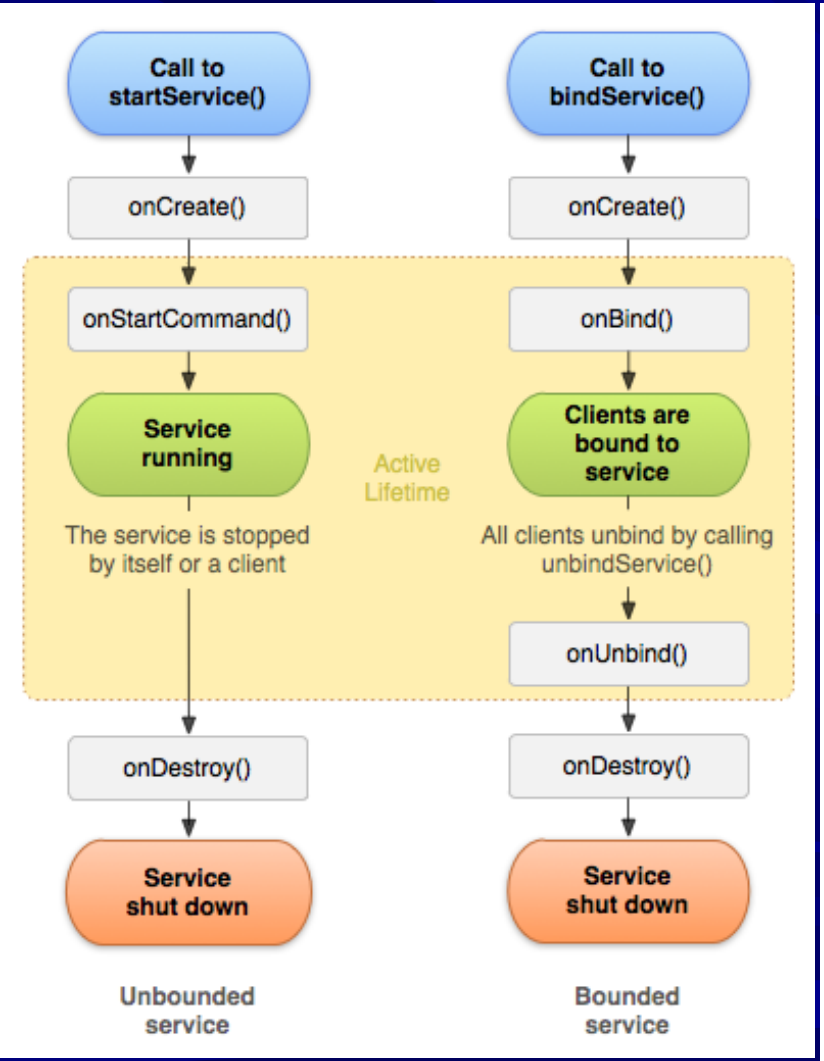
Intent

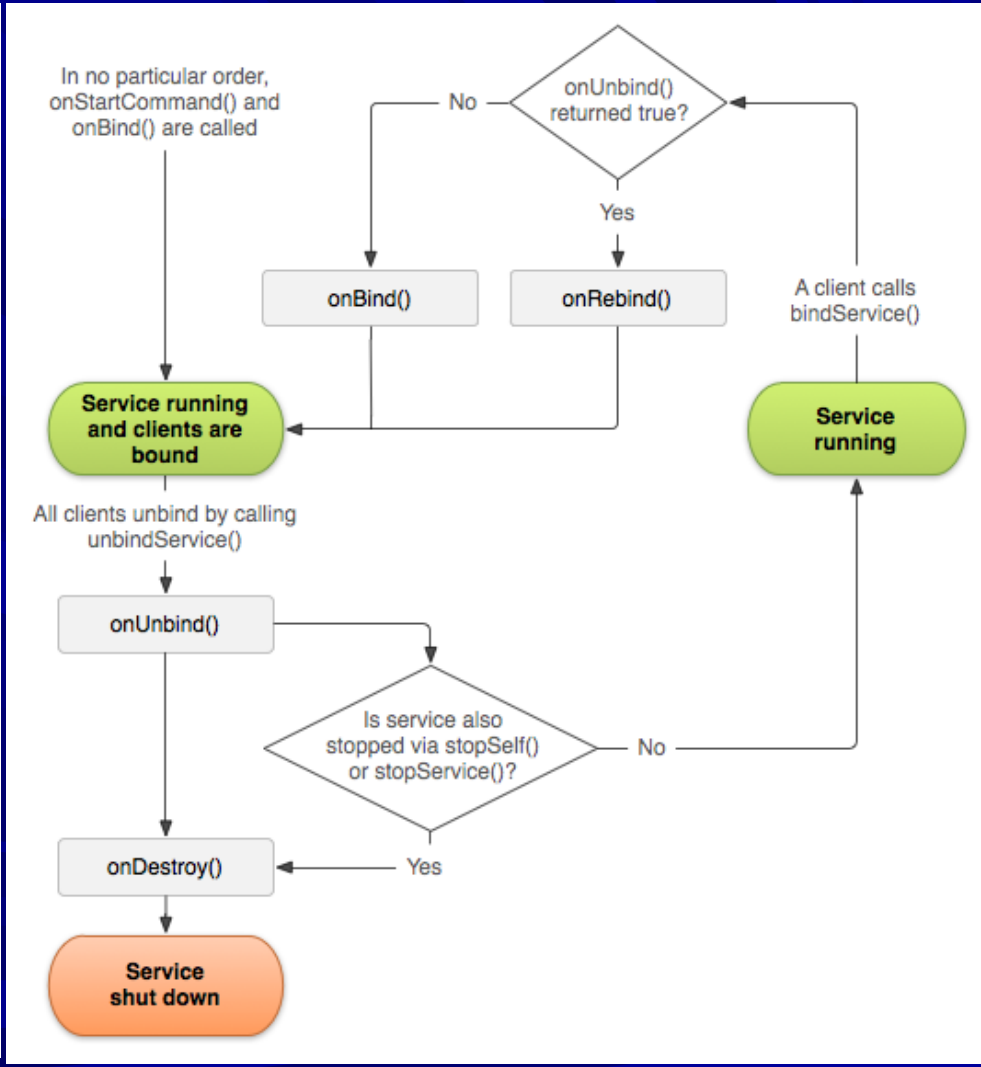
- Explicit intents
- Implicit intents
- Extras:simple key/value pairs
  - putExtra()/getExtra()
- Flags
  - setFlags()/ getFlags()

Services

- Faceless components that run in the background
  - Example: music player, network downlaod, etc.
- Bind your code to a running service via a remote-able interface defined in an IDL
- Can run in your own process or separate process

The Service Lifecycles





Storage Options

Android provides several options for you to save persistent application data. The solution you choose depends on your specific needs, such as whether the data should be private to your application or accessible to other applications (and the user) and how much space your data requires.

Your data storage options are the following:

Shared Preferences
: Store private primitive data in key-value pairs.

Internal Storage
: Store private data on the device memory.

