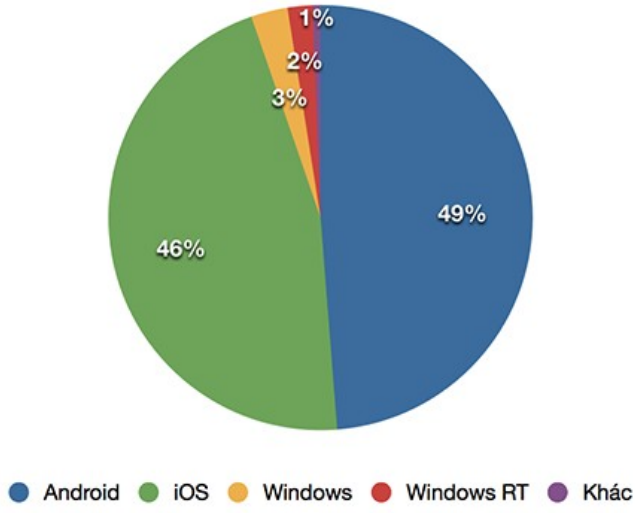
Chương 1: Các kiến thức cơ bản dùng cho lập trình ứng dụng trên Android

1.1 Hệ điều hành Android

1.1.1 Hệ điều hành Android. Các loại thiết bị cài đặt Android

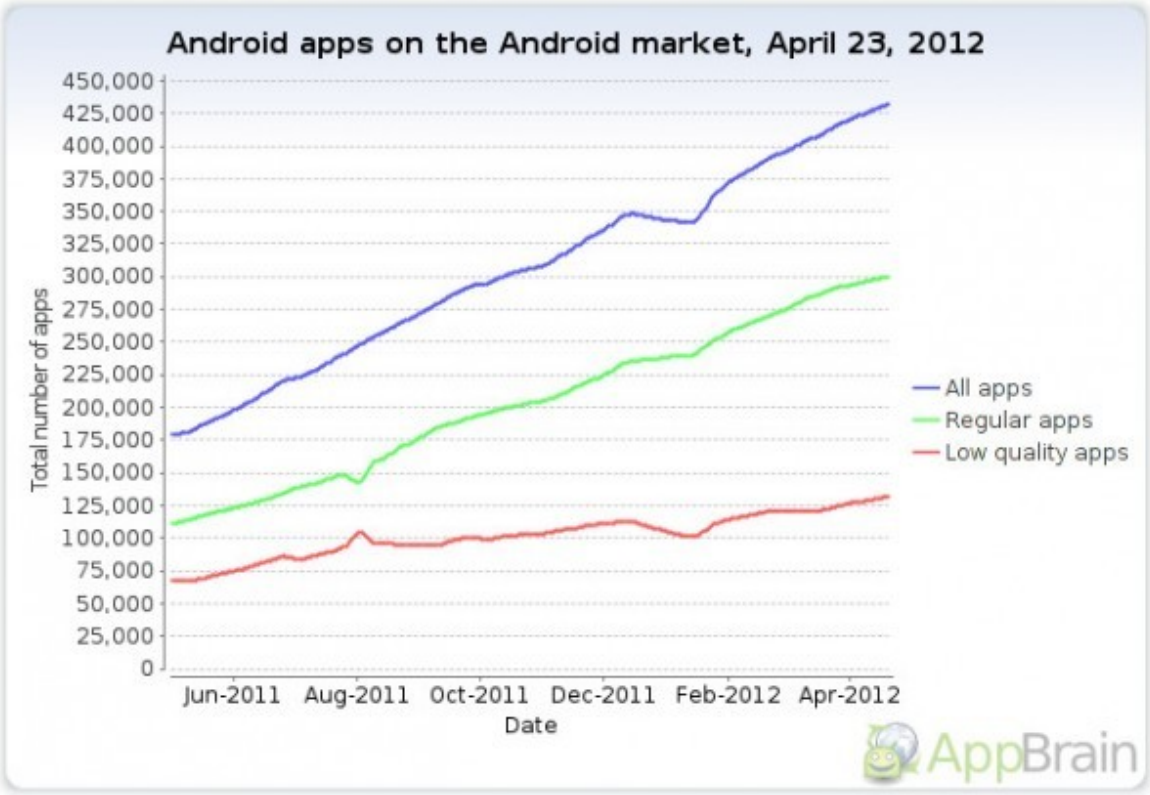
Android là một hệ điều hành dành cho thiết bị di động như điện thoại, máy tính bảng và netbooks. Android được phát triển bởi Google, dựa trên nền tảng Linux và các phần mềm mã nguồn mở. Android được phát triển nhằm cạnh tranh với các hệ điều hành di động khác như iOS (Apple), BlackBerry OS (BlackBerry), Windows Mobile (Microsoft), Symbian(Nokia). Bada (Samsung) ... Trên điện thoại di động *Công ty nghiên cứu thị trường Canalys đã ước tính trong quý 2 năm 2009 rằng Android có 2,8% thị phần điện thoại thông minh được bán ra toàn cầu. Đến quý 4 năm 2010 con số này tăng lên 33% thị phần, trở thành nền tảng điện thoại thông minh bán chạy hàng đầu. Đến quý 3 năm 2011 Gartner ước tính rằng hơn một nửa (52,5%) thị trường điện thoại thông minh thuộc về Android Đến quý 3 năm 2012 Android đã có 75% thị phần điện thoại thông minh toàn cầu theo nghiên cứu của hãng IDC. (Nguồn* ***http://vi.wikipedia.org/wiki/Andoid****).* Trên lãnh vực máy tính bảng, Công ty nghiên cứu thị trường IDC vừa đưa ra dự báo rằng thị phần của máy tính bảng Android có thể sẽ đạt mức cao nhất là 48,8% trong năm nay (2013) (Theo http://tintuc.vnn.vn/)



Hệ điều hành máy tính bảng	Thị phần năm 2013	Thị phần năm 2017
Android	48.80%	46.00%
iOS	46.00%	43.50%
Windows	2.80%	7.40%
Windows RT	1.90%	2.70%
Khác	0.60%	0.40%
Tổng	**100.00%**	**100.00%**

Dự báo thị phần của các hệ điều hành dành cho máy tính bảng trong năm 2013 và 2017 (nguồn IDC)

Android có một cộng đồng phát triển ứng dụng lớn. AppBrain - trang web cung cấp các ứng dụng Android tốt nhất thống kê rằng tính đến tháng 04/2012 Google Play đang sở hữu 430.000 ứng dụng, trong đó 311.000 (72%) ứng dụng miễn phí.



Các thiết bị cài đặt Android rất đa dạng về kích thước và chủng loại. Hệ điều hành Android có thể chạy trên các loại thiết bị sau

- Điện thoại thông minh (Smartphone)
- Máy tính bảng (Tablet)
- Các thiết bị đọc điện tử (E-reader devices)
- Netbook
- Máy nghe nhạc MP4

- Internet Tivi, Smart TV

1.1.2 Các phiên bản Android

(Tham khảo nguồn: *http://en.wikipedia.org/wiki/Android_version_history; http://nhipsongso.tuoitre.vn/Kien-thuc-Cong-nghe/519360/Luoc-su-Android-qua-cac-phien-ban.html)*

- **Phiên bản 1.0.**
  - Ngày công bố: 23-11-2008.
  - Tên mã : Astro Boy hay Bender.
  - Phiên bản Linux kernel 2.6.25
  - Tính năng:
    + Android 1.0 rất nguyên sơ, tích hợp sẵn khả năng đồng bộ dữ liệu với các dịch vụ trực tuyến của Google như Gmail, Google Calendar và Contacts, một trình phát media, hỗ trợ Wi-Fi và Bluetooth, thanh trạng thái hiển thị các thông báo ứng dụng và một ứng dụng chụp ảnh (camera) tuy chưa cho phép thay đổi độ phân giải và chất lượng ảnh.
- **Phiên bản 1.5.**
  - Ngày công bố: 30-4-2009.
  - Tên mã : Cupcake..
  - Phiên bản Linux kernel 2.6.27
  - Tính năng:
    + Cupcake mang nhiều tính năng mới như bàn phím ảo có khả năng dự đoán từ đang gõ, từ điển từ ngữ do người dùng đặt ra, hỗ trợ widget trên giao diện, quay phim và phát lại video clip, lược sử thời gian cuộc gọi, chế độ tự động xoay màn hình theo hướng sử dụng. Trình duyệt web trong Cupcake có thêm khả năng sao copy/paste.
- **Phiên bản 1.6.**
  - Ngày công bố: 30-9-2009.
  - Tên mã : Donut.
  - Phiên bản Linux kernel 2.6.29
  - Tính năng:
    + Cung cấp chức nanưg tìm kiếm nhanh; đọc văn bản.; xóa nhiều file ảnh cùng lúc, ...

- **Phiên bản 2.0.**
    - Ngày công bố: 26-10-2009.
    - Tên mã : Éclair.
    - Phiên bản Linux kernel 2.6.29
    - Tính năng:
        + Cho phép quản lý nhiều tài khoản Email, tìm kiếm tin nhắn, Tìm kiếm bằng giọng nói; Google map; Hướng đến các thế hệ smartphone màn hình lớn.
- **Phiên bản 2.2.**
    - Ngày công bố: 20-5-2010.
    - Tên mã : Froyo.
    - Phiên bản Linux kernel 2.6.32
    - Tính năng:
        + Hỗ trợ Flash; cho phép biến chiếc smartphone thành thiết bị phát sóng Wi-Fi; cho phép cài đặt ứng dụng lên thẻ nhớ SD thay vì mặc định cài ngay vào bộ nhớ trong của thiết bị; mật khẩu đã hỗ trợ số và chữ số.
- **Phiên bản 2.3.**
    - Ngày công bố: 6-12-2010.
    - Tên mã : Gingerbread.
    - Phiên bản Linux kernel 2.6.35
    - Tính năng:
        + Hỗ trợ thiết kế giao diện đơn giản và hiệu quả; Nhập văn bản thông minh và nhanh hơn; hỗ trợ chức năng copy và dán; hỗ trợ tính năng dọn rác. Đến cuối năm 2012, Gingerbread vẫn đang "phủ sóng" trên rất nhiều thiết bị dùng Android, chiếm đến hơn phân nửa (54%)
- **Phiên bản 3.0.**
    - Ngày công bố: 6-12-2010.
    - Tên mã : Honeycomb.
    - Phiên bản Linux kernel 2.6.36
    - Tính năng:
        + Đây có thể xem là một thế hệ Android đầu tiên dành riêng cho máy tính bảng (tablet). Android 3.0 cải tiến giao diện phù

hợp với cách sử dụng máy tính bảng, bàn phím ảo thân thiện hơn, hỗ trợ xử lý đa tác vụ (multi-tasking), cho phép chuyển đổi qua lại các ứng dụng đang cùng chạy. Phần lõi hệ thống có các cải tiến tương thích với phần cứng như hỗ trợ chip xử lý (CPU) đa lõi, tăng tốc phần cứng.

