

Lập trình ứng dụng trên đầu cuối di động

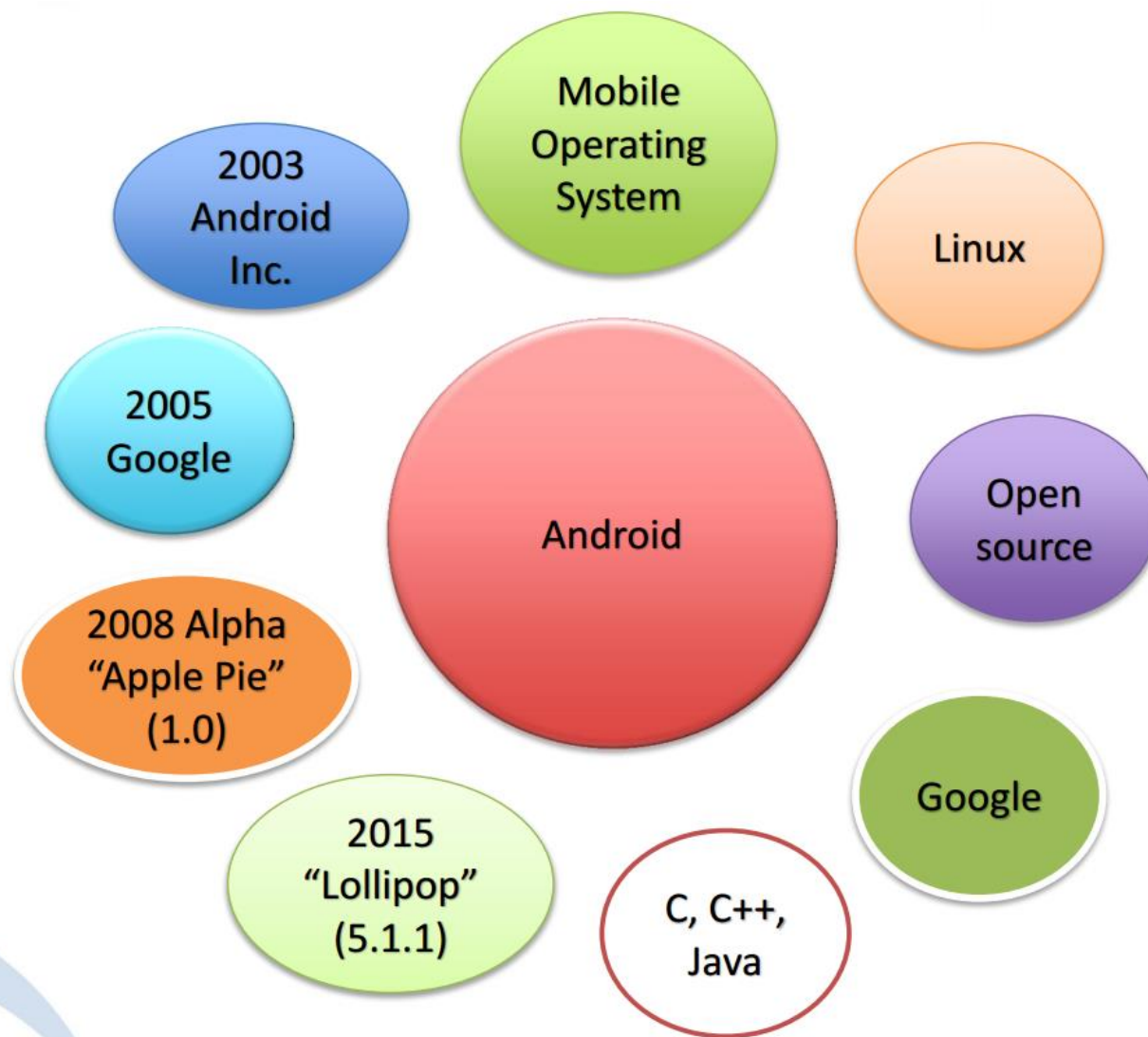
CHƯƠNG 2. MÔI TRƯỜNG LẬP TRÌNH ỨNG DỤNG ANDROID

Nội dung

- Tổng quan về hệ điều hành Android
- Thiết lập môi trường lập trình ứng dụng Android
- Bổ trợ kiến thức cơ bản về ngôn ngữ Java cho lập trình ứng dụng Android
- Quy trình lập trình ứng dụng di động Android

➤ Tổng quan về hệ điều hành Android

Hệ điều hành Android là gì?



Quá trình phát triển của HĐH Android

- Alpha “Apple Pie” (1.0)
- Beta “Petit Four” (1.1)
- Cupcake (1.5)
- Doughnut (1.6)
- Eclair (2.0–2.1)
- Froyo (2.2–2.2.3)
- Gingerbread (2.3–2.3.7)
- Honeycomb (3.0–3.2.6)
- Ice Cream Sandwich (4.0–4.0.4)
- Jelly Bean (4.1–4.3.1)
- KitKat (4.4–4.4.4)
- Lollipop (5.0)