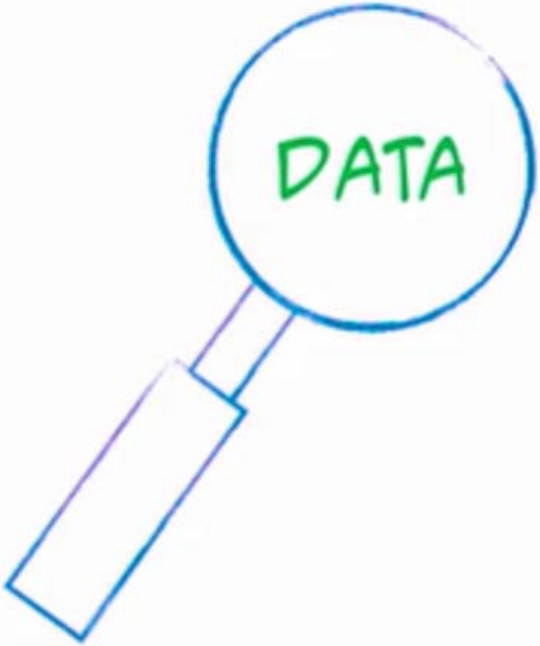
External Storage
: Store public data on the shared external storage.

SQLite Databases
: Store structured data in a private database.

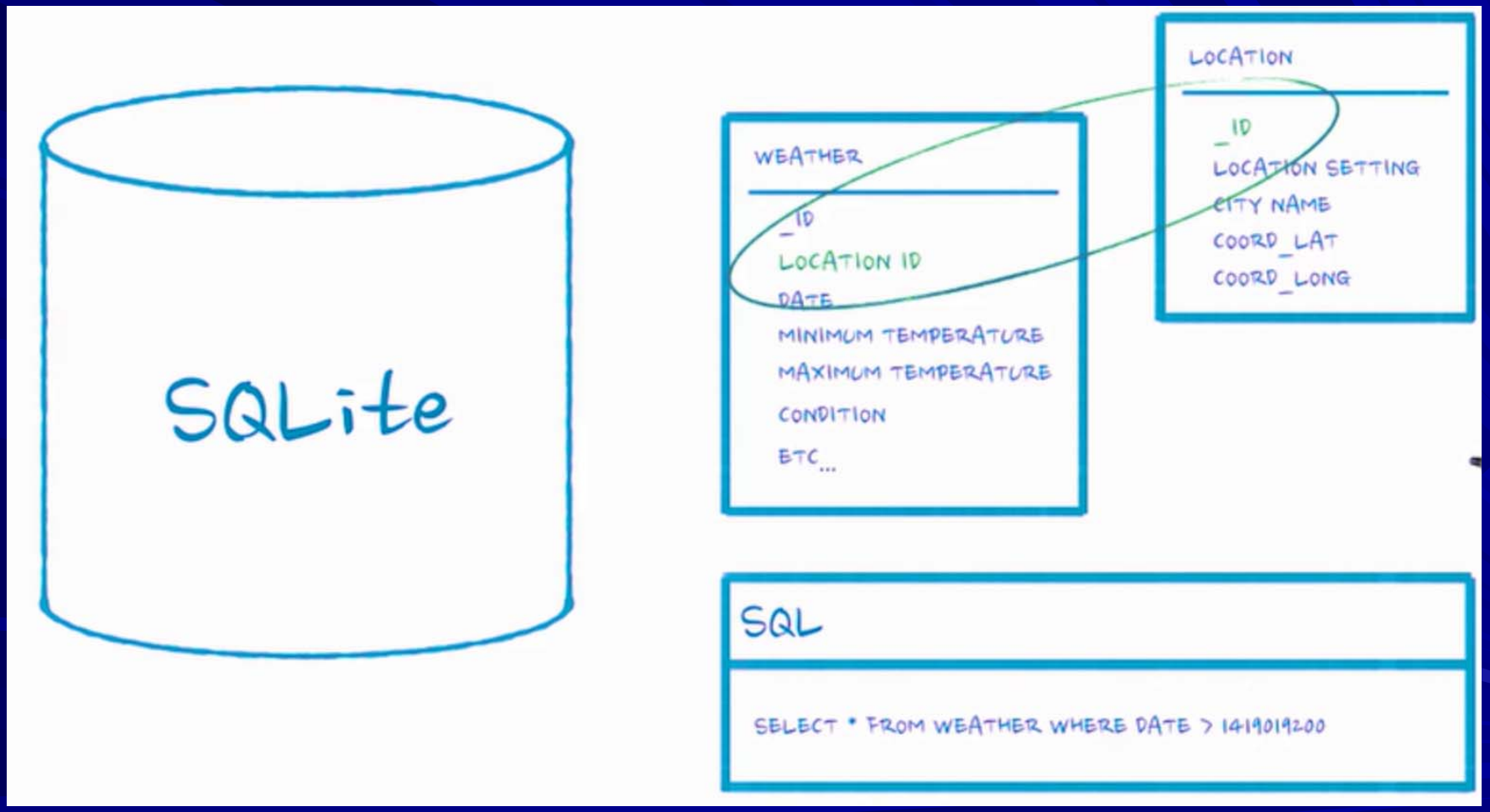
Network Connection
: Store data on the web with your own network server.