- **Phiên bản 4.0.**
  - Ngày công bố: 6-12-2010.
  - Tên mã : Ice Cream Sandwich.
  - Phiên bản Linux kernel 3.0.1
  - Tính năng:
    + Đây là sự kết hợp của phiên bản 3.x cho máy tính bảng và 2.x cho điện thoại di động. Phiênbản này có các tính năng ưu việt như: giao diện đẹp hơn, widget có thể thay đổi kích thước, cho phép khóa màn hình, ...
- **Phiên bản mới nhất 4.2.**
  - Ngày công bố: 6-12-2010.
  - Tên mã : Jelly Bean.
  - Phiên bản Linux kernel 3.4
  - Tính năng:
    + Android 4.2 tiếp tục mang đến những cải tiến hấp dẫn cho ứng dụng chụp ảnh (Camera) như chụp ảnh trung thực HDR (High Dynamic Range), chụo ảnh rộng Photo Sphere, hiệu ứng ảnh, tìm kiếm thông minh và đẹp hơn Google Now, đưa tính năng lướt chọn từ rất hay trong bàn phím ảo. Chức năng hỗ trợ nhiều tài khoản người dùng (multi-user profile) lần đầu tiên được áp dụng trong Android 4.2 nhưng chỉ có người dùng máy tính bảng thừa hưởng chức năng này.

Đến cuối năm 2012 vẫn còn đến 54% thiết bị Android dùng Gingerbread (Android 2.3), Ice Cream Sandwich (Android 4.0) theo sau với 25,8%. Thế hệ Jelly Bean mới nhất còn khá ít ỏi với 2,7% thiết bị sử dụng.

1.1.3 Ưu và nhược điểm của Android.

Vì Android là hệ điều hành mã nguồn mở, nên có những ưu và nhược điểm sau:

1.1.3.1 Ưu điểm

- **An ninh:** Các lỗi nhanh chóng được phát hiện và sửa đổi
- **Chất lượng:** Các ứng dụng không ngừng được cải tiến, phù hợp với nhu cầu sử dụng của nhiều người.
- **Khả năng tùy biến:** Những đoạn mã trong chương trìn được công khai, nên người dùng có thể thêm bớt các chức năng túy ý muốn.
- **Chi phí:** Sử dụng sản phẩm mã nguồn mở hoàn toàn không tốn phí, tiết kiệm kinh phí.

1.1.3.2. Nhược điểm

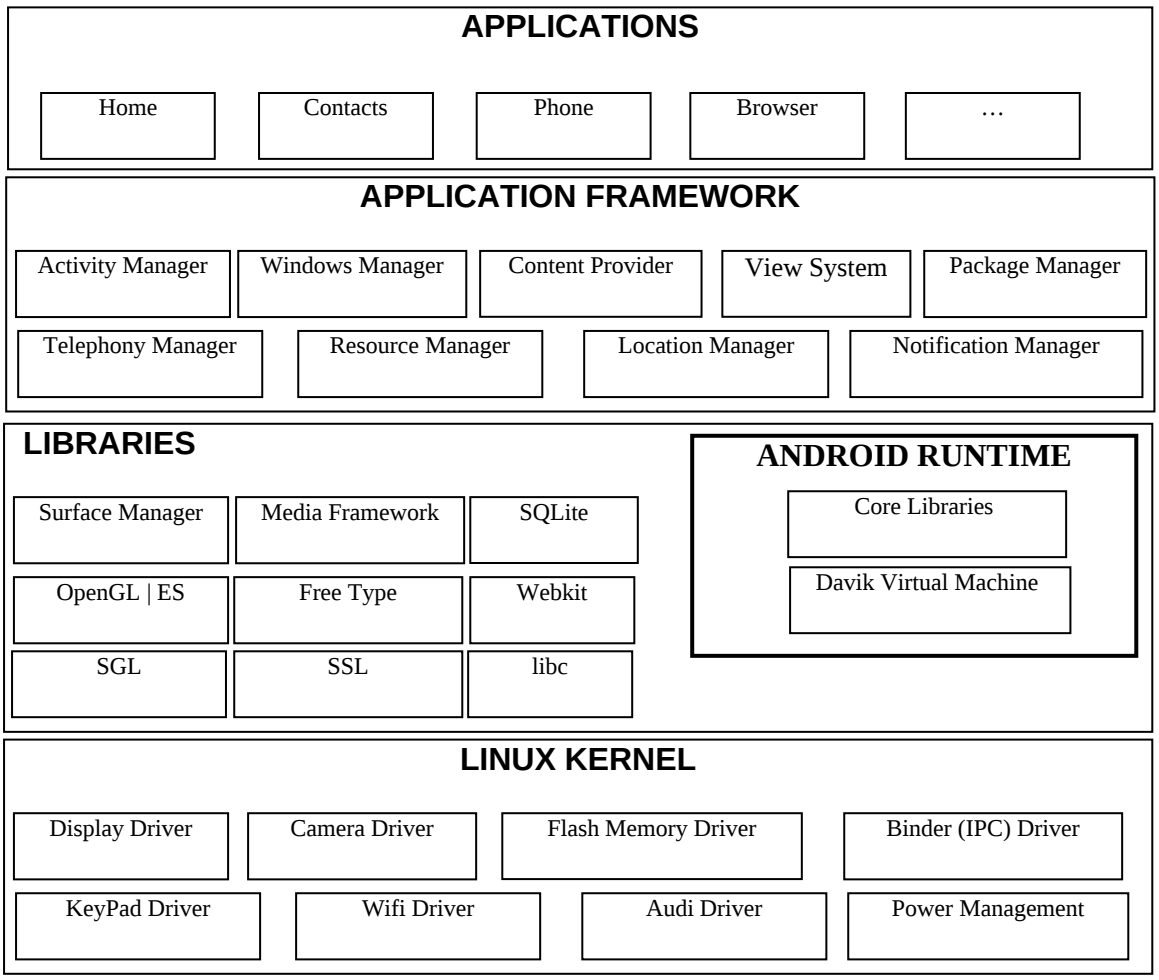
- **Khả năng bảo mật:** Vì Android là hệ điều hành mã nguồn mở nên tất cả các thông tin về hệ thống mọi người đều nắm được. Đây là ưu điểm và cũng là nhược điểm, bởi vì các hacker có thể tìm kiếm những lỗ hổng hệ thống và tạo ra những mã độc.

1.1.4 Nền tảng hệ điều hành Android

https://docs.google.com/document/preview?hgd=1&id=1ehWmLcA4DmeQ2GDYqzBvUyby3BwdUHRMuaP_6tX7dOM&pli=1#

1.1.4.1 Các thành phần của Android

Lược đồ sau thể hiện các thành phần của hệ điều hành Android:



Có 5 tầng phân biệt trong hệ thống Android.

- **Applications**

Hệ điều hành Android tích hợp sẵn một số ứng dụng cơ bản như email client, SMS, lịch điện tử, bản đồ, trình duyệt web, sổ liên lạc và một số ứng dụng khác. Ngoài ra tầng này cũng chính là tầng chứa các ứng dụng được phát triển bằng ngôn ngữ Java.

- **Application Framework**

Trong tầng này thành phần quan trọng nhất là Acitivity Manager vì nó quản lý vòng đời của một Activity.

Tầng này chứa các thư viện Java hỗ trợ người dùng giao tiếp với tầng Android Framework. Một phần của thư viện này do Google cung cấp sẵn, một phần do ta tạo ra. Nhờ vậy các nhà phát triển ứng dụng có khả năng tạo ra các ứng dụng vô cùng sáng tạo và phong phú. Các nhà phát triển ứng dụng được tự do sử dụng các tính năng cao cấp của thiết bị phần cứng như: thông tin định vị

địa lý, khả năng chạy dịch vụ dưới nền, thiết lập đồng hồ báo thức, thêm chú thích (notification) vào thanh trạng thái (status bar) của màn hình thiết bị…

Người phát triển ứng dụng được phép sử dụng đầy đủ bộ API được dùng trong các ứng dụng tích hợp sẵn của Android. Kiến trúc ứng dụng của Android được thiết kế nhằm mục đích đơn giản hóa việc tái sử dụng các thành phần (component). Qua đó bất kì ứng dụng nào cũng có thể công bố các tính năng mà nó muốn chia sẻ cho các ứng dụng khác (Ví dụ: Ứng dụng email muốn các ứng dụng khác có thể sử dụng tính năng gởi mail của nó).

Tầng này bao gồm một tập các services và thành phần sau:

+ **View System**: dùng để xây dựng ứng dụng có các đối tượng như: list, grid, text box, button và thậm chí là một trình duyệt web có thể nhúng vào ứng dụng
+ **Content Provider**: Cho phép các ứng dụng có thể truy xuất dữ liệu từ các ứng dụng khác hoặc chia sẽ dữ liệu của chúng.
+ **Resource Manager:** Cung cấp khả năng truy xuất các tài nguyên non-code như hình ảnh hoặc file layout.
+ **Notification Manager**: Cung cấp khả năng hiển thị custom alert trên thanh status bar.
+ **Activity Manager:** Đây là thành phần quan trọng nhất giúp ứng dụng quản lý vòng đời của một Activity.
+ **Telephony Manager:** Cung cấp thư viện để truy xuất đến các dịch vụ điện thoại cũng như thông tin thuê bao.
+ **Location Manager:** Cung cấp thư viện hỗ trợ người dùng định vị ví trí của thiết bị.

