MOBILE PHONE EVOLUTION

In 1876, Alexander Graham Bell became the first to receive a patent for the electric phone

In 1936, Alfred Gross. Case Tech OH (Case Western Reserve University). Invented/Patented Walkie-talkie, CB radio, Telephone Pager

In 1975, Dr. Martin Cooper invented first commercial portable Motorola radio phone

In 2007, iPhone and Android appeared

WHAT IS ANDROID?

Android OS is an open-source Linux-based operating system for mobile devices

It is being developed by the Open Handset Alliance and Google Inc

The operating system has several native applications supporting telephony, messaging, etc.

3rd party Java developers can use Android API to extend functionality of the devices

Google provides an on-line electronic market for third-party developers to sell-distribute their applications

Open Handset Alliance is a consortium of 80+ technology and mobile business companies

Quoting from www.OpenHandsetAlliance.com site (2/25/2012)

> Today, there are 1.5 billion television sets in use around the world. 1 billion people are on the Internet. But nearly 3 billion people have a mobile phone, making it one of the world's most successful consumer products. Building a better mobile phone would enrich the lives of countless people across the globe. The Open Handset Alliance™ is a group of mobile and technology leaders who share this vision for changing the mobile experience for consumers.

SOFTWARE: WHAT IS ANDROID? (Open handset alliance members)

Mobile operators	Handset manufacturers	Semiconductor companies	Software companies	Commercialization companies
1. Bouygues Telecom China Mobile 2. Communications Corporation 3. China Telecommunications 4. China United Network Communications 5. KDDI 6. LG Uplus 7. NTT Docomo 8. Softbank mobile 9. Sprint Nextel 10. T-Mobile 11. Telecom Italia 12. Telefónica 13. Telus 14. Vodafone	1. Acer 2. Alcatel mobile phones 3. ASUSTeK Computer 4. CCI 5. Dell 6. Foxconn International Holdings Limited 7. Fujitsu limited 8. Garmin International 9. Haier Telecom (Qingdao) 10. HTC 11. Huawei Technologies 12. Kyocera 13. Lenovo Mobile Communication Technology 14. LG Electronics 15. Motorola 16. NEC 17. OPPO Mobile Telecommunications 18. Pantech 19. Samsung Electronics...	1. AKM 2. Audience 3. ARM 4. Atheros Communications 5. Broadcom Corporation 6. Cypress Semiconductor Corporation 7. Freescale Semiconductor 8. Gemalto 9. Imagination Technologies 10. Intel Corporation 11. Marvell Semiconductor 12. MediaTek 13. MIPS Technologies 14. NVIDIA Corporation 15. Qualcomm 16. Renesas Electronics Corporation 17. ST-Ericsson 18. Synaptics 19. Texas Instruments Incorporated 20. Via Telecom	1. Ándago Ingeniería S.L. 2. ACCESS 3. Ascender 4. Cooliris 5. eBay 6. Google 7. LivingImage 8. Myriad 9. MOTOYA 10. Nuance Communications 11. NXP Software 12. OMRON Software 13. PacketVideo 14. SkyPop 15. SONiVOX 16. SVOX 17. VisualOn	1. Accenture 2. Aplix Corporation 3. Borqs 4. Intrinsyc Software International 5. L&T Infotech 6. Noser Engineering 7. Sasken Communication Technologies 8. SQLStar International 9. The Astonishing Tribe AB 10. Teleca AB 11. Wind River 12. Wipro Technologies

SOFTWARE: WHAT IS ANDROID? (The mobile revolution)

Electronic tools commonly carried by a typical business warrior

Not so long ago ...	Today
1. Phone 2. Pager 3. PDA Organizer 4. Laptop 5. MP3 Portable music player 6. Wired modem 7. No internet access / limited access	1. Smartphone 2. Laptop

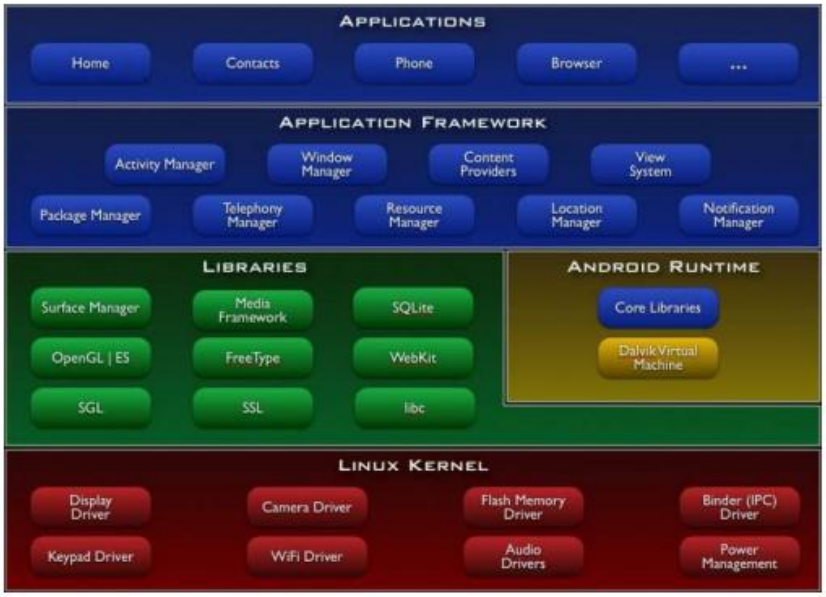
Tomorrow?

I want my 2015 Smartphone to be: Phone, Pager, PDA Organizer, high quality camera, laptop, cash, ...