SharedPreferences

= location



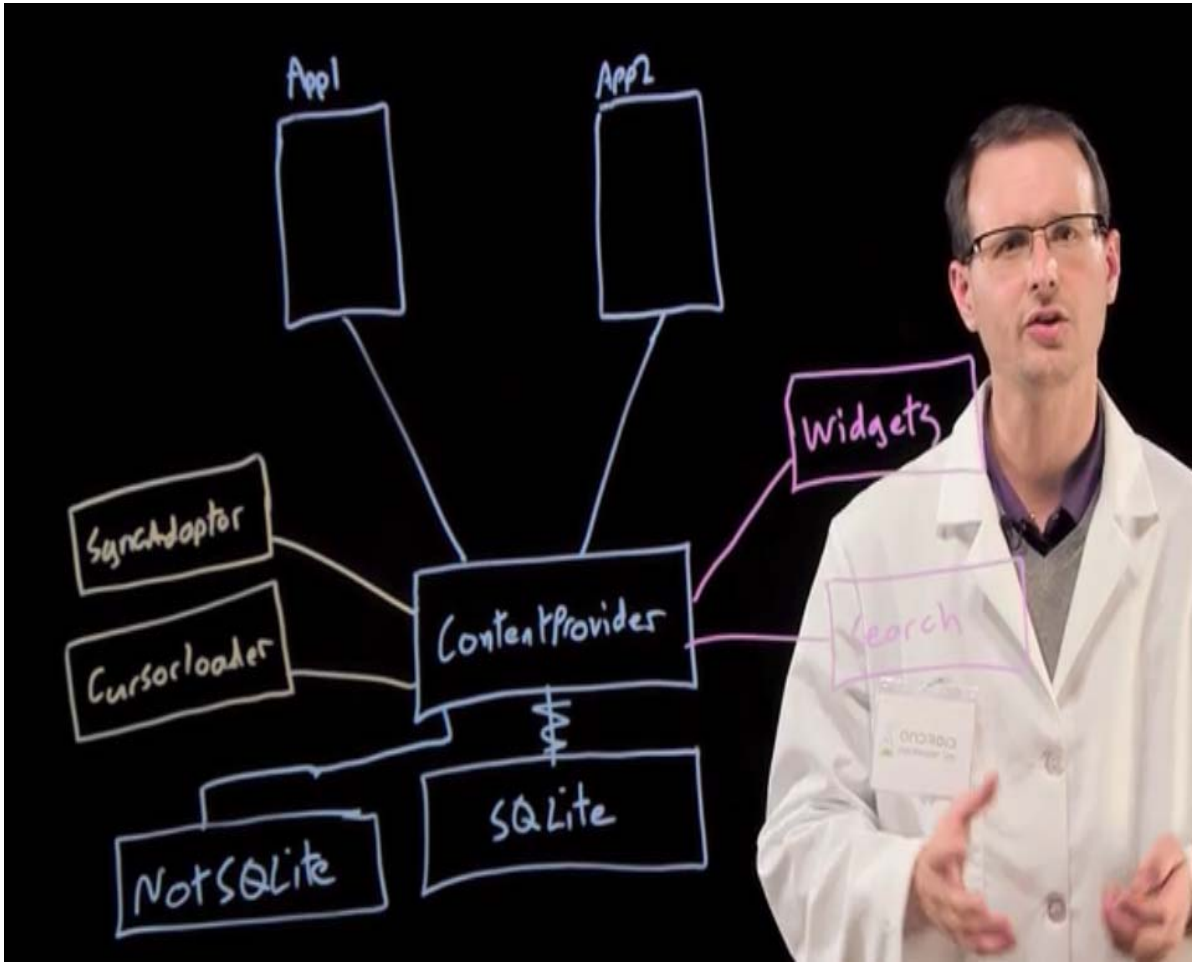
= 94043


SQLite
LOCATION
_ID
LOCATION SETTING
CITY NAME
COORD_LAT
COORD_LONG
WEATHER
_ID
LOCATION ID
DATE
MINIMUM TEMPERATURE
MAXIMUM TEMPERATURE
CONDITION
ETC...
SQL
SELECT * FROM WEATHER WHERE DATE > 1419019200

ContentProviders

- Enables sharing of data across applications
  - Examples: address book, photo gallery, etc.
- Provides uniform APIs for:
  - querying (returns a Cursor)
  - delete, update, and insert rows
- Content is represented by URI and MIME type

Why ContentProvider



1) Determine URIs

2) Update Contract

```
WEATHER = 100
content://com.example.android.sunshine.app/
weather

WEATHER_WITH_LOCATION = 101
content://com.example.android.sunshine.app/
weather/[location_query]

WEATHER_WITH_LOCATION_AND_DATE = 102
content://com.example.android.sunshine.app/
weather/[location_query]/[date]

LOCATION = 300
content://com.example.android.sunshine.app/
location
```

3) Fill out UriMatcher

"path" - matches "path" exactly
"path/#" - matches "path" followed by a number
"path/*" - matches "path" followed by any string
"path/*/other/#" - matches "path" followed by a string followed by "other" followed by a number

4) Implement Functions

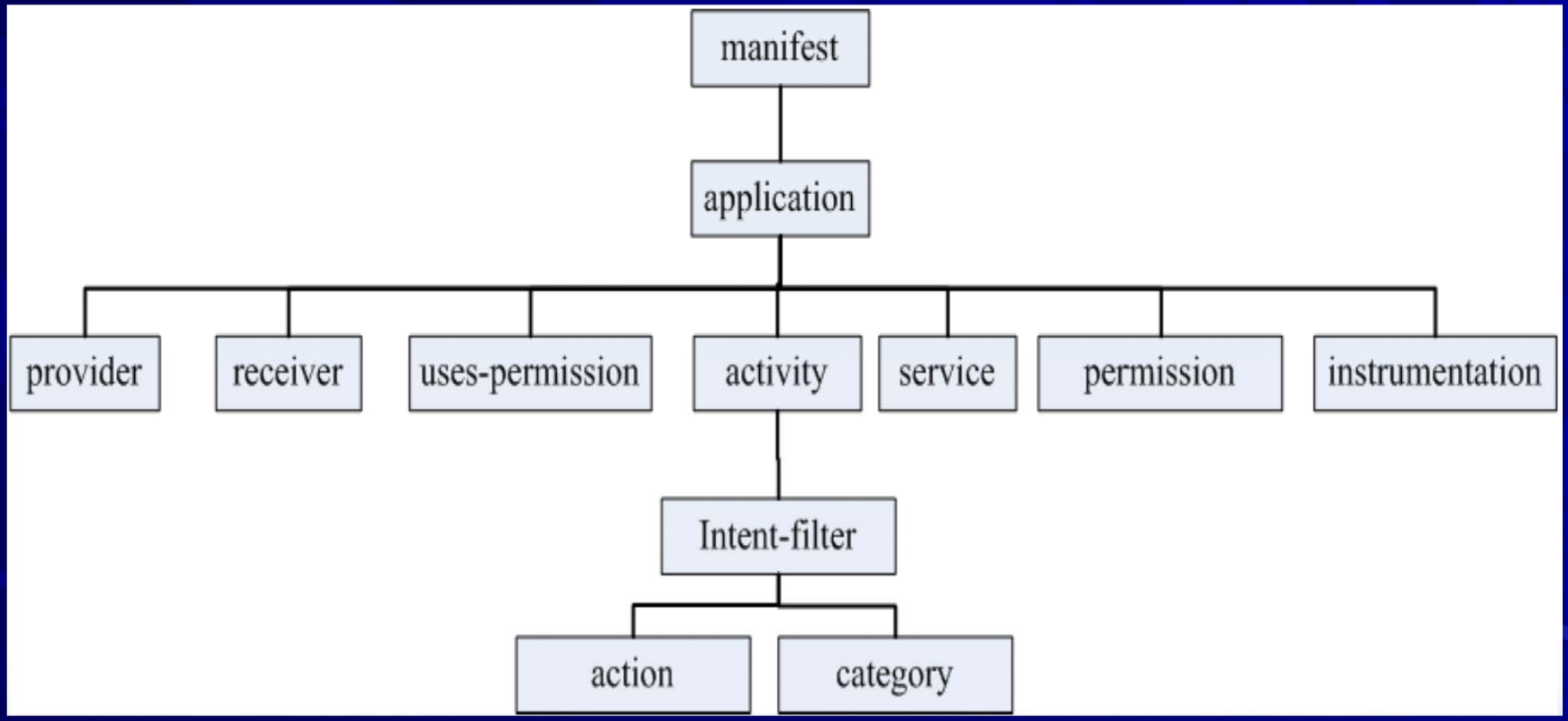
```
onCreate()
query(Uri, String[], String, String[], String)
insert(Uri, ContentValues)
update(Uri, ContentValues, String, String[])
delete(Uri, String, String[])
getType(Uri)
```

SQLite & SQL

- What is SQLite
- What is SQL
- Cursor

CursorLoader	SQLiteCursor
CursorAdapter	SQLiteDatabase
CursorJoiner	SQLiteOpenHelper
ContentProvider	SQLiteQueryBuilder
ContentValues	SQLiteQuery
DatabaseUtils	SQLiteStatement

The AndroidManifest.xml File



The AndroidManifest.xml File

- names the Java package for the application.
- describes the components of the application
  - activities, services, broadcast receivers, and content providers
- determines which processes will host application components.
- declares which permissions the application must have
- declares the permissions that others are required to have
- lists the Instrumentation classes
- declares the minimum level of the Android API
- lists the libraries that the application must be linked against.