ANDROID



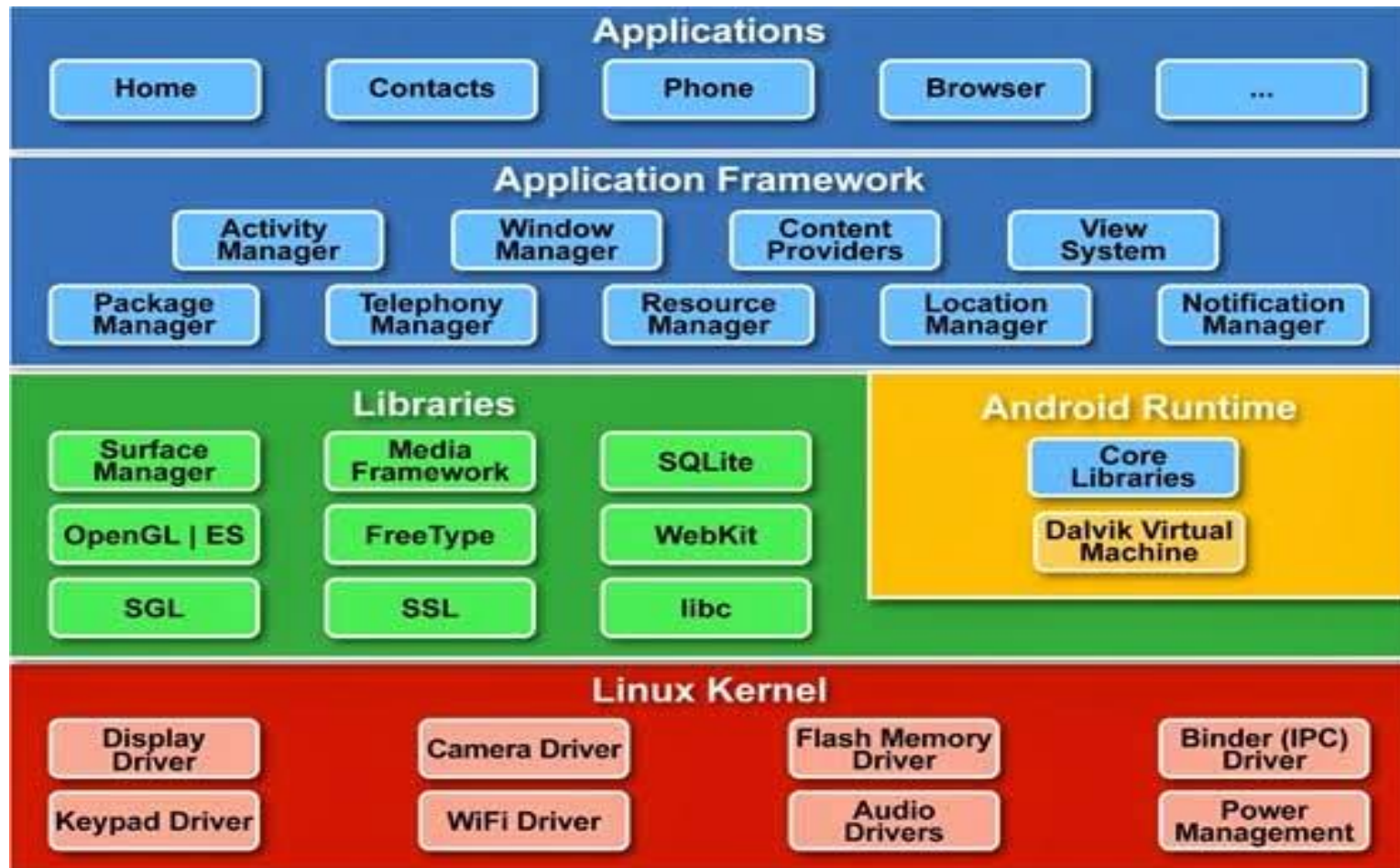
Một số thiết bị chạy HĐH Android



Các đặc trưng cơ bản của Android OS

Đặc trưng	Miêu tả
Kết nối	GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wifi, LTE, NFC, WiMAX
Lưu trữ	Sử dụng cơ sở dữ liệu quan hệ SQLite ; Lưu trữ dữ liệu trong bộ nhớ thiết bị ; Lưu trữ dữ liệu kiểu key/value
Hỗ trợ media	Theo chuẩn: H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, BMP
Tin nhắn	SMS, MMS
Web browser	Dựa trên bộ thư viện mã nguồn mở WebKit
Hỗ trợ thiết bị phần cứng	Accelerometer sensor, camera, digital compass, Proximity sensor, GPS
Đa chạm	Hỗ trợ màn hình đa chạm
Đa nhiệm	Ứng dụng có khả năng hoạt động đa nhiệm
Chia sẻ kết nối Internet	Hỗ trợ chia sẻ kết nối Internet như điểm truy nhập có dây / không dây
Hỗ trợ flash	Android 2.3 hỗ trợ Flash 10.1
Hỗ trợ widget tùy biến	Widget có khả năng tùy biến cho phép người dùng mở rộng hoặc thu nhỏ lại để xem thông tin cho phù hợp.
Đa ngôn ngữ	Hỗ trợ hiển thị bằng đa ngôn ngữ