SOFTWARE: WHAT IS ANDROID? (The mobile revolution)

Android Software/Hardware Components

- Dalvik virtual machine (soon to be replaced by ART)
- Integrated browser (WebKit)
- Graphic Capabilities (hardware acceleration)
- SQLite for structured data storage
- Media support (audio/video)
- GSM Telephony (hardware dependent)
- Bluetooth, EDGE, 3G, 4G, NFC, and Wi-Fi (hardware manufacturer dependent)
- Camera, GPS, compass, accelerometer, gyroscope, proximity/ambient light, barometric pressure, fingerprint reader, heart rate sensor (hardware dependent)
- Software Development Tools & Application framework (device emulator, debugging, profiling, plugin for the Eclipse IDE, resource managers, Android Studio)



ANDROID'S SOFTWARE ARCHITECTURE

ANDROID INTENTS

An Intent is a request for services offered by an Android based device

An Intent is made up of various pieces including:

- desired action or service
- data
- category of component that should handle the intent and instructions on how to launch a target activity

Action	Data
The general action to be performed, such as: ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.	The data to operate on, such as a person record in the contacts database, expressed as a Uri

ANATOMY OF ANDROID APPLICATIONS

Core components are the primordial classes or building blocks from which apps are made.

An application consists of one or more core component objects, working in a cooperative mode, each contributing somehow to the completion of the tasks

Each core component provides a particular type of functionality and has a distinct lifecycle.

A lifecycle defines how the component is created, transitioned, and destroyed.

There are four type of core components

- Activities
- Services
- Broadcast Receiver
- Content Provider

ANATOMY OF ANDROID APPLICATIONS (Android's core components – service)

Services are a special type of activity that do not have a visual user interface. A service object may be active without the user noticing its presence.

Services are analogous to secondary threads, usually running some kind of background 'busy-work' for an indefinite period of time.

Applications start their own services or connect to services already active.

Examples:

- Your background GPS service could be set to quietly run in the background detecting location information from satellites, phone towers or wi-fi routers.
- The service could periodically broadcast location coordinates to any app listening for that kind of data.
- An application may opt for binding to the running GPS service and use the data that it supplies.

ANATOMY OF ANDROID APPLICATIONS (Android's core components – service)

In this example a music service (say Pandora Radio) and GPS location run in the background.

The selected music station is heard while other GUIs are show on the device's screen. For instance, our user –an avid golfer- may switch between occasional golf course data reading (using the GolfShot app) and "Angry Birds"

ANATOMY OF ANDROID APPLICATIONS (Android's core components – broadcast receiver)

A broadcast receiver is a dedicated listener that waits for a triggering system-wide message to do some work. The message could be something like low-battery, wi-fi connection available, earth-quakes in California, speed-camera nearby.

Broadcast receivers do not display a user interface.

They typically register with the system by means of a filter acting as a key. When the broadcasted message matches the key the receiver is activated.

A broadcast receiver could respond by either executing a specific activity or use the notification mechanism to request the user's attention.

ANATOMY OF ANDROID APPLICATIONS (Android's core components – content provider)

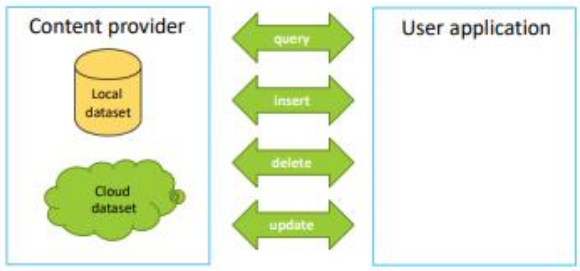
A content provider is a data-centric service making persistent datasets available to any applications.

Common global datasets include contacts, pictures, messages, audio files, emails.

The global datasets are stored in an SQLite database (however developer doesn't have to be an SQL expert)

The content provider class offers a standard set of parametric methods to enable other applications to retrieve, delete, update, and insert data items.

Content provider is a wrapper hiding the actual physical data. Users interact with their data through a common object interface.



(Activity stack)

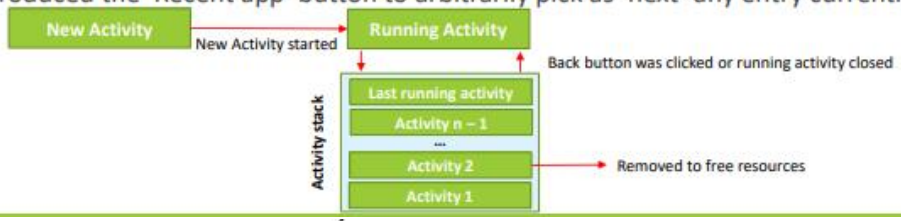
Activities in the system are scheduled using an activity stack.

When a new activity is started, it is placed on top of the stack to become the running activity

Previous activity is pushed-down one level in stack and may come back to foreground once new activity finishes.