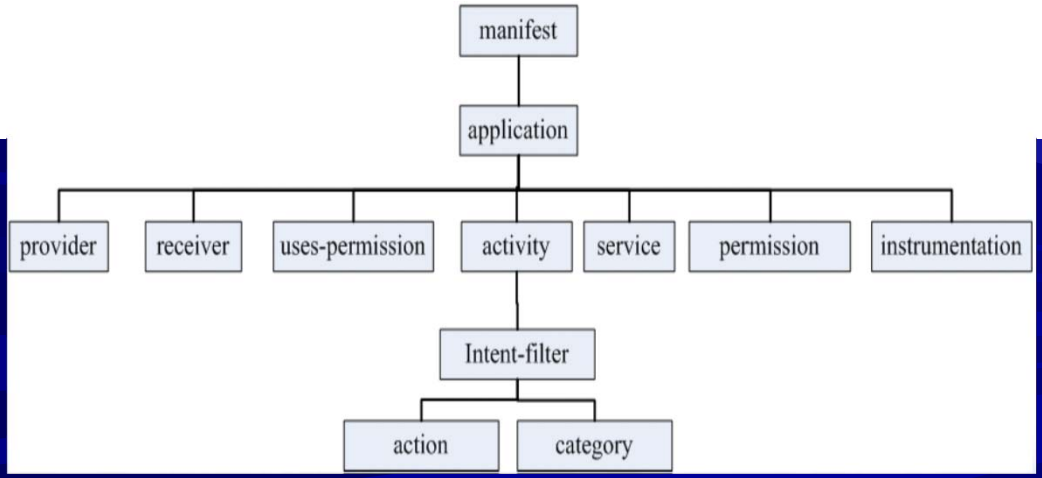
AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.edu.xmu.lxzheng.layouttest" >

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="LayoutTest"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="LayoutTest" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```



MainActivity.java

```
public class MainActivity extends ActionBarActivity
    {

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

R.java

- A project's R.java file is an index into all the resources defined in the file
- contains resource IDs for all the resources in your res/ directory.
- For each type of resource, there is an R subclass

R.java

```
public final class R {
    public static final class attr {
    }
    public static final class dimen {
        public static final int activity_horizontal_margin=0x7f040000;
        public static final int activity_vertical_margin=0x7f040001;
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int action_settings=0x7f080000;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
。 。 。 。 。 。
```

Resource

- Res
  - Drawable
  - Layout
  - Values

res/values

- XML files describing additional resources such as strings, colors, and styles
  - arrays
  - classes.xml
  - colors.xml
  - dimens.xml
  - strings.xml
  - styles.xml
  - values.xml

res/values/strings.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name" >Hello
    </string>
    <string name="action_settings" >Settings
    </string>
    <string name="hello_world" >Hello world!
    </string>

</resources>
```

res/layout

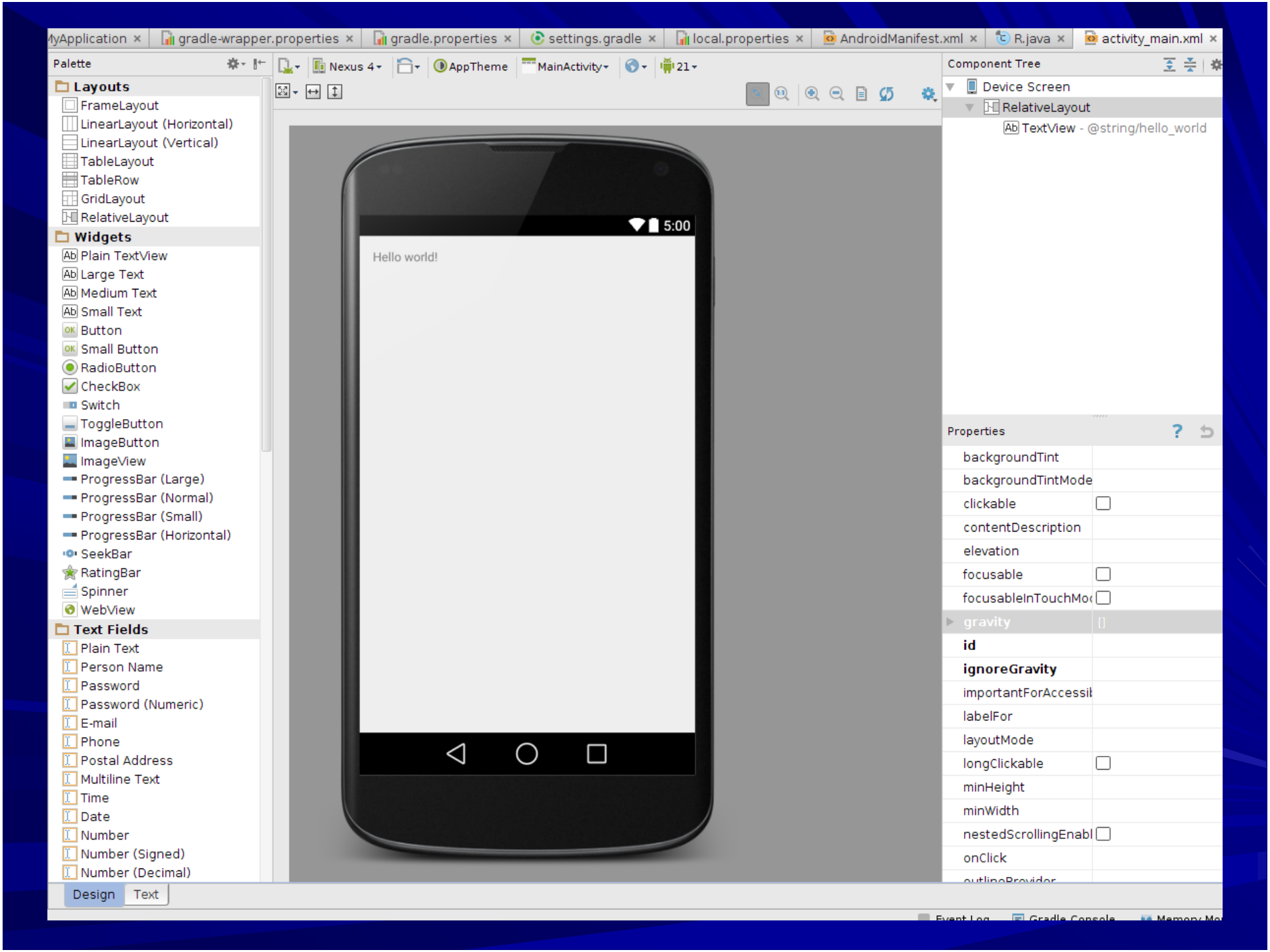
- Holds all the XML files describing screens or parts of screens.
  - FrameLayout
    - simplest type of layout objec
  - LinearLayout
    - aligns all children in a single direction
  - RelativeLayout
    - lets child views specify their position relative to the parent view or to each other
  - TableLayout
    - positions its children into rows and columns