Kiến trúc của HĐH Android



Kiến trúc của HĐH Android

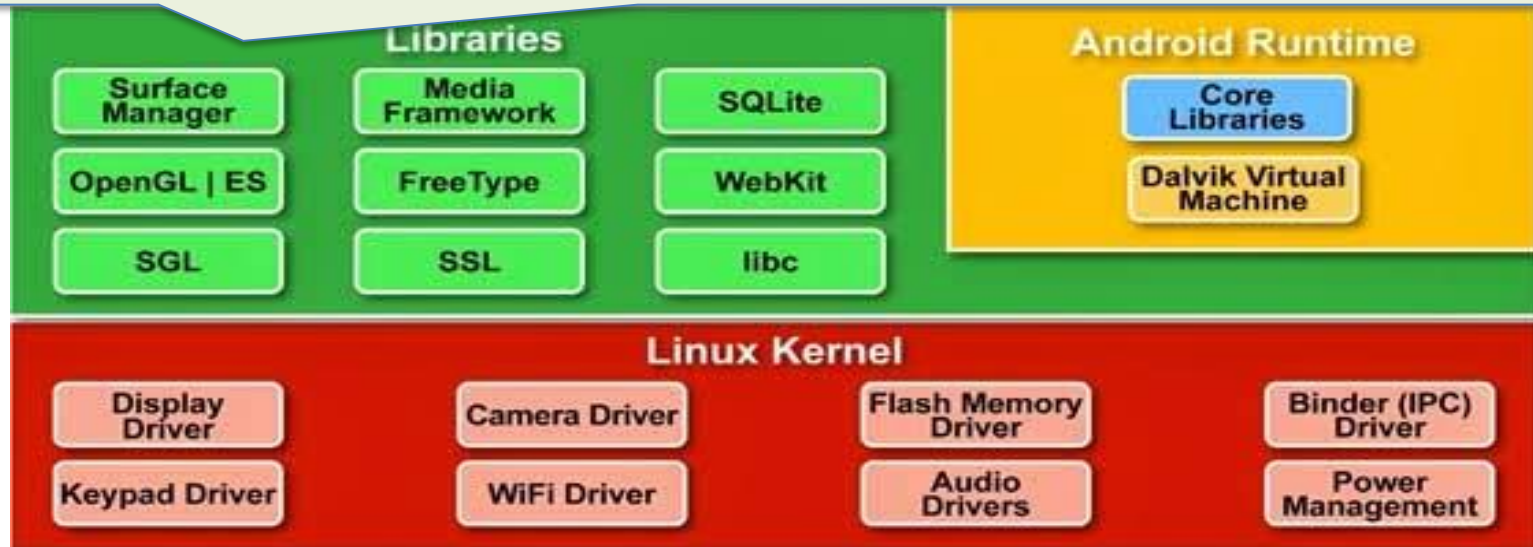
- Android sử dụng nhân Linux 2.6.
- Tầng này cung cấp chức năng hệ thống cơ bản như giao tiếp với tầng trên, bảo mật, quản lý bộ nhớ, quản lý các tiến trình, ngăn xếp mạng và trình điều khiển thiết bị (máy ảnh, bàn phím, màn hình hiển thị, giao tiếp USB, giao tiếp hồng ngoại, không dây, v.v...).
- Ngoài ra, nhân Linux này cũng có vai trò như một lớp trừu tượng giữa phần cứng và phần mềm.



Kiến trúc của HĐH Android

Bộ thư viện: Android cung cấp một tập hợp các thư viện C/C++ như:

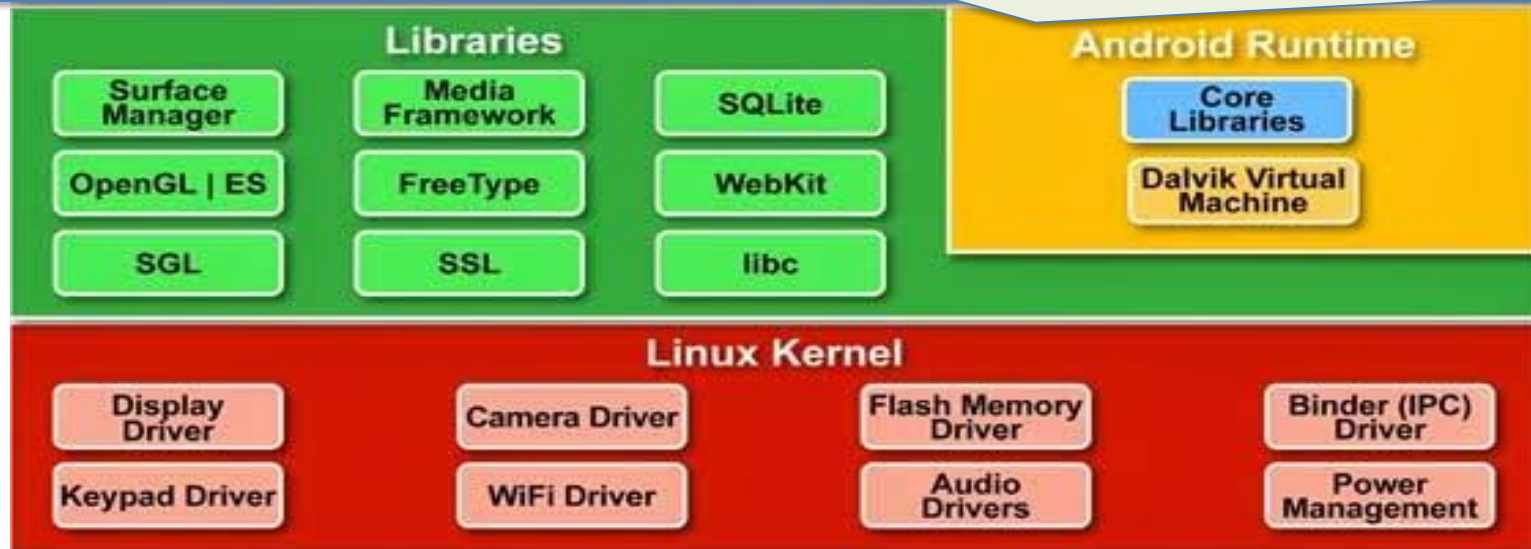
- **OpenGL:** thư viện dùng để tạo ra các đồ họa 3D dựa vào chuẩn OpenGL ES 1.0.
- **FreeType:** thư viện hỗ trợ xử lý bitmap, font, vector.
- **SGL:** thư viện cơ bản cung cấp các engine đồ họa 2D.
- **Libc** : thư viện C chuẩn, được tối ưu cho các thiết bị Linux-based.
- **SQLite:** thư viện thao tác với cơ sở dữ liệu quan hệ nhỏ gọn SQLite.
- **SSL** : thư viện hỗ trợ sử dụng giao thức mã hóa SSL (Secure Sockets Layer) trong bảo mật truyền thông Internet.