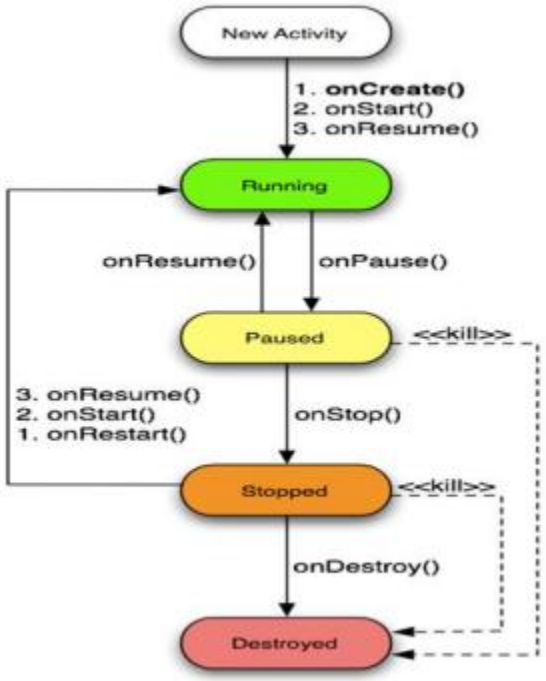
If the user presses the Back Button the current activity is terminated and previous activity on stack moves up to become active.

Android 4.0 introduced the 'Recent app' button to arbitrarily pick as 'next' any entry currently in stack



When progressing from one state to the other, OS notifies application of the changes by issuing calls to the following protected transition methods:

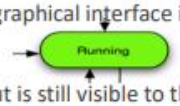
- void onCreate( ): The activity is being created.
- void onStart( ): The activity is about to become visible.
- void onResume( ): The activity has become visible (it is now "resumed")
- void onPause( ): Another activity is taking focus (this activity is about to be "paused")
- void onStop( ): The activity is no longer visible (it is now "stopped")
- void onDestroy( ): The activity is about to be destroyed



(Activity state: running & paused & stopped)

Your activity is active or running when it is in foreground of screen (seating on top of the activity stack)

This is the activity that has "focus" and its graphical interface is responsive to the user's interactions.

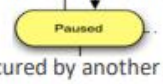


Your Activity is paused if it has lost focus but is still visible to the user.

That is, another activity seats on top and new activity either is transparent or doesn't cover full screen.

A paused activity is alive (maintaining its state information and attachment to the window manager).

Paused activities can be killed by the system when available memory becomes extremely low.



Your Activity is stopped if it is completely obscured by another activity.

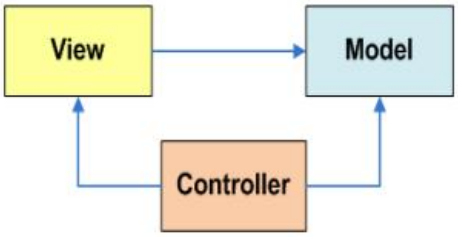
Although stopped, it continues to retain all its state information.

It is no longer visible to the user (its window is hidden, and its life cycle could be terminated at any point by the system if the resources that it holds are needed elsewhere).

The function below allows to obtain the current ORIENTATION of the device as NORTH(0), WEST(1), SOUTH(2) and EAST(3).

```
private int getOrientation(){
 // the TOP of the device points to [0:North, 1:West, 2:South, 3:East]
 Display display = ((WindowManager) getApplication().getSystemService(Context.WINDOW_SERVICE)).getDefaultDisplay();
 display.getRotation();
 return display.getRotation();
}
```

GRAPHICAL USER INTERFACES
(The model-view-control pattern - MVC)



The Model-View-Controller (MVC) is an important software design pattern first introduced with the Xerox-Smalltalk80 system whose main goal is to separate the (1) user interface, (2) business, and (3) input logic.

How is this pattern seen by the Android developer?

- Model. Consists of the Java code and API objects used to represent the business problem and manage the behavior and data of the application.
- View. Set of screens the user sees and interacts with.
- Controller. Implemented through the Android OS, responsible for interpretation of the user and system inputs. Input may come from a variety of sources such as the trackball, keyboard, touch-screen, GPS chip, proximity sensor, accelerometer, etc, and tells the Model and/or the View (usually through callbacks and registered listeners) to change as appropriate.

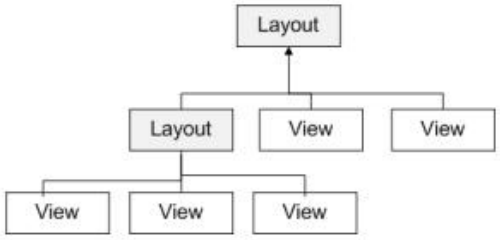
(The VIEW class)

The View class is the Android's most basic component from which user interfaces can be created. It acts as a container of displayable elements.

A View occupies a rectangular area on screen and is responsible for drawing & event handling.

Widgets are subclasses of View. They are used to create interactive UI components such as buttons, checkboxes, labels, text fields, etc.

Layouts are invisible structured containers used for holding other Views and nested layouts.



The LAYOUT

- Android GUI Layouts are containers having a predefined structure and placement policy such as relative, linear horizontal, grid-like, etc.
- Layouts can be nested, therefore a cell, row, or column of a given layout could be another layout.

INTRODUCTION (FRAGMENTS)

Android is a multitasking OS and its hardware specs allow for real parallelism. However, at any given time only one activity per app can be 'visible' and 'active'. This fact is rather limiting considering the extensive screen area offered by larger devices (tablets, phablets, TV sets, etc). Fragments offer an escape solution.

The Fragment class produces visual objects that can be dynamically attached to designated portions of the app's GUI. Each fragment object can expose its own views and provide means for the users to interact with the application.

Fragments must exist within the boundaries of an Activity that acts as a 'home-base' or host.