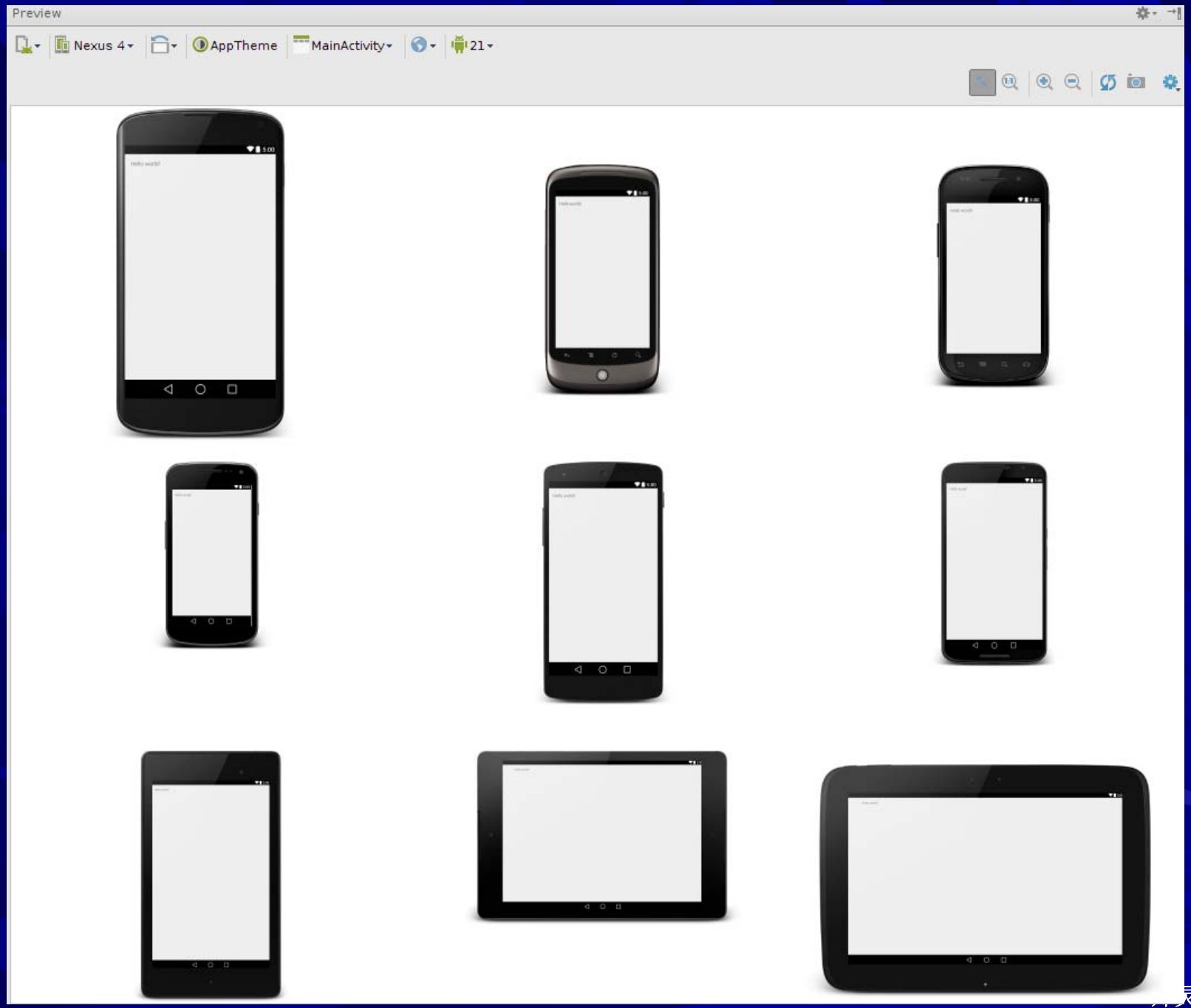
res/layout/activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello world!" />