Kiến trúc của HĐH Android

Android runtime cung cấp một tập **các thư viện lõi** cho phép các nhà phát triển viết các ứng dụng Android sử dụng ngôn ngữ lập trình Java.

Ngoài ra, Android runtime cung cấp **máy ảo Dalvik** – một loại máy ảo Java được thiết kế đặc biệt và tối ưu hóa cho Android. Máy ảo Dalvik sử dụng các tính năng cốt lõi của Linux như quản lý bộ nhớ và đa luồng. Máy ảo Dalvik cũng cho phép tất cả các ứng dụng Android chạy trong tiến trình riêng của nó.



Danh sách các gói API cơ bản được cung cấp bởi Android runtime

Gói thư viện	Miêu tả
android.util	Gói tiện ích cơ bản bao gồm nhiều lớp mức thấp như các lớp quản lý (List, Stack...), lớp xử lý chuỗi, lớp xử lý XML.
android.os	Gói hệ điều hành cho phép truy cập đến các dịch vụ cơ bản như: tin nhắn, đồng hồ và gỡ lỗi.
android.graphics	Gói đồ họa cung cấp các lớp đồ họa mức thấp thực hiện các chức năng đồ họa, màu, vẽ cơ bản.
android.text	Công cụ hiển thị và xử lý văn bản.
android.database	Cung cấp các lớp mức thấp làm việc với cơ sở dữ liệu như SQLite.
android.content	Các giao tiếp lập trình nội dung được dùng để quản lý truy cập dữ liệu và xuất bản bằng cách cung cấp các dịch vụ thao tác với tài nguyên, nhà cung cấp nội dung và các gói.
android.view	Cung cấp các lớp giao diện người dùng cơ bản nhất. Tất cả giao diện người dùng được tạo ra đều phải sử dụng một tập các view.

Gói thư viện	Miêu tả
android.widget	Xây dựng kế thừa từ gói android.view. Widget bao gồm những thành phần giao diện tạo sẵn để tạo nên giao diện người dùng. Các widget bao gồm danh sách, nút bấm, hộp nhập, các kiểu trình bày (layout).
com.google.android.maps	Bộ API mức cao cho phép ứng dụng truy cập đến bản đồ Google sẵn có của Android, bao gồm các lớp MapView, Overlay, MapController để tương tác với bản đồ bên trong ứng dụng.
android.app	Gói ứng dụng là một gói thư viện bậc cao, cho phép truy cập đến dữ liệu của ứng dụng. Gói ứng dụng cũng bao gồm Activity, Service là thành phần cơ bản của mọi ứng dụng Android.
android.provider	Bao gồm một tập các lớp cho phép ứng dụng truy cập đến dữ liệu tiêu chuẩn trong tất cả các bản phân phối Android (như dữ liệu danh bạ).
android.telephony	Cho phép ứng dụng tạo, nhận, theo dõi cuộc gọi, tình trạng cuộc gọi và tin nhắn SMS.
android.webkit	Bao gồm một lớp WebView để tạo giao diện web nhúng trong ứng dụng, làm việc với các nội dung web và một trình quản lý cookie.

Ngoài các thư viện chuẩn của Android, để đáp ứng tiêu chí phù hợp với nhiều thiết bị khác nhau, Android còn có thể có các API phụ thuộc thiết bị như: android.location, android.media, android.opengl, android.hardware, android.bluetooth, android.net.wifi.

Kiến trúc của HĐH Android

Khung ứng dụng Android cung cấp nhiều dịch vụ cấp cao thông qua các API ở dạng các lớp Java nằm trong Android SDK (Ví dụ: dịch vụ kết nối Internet, dịch vụ SMS...).