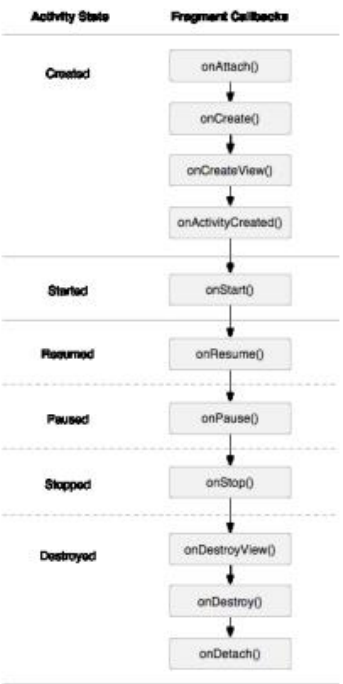
A host activity's GUI may expose any number of fragments. In this GUI each fragment could be visible and active.

Fragments behave as independent threads, usually they cooperate in achieving a common goal; however each can run its own I/O, events and business logic.

Fragments could access 'global data' held in the main activity to which they belong. Likewise, they could send values of their own to the main activity for potential dissemination to other fragments.

Fragments have their own particular Life-Cycle, in which the onCreateView method does most of the work needed to make them.

Fragments were first introduced in the Honeycomb SDK (API 11)



FRAGMENTS (LIFECYCLE)

onAttach() Invoked when the fragment has been connected to the host activity.

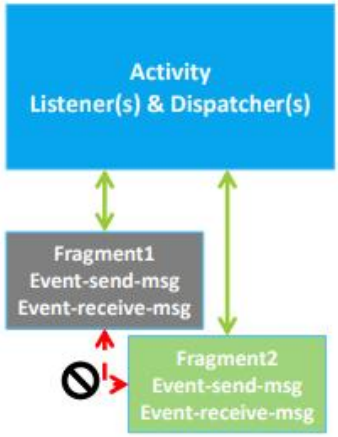
onCreate() Used for initializing non-visual components needed by the fragment.

onCreateView() Most of the work is done here. Called to create the view hierarchy representing the fragment. Usually inflates a layout, defines listeners, and populates the widgets in the inflated layout.

onPause() The session is about to finish. Here you should commit state data changes that are needed in case the fragment is reexecuted.

onDetach() Called when the inactive fragment is disconnected from the activity.

FRAGMENTS
(Inter-fragment communication)



All Fragment-to-Fragment communication is done in a centralized mode through the home-base Activity.

As a design principle; two Fragments should NEVER communicate directly.

The home-base Activity and its fragments interact through listeners and events.

When a fragment has some data for another fragment, it sends it to a listener in the main which in turn dispatches to a listener of the second fragment.

FRAGMENTS
(Integrating the home Activity and its fragments)

In general fragments appear on their enclosing Activity's GUI using one of the following attachment approaches

- Dynamic Binding: The main activity defines a particular place on its GUI for fragments to be plugged in (or attached). Occupancy of designated areas is not permanent. Later on, the hosting Activity may replace a fragment with another (see Example-1)
- Static Binding: the Activity's GUI declares a portion of its layout as a < fragment > and explicitly supplies a reference to the first type of fragment to be held there using the "android:name=fragmentName" clause. This simple association does not require you to call the constructors (or pass initial parameters). A static binding is permanent, fragments cannot be replaced at run time (see Example-2)
- Multiple Fragments: the hosting activity may simultaneously expose any number of fragments using a combination of the strategies describe above. Fragments may interact with each other using the enclosing activity as a central store-and-forward unit (Example-3).

FRAGMENTS (DYNAMIC BINDING)

Fragments must be created inside a secure FragmentTransaction block.

You may use the method add() to aggregate a fragment to the activity. Optionally any view produced by the fragment is moved into an UI container of the host activity.

When you use the replace method to refresh the UI, the current view in the target area is removed and the new fragment is added to the activity's UI.

A faceless fragment may also be added to an activity without having to produce a view hierarchy.

STEPS

- 1. Obtain a reference to the FragmentManager, initiate a transaction:
  FragmentTransaction ft= getFragmentManager().beginTransaction();
- 2. Create an instance of your fragment, supply arguments if needed:
  FragmentBlue blueFragment= FragmentBlue.newInstance("some-value");
- 3. Place the fragment's view on the application's GUI: ft.replace(R.id.main_holder_blue, blueFragment);
- 4. Terminate the transaction: ft.commit();

FRAGMENTS (Dynamic binding – MainActivity - comments)

Each fragment is safely created inside a TRANSACTION frame demarcated by: beginTransaction … commit.

An invocation to the special newInstance constructor is used to supply any arguments a fragment may need to begin working.

Once created, the new fragment is used to replace whatever is shown at a designated area of the GUI (as defined in the activity_main.xml layout).

The method onMsgFromFragToMain implements the MainCallbacks interface. It accepts messages asynchronously sent from either redFragment or blueFragment to the enclosing MainActivity.

In our example, the row number selected from the blueFragment is forwarded to the redFragment using the fragment's callback method onMsgFromMainToFragment.

FRAGMENTS
(Dynamic binding – FragmentBlue - comments)

The Class.newInstance(…) construction is a reflective method commonly used for creating instances of classes (regardless of the number of parameters).

Creating an Android fragment begins with the making of a new Bundle in which each of the supplied arguments is stored as a <key,value> entry. Then the bundle is bound to the fragment trough the .setArguments(…) method. Finally the newly created fragment is returned.

In our example, the onCreate method verifies that the MainActivity implements the Java Interface defining methods needed to send data from the fragment to the MainActivity.

Fragments do most of their work inside of onCreateView. In this example, the predefined layout_blue.xml is inflated and plumbing is done to access its internal widgets (a TextView and a ListView).