</RelativeLayout>
```


MyApplication
gradle-wrapper.properties
gradle.properties
settings.gradle
local.properties
AndroidManifest.xml
R.java
activity_main.xml
Palette
Layouts
FrameLayout
LinearLayout (Horizontal)
LinearLayout (Vertical)
TableLayout
TableRow
GridLayout
RelativeLayout
Widgets
Plain TextView
Large Text
Medium Text
Small Text
Button
Small Button
RadioButton
CheckBox
Switch
ToggleButton
ImageButton
ImageView
ProgressBar (Large)
ProgressBar (Normal)
ProgressBar (Small)
ProgressBar (Horizontal)
SeekBar
RatingBar
Spinner
WebView
Text Fields
Plain Text
Person Name
Password
Password (Numeric)
E-mail
Phone
Postal Address
Multiline Text
Time
Date
Number
Number (Signed)
Number (Decimal)
Nexus 4
AppTheme
MainActivity
21
5:00
Hello world!
Component Tree
Device Screen
RelativeLayout
TextView - @string/hello_world
Properties
backgroundTint
backgroundTintMode
clickable
contentDescription
elevation
focusable
focusableInTouchMo
gravity
id
ignoreGravity
importantForAccessil
labelFor
layoutMode
longClickable
minHeight
minWidth
nestedScrollingEnabl
onClick
Design
Text


Preview
Nexus 4
AppTheme
MainActivity
21

VIEW GROUPS

FrameLayout

LinearLayout

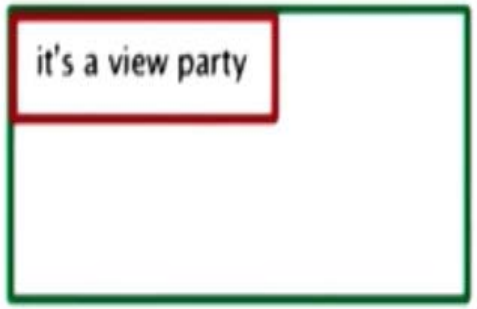
RelativeLayout

GridLayout

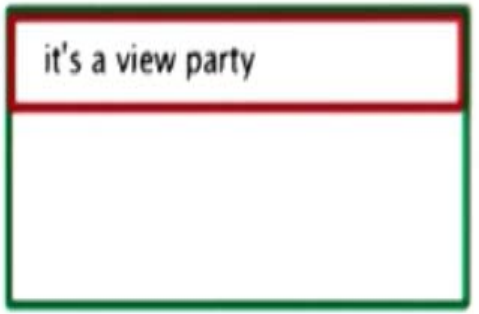
VIEW WIDTH/HEIGHT

For TextView

width=
wrap_content
height=
wrap_content

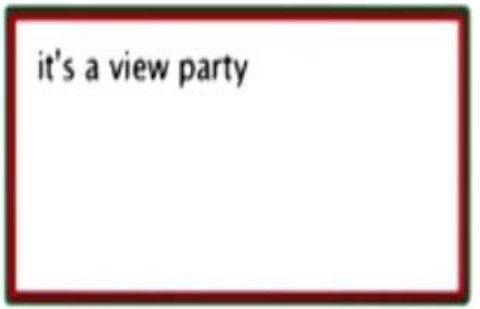


width=
match_parent
height=
wrap_content



width=
wrap_content
height=
match_parent

width=
match_parent
height=
match_parent



GRAVITY

For this TextView

No gravity set

gravity = center

Center within TextView

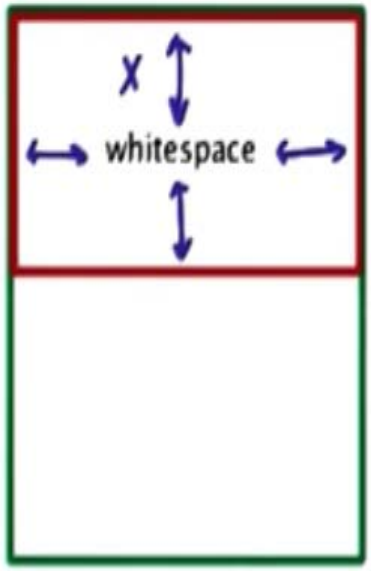
layout_gravity = center

center within parent

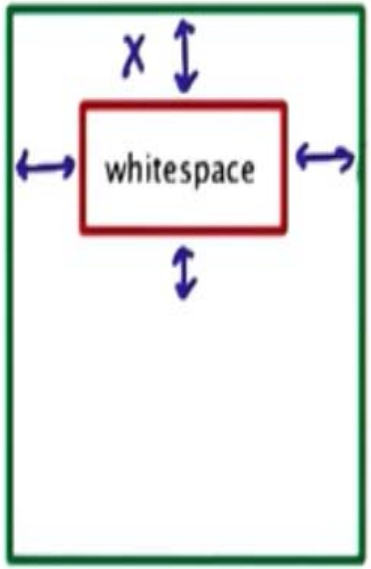
PADDING VS. MARGIN

padding = X

For this TextView



layout_margin = X



VIEW VISIBILITY

For Image View

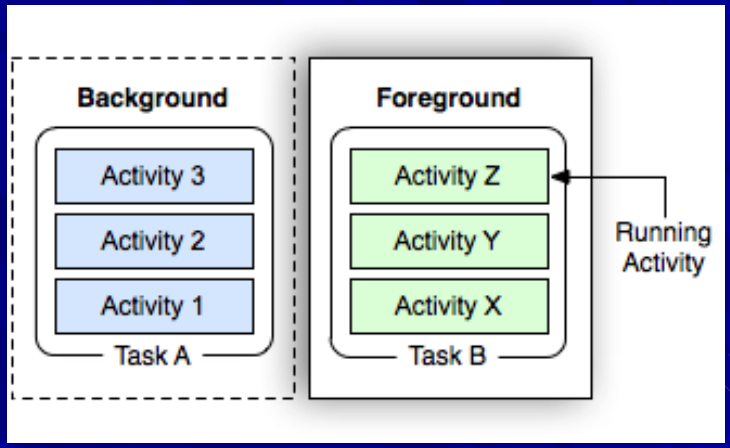
visible

invisible

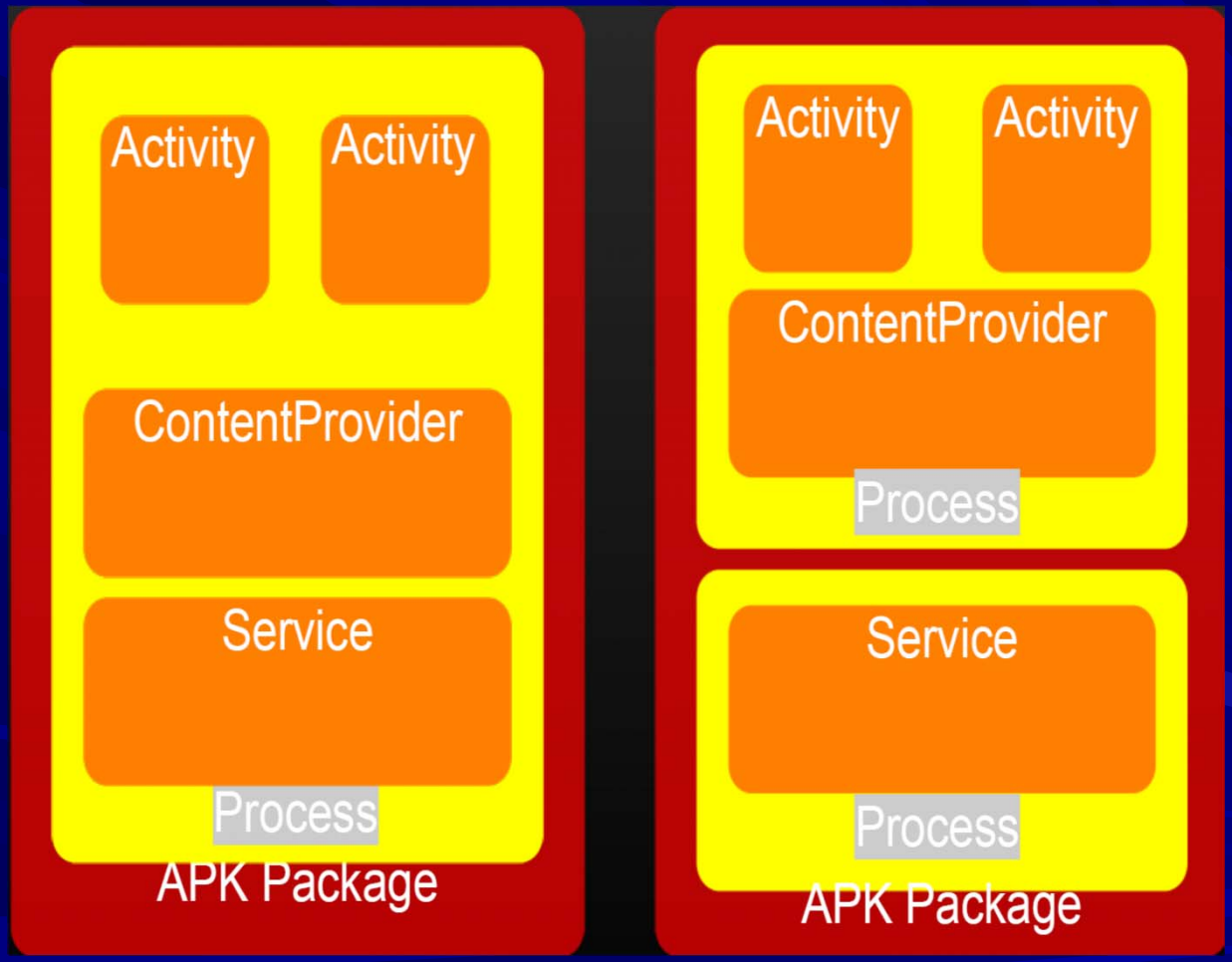
gone

Application, Activity, Stack, Task, Process

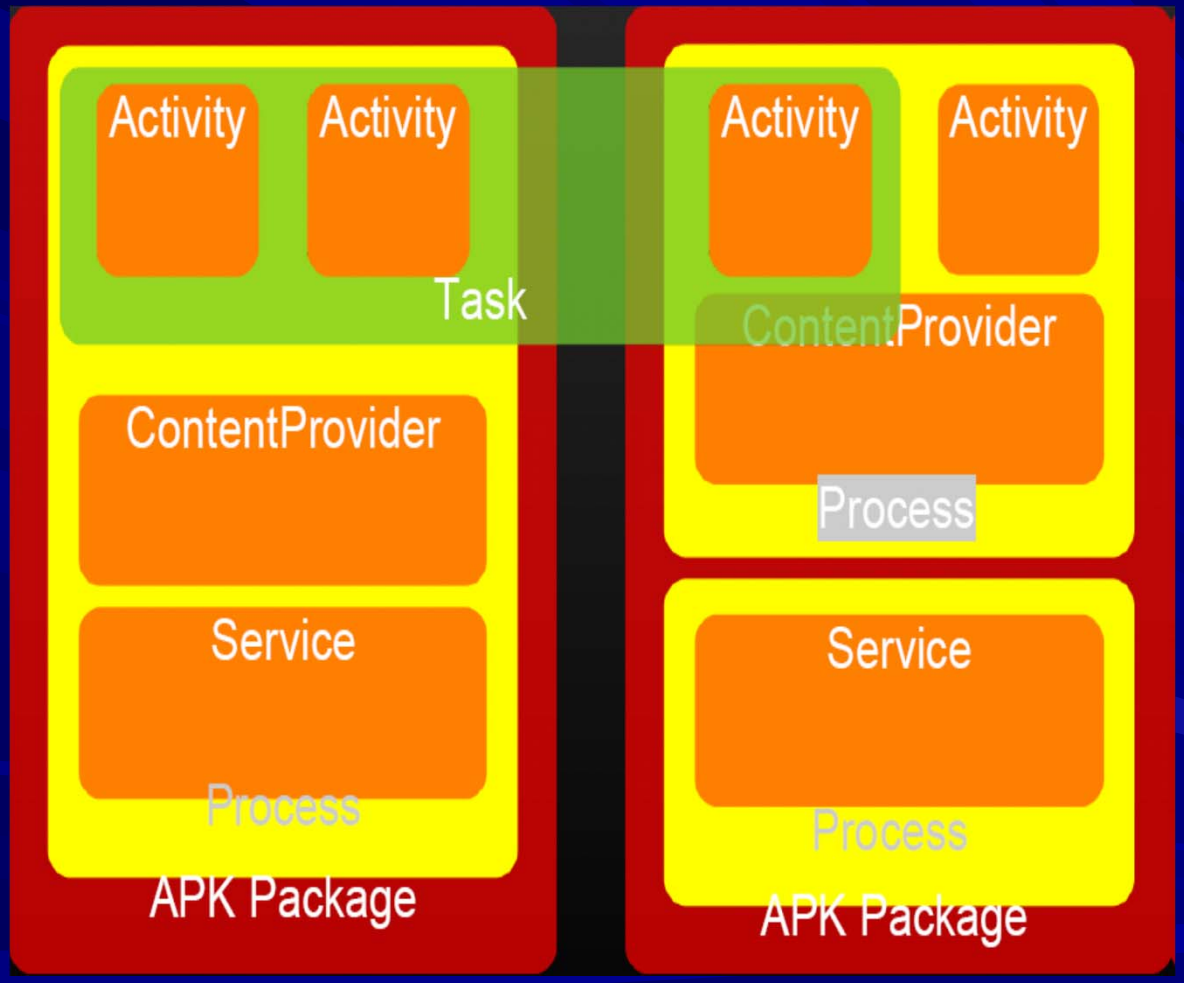
- Application
  - one or more related, loosely bound activities
  - bundled up in a single apk file
- Activity
  - the main building blocks of Android applications
- Stack
  - a linear navigation history of activities the user has visited
- Task
  - A task is the sequence of activities the user follows to accomplish an objective
- Process
  - A “process” is a standard Linux process
  - every application runs in its own Linux process



Activities and Tasks



Activities and Tasks



What class does Android use to simplify background thread creation and UI thread synchronization?

郑灵翔

AsyncTask

extends Object

java.lang.Object
  ↳android.os.AsyncTask<Params, Progress, Result>

Class Overview

AsyncTask enables proper and easy use of the UI thread. This class allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.