▪ **Libraries**

+ **System C library**: Tập thư viện hệ thống C|C^{++} chuẩn (libc) có thể gọi thông qua giao diện Java.
+ **Media FrameWork:** Bộ thư viện hỗ trợ trình diễn và ghi các định dạng âm thanh và hình ảnh phổ biến.
+ **Surface manager:** Cho phép tạo các cửa sổ giao diện
+ **OpenGL:** Hỗ trợ xây dựng các ứng dụng đồ họa 2D và 3D.
+ **SSL:** Cung cấp chức năng bảo mật thiết bị.
+ **SGL:** Engine hỗ trợ đồ họa 2D.
+ **Free Type:** Hỗ trợ các Font Bitmap và vector.
+ **SQLite:** Cung cấp bộ máy cơ sở dữ liệu được nhúng trong thiết bị.

+ **Webkit:** hỗ trợ hiển thị nội dung website.

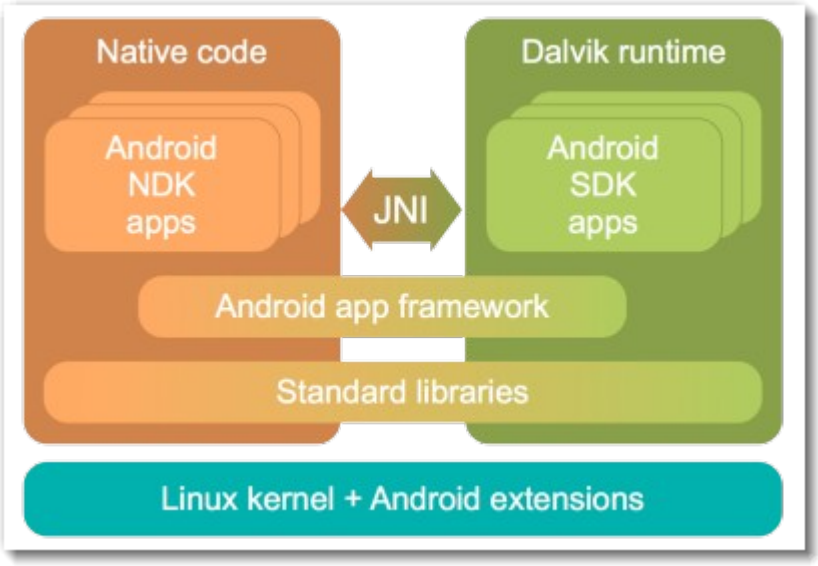
- **Android Runtime**

Hệ điều hành Android tích hợp sẵn một tập hợp các thư viện cốt lõi cung cấp hầu hết các chức năng có sẵn trong các thư viện lõi của ngôn ngữ lập trình Java. Mọi ứng dụng của Android chạy trên một tiến trình của riêng nó cùng với một thể hiện của máy ảo Dalvik. Máy ảo Dalvik thực tế là một biến thể của máy ảo Java được sửa đổi, bổ sung các công nghệ đặc trưng của thiết bị di động. Nó được xây dựng với mục đích làm cho các thiết bị di động có thể chạy nhiều máy ảo một cách hiệu quả. Trước khi thực thi, bất kì ứng dụng Android nào cũng được chuyển đổi thành file thực thi với định dạng nén Dalvik Executable (.dex). Định dạng này được thiết kế để phù hợp với các thiết bị hạn chế về bộ nhớ cũng như tốc độ xử lý. Ngoài ra máy ảo Dalvik sử dụng bộ nhân Linux để cung cấp các tính năng như luồng (thread), quản lý bộ nhớ thấp (low-level memory management).

- **Linux Kernel**

Hệ điều hành Android được xây dựng trên bộ nhân Linux 2.6 cho những dịch vụ hệ thống cốt lõi như: security, memory management, process management, network stack, driver model. Bộ nhân này làm nhiệm vụ như một lớp trung gian kết nối phần cứng thiết bị và phần ứng dụng.

Dưới đây là mô hinh hợp tác giữa máy ảo Dalvik và Navite code:



Hình 3 - Mô hình hợp tác giữa máy ảo Dalvik và Navite code

JNI: Java Native Interface (Tương tự khái niệm Application Programming Interface).

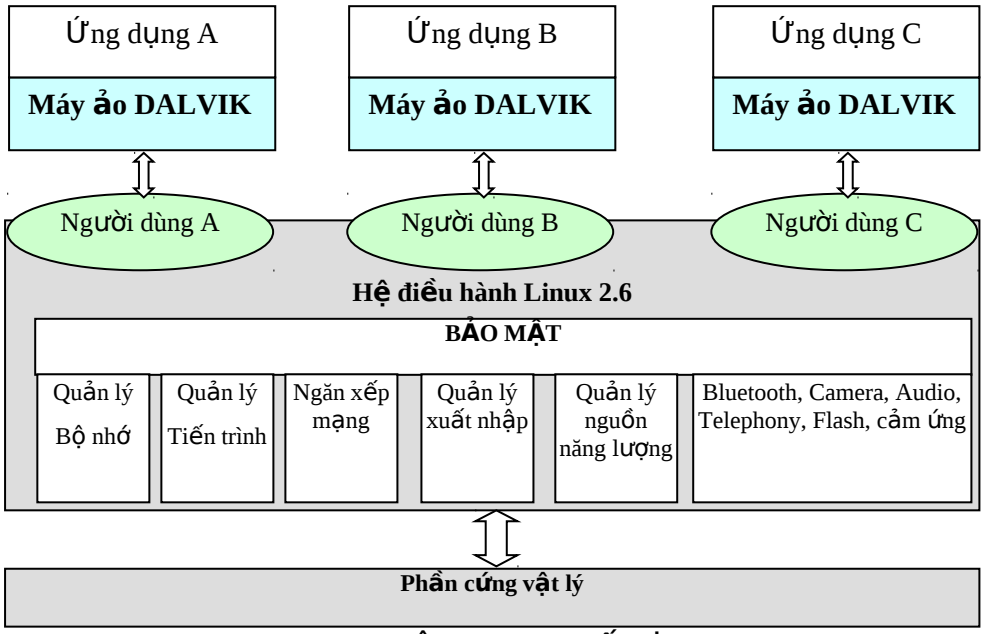
Java Native Interface là một bộ framework cho phép mã lệnh viết bằng Java chạy trên máy ảo java có thể gọi hoặc được gọi bởi một ứng dụng viết bằng native code (Ứng dụng được viết cho một phần cứng cụ thể và trên một hệ điều hành cụ thể) hoặc những bộ thư viện viết bằng C, C++ hoặc Assembly. Bằng cách sử dụng JNI, Android cho phép các ứng dụng chạy trên máy ảo Dalvik có thể sử dụng những phương thức được viết bằng các ngôn ngữ cấp thấp như: C, C++, Assembly. Qua đó các nhà phát triển ứng dụng có thể xây dựng ứng dụng dựa trên các bộ thư viện viết bằng C, C++, Assembly nhằm tăng tốc độ thực thi của ứng dụng hoặc sử dụng những tính năng mức thấp mà ngôn ngữ Java không hỗ trợ. Tuy nhiên người phát triển ứng dụng cần phải cân nhắc sự gia tăng độ phức tạp của ứng dụng khi quyết định sử dụng các bộ thư viện này.

1.1.4.2. Kết nối mạng (xem tài liệu Lập trình Android Nguyễn Thị Ngọc Tú)

1.1.4.3. Bảo mật (xem Nguyễn thị Ngọc Tú)

Android là một hệ thống đa tiến trình, trong đó mỗi ứng dụng chạy trong một tiến trình riêng biệt. Mỗi ứng dụng trong Android được gán một ID và mặc định một ứng dụng không có bất cứ quyền hạn gì tác động đến hệ điều hành, người dùng hoặc ứng dụng khác. Việc cho phép trao đổi thông tin và tương tác qua lại giữa các tiến trình và ứng dụng trong Android phải được định nghĩa trước trong ứng dụng để khi cài đặt hệ điều hành sẽ nhận diện được thông tin này.

Cơ chế bảo mật và ứng dụng Android có thể mô tả bằng hình ảnh sau



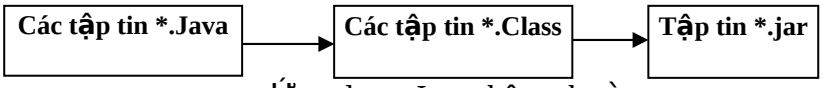
Hoạt động trong thiết bị Android

1.1.4.4. Tập tin DEX (xem Nguyễn thị Ngọc Tú)

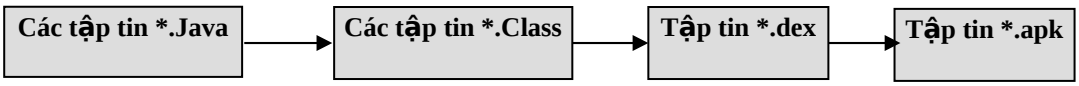
Trong môi trường Java chuẩn, mỗi lớp chứa mã nguồn Java được biên dịch thành các tập tin .class chứa mã nhị phân. Máy ảo Java có thể đọc được các tập tin .class này. Các tập tin .class được đóng gói thành tập tin .jar.

Trong nền tảng Android, mã nguồn Java cũng được biên dịch thành các tập tin .class. Máy ảo DALVIK không đọc được các tập tin .class này. Các tập tin .class được tổ chức thành tập tin .dex (Dalvik Executable). Máy ảo DALVIK sẽ thực thi tập tin .dex này. Tập tin .dex được đóng gói thành tập tin .apk. Người dùng tải tập tin .apk và Android sẽ cài đặt ứng dụng lên máy từ tập tin này.

Có thể so sánh và hình dung quá trình biên dịch và đóng gói các thành phần trong một dự án Java thông thường và dự án Android như sau:



Ứng dụng Java thông thường



Ứng dụng Android

1.2 Các nguyên tắc nguyên tắc lập trình tạo các ứng dụng trên Android

1.2.1 Những giới hạn của thiết bị điện thoại di động. Khác với lập trình tạo ứng dụng trên máy tính thông thường, khi tạo ứng dụng trên điện thoại di động chúng ta cần chú ý những giới hạn của nó.

- Bộ nhớ giới hạn.
- Khả năng xử lý giới hạn.
- Nguồn năng lượng giới hạn.
- Bàn phím nhỏ hoặc ít phím.
- Công nghệ và nền tảng khác nhau.
- Kích thước màn hình bé.
- Giao diện người dùng đơn giản
- Băng thông giới hạn
- Kết nối mạng không ổn định

1.2.2 Các yêu cầu để phát triển phần mềm trên nền tảng Android

+ **Kiến thức lập trình Java**. Ngôn ngữ Java sử dụng trong Android không phải là toàn bộ thư viện J2EE (Java 2 Platform, Enterprise Edition) mà những nhà phát triển Java thường sử dụng mà chỉ là một phần nhỏ được xem là máy ảo Dalvik.