A simple ArrayAdapter is used to fill the rows of the ListView.

An event handler is set on the ListView, so when the user clicks on a row its position is sent to the MainActivity's listener onMsgFromFragToMain.

FRAGMENTS
(Dynamic binding – Callbacks)

These Java Interfaces are used to enforce a formal Inter-Process-Communication behavior between the Fragments and the MainActivity. During their onCreate method each participant can check that the other has implemented the required listeners.

```
// method(s) to pass messages from fragments to MainActivity
public interface MainCallbacks
{
  public void onMsgFromFragToMain (String sender, String strValue);
}
```

```
// method(s) to pass messages from MainActivity to Fragments
public interface FragmentCallbacks
{
  public void onMsgFromMainToFragment(String strValue);
}
```

FRAGMENTS (STATIC BINDING)

Static binding is simple and requires less programming than dynamic binding.

This approach is appropriate for apps in which the interface retains the same fragment(s) for their entire session.

Statically attached fragments cannot be removed (however other fragments can be added to the interface).

The Activity's layout file uses the <fragment> element to mark the position and size of the area on which a fragment instance is to be injected.

The following attributes can be used to identify the fragment in case it needs to be restarted (if none is provided the fragment is identified by the system's id of the fragment's container id)

- 1. android:name="AppPackageName.FragmentClassName"
- 2. android:id="@id+/uniqueName" or android:tag="string"

FRAGMENTS (STATIC BINDING - COMMENTS)

In this example the onCreate method has nothing to do. Moreover, onCreateView is not even called, observe that the XML-layout clause android:name="csu.matos.fragments.FragmentXYZ" defines the specific fragment to be plugged in the activity's screen.

When a fragment is moved to the screen the onAttachFragment method is executed. This event is used here to keep a reference to the redFragment and the blueFragment.

Messages sent by the blueFragment to the MainActivity are caught in the onMsgFromFragToMain listener. As in the previous example, blueFragment messages are forwarded to the redFragment.

OPERATIONS ON FRAGMENTS

There are various operations that affect the presence and visibility of fragments dynamically bound to an activity. Those operations must be applied inside the scope of a FragmentTransaction object.

- add() Add a fragment to an activity (generally showing a view). If the activity is re-started due to a configuration-change, previously created fragments that appeared on the UI via add() can be reused (better performance, no need to re-create the fragment).
- remove() Remove a fragment from the activity. Fragment is destroyed (unless it was also added to the BackStack).
- replace() A fragment currently on the UI is destroyed and replace by another fragment.
- show() / hide() Shows a previously hidden fragment (hidden but not destroyed).
- attach() / detach() Attaches a fragment previously separated (detached) from the UI. Detached fragments are invisible but not destroyed.

OPERATIONS ON FRAGMENTS (Using the BackStack to recreate state)

Android-OS introduced a special stack to help fragments keep state when the user navigates from one UI to the other.

The artifact is called the BackStack and allows push/pop operations to manage FragmentTransactions. The BackStack mirrors the behavior of the

activity stack within a single activity

Remember that all Android devices include a Back button. When this button is pressed in succession the app transitions to the previous screen shown by the app until it ends. This mechanism provides a natural historical navigation (also known as Back-Navigation). Another important pattern of navigation known as Child-to-HighAncestor is discussed later.

Why should BackStack be used?

When the BackStack is used, the retrieved fragment is re-used (instead of recreated from scratch) and its state data transparently restored (no need for input/output state bundles). This approach leads to simpler and more efficient apps.

LIST-BASED WIDGETS (GUI design for selection making)

RadioButtons and CheckButtons are widgets suitable for selecting options offered by a small set of choices. They are intuitive and uncomplicated; however they occupy a permanent space on the GUI (which is not a problem when only a few of them are shown)

When the set of values to choose from is large, other Android List-Based Widgets are more appropriate.

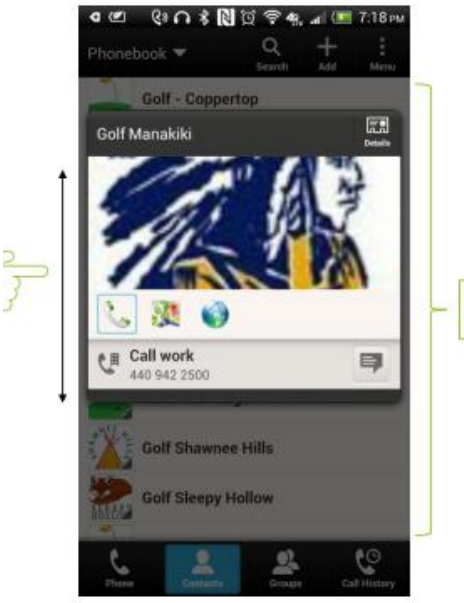
Example of List-Based Widgets include:

- ListViews
- Spinner
- GridView
- Image Gallery
- ScrollViews, etc.

LIST-BASED WIDGETS (Showing a large set of choices on the GUI)

The Android DataAdapter class is used to feed a collection of data items to a List-Based Widget.

The Adapter's raw data may come from a variety of sources, such as small arrays as well as large databases.



LIST-BASED WIDGETS (ListView)

The Android ListView widget is the most common element used to display data supplied by a data adapter.

ListViews are scrollable, each item from the base data set can be shown in an individual row.

Users can tap on a row to make a selection.

A row could display one or more lines of text as well as images.