AsyncTask is designed to be a helper class around `Thread` and `Handler` and does not constitute a generic threading framework. AsyncTasks should ideally be used for short operations (a few seconds at the most.) If you need to keep threads running for long periods of time, it is highly recommended you use the various APIs provided by the `java.util.concurrent` package such as `Executor`, `ThreadPoolExecutor` and `FutureTask`.

An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called `Params`, `Progress` and `Result`, and 4 steps, called `onPreExecute`, `doInBackground`, `onProgressUpdate` and `onPostExecute`.

The 4 steps

When an asynchronous task is executed, the task goes through 4 steps:

1. `onPreExecute()`, invoked on the UI thread before the task is executed. This step is normally used to setup the task, for instance by showing a progress bar in the user interface.
2. `doInBackground(Params...)`, invoked on the background thread immediately after `onPreExecute()` finishes executing. This step is used to perform background computation that can take a long time. The parameters of the asynchronous task are passed to this step. The result of the computation must be returned by this step and will be passed back to the last step. This step can also use `publishProgress(Progress...)` to publish one or more units of progress. These values are published on the UI thread, in the `onProgressUpdate(Progress...)` step.
3. `onProgressUpdate(Progress...)`, invoked on the UI thread after a call to `publishProgress(Progress...)`. The timing of the execution is undefined. This method is used to display any form of progress in the user interface while the background computation is still executing. For instance, it can be used to animate a progress bar or show logs in a text field.
4. `onPostExecute(Result)`, invoked on the UI thread after the background computation finishes. The result of the background computation is passed to this step as a parameter.

Threading rules

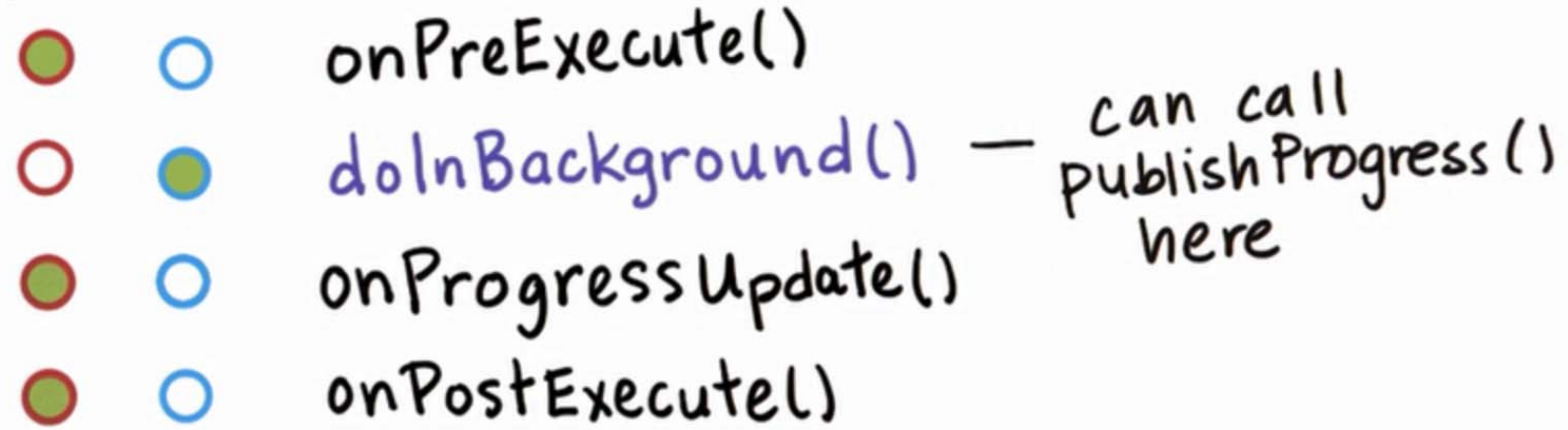
There are a few threading rules that must be followed for this class to work properly:

- The AsyncTask class must be loaded on the UI thread. This is done automatically as of `JELLY_BEAN`.
- The task instance must be created on the UI thread.
- `execute(Params...)` must be invoked on the UI thread.
- Do not call `onPreExecute()`, `onPostExecute(Result)`, `doInBackground(Params...)`, `onProgressUpdate(Progress...)` manually.
- The task can be executed only once (an exception will be thrown if a second execution is attempted.)

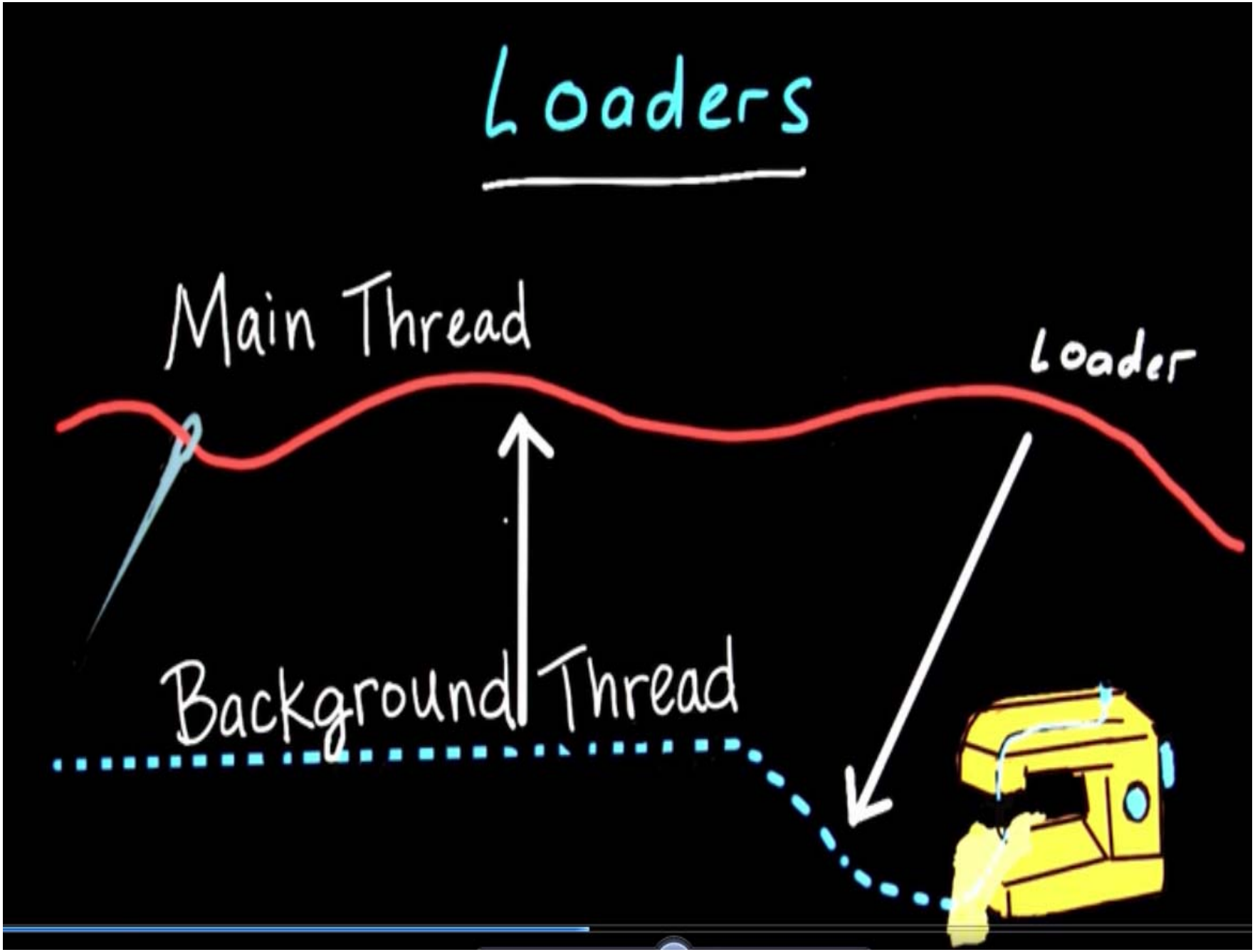
Protected Methods	
abstract Result	doInBackground (Params... params) Override this method to perform a computation on a background thread.
void	onCancelled (Result result) Runs on the UI thread after `cancel(boolean)` is invoked and `doInBackground(Object[])` has finished.
void	onCancelled () Applications should preferably override `onCancelled(Object)`.
void	onPostExecute (Result result) Runs on the UI thread after `doInBackground(Params...)`.