Các ứng dụng cơ bản của Android :

- **Activity manager, fragment manager**: quản lý chu trình sống và điều hướng các activity, fragment. Đồng thời quản lý activity stack.
- **View system** : một tập hợp rất nhiều các view sử dụng để tạo lập giao diện người dùng cho các activity, fragments.
- **Notification manager**: cung cấp cơ chế ổn định và quy củ cho việc gửi các thông báo đến người dùng thông qua các alerts trong thanh trạng thái (status bar).
- **Content provider**: cho phép các ứng dụng có thể truy cập và chia sẻ dữ liệu với ứng dụng khác (như ứng dụng danh bạ sẵn có trong điện thoại).
- **Resource manager**: cho phép truy xuất các thành phần không thuộc mã nguồn, chẳng hạn như: chuỗi ký tự (localized strings), đồ họa (graphics), file giao diện (layout files).



Kiến trúc của HĐH Android

Các ứng dụng:

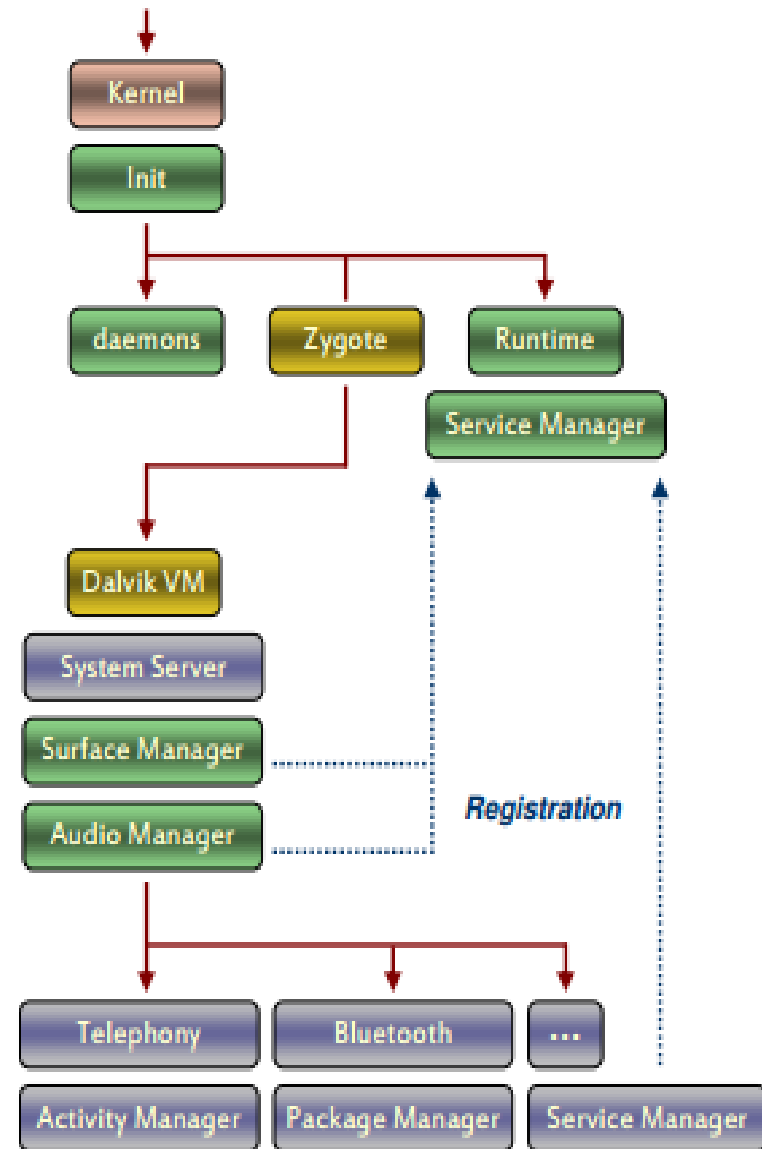
- Đây là tầng trên cùng trong kiến trúc của hệ điều hành Android. Mặc định Android tích hợp sẵn một số ứng dụng cần thiết cơ bản như: home, contacts, phone, browser, camera...
- Tất cả các ứng dụng được viết bằng ngôn ngữ Java và được quản lý bởi máy ảo Dalvik.
- Ứng dụng được phát hành trên các kho ứng dụng (ví dụ: Google Play, Amazon Appstore), người dùng đầu cuối có thể vào kho tải ứng dụng về máy.



Quá trình khởi động HĐH vào điện thoại Android



- **Phần Daemons** làm nhiệm vụ quản lý phần cứng cấp thấp (ví dụ: usb, adb, debugger, radio...).
- **Tiến trình môi Zygote** được gọi nhằm khởi động máy ảo Dalvik (Dalvik VM) để chạy các ứng dụng.
- **Tiến trình Runtime** được gọi nhằm khởi động trình quản lý dịch vụ (Service Manager). Bất cứ một dịch vụ chạy ngầm nào đều phải được đăng ký với Service Manager (Ví dụ: Surface Manager, Audio Manager...)



Các thành phần tạo nên ứng dụng Android

→ Các thành phần chính

Thành phần	Miêu tả
Activities	<ul style="list-style-type: none">Quản lý màn hình giao diện, xử lý tương tác của người dùng.Trong một ứng dụng Android có thể có một hoặc nhiều Activity, trong đó có một main Activity để khởi chạy ứng dụng, Activity này sẽ gọi các Activity khác trong quá trình ứng dụng hoạt động.
Services	Chạy ngầm trong hệ thống mà người sử dụng không thấy được. Service sử dụng để cập nhật dữ liệu, đưa ra thông báo (Notification).
Broadcast Receivers	Quản lý sự kiện, chuyển tiếp sự kiện trong hệ thống (Ví dụ: xử lý truyền thông giữa hệ điều hành và ứng dụng Android).
Content Provider	Cung cấp cách thức quản lý và chia sẻ tài nguyên <u>giữa các ứng dụng</u> .