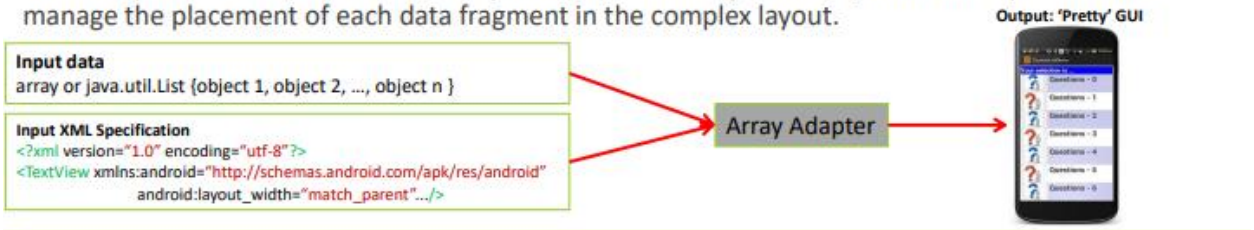
LIST-BASED WIDGETS (ListView - ArrayAdapter)

An ArrayAdapter<T> accepts for input an array (or ArrayList) of objects of some arbitrary type T.

The adapter works on each object by (a) applying its toString() method, and (b) moving its formatted output string to a TextView.

The formatting operation is guided by a user supplied XML layout specification which defines the appearance of the receiving TextView.

For ListViews showing complex arrangement of visual elements –such as text plus images- you need to provide a custom made adapter in which the getView(…) method explains how to manage the placement of each data fragment in the complex layout.



SPINNER

Android's Spinner is equivalent to a drop-down selector.

Spinners have the same functionality of a ListView but take less screen space.

An Adapter is used to supply its data using setAdapter(…)

A listener captures selections made from the list with setOnItemSelectedListener(…).

The setDropDownViewResource(…) method shows the drop-down multi-line window

GRIDVIEW

GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

Data items shown by grid are supplied by a data adapter.

Grid cells can show text and/or images

Some properties used to determine the number of columns and their sizes:

- android:numColumns: indicates how many columns to show. When used with option "auto_fit", Android determines the number of columns based on available space and the properties listed below.
- android:verticalSpacing and android:horizontalSpacing: indicate how much free space should be set between items in the grid.
- android:columnWidth: column width in dips.
- android:stretchMode: indicates how to modify image size when there is available space not taken up by columns or spacing .

HORIZONTALSCROLLVIEW

HorizontalScrollViews allow the user to graphically select an option from a set of small images called thumbnails

The user interacts with the viewer using two simple actions:

- 1. Scroll the list (left ↔ right)
- 2. Click on a thumbnail to pick the option it offers.

In our example, when the user clicks on a thumbnail the app responds by displaying a high-resolution version of the image

HORIZONTALSCROLLVIEW
(Example - Populating the HorizontalScrollView)

Our HorizontalScrollView will expose a list of frames, each containing an icon and a caption below the icon.

The frame_icon_caption.xml layout describes the formatting of icon and its caption. This layout will be inflated in order to create run-time GUI objects.

After the current frame is filled with data, it will be added to the growing set of views hosted by the scrollViewgroup container (scrollViewgroup is nested inside the horizontal scroller).

Each frame will receive an ID (its current position in the scrollViewgroup) as well as an individual onClick listener.

(Ghi chú: Tham khảo tài liệu GIẤY và chọn câu hợp lý nhất)

1. Hệ điều hành nền tảng cho Android là:

a. Linux b. Mac OS c. Ubuntu d. Windows

2. Theo mặc định trong android studio, trong quá trình phát triển ứng dụng, tập tin chứa thông tin về các tính năng và thành phần cơ bản của ứng dụng là:

a. res/layout b. AndroidManifest.xml c. res/values d. Build.gradle

3. Điện thoại đầu tiên sử dụng hệ điều hành Android là?

a. LG Optimus One b. Samsung Galaxy S c. Motorola Droid d. T-Mobile G1

4. Để mở máy ảo Android Emulator ta chọn mục nào?

a. AVD Manager b. SDK Manager c. JDK Manager d. AD VManager

5. Thành phần nào để truyền dữ liệu giữa các activities trong Android?

a. Fragment b. Broadcast receiver c. Content Provider d. Intent

6. Để sắp xếp các view trên giao diện theo chiều đứng hoặc ngang thì chọn Layout nào?

a. Linear Layout b. Relative Layout c. Table Layout d. Grid View

7. Tùy chọn nút (button) có thể được chọn từ danh mục bảng nào?

a. textfields b. layouts c. containers d. widgets

8. AndroidManifest có chức năng gì trong màn hình Android Studio?

a. Là tập tin thiết lập các quyền cho Activity, Service, ...

b. Là thư mục chứa Activity, Service, ...

c. Là nơi lưu trữ mã XML cho giao diện d. Là chương trình mặc định trong Android Studio

9. Android chủ yếu đang sử dụng bằng:

a. Python code b. Java code c. C code d. C# code

10. Công ty nào phát triển Android đầu tiên?

a. Nokia b. Android Inc c. Apple d. Google

11. Một loại phần tử bố trí cho phép mô tả vị trí tương đối các con của nó là:

a. RelativeLayout b. TextviewLayout c. ConstraintLayout d. LinearLayout

12. Một trong các thành phần ứng dụng, quản lý các dịch vụ nền của ứng dụng gọi là:

a. Content Providers b. Services c. Broadcast Receivers d. Activities

13. Những layouts nào không có trong android?

a. Frame Layout b. Farme Layout c. AbsoluteLayout d. Linear Layout

14. Thuộc tính nào sau đây được sử dụng để đặt màn hình activity theo hướng ngang?

a. screenorientation = "landscape" b. android:screenOrientation= "landscape"

c. screenOrientation= "landscape" d. android:ScreenOrientation= "landscape"