void	onPreExecute () Runs on the UI thread before `doInBackground(Params...)`.
void	onProgressUpdate (Progress... values) Runs on the UI thread after `publishProgress(Progress...)` is invoked.
final void	publishProgress (Progress... values) This method can be invoked from `doInBackground(Params...)` to publish updates on the UI thread while the background computation is still running.

AsyncTask

MAIN or BACKGROUND thread?



郑灵翔


Loaders
Main Thread
Loader
Background Thread

AsyncTask Lifecycle

onCreate

asyncTask.execute()

asyncTask.doInBackground()

asyncTask.onPostExecute()

activity restarts

onCreate

asyncTask.execute()

asyncTask.doInBackground()

AsyncTaskLoader Lifecycle

onCreate

initLoader()

onCreateLoader()

loadInBackground()

.....activity restarts....

initLoader()

onLoadFinished()

Reference

- Application Fundamentals
  - https://developer.android.com/guide/components/fundamentals.htm
- Intents and Intent Filters
  - https://developer.android.com/guide/components/intents-filters.html
- Activities
  - https://developer.android.com/guide/components/activities.html
- Services
  - https://developer.android.com/guide/components/services.html
- Content Providers
  - https://developer.android.com/guide/topics/providers/content-providers.html

Reference

- Building Your First App
  - Creating an Android Project
    - https://developer.android.com/training/basics/firstapp/creating-project.html
  - Running Your App
    - https://developer.android.com/training/basics/firstapp/running-app.html
  - Building a Simple User Interface
    - https://developer.android.com/training/basics/firstapp/building-ui.html
  - Starting Another Activity
    - https://developer.android.com/training/basics/firstapp/starting-activity.html

Reference

- Managing the Activity Lifecycle
  - Starting an Activity
    - https://developer.android.com/training/basics/activity-lifecycle/starting.html
  - Pausing and Resuming an Activity
    - https://developer.android.com/training/basics/activity-lifecycle/pausing.html
  - Stopping and Restarting an Activity
    - https://developer.android.com/training/basics/activity-lifecycle/stopping.html
  - Recreating an Activity
    - https://developer.android.com/training/basics/activity-lifecycle/recreating.html

Reference

- Sending Simple Data to Other Apps
  - https://developer.android.com/training/sharing/send.html
- Receiving Simple Data from Other Apps
  - https://developer.android.com/training/sharing/receive.html

Reference

- Saving Data
  - Saving Key-Value Sets
    - https://developer.android.com/training/basics/data-storage/shared-preferences.html
  - Saving Files
    - https://developer.android.com/training/basics/data-storage/files.html
  - Saving Data in SQL Databases
    - https://developer.android.com/training/basics/data-storage/databases.html

Reference(中文)

http://hukai.me/android-training-course-in-chinese/

http://wiki.eoeandroid.com/Android_API_Guides