Các thành phần tạo nên ứng dụng Android

-- Các thành phần khác --

Thành phần	Miêu tả
Fragments	Là thành phần giao diện được nhúng vào Activity. Fragment được thêm vào phiên bản Android 3.0 (API level 11). Sử dụng Fragment cho phép ứng dụng chạy trên các dòng thiết bị có độ phân giải khác nhau có màn hình giao diện ứng dụng phù hợp.
Views	Là các đối tượng hiển thị trên giao diện ứng dụng (Ví dụ: button, list, textbox...).View chỉ được hiển thị lên màn hình khi được đặt trong Activity.
Layouts	Cho phép bố trí sắp xếp các views lên giao diện ứng dụng theo nhu cầu.
Intents	Dùng để truyền các thông báo từ Activity này tới service hoặc Activity khác (Ví dụ: khi mở 1 trang web, bạn gửi 1 Intent đi để tạo 1 Activity mới hiển thị trang web đó).
Resources	Lưu trữ tài nguyên của ứng dụng, như: strings, constants, drawables pictures.
Manifest	File cấu hình của ứng dụng.

Các thành phần tạo nên ứng dụng Android

-- Chú ý --

- Tất cả các thành phần vừa nêu **không nhất thiết** phải có mặt đầy đủ trong ứng dụng Android.
- Mỗi một thành phần trên muốn được triệu gọi trong ứng dụng thì bắt buộc nó **PHẢI** được khai báo trong `AndroidManifest.xml`.
- Chúng ta có thể xem các thành phần nào được sử dụng trong ứng dụng bằng việc xem khai báo trong file **`AndroidManifest.xml`**.

➤ **Thiết lập môi trường lập trình ứng dụng Android**

4 bước thiết lập ...

1. Cài đặt môi trường lập trình Java – JDK (Java Development Kit).
2. Cài đặt Android SDK (Android Software Development Kit).
3. Cập nhật ADT plugin (Android Development Tool) cho Eclipse.
4. Tạo máy ảo Android - AVD (Android Virtual Device).

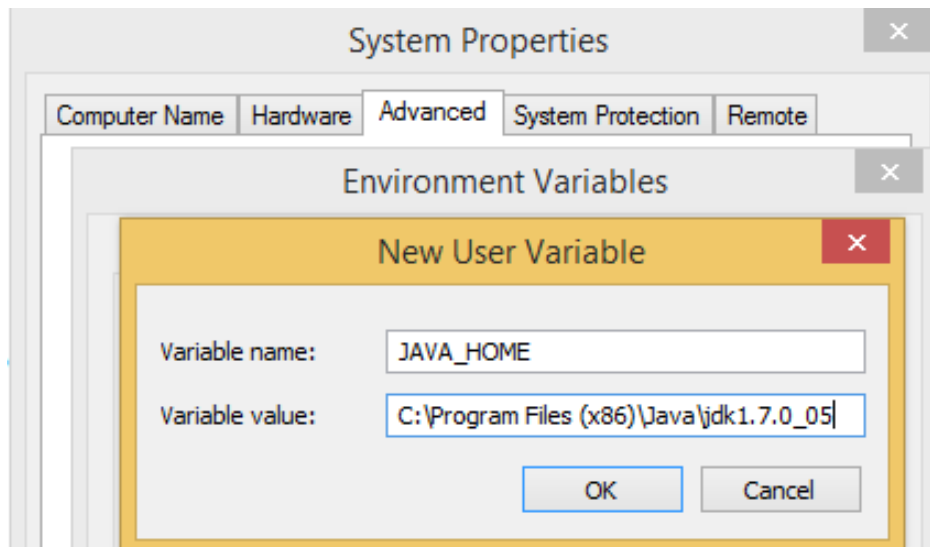
4 bước thiết lập ...