Vì vậy, ta chỉ quan tâm đến những thư viện được dùng để viết các ứng dụng di động.

+ **Kiến thức về XML.** Ngôn ngữ XML được dùng để định nghĩa các tài nguyên cho ứng dụng: layout, menu, các trị (values), tập tin thông tin hệ thống Androidmanifest.XML.

+ **Các công cụ và môi trường phát triển.**

- **JDK (Java Development Kit)**. JDK là môi trường dùng để viết các ứng dụng Java. JDK gồm JRE (Java Runtime Environment) và các công cụ như: trình biên dịch (javac.exe), trình thực thi mã (java.exe), gỡ lỗi, thư viện phần mềm, bộ giả lập điện thoại,...
- **Eclipse.** Có nhiều môi trường phát triển tích hợp (IDE –Integrated Development Environment) miễn phí cho Java như Jgraph, Oracle JDeveloper, JEdit, NetBeans. Trong trường hợp Android, IDE được đề nghị là Eclipse.
- **Android SDK (Software Development Kit).** SDK là bộ công cụ phát triển ứng dụng Android. Sản phẩm này do Google xây dựng và phát hành miễn phí.
- **ADT (Android Development Tools).** Bộ công cụ mở rộng tính năng của Eclipse gồm:
  - **Dx (Dalvik Cross- Assembler).** Dùng để chuyển các lớp Java (đã biên dịch) thành một file nhị phân (*.dex) chạy trên máy ảo Dalvik
  - **Aapt (Android Asset Packing Tool).** Dùng để đóng gói (nén) các tập tin dạng *.dex thành file *.apk cho phép người dùng tải và cài đặt trên thiết bị.
  - **Adb (Android Debug Bridge).** Tạo cầu nối để chuyển và cài đặt mã nguồn của ứng dụng lên trình giả lập (Emulator) hoặc thiết bị Android.
  - **Ddms (Dalvik Debug Monitor Service).** Cung cấp những dịch vụ như: quản lý thông tin tiến trình và ngăn xếp, logcat, ...

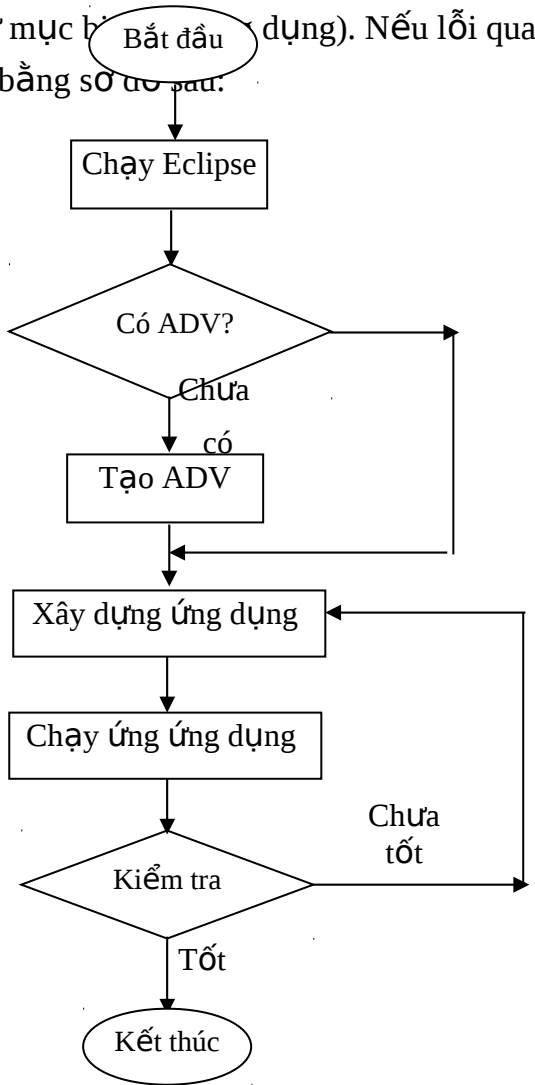
1.2.3 Qui trình cài đặt môi trường phát triển ứng dụng Android

Cài đặt JDK → Cài đặt Eclipse → Cài đặt Android SDK → Bổ sung ADT cho Eclipse

1.2.4 Qui trình tạo một ứng dụng Android

- **Bước 1:** Chạy Eclipse.
  - Chọn dự án Android (Android Application Project)
  - Khai báo tên ứng dụng, tên dự án, tên gói (package), chọn phiên bản Android (build SDK), phiên bản thấp nhất (Minimum Required SDK), nơi lưu ứng dụng (location)
  - Chọn hình biểu tượng (icon)
  - Tạo Activity rỗng (Blank Activity)
- **Bước 2:** Từ môi trường Eclipse, thiết lập máy ảo (Android Virtual Device) tương ứng với phiên bản SDK đã chọn ở bước 1 (nếu chưa thiết lập)
  - Chạy chức năng AVD Manager trong menu Windows.
  - Khai báo tên máy ảo (Name), phiên bản Android (Target), độ phân giải màn hình (Skin), ...
- **Bước 3:** Xây dựng ứng dụng: khai báo tài nguyên (Resource), tạo giao diện (Layout), tạo các thành phần ứng dụng (Component), viết mã xử lý (class), ...
- **Bước 4:** Chạy (run as), kiểm tra ứng dụng và kết thúc (sản phẩm là file *.apk trong thư mục b[illegible] dụng). Nếu lỗi quay lại bước 3

Có thể mô tả qui trình bằng sơ đồ sau:



1.2.5. Các thành phần cấu thành một dự án Android

Bất kỳ một ứng dụng Android nào cũng được cấu thành từ 3 thành phần chính: Application Components, tập tin AndroidManifest.XML, Application Resources

1.2.5.1 Application Components: Có 4 loại Application Components:

- **Activities:** Một Activity đại diện cho một cửa sổ chứa giao diện ứng dụng mà người dùng có thể tương tác trực tiếp. Trong mỗi Activity, ngoài việc thiết lập giao diện, nó còn phải xử lý những tương tác giữa người dùng với giao diện: như sự kiện touch, click ...Mỗi Activity được tạo ra trong ứng dụng sẽ là một lớp con kết thừa (extends) từ lớp Activity của nền tảng Android (**android.app.Activity**).
- **Services:** Đây là loại Application Component chạy nền để thực hiện những công việc liên tục và kéo dài. Service không có giao diện. Một Service được tạo ra trong ứng dụng là một lớp con kết thừa từ lớp Service của nền tảng Android (**android.app.Service**)
- **Content Providers:** Đây là thành phần dùng để quản lý một tập các dữ liệu chia sẻ được. Dữ liệu có thể lưu trữ dưới dạng tập tin, cơ sở dữ liệu SQLite hoặc trên trang web. Thông qua Content Provider, những ứng dụng có thể truy vấn hay chỉnh sửa dữ liệu nếu Content Providers cho phép. Một Content Provider được tạo ra trong ứng dụng là một lớp con kết thừa từ lớp Content Provider của nền tảng Android (**android.content.ContentProvider**)
- **Broadcast Receivers:** Đây là thành phần trong ứng dụng dùng để lắng nghe các thông điệp (Broadcast) được gởi đi từ hệ thống. Ví dụ hệ thống thông báo nguồn năng lượng trong máy gần cạn kiệt thì Broadcast Receivers sẽ lắng nghe và nhận thông điệp này để có những ứng xử thích hợp. Bản thân một ứng dụng cũng có thể gởi đi những thông điệp để những ứng dụng khác biết. Một Broadcast Receivers được tạo ra trong ứng dụng là một lớp con kết thừa từ lớp Broadcast Receivers của nền tảng Android (**android.content.BroadcastReceiver**)

1.2.5.2 Tập tin AndroidManifest.xml: Đây được xem là tập tin quan trọng nhất của dự án Android. Tập tin này chứa tất cả những thông tin của dự án. Trước khi ứng dụng được thực thi, hệ thống sẽ đọc những thông tin này. Tập tin AndroidManifest.xml chứa những thông tin cơ bản sau:

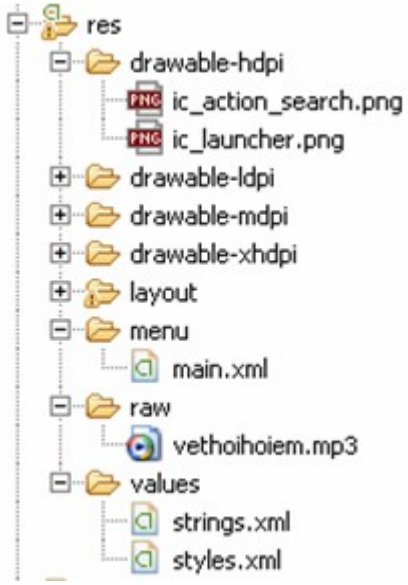
- Tên Package
- Các thành phần của ứng dụng hiện có: Activity, Services, Content Providers hoặc Broadcast Receiver.
- Quyền hạn của ứng dụng
- ....