15. Thuộc tính android:capitalize trong view Textview có chức năng gì?

a. Hiển thị chữ viết hoa đầu tiên của text b. Hiển thị chữ viết thường của text

c. Chỉ định kiểu text hiển thị d. Viết hoa toàn bộ text

16. Một class cho phép hiển thị thông báo trên cửa sổ logcat là?

a. Log class b. Show class c. makeTest class d. Toast Class

17. Android hỗ trợ bao nhiêu định hướng?

a. 8 b. 2 c. 6 d. 4

18. Quá trình chuyển đổi Java thành dạng có thể đọc được của Android được gọi là:

a. debugging b. compiling c. linking d. testing

19. Liên minh các công ty phát triển các tiêu chuẩn mở cho thiết bị di động được công bố vào?

a. 2006 b. 2005 c. 2007 d. 2008

20. Chiếc điện thoại chạy hệ điều hành Android đầu tiên được bán ra là?

a. Nesux b. HTC One c. HTC Dream d. HTC Dearm

21. Thuộc tính nào bắt buộc phải khai báo khi sử dụng Layout?

a. Layout_height b. Layout_width c. id d. Tất cả đều đúng

22. Thành phần của android studio hoạt động như một trình giả lập cho các thiết bị được gọi là:

a. firmware b. emulator c. driver d. stub

23. Thành phần Android hiển thị một phần của activity trên màn hình được gọi là:

a. manifest b. fragment c. intent d. view

24. Môi trường Android cần gì để người lập trình phát triển:

a. JDK b. IDE c. APK d. SDK

25. Theo mặc định trong android studio trong quá trình phát triển ứng dụng, tệp chứa thông tin về SDK, phiên bản, id ứng dụng, ... là:

a. res/values b. AndroidManifest.xml c. Build.gradle d. res/layout

26. Phương thức nào được gọi trong một Activity khi một Activity khác được gọi?

a. onPause( ) b. onStop( ) c. onDestroy( ) d. onStart( )

27) Phương thức nào dùng để ánh xạ đến các view của Android qua thuộc tính id?

a. findViewById(String id) b. findViewById(int id)

c. retrieveResourceById(int id) d. findViewByReference(int id);

28. Trong android studio, các tùy chọn nhanh có thể được truy cập từ:

a. Editor tab b. Menu bar c. Navigation bar d. Tool bar

29. Một lớp dùng để hiển thị thông điệp cho người dùng là:

a. makeTest class b. Log class c. Toast class d. Show class

30. Thư viện Android cung cấp thao tác văn bản và hiển thị ứng dụng là:

a. android.os b. android.text c. android.webkit d. android.view

31. Trong android studio, mỗi activity mới được tạo phải được định nghĩa bởi:

a. AndroidManifest.xml b. Build.gradle c. res/values d. res/layout

32. Đâu là layout trong Android?

a. TableLayout b. RelativeLayout c. LinearLayout d. Tất cả đều đúng

33. Thành phần quản lý giao diện và định dạng trên màn hình trong Android gọi là:

a. fragment b. intent c. view d. layout

34. Thuộc tính android:editable trong EditText có chức năng gì?

a. Điều chỉnh text trong Table b. Cho phép điều chỉnh text

c. Không cho phép điều chỉnh text d. Đáp án (b) hoặc (c) đều ĐÚNG

35. Đâu là IDE chính thức để phát triển Android?

a. Net beans b. Android studio c. Java d. Eclipse

36. Một kiểu của bố cục trình bày cho phép bố trí tất cả các phần tử theo thứ tự là:

a. TextviewLayout b. ConstraintLayout

c. RelativeLayout d. LinearLayout

37. Layout hoặc thiết kế của ứng dụng android được lưu trong file:

a. * dex b. * text c. * java d. * xml

38. Muốn sắp xếp các view theo dạng cột và dòng thì ta dùng Layout nào?

a. Linear Layout b. Table Layout c. Grid View d. List View

39. Khái niệm smartphone xuất hiện lần đầu vào năm nào?

a. 1980 b. 1995 c. 1990 d. 1997

40. Sự khác biệt cơ bản giữa smartphone (SP) và feature phone (FP)

a. SP có API, FP không có b. SP có nhiều ứng dụng hơn FP

c. SP có HĐH, FP không có d. Tất cả đều sai

41. Ý nghĩa từ cell trong "cell phone" là gì?

a. Mạng chia ô b. Thuật ngữ mô tả bộ phận bên trong điện thoại

c. Tên thương mại d. Tất cả các câu đều sai

42. Các bên thứ 3 sử dụng gì để mở rộng chức năng cho thiết bị Android?

a. Android API b. Android Framework c. Android Library d. Android Core

43. SGL là thư viện về?

a. Xử lý chuỗi b. Xử lý ngày tháng c. Xử lý đồ họa d. Xử lý tập tin

44. Content Providers thuộc lớp nào trong kiến trúc Android?

a. Application b. Application Framework c. Libraries d. Linux Kernel

45. Các thành phần trong lớp Libraries được viết bằng ngôn ngữ nào?

a. Java b. C/C++ c. Visual Basic d. Mã máy

46. IPC là viết tắt của cụm từ nào?

a. Inter-process communication b. Internet protocol communication

c. Inter-process component d. Internet processing component

47. Dalvik là gì?

a. Ngôn ngữ lập trình b. Tên khác của Android c. Máy ảo trong Android d. Thư viện