Bước 1. Cài đặt Java JDK

1. Tải và cài đặt Java JDK

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

2. Cập nhật biến môi trường (PATH) giúp việc thực thi và biên dịch mã Java có thể tiến hành ở bất cứ thư mục nào.



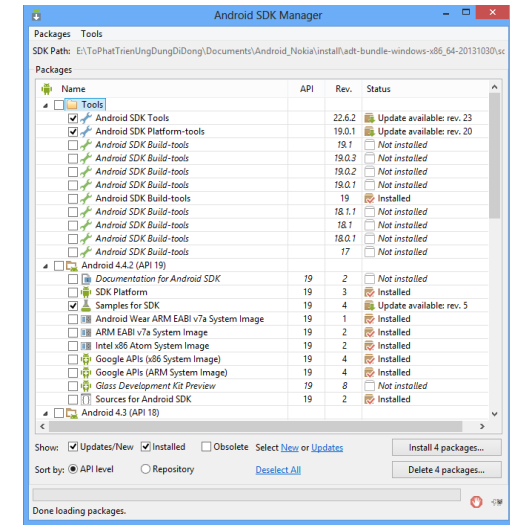
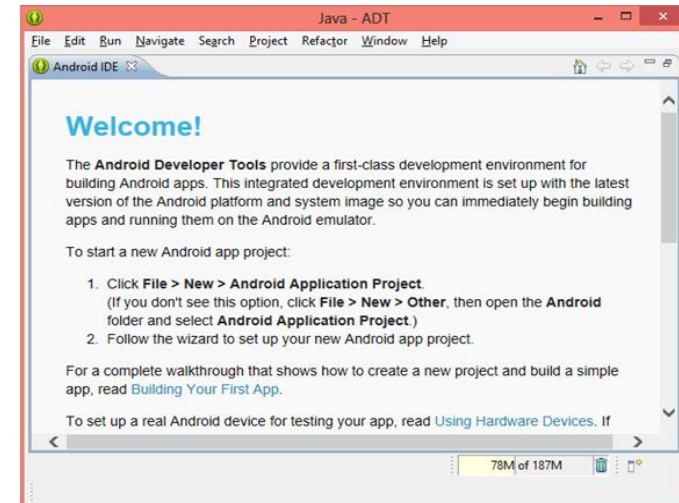
4 bước thiết lập ...

Bước 2. Cài đặt Android SDK

Android SDK: thư viện API, môi trường test (emulator), môi trường phát triển (debug, build, deploy), tài liệu, code mẫu cho ứng dụng Android.

1. Tải và cài đặt Android SDK (gói **ADT bundle for Windows**)

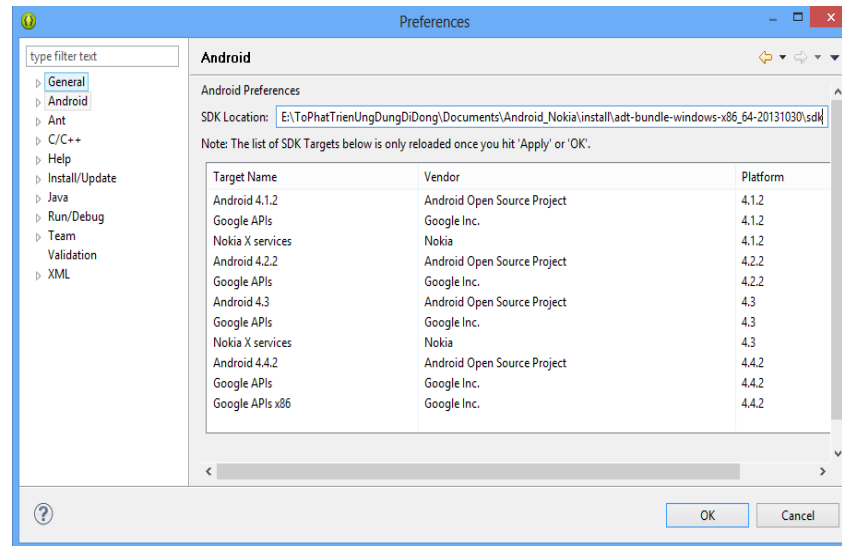
2. Cấu hình Android SDK Manager (vào menu **Windows/Android SDK Manager**)



4 bước thiết lập ...

Bước 3. Cập nhật ADT cho Eclipse

1. Vào menu Help / chọn Install New Software
2. Nhập đường dẫn <https://dl-ssl.google.com/android/eclipse/> và bấm nút Add ở góc bên phải
3. Chọn các thông số thích hợp để tiến hành cập nhật.
4. Tham chiếu Eclipse tới Android SDK: vào menu **Windows/ Preferences/ chọn Android và browser tới SDK location** mà bạn lưu trữ.

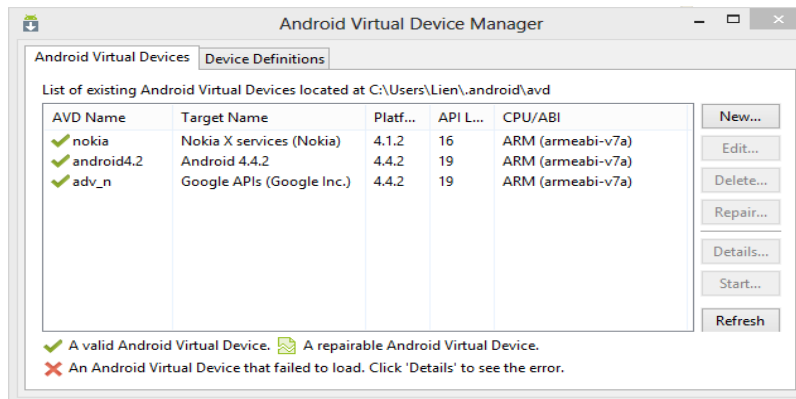


4 bước thiết lập ...