Một tập tin AndroidManifest .xml mẫu

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="android.cook.test"
    android:versionCode="1"
    android:versionName="1.0">

    <uses-sdk android:minSdkVersion="9"
android:targetSdkVersion="15" />

    <application android:label="@string/app_name"
        android:icon="@drawable/ic_launcher"
        android:theme="@style/AppTheme">
          <receiver
android:name=".HelloAndroidWidgetProvider"
android:permission="android.permission..SET_WALLPAPER">
              <intent-filter >
                  <action
android:name="android.appwidget.action.APPWIDGET_UPDATE" />
              </intent-filter>
              <meta-data
android:name="android.appwidget.provider"

     android:resource="@xml/helloandroidprovider"/>
          </receiver>
    </application>
</manifest>
```

+ Application Resources

1.2.5.2 Application Resources (tài nguyên ứng dụng): được sử dụng để thiết kế giao diện, cung cấp hình ảnh, âm thanh cho ứng dụng, ... Tất cả các tài nguyên ứng dụng được lưu trong thư mục res/

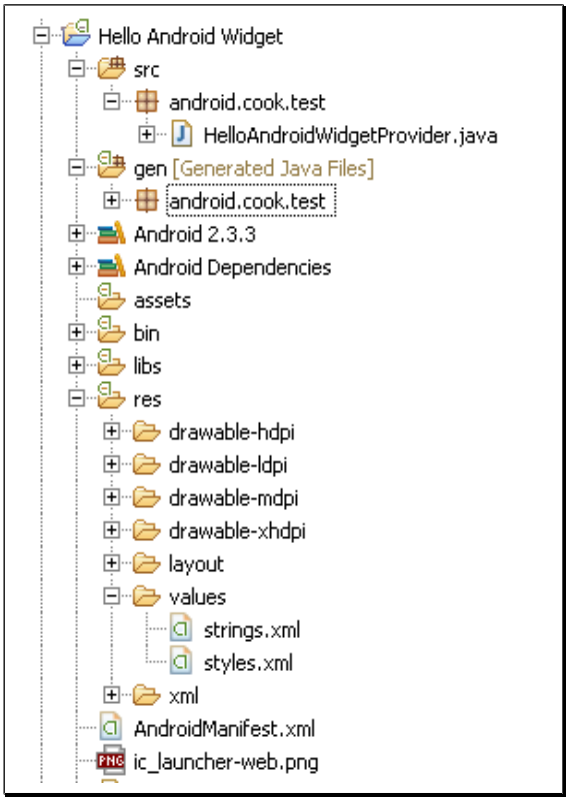


Khi một tài nguyên được thêm vào, Android SDK sẽ phát sinh một số ID tương ứng trong tập tin R.java. Ví dụ tập tin R.java ứng với tài nguyên mẫu trên