48. Android dùng công cụ gì để lưu trữ dữ liệu có cấu trúc?

a. DB2 b. Oracle c. SQL d. SQLite

49. Kiến trúc Android có bao nhiêu lớp?

a. 1 b. 2 c. 3 d. 4

50. Webkit là gì?

a. Mã nguồn mở b. Ngôn ngữ lập trình c. a và b đúng d. a và b sai

51. Lớp Application Framework viết bằng ngôn ngữ gì?

a. Java b. C/C++ c. Python d. Visual Basic

52. System process chứa gì?

a. Activity Manager b. Data Provider c. Backstack d. Activity

53. Activity Manager làm nhiệm vụ gì?

a. Chứa Backstack b. Quản lí chu kì sống ứng dụng c. a và b đúng d. a và b sai

54. Package Manager làm nhiệm vụ gì?

a. Quản lí việc cài đặt ứng dụng b. Quản lí bộ nhớ ứng dụng

c. Quản lí nguồn năng lượng d. Tất cả đều sai

55. Mục tiêu của intent receiver?

a. Đăng kí một đoạn mã chờ được kích hoạt b. Kích hoạt một đoạn mã đã đăng kí

c. a và b đúng d. a và b sai

56. Khi muốn thực hiện một tác vụ ngầm, ta dùng

a. Activity b. Service c. Content Provider d. Intent Receiver

57. Tầng Application trong kiến trúc Android là?

a. Các ứng dụng b. Các thư viện c. a và b đều đúng d. a và b đều sai

58. Nguyên lí thiết kế của Android là?

a. Tái sử dụng b. Thay thế c. a và b đều đúng d. a và b đều sai

59. Có thể hiểu đơn giản Intent là gì?

a. Yêu cầu dịch vụ b. Thư viện lập trình c. Tất cả đều đúng d. Tất cả đều sai

60. Cấu tạo thành phần của Intent

a. Action, Data và Category b. Action, Service, Data, và Category

c. Service, Data d. Tất cả đều đúng

61. Ý nghĩa của "ACTION_CALL tel:123"?

a. Hiển thị bàn phím số điện thoại b. Thực hiện cuộc gọi c. a và b đúng d. a và b sai

62. Khi không đặc tả category, activity sẽ có loại?

a. android.intent.category.LAUNCHER b. android.intent.category.BROWSABLE

c. android.intent.category.HOME d. android.intent.category.DEFAULT

63. Muốn dùng các tính năng của thiết bị, cần khai báo thẻ nào trong AndroidManifest?

a. <meta-data> b. <intent-filter> c. <uses-permission> d. <data>

64. Tập tin apk là?

a. Tập tin cài đặt ứng dụng b. Chứa bytecode c. a và b đúng d. a và b sai

65. Tập tin apk được thực thi trong?

a. Java virtual machine b. Dalvik virtual machine c. a và b đúng d. a và b sai

66. Ta có thể lập trình Android trên các hệ điều hành nào?

a. Windows b. Linux c. MAC OS X d. Tất cả đều đúng

67. Ứng dụng Android có thể bao gồm các thành phần nào?

a. Activity b. Service, Content provider c. Broadcast receiver d. Tất cả đều đúng

68. Một activity có thể hiểu như?

a. Một windows-Form b. Một tiểu trình c. Chu kì sống ứng dụng d. Tất cả đều sai

69. Một ứng dụng có thể chọn bao nhiêu Activity làm màn hình chính?

a. 1 b. 2 c. 3 d. Tất cả đều sai

70. Service là gì?

a. Activity không có giao diện b. Thành phần chạy ngầm

c. a và b đúng d. a và b sai

71. Broadcast receiver có thể làm gì?

a. Khởi động một activity b. Dùng cơ chế notification c. a và b sai d. a và b đúng

72. Các tập dữ liệu toàn cục (Global Dataset) được lưu trong?

a. SQL Database b. DB2 Database c. SQLite Database d. Tất cả đều sai

73. Các activity trong hệ thống được lập lịch bởi?

a. Dalvik virtual machine b. Activity Stack c. Tất cả đều đúng d. Tất cả đều sai

74. Khi người dùng nhấn nút "Back", activity hiện hành sẽ?

a. Bị ngắt b. Bị loại bỏ khỏi stack c. Tất cả đều sai d. Tất cả đều đúng

75. Activity có bao nhiêu phương thức chuyển trạng thái cơ bản?

a. 5 b. 6 c. 7 d. 8

76. Các trạng thái nào của activity có khả năng bị hệ thống hủy cao nhất?

a. Running b. Stopped c. Paused d. Tất cả đều sai

77. Khi activity hiện hành bị một activity khác che một phần, nó sẽ rơi vào trạng thái nào?

a. Paused b. Stopped c. a và b sai d. a và b đúng

78. Khi activity hiện hành bị một activity khác che khuất hoàn toàn, nó sẽ rơi vào trạng thái nào?

a. Paused b. Stopped c. a và b sai d. a và b đúng

79. Khi muốn lưu dữ liệu theo dạng danh sách "key – value", ta nên chọn

a. Preferences b. SQLite c. a và b đúng d. a và b sai

80. Khi tạo Preferences bằng phương thức getSharedPreferences của activity, tập tin Preferences đó có thể được sử dụng bởi?

a. Nội bộ trong ứng dụng b. Các ứng dụng khác c. a và b đúng d. a và b sai