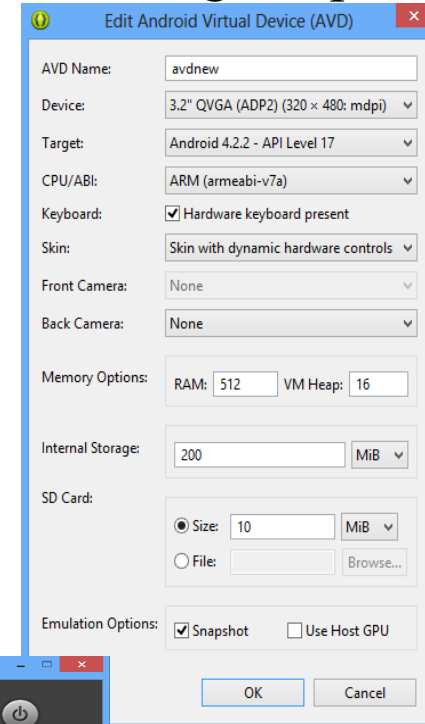
Bước 4. Tạo máy ảo Android - AVD

AVD (Android Virtual Device) cho phép chạy ứng dụng trên thiết bị giả lập.

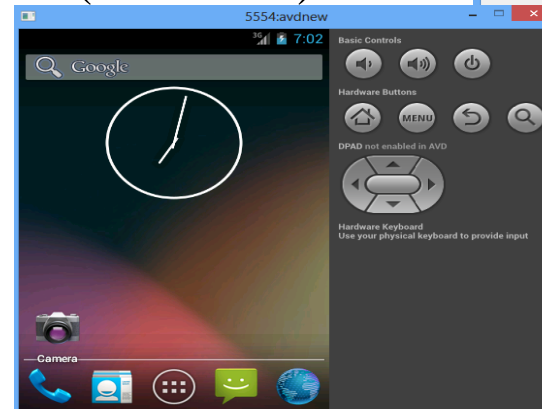
1. Vào menu **Windows/ Android Virtual Device Manager**



2. Click nút **New**, thiết lập các thông số: tên máy ảo, độ phân giải, API, bộ nhớ trong, bộ nhớ ngoài (SD Card)...



3. Chọn tên máy ảo cần chạy trong Android Virtual Device Manager, nhấn **Start** để chạy Android Emulator



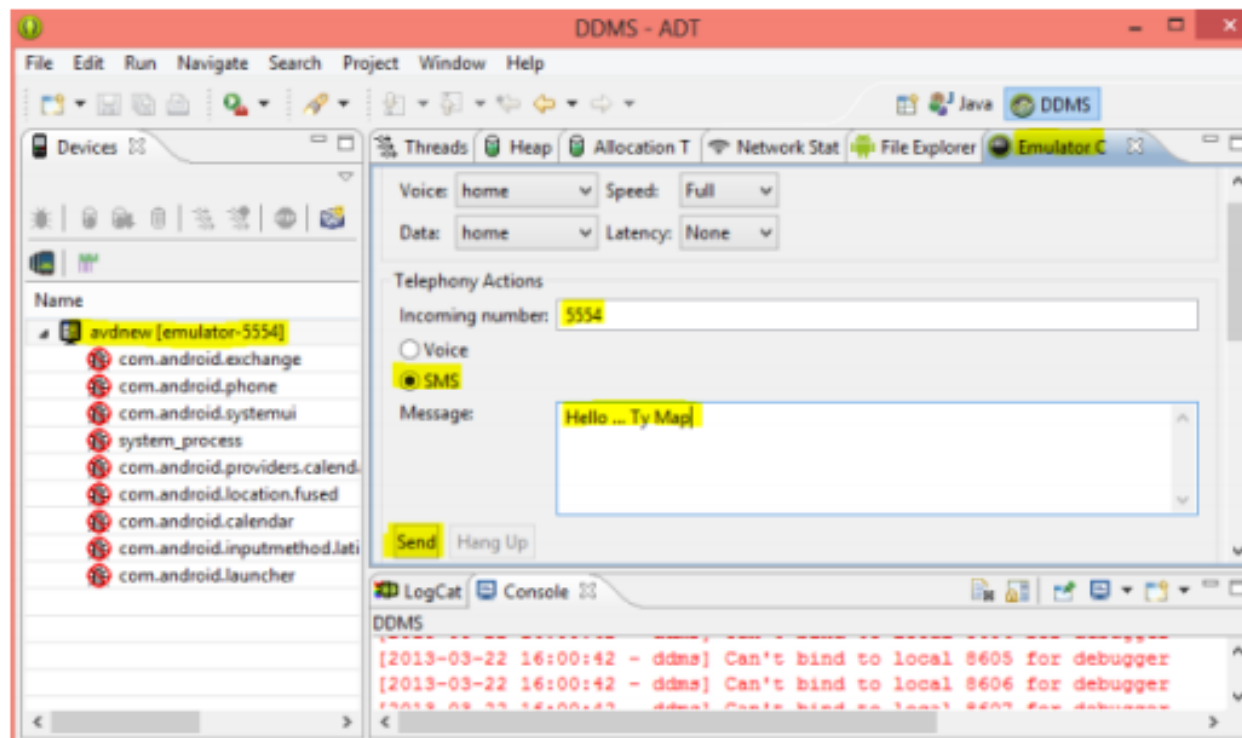
Android Emulator

- Mỗi một máy ảo được khởi động sẽ đi với một port nào đó (ví dụ 5554). Port này chính là đại diện cho số điện thoại của máy ảo đó
- Các phím chức năng: tương tự thiết bị thật
- Nhấn tổ hợp phím **Ctrl + F11** để xoay màn hình

Sử dụng Android Emulator

-- Gửi tin nhắn/ gọi điện thoại trong máy ảo --