```
package android.app.cook;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_action_search=0x7f020000;
        public static final int ic_launcher=0x7f020001;
    }
    public static final class id {
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class menu {
    public static final class raw {
        public static final int vethoihoiem=0x7f040000;
    }
    public static final class string {
    public static final class style {
}
```

1.2.5 Cấu trúc lưu trữ một dự án Android

Hình sau minh họa cấu trúc lưu trữ một dự án Android.



Có một số thư mục và tập tin quan trọng cần quan tâm sau:

- Thư mục **RES**: lưu trữ các tập tin tài nguyên.
- Thư mục **SRC:** lưu trữ toàn bộ tập tin Java trong ứng dụng. Các tập tin (class) được tổ chức thành các gói (package) java.
- Thư mục **GEN:** chứa tập tin R.java được dùng để truy xuất các tài nguyên khi viết mã
- Tập tin **AndroidManifest.xml** chứa thông tin về ứng dụng.

✿Chương 2 : Tổng hợp một số lớp quan trọng trên nền tảng Android

Các lớp này được giới thiệu *developer.android.com/reference/packages.html*. Trong chương trình này, tác giả tổng hợp một số lớp, phương thức cơ bản được dùng trong giai đoạn nghiên cứu này.

2.1 Lớp android.app.Activity

Kiểu	Tên phương thức	Chức năng
void	**addContentView()**	Thêm một view vào Activity
void	**closeContextMenu()**	Đóng menu ngữ cảnh hiện hành (nếu đang mở)
void	**closeOptionsMenu()**	Đóng menu options.
view	**findViewById()**	Tìm một view theo thuộc tính ID của file XML
void	**finish()**	Gọi khi Activity đóng lại
void	**finishActivity()**	Buộc một activity khác đóng lại
Intent	**getIntent()**	Trở lại Intent khởi động Activity
MenuInflater	**getMenuInflater()**	Tạo một đối tượng MenuInflater
void	**onCreateContextMenu()**	Gọi khi tạo menu context
boolean	**onCreateOptionsMenu()**	Khởi tạo nội dung của menu options
void	**setContentView()**	Đưa một view để Activity hiển thị.
void	**onCreate()**	Được gọi khi Activity được tạo lần đầu
void	**onDestroy**	Được gọi trước khi Activity được hủy hoàn toàn
void	**onPause()**	Được gọi khi Activity bị che khuất
void	**onRestart()**	Hiển thị và focus lại Activity ở trạng thái stopped
void	**onResume()**	Đượcgọi khi Activity bắt đầu tương tác với user
void	**onStart()**	Được gọi khi Activity đang hiển thị
void	**onStop()**	Được gọi khi Activity bị che khuất hoàn toàn
void		

2.1 Lớp android.app.AlertDialog

- **Chức năng:** Đây là một lớp con của lớp Dialog có thể trình bày hộp thoại có một, hai hoặc ba buttons.
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
Button	**getButton(int whichButton)** Lấy ra một nút trong các nút được sử dụng trong dialog
Listview	**getListView()** Lấy ra một listview được sử dụng trong dialog
boolean	**onKeyDown(int keyCode, KeyEvent event)** Được gọi khi một phím được nhấn
boolean	**onKeyUp(int keyCode, KeyEvent event)** Được gọi khi một phím được thả.
void	**setButton(int whichButton, CharSequence text, DialogInterface.OnClickListener listener)** Thiết đặt sự kiện lắng nghe khi nút dương của dialog được nhấn
void	**setButton(int whichButton, CharSequence text, Message msg)** Thiết đặt thông báo khi nút được nhấn
void	**setIcon(int resId)** Thiết đặt Icon (0 khi không muốn có Icon)
void	**setIconAttribute(int attrId)** Thiết đặt thuộc tính bổ sung cho icon.
void	**setTitle(CharSequence title)** Thiết đặt tiêu đề cho hộp thoại.
void	**setView(View view)** Thiết đặt một view trình bày trong dialog

2.2 Lớp android.app.DatePickerDialog:

- **Chức năng:** Đây là một lớp con của lớp Dialog có thể chứa DatePicker (bộ chọn ngày)
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
DatePicker	**getDatePicker()** **Gets the DatePicker contained in this dialog.**
void	**onClick(DialogInterface dialog, int which)** Được gọi khi kích một nút trong dialog.
void	**onDateChanged(DatePicker view, int year, int month, int day)** Được gọi khi ngày thay đổi.
void	**onRestoreInstanceState(Bundle savedInstanceState)** Phục hồi trạng thái của dialog từ Bundle.

Bundle	**onSaveInstanceState()** Lưu trạng thái của dialog vào Bundle
void	**updateDate(int year, int monthOfYear, int dayOfMonth)** Thiết đặt ngày hiện hành

2.3 Lớp android.app.Dialog:

- **Chức năng:** Đây là lớp Base của các lớp con Dialog.
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
void	**addContentView**(View view, ViewGroup.LayoutParams params) Thêm một view vào màn hình
void	**cancel()** Hủy Dialog
void	**closeOptionsMenu()** Đóng menu Options
void	**dismiss()** Loại bỏ hộp thoại ra khỏi màn hình
boolean	**dispatchGenericMotionEvent(MotionEvent ev)** Gọi để xử lý các sự kiện chuyển động chung.
boolean	**dispatchKeyEvent(KeyEvent event)** Gọi để xử lý các sự kiện phím.
boolean	**dispatchKeyShortcutEvent(KeyEvent event)** Gọi để xử lý các sự kiện phím nóng
boolean	**dispatchTouchEvent(MotionEvent ev)** Gọi để xử lý các sự kiện Touch
boolean	**dispatchTrackballEvent(MotionEvent ev)** Gọi để xử lý các sự kiện trackball
View	**findViewById(int id)** Tìm một view theo thuộc tính ID
ActionBar	**getActionBar()** Lấy ActionBar gắn vào hộp thoại, nếu có.
Context	**getContext()** Lấy context mà hộp thoại đang chạy trong đó
View	**getCurrentFocus()** Gọi phương thức getCurrentFocus() trên cửa sổ.
Activity	**getOwnerActivity()** Trả về Activity có hộp thoại này
window	**getWindow()** Lấy cửa sổ hiện hành cho Activity
void	**hide()** Làm ẩn dialog nhưng không loại bỏ nó.
void	**onActionModeFinished(ActionMode mode)** Được gọi khi kiểu action đã hoàn thành.
void	**onActionModeStarted(ActionMode mode)**

	Được gọi khi kiểu action đã bắt đầu.

void	**onAttachedToWindow()** Được gọi khi một cửa sổ được gắn vào trình quản lý cửa sổ.
void	**onBackPressed()** Được gọi khi dialog phát hiện phím back được nhấn.
void	**onContentChanged()** Được gọi bất cứ lúc nào nội dung view của màn hình thay đổi.
void	**onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo)** Được gọi khi tạo menu context của view.
boolean	**onCreateOptionsMenu(Menu menu)** Được gọi khi tạo menu Options.
boolean	**onCreatePanelMenu(int featureId, Menu menu)** Khởi tạo nội dung của menu cho panel.
View	**onCreatePanelView(int featureId)** Khởi tạo view để trình bày trong panel.
void	**onDetachedFromWindow()** Được gọi khi một cửa sổ được gắn vào trình quản lý cửa sổ.
boolean	**onKeyDown(int keyCode, KeyEvent event)** Được gọi khi một phím được nhấn.
boolean	**onKeyLongPress(int keyCode, KeyEvent event)** Được gọi khi một phím được nhấn và giữ
boolean	**onKeyMultiple(int keyCode, int repeatCount, KeyEvent event)** Được gọi khi phím được nhấn nhiều lần.
boolean	**onKeyShortcut(int keyCode, KeyEvent event)** Được gọi khi phím nóng được nhấn.
boolean	**onKeyUp(int keyCode, KeyEvent event)** Được gọi khi một phím được thả
boolean	**onMenuItemSelected(int featureId, MenuItem item)** Được gọi khi một mục trên menu được chọn.
boolean	**onMenuOpened(int featureId, Menu menu)** Được gọi khi menu trên panel được mở bởi người dùng.
void	**onPanelClosed(int featureId, Menu menu)** Được gọi khi panel đóng.
boolean	**onPrepareOptionsMenu(Menu menu)** Được gọi khi menu Options có sửa đổi.
boolean	**onPreparePanel(int featureId, View view, Menu menu)** Được gọi khi panelcó sửa đổi.
void	**onRestoreInstanceState(Bundle savedInstanceState)** Phục hồi trạng thái của dialog từ bundle
Bundle	**onSaveInstanceState()**

	Lưu trạng thái của dialog vào bundle.
boolean	**onSearchRequested()** Được gọi khi bắt đầu thực hiện yêu cầu tìm kiếm.

boolean	**onTouchEvent(MotionEvent event)** Được gọi khi có sự kiện touch.
boolean	**onTrackballEvent(MotionEvent event)** Được gọi khi có sự kiện Trackball.
void	**onWindowAttributesChanged(WindowManager.LayoutParams params)** Được gọi khi có sự thay đổi thuộc tính của cửa sổ hiện hành.
void	**onWindowFocusChanged(boolean hasFocus)** Được gọi khi focus của cửa sổ thay đổi
ActionMode	**onWindowStartingActionMode(ActionMode.Callback callback)** Được gọi khi kiểu action của cửa sổ bắt đầu
void	**setCancelMessage(Message msg)** Thiết đặt thông báo khi dialog đóng
void	**setCancelable(boolean flag)** Thiết đặt thông báo khi dialog bị hủy
void	**setContentView(View view)** Đặt nội dung màn hình vào view.
void	**setDismissMessage(Message msg)** Thiết đặt thông báo khi dialog bị hủy
void	**setOnCancelListener(DialogInterface.OnCancelListener listener)** Thiết đặt sự kiện lắng nghe khi dialog bị hủy
void	**setOnShowListener(DialogInterface.OnShowListener listener)** Thiết đặt sự kiện lắng nghe khi dialog hiển thị. **Sets a listener to be invoked when the dialog is shown.**
void	**setOwnerActivity(Activity activity)** Gán tên Activity có dialog này
void	**setTitle(int titleId)** Gán tiêu đề cho dialog
void	**show()** Khởi động dialog và trình bày nó trên màn hình.

2.4 Lớp android.app.ProgressDialog:

- **Chức năng:** Đây là một lớp con Dialog trình bày hộp thoại tiến trình.
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
void	**onStart()** Được gọi khi dialog khởi tạo

void	**setProgressNumberFormat(String format)** Thay đổi dạng số của hộp tiến trình.
void	**setProgressPercentFormat(NumberFormat format)** Thay đổi dạng số phần trăm của hộp tiến trình.

2.5 Lớp android.app.Service:

- **Chức năng:** Đây là lớp xử lý thành phần services của ứng dụng
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
Application	**getApplication()** Trả về ứng dụng có service này.
IBinder	**onBind(Intent intent)** Trả về kênh truyền thông đến service
void	**onConfigurationChanged(Configuration newConfig)** Được gọi bởi hệ thống khi cấu hình thiết bị thay đổi.
void	**onCreate()** Được gọi bởi hệ thống khi service được tạo lần đầu
void	**onDestroy()** Được gọi bởi hệ thống khi service bị hủy bỏ.
void	**onLowMemory()** Được gọi bởi hệ thống khi bộ nhớ suy giảm
void	**onRebind(Intent intent)** Được gọi khi có một người dùng mới kết nối với service
void	**onStartCommand(Intent intent, int flags, int startId)** Được gọi bởi hệ thống khi có một người dùng bắt đầu sử dụng service một cách tường minh.
void	**onTaskRemoved(Intent rootIntent)** Được gọi nếu service đang chạy và người dùng đã loại tác vụ có từ ứng dụng của service.
void	**onTrimMemory(int level)** Được gọi khi hệ điều hành phát hiện ra thời điểm cần để cắt bớt bộ nhớ không cần thiết ra khỏi tiến trình.
boolean	**onUnbind(Intent intent)** Được gọi khi tất cả người dùng mới không kết nối với service
void	**startForeground(int id, Notification notification)** Bắt đầu chạy chế độ nền
void	**stopForeground(boolean removeNotification)** Kết thúc chạy chế độ nền
void	**stopSelf()** Tự dừng service

2.6 Lớp android.app.TimePickerDialog:

- **Chức năng:** Đây là một lớp con của lớp Dialog có thể chứa TimePicker

- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
void	**onClick(DialogInterface dialog, int which)** Được gọi khi kích một nút trong dialog.
void	**onRestoreInstanceState(Bundle savedInstanceState)** Phục hồi trạng thái của dialog từ Bundle.
Bundle	**onSaveInstanceState()** Lưu trạng thái của dialog vào Bundle
void	**onTimeChanged(TimePicker view, int hourOfDay, int minute)** Được gọi khi giờ thay đổi
void	**updateTime(int hourOfDay, int minutOfHour)** Cập nhật giờ

2.7 Lớp android.appwidget.AppWidgetManager

- **Chức năng:** Cập nhật trạng thái AppWidget; cung cấp thông tin về AppWidget provider đã cài đặt và các trạng thái liên quan khác.
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
boolean	**bindAppWidgetIdIfAllowed(int appWidgetId, ComponentName provider)** Thiết đặt component cho một appWidgetId
int[]	**getAppWidgetIds(ComponentName provider)** Cung cấp danh sách appWidgetIds ràng buộc với AppWidget provider.
AppWidget_ ProviderInfo	**getAppWidgetInfo(int appWidgetId)** Cung cấp thông tin về AppWidget
Bundle	**getAppWidgetOptions(int appWidgetId)** Cung cấp extras kết hợp với thục thể Widget
AppWidget_ Manager	**getInstance(Context context)** Cung cấp thực thể AppWidgetManager để dùng cho đối tượng Context.
void	**notifyAppWidgetViewDataChanged(int[] appWidgetIds, int viewId)** Thông báo một tập các view trong tất cả các thực thể AppWidget xác định để làm mất hiệu lực dữ liệu hiện hành của chúng.
void	**partiallyUpdateAppWidget(int appWidgetId, RemoteViews views)** Thực hiện cập nhật trên một widget được chỉ định bởi appWidgetId
void	**updateAppWidget(int[] appWidgetIds, RemoteViews views)** **Set the RemoteViews to use for the specified appWidgetIds.**
void	**updateAppWidget(ComponentName provider, RemoteViews views)**

	Thiết đặt RemoteViews dùng cho tất cả các thục thể AppWidget
void	**updateAppWidgetOptions(int appWidgetId, Bundle options)** Cập nhật extras đối với thục thể widget đã cho

2.8 Lớp android.appwidget.AppWidgetProvider

- **Chức năng:** Cung cấp cách cài đặt một AppWidget provider.
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
void	**onAppWidgetOptionsChanged(Context context, AppWidgetManager appWidgetManager, int appWidgetId, Bundle newOptions)** Được gọi để đáp ứng thông báo ACTION_APPWIDGET_OPTIONS_CHANGED khi widget này thay đổi kích thước.
void	**onDeleted(Context context, int[] appWidgetIds)** Được gọi để đáp ứng thông báo ACTION_APPWIDGET_DELETED khi có một hay nhiều widget bị hủy.
void	**onDisabled(Context context)** Được gọi để đáp ứng thông báo ACTION_APPWIDGET_DISABLED khi widget cuối cùng bị hủy.
void	**onEnabled(Context context)** Được gọi để đáp ứng thông báo ACTION_APPWIDGET_ENABLED khi có widget được khởi tạo.
void	**onReceive(Context context, Intent intent)** Cài đặt onReceive(Context, Intent) để gửi các lời gọi đến các phương thức khác nhau trên AppWidgetProvider.
void	**onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds)** Được gọi để đáp ứng thông báo ACTION_APPWIDGET_UPDATE khi AppWidget provider này được yêu cầu cung cấp RemoteViews cho tập các AppWidgets.

2.9 Lớp android.content.BroadcastReceiver

- **Chức năng:** Lớp Base cho mã xử lý intents được gởi bởi sendBroadcast().
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
void	**abortBroadcast()** Thiết lập cờ biểu thị receiver (intent) bỏ qua thông báo hiện tại.
void	**clearAbortBroadcast()** Xóa cờ biểu thị receiver (intent) bỏ qua thông báo hiện tại.
boolean	**getAbortBroadcast()** Trả về trạng thái của cờ chỉ thị
int	**getResultCode()**

	Cung cấp mã kết quả hiện hành
String	**getResultData()** Cung cấp dữ liệu kết quả hiện hành
Bundle	**getResultExtras(boolean makeMap)** Cung cấp dữ liệu extras kết quả hiện hành
boolean	**isOrderedBroadcast()** Trả về true nếu receiver đang xử lý một thông báo có thứ tự
void	**onReceive(Context context, Intent intent)** Được gọi khi BroadcastReceiver đang nhận thông báo Intent
void	**setResult(int code, String data, Bundle extras)** Thay đổi tất cả dữ liệu kết quả trả về từ thông báo.
void	**setResultCode(int code)** Thay đổi mã kết quả hiện hành của thông báo.
void	**setResultData(String data)** Thay đổi dữ liệu kết quả hiện hành của thông báo.
void	**setResultExtras(Bundle extras)** Thay đổi extras kết quả hiện hành của thông báo.

2.10 Lớp android.content.Context

- **Chức năng:** Xử lý giao diện thông tin toàn cục về môi trường ứng dụng
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
boolean	**bindService(Intent service, ServiceConnection conn, int flags)** Kết nối với ứng dụng service, tạo ra nó nếu cần
int	**checkCallingOrSelfPermission(String permission)** Xác định đối tượng gán quyền đặc biệt (IPC hay người dùng)
int	**checkCallingOrSelfUriPermission(Uri uri, int modeFlags)** Xác định đối tượng gán quyền truy cập URI
int	**checkCallingPermission(String permission)** Xác đinh có phải IPC được gán quyền đặc biệt
int	**checkCallingUriPermission(Uri uri, int modeFlags)** Xác đinh có phải tiến trình đang gọi và User được gán quyền truy cập URI
int	**checkUriPermission(Uri uri, String readPermission, String writePermission, int pid, int uid, int modeFlags)** Kiểm tra cả hai quyền URI và quyền normal
Context	**createConfigurationContext(Configuration overrideConfiguration)** Trả về đối tượng Context mới nhưng tài nguyên của nó được điều chỉnh phù hợp với cấu hình đã cho.

Context	**createDisplayContext(Display display)** Trả về đối tượng Context mới nhưng tài nguyên của nó được điều chỉnh phù hợp với kích thước mà hình đã cho.
Context	**createPackageContext(String packageName, int flags)** Trả về đối tượng Context mới với tên ứng dụng đã cho.
String[]	**databaseList()** Trả về mảng tên các CSDL kết hợp với gói ứng dụng của Context.
boolean	**deleteDatabase(String name)** Hủy một SQLiteDatabase kết hợp với gói ứng dụng của Context.
boolean	**deleteFile(String name)** Hủy một File kết hợp với gói ứng dụng của Context.
String[]	**fileList()** Trả về mảng tên các files kết hợp với gói ứng dụng của Context
Context	**getApplicationContext()** Trả về nội dung đối ứng dụng đơn, toàn cục của tiến trình hiện hành.
Application Info	**getApplicationInfo()** Trả về thông tin ứng dụng đối với gói context
File	**getCacheDir()** Trả về đường dẫn tuyệt đối đến thư mục ứng dụng.
File	**getDatabasePath(String name)** Trả về đường dẫn tuyệt đối đến CSDL ứng dụng.
File	**getDir(String name, int mode)** Lấy ra, tạo mới thư mục mà ứng dụng có thể đặt vào các file dữ liệu riêng.
File	**getFileStreamPath(String name)** Trả về đường dẫn tuyệt đối của trên filesystem
Resources	**getResources()** Trả về thực thể Resources của gói ứng dụng.
void	**grantUriPermission(String toPackage, Uri uri, int modeFlags)** Gán quyền truy cập Uri cho gói ứng dụng khác.
boolean	**isRestricted()** Cho biết Context bị hạn chế không?
FileInput Stream	**openFileInput(String name)** Mở file kết hợp với gói ứng dụng Context để đọc
FileOutput Stream	**openFileOutput(String name, int mode)** Mở file kết hợp với gói ứng dụng Context để viết

SQLite Database	**openOrCreateDatabase(String name, int mode, SQLiteDatabase.CursorFactory factory)** Mở SQLiteDatabase kết hợp với gói ứng dụng Context.
void	**revokeUriPermission(Uri uri, int modeFlags)** Loại bỏ tất cả các quyền truy cập content provider Uri
void	**sendBroadcast(Intent intent)** Truyền intent đến tất cả các BroadcastReceivers được quan tâm.
void	**setTheme(int resid)** Thiết đặt theme cơ sở cho context
void	**startActivities(Intent[] intents, Bundle options)** Khởi động các Activites mới
void	**unbindService(ServiceConnection conn)** Tắt kết nối với mọt ứng dụng service

2.11 Lớp android.content.Intent

- **Chức năng:** Cung cấp thông tin về cách tạo và xử lý các intents
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
Intent	**addCategory(String category)** Thêm một category cho intent
Intent	**addFlags(int flags)** Thêm một flags cho intent
Object	**clone()** Tạo và trả về một bản sao của đối tượng (this)
Intent	**cloneFilter()** Tạo và trả về một bản sao của đối tượng (this) có lọc
int	**fillIn(Intent other, int flags)** Copy nội dung của *other* vào trong intent này, nhưng chỉ những trường không được định nghĩa bởi intent.
boolean	**filterEquals(Intent other)** Xác định xem hai intents có giống nhau về mục đích không?
int	**filterHashCode()** Phát sinh mã hash phùc hợp với ngữ nghĩa của filterEquals().
String	**getAction()** Lấy ra hành động tổng quát được thực hiện, như là ACTION_VIEW.
boolean[]	**getBooleanArrayExtra(String name)** Lấy ra dữ liệu mở rộng từ intent

ClipData	**getClipData()** Trả về ClipData kết hợp với Intent.
Component Name	**getComponent()** Lấy ra thành phần cụ thể kết hợp với intent
Uri	**getData()** Lấy ra dữ liệu mà intent đang xử lý
String	**getDataString()** Giống như getData(), nhưng trả về URI như là chuỗi mã.
Bundle	**getExtras()** Lấy ra dữ liệu extras mở rộng từ intent
int	**getFlags()** Lấy ra cờ đặc biệt bất kỳ kết hợp với intent.
String	**getScheme()** Trả về phần sơ đồ dữ liệu của intent
Intent	**getSelector()** Trả về selector xác định kết hợp với Intent.
Rect	**getSourceBounds()** Lấy về ràng buộc người gởi intent theo tọa độ màn hình.
String	**getType()** Lấy ra kiểu MIME tường minh bao gồm trong intent.
boolean	**hasCategory(String category)** Kiểm tra xem category có tồn tại trong intent.
boolean	**hasExtra(String name)** Trả về true nếu giá trị extra value được kết hợp với name
Intent	**makeMainActivity(ComponentName mainActivity)** Tạo ra một intent để trình bày activity chính
Intent	**makeMainSelectorActivity(String selectorAction, String selectorCategory)** Tạo ra một Intent cho activity chính.
String	**normalizeMimeType(String type)** Chuẩn hóa kiểu dữ liệu MIME.
Intent	**parseIntent(Resources resources, XmlPullParser parser, AttributeSet attrs)** Đổi phần tử "intent" element (và các children) từ XML và tạo ra đối tượng Intent.
Intent	**parseUri(String uri, int flags)** Tạo một intent từ URI.

Intent	**putExtra(String name, double[] value)** Thêm dữ liệu mở rộng cho intent
Intent	**putExtras(Intent src)** Copy tất cả extras trong 'src' vào intent
Intent	**putExtras(Bundle extras)** Thêm tập dữ liệu mở rộng cho intent
void	**removeCategory(String category)** Loại category ra khỏi intent.
void	**removeExtra(String name)** Loại dữ liệu mở rộng ra khỏi intent.
Intent	**replaceExtras(Bundle extras)** Thay thể hoàn toàn extras trong Intent bằng extras trong Bundle
Component Name	**resolveActivity(PackageManager pm)** Trả về thành phần Activity component được sử dụng bởi intent.
String	**resolveType(ContentResolver resolver)** Trả về kiểu dữ liệu MIME của intent.
Intent	**setAction(String action)** Thiết đặt action chung được thực hiện.
void	**setClipData(ClipData clip)** Thiết đặt ClipData kết hợp với Intent.
Intent	**setComponent(ComponentName component)** Thiết đặt Component kết hợp với Intent.
Intent	**setData(Uri data**) Thiết đặt dữ liệu kết hợp với Intent.
Intent	**setDataAndNormalize(Uri data)** Chuẩn hóa và thiết đặt dữ liệu kết hợp với Intent.
Intent	**setDataAndType(Uri data, String type)** Thiết đặt dữ liệu cho Intent cùng với kiểu dữ liệu MIME.
Intent	**setDataAndTypeAndNormalize(Uri data, String type)** Chuẩn hóa và thiết đặt cả dữ liệu Uri và kiểu dữ liệu MIME.
Intent	**setFlags(int flags)** Thiết đặt cờ điều khiển intent.
void	**setSelector(Intent selector)** Thiết đặt selector cho Intent.
Intent	**setType(String type)** Thiết đặt kiểu dữ liệu MIME.
Intent	**setTypeAndNormalize(String type)**

	Chuẩn hóa và thiết đặt kiểu dữ liệu MIME.
String	**toUri(int flags)** Đổi Intent thành chuỗi biểu diễn URI.

2.11 Lớp android.content.res.Resources

- **Chức năng:** Cung cấp thông tin về cách tạo và xử lý các intents
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
void	**flushLayoutCache()** Loại tất cả các resources trong bộ nhớ cache khỏi đối tượng Resources
boolean	**getBoolean(int id)** Trả về một giá trị boolean kết hợp với resource ID
int	**getColor(int id)** Trả về một số nguyên màu kết hợp với resource ID
Color StateList	**getColorStateList(int id)** Trả về một danh sách trạng thái màu kết hợp resource ID
Configuration	**getConfiguration()** Trả về cấu hình hiện hành đối với đối tượng Resources
float	**getDimension(int id)** Lấy ra dimensional đối với resource ID.
Display Metrics	**getDisplayMetrics()** Trả về kích thước màn hình hiện hành
Drawable	**getDrawable(int id)** Trả về một đối tượng drawable kết hợp với resource ID
float	**getFraction(int id, int base, int pbase)** Lấy ra đơn vị fractional đối với resource ID
int	**getIdentifier(String name, String defType, String defPackage)** Lấy ra identifier ứng với name của resource
int	**getInt(int id)** Trả về một số nguyên kết hợp với resource ID
Movie	**getMovie(int id)** Trả về một đối tượng Movie kết hợp với resource ID
String	**getResourceEntryName(int resid)** Trả về tên ứng với identifier của resource
void	**getValue(String name, TypedValue outValue, boolean resolveRefs)** Trả về dữ liêu thô kết hợp với resource ID

void	**getValueForDensity(int id, int density, TypedValue outValue, boolean resolveRefs)** Trả về giá trị thô kết hợp với resource có density kết hợp.
Resources.Theme	**newTheme()** Phát sinh một đối tượng Theme ứng với tập Resources.
Typed Array	**obtainAttributes(AttributeSet set, int[] attrs)** lấy ra một tập các giá trị thuộc tính cơ bản từ AttributeSet
Input Stream	**openRawResource(int id, TypedValue value)** Mở data stream để đọc raw resource.
void	**updateConfiguration(Configuration config, DisplayMetrics metrics)** Chứa cấu hình đã cập nhật mới nhất

2.12 Lớp android.graphics.Color

- **Chức năng:** Định nghĩa các phương thức để tạo và đổi các số nguyên color.
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
int	**HSVToColor(float[] hsv)** Đổi các thành phần HSV thành màu ARGB.
void	**RGBToHSV(int red, int green, int blue, float[] hsv)** Đổi các thành phần RGB thành HSV.
int	**alpha(int color)** Trả về thành phần alpha của số nguyên màu
int	**argb(int alpha, int red, int green, int blue)** Trả về số nguyên color từ các thành phần alpha, red, green, blue.
int	**blue(int color)** Trả về thành phần blue của số nguyên màu.
void	**colorToHSV(int color, float[] hsv)** Đổi màu ARGB thành HSV.
int	**blue(int color)** Trả về thành phần green của số nguyên màu.
int	**blue(int color)** Trả về thành phần red của số nguyên màu.
int	**rgb(int red, int green, int blue)** Trả về số nguyên color từ các thành phần red, green, blue.

2.13 Lớp android.media.MediaPlayer

- **Chức năng:** Lớp MediaPlayer có thể được sử dụng để điều khiển việc phát lại các tập tin audio / video và stream.
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
void	**addTimedTextSource(Context context, Uri uri, String mimeType)** Thêm file nguồn dạng text thay đổi theo thời gian bên ngoài(Uri)
void	**attachAuxEffect(int effectId)** Gắn hiệu ứng phụ vào trình player
void	**deselectTrack(int index)** Bỏ chọn track
int	**getAudioSessionId()** Trả về ID của audio session.
int	**getCurrentPosition()** Cho vị trị playback hiện hành.
int	**getDuration()** Cho duration của file.
TrackInfo[]	**getTrackInfo()** Cho mảng thông tin track.
int	**getVideoHeight()** Trả về độ cao của video
int	**getVideoWidth()** Trả về độ rộng của video
boolean	**isLooping()** Kiểm tra xem MediaPlayer có lặp lại hay không.
boolean	**isPlaying()** Kiểm tra xem MediaPlayer có đang phát hay không.
void	**pause()** Tạm dừng playback.
void	**prepare()** Chuẩn bị trình player để playback
void	**release()** Giải phóng resources gắn với đối tượng MediaPlayer.
void	**reset()** Reset MediaPlayer trở lại trạng thái chưa khởi tạo.

void	**seekTo(int msec)** Tìm vị trí time xác định.
void	**selectTrack(int index)** Chọn track
void	**setAudioSessionId(int sessionId)** Thiết đặt ID cho audio session
void	**setAudioStreamType(int streamtype)** Thiết đặt kiểu audio stream cho MediaPlayer.
void	**setAuxEffectSendLevel(float level)** Thiết đặt level gởi player cho hiệu ứng phụ thêm vào.
void	**setDataSource(String path)** Thiết đặt nguồn dữ liệu (file-path or http/rtsp URL)
void	**setDataSource(Context context, Uri uri)** Thiết đặt nguồn dữ liệu như content Uri
void	**setDisplay(SurfaceHolder sh)** Thiết đặt SurfaceHolder để trình bày phần video của media..
void	**setLooping(boolean looping)** Thiết đặt trình player lặp lại hay không.
void	**setNextMediaPlayer(MediaPlayer next)** Thiết đặt MediaPlayer khởi động khi MediaPlayer hoàn thành playback.
void	**setSurface(Surface surface)** Thiết đặt Surface của phần video.
void	**setVideoScalingMode(int mode)** Thiết đặt kiểu tỉ lệ video (video scaling mode).
void	**setVolume(float leftVolume, float rightVolume)** Thiết đặt mức âm thanh (volume) của player.
void	**setWakeMode(Context context, int mode)** Thiết đặt quản lý nguồn mức thấp cho MediaPlayer.
void	**start()** Bắt đầu hay resumes playback.
void	**stop()** Dừng playback sau khi playback đã bị dừng hay tạm dừng.

2.14 Lớp android.net.Uri

- **Chức năng:** Dùng để tham chiếu Uri.
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
int	**compareTo(Uri other)** So ánh biểu diễn chuỗi của Uri này với Uri khác
String	**decode(String s)** Giải mã *'%'-escaped octets* trong chuỗi sử dụng bộ mã UTF-8.
String	**encode(String s, String allow)** Mã hóa các ký tự trong chuỗi dạng *'%'-escaped octets* sử dụng bộ mã UTF-8
boolean	**equals(Object o)** So sánh Uri với đối tượng khác xem có bằng không
Uri	**fromFile(File file)** Tạo Uri từ file
Uri	**fromParts(String scheme, String ssp, String fragment)** Tạo Uri từ các thành phần đã cho trong tham số
String	**getAuthority()** Cho phần authority đã giải mã của Uri
boolean	**getBooleanQueryParameter(String key, boolean defaultValue)** Tìm chuỗi truy vấn giá trị đầu tiên với phím đã cho và diễn giải nó như một giá trị boolean
String	**getEncodedAuthority()** Cho phần authority đã mã hóa của Uri
String	**getEncodedFragment()** Cho phần fragment đã mã hóa của Uri
String	**getEncodedPath()** Cho phần path đã mã hóa của Uri
String	**getEncodedQuery()** Cho phần query đã mã hóa của Uri
String	**getEncodedUserInfo()** Cho phần thông tin user từ authority đã mã hóa
String	**getFragment()** Cho phần fragment đã giải mã của Uri
String	**getHost()** Cho phần thông tin host từ authority đã mã hóa
String	**getLastPathSegment()** Cho phần segment cuối cùng đã giải mã của Uri
String	**getPath()** Cho phần path đã giải mã

int	**getPort()** Cho phần port từ authority.
String	**getQuery()** Cho phần query đã giải mã của Uri
String	**getQueryParameter(String key)** Tìm chuỗi truy vấn giá trị đầu tiên với phím đã cho.
String	**getScheme()** Gets the scheme of this URI.
String	**getUserInfo()** Cho phần thông tin user từ authority đã giải mã
boolean	**isAbsolute()** Trả về true nếu URI là absolute
boolean	**isHierarchical()** Trả về true nếu URI là absolute hierarchical như "http://google.com".
boolean	**isOpaque()** Trả về true nếu URI là opaque như "mailto:nobody@google.com".
boolean	**isRelative()** Trả về true nếu URI là relative.
Uri	**normalizeScheme()** Trả về URI tương đương với thành phần scheme viết thường.
Uri	**parse(String uriString)** Tạo Uri bằng cách chuyển chuỗi URI đã mã hóa
String	**toString()** Trả về chuỗi mã hóa biểu diễn URI.
Uri	**withAppendedPath(Uri baseUri, String pathSegment)** Tạo một Uri mới bằng cách bổ sung một đoạn Path đã mã hóa vào Uri.
void	**writeToParcel(Parcel out, Uri uri)** Viết Uri thành một Parcel

2.14 Lớp android.os.Handler

- **Chức năng:** Lớp xử lý Handler cho phép gởi vaà xử lý Message và các đối tượng Runnable kết hợp với một thread.
- **Các phương thức:**

Kiểu	Tên phương thức và chức năng
void	**dispatchMessage(Message msg)** Gởi thông điệp hệ thống.

String	**getMessageName(Message message)** Trả về chuỗi biểu diễn tên của message.
void	**handleMessage(Message msg)** Lớp con phải cài đặt để nhận message
boolean	hasMessages(int what, Object object) Kiểm tra xem có message với mã 'what' và đối tượng 'object' trong hàng đợi không.i
Message	**obtainMessage()** Trả về message mới từ pool message toàn cục
boolean	**post(Runnable r)** Tạo ra đối tượng Runnable r thêm vào hàng đợi message.
boolean	**postAtFrontOfQueue(Runnable r)** Gởi message đến một đối tượng có cài đặt Runnable.
void	**removeCallbacks(Runnable r)** Loại bỏ bất kỳ Runnable r có trong hàng đợi message.
void	**removeMessages(int what**) Loại bỏ các message với mã 'what' trong hàng đợi không.
boolean	**sendEmptyMessage(int what)** Gởi message chỉ chứa giá trị what
boolean	**sendEmptyMessageAtTime(int what, long uptimeMillis)** Gửi một message chỉ chứa giá trị what , được giao vào một thời điểm cụ thể
boolean	**sendMessage(Message msg)** Gởi message vào cuối hàng đợi message
String	**toString()** Trả về chuỗi chứa phần mô tả ngắn của đối tượng.

2.15 Lớp android.os.Message

- **Chức năng:** Định nghĩa `message` chứa phần mô tả và đối tượng dữ liệu bất kỳ
- **Các phương thức:**

Kiểu	**Tên phương thức và chức năng**
void	**copyFrom(Message o)** Tạo message cho bởi o
Runnable	**getCallback()** Lấy đối tượng callback mà sẽ thực thi khi message được gởi.

Bundle	**getData()** Nhận dữ liệu kết hợp với sự kiện
Handler	**getTarget()** Lấy ra phần cài đặt Handler nhận message
long	**getWhen()** Trả về thời điểm chuyển message đến đích (tính bằng milliseconds.)
Message	**obtain()** Trả về thực thể Message từ global pool.
void	**sendToTarget()** Gởi Message đến Handler được chỉ ra bởi phương thức getTarget().
void	**setData(Bundle data)** Đặt dữ liệu dạng Bundle
String	**toString()** Trả về chuỗi chứa phần mô tả ngắn của đối tượng.
void	**writeToParcel(Parcel dest, int flags)** Flatten đối tượng thành Parcel.

2.16 Lớp android.view.ContextMenu

- **Chức năng:** Cung cấp thông tin về cách tạo Context Menu
- **Các phương thức:**

Kiểu	**Tên phương thức và chức năng**
void	**clearHeader()** Xóa phần header của context menu .
Context Menu	**setHeaderIcon(Drawable icon)** Thiết đặt icon của phần header
Context Menu	**setHeaderTitle(CharSequence title)** Thiết đặt tiêu đề của phần header
Context Menu	**setHeaderView(View view)** Thiết đặt header của context menu cho View.

2.1 Lớp android.view.Gravity;

2.1 Lớp android.view.Menu;

2.1 Lớp android.view.MenuInflater;

2.1 Lớp android.view.MenuItem;

2.1 Lớp android.view.View;

2.1 Lớp android.view.ViewGroup.LayoutParams;

2.1 Lớp android.view.ViewGroup.LayoutParams;

2.1 Lớp android.widget.ArrayAdapter;

2.1 Lớp android.widget.Button;

2.1 Lớp android.widget.DatePicker;

2.1 Lớp android.widget.EditText;

2.1 Lớp android.widget.FrameLayout;

2.1 Lớp android.widget.ImageView.ScaleType;

2.1 Lớp android.widget.ImageView;

2.1 Lớp android.widget.LinearLayout;

2.1 Lớp android.widget.ListView;

2.1 Lớp android.widget.RelativeLayout.LayoutParams;

2.1 Lớp android.widget.RelativeLayout;

2.1 Lớp android.widget.TableLayout;

2.1 Lớp android.widget.TableRow;

2.1 Lớp android.widget.TextView;

2.1 Lớp android.widget.TimePicker;

2.1 Lớp android.widget.Toast;

2.1 Lớp java.io.BufferedReader;

2.1 Lớp java.io.InputStreamReader;

2.1 Lớp java.net.URL;

2.1 Lớp java.util.Calendar;

Chương 3 : Xây dựng một số ứng dụng dựa trên các lớp của Android.

Kết luận và hướng phát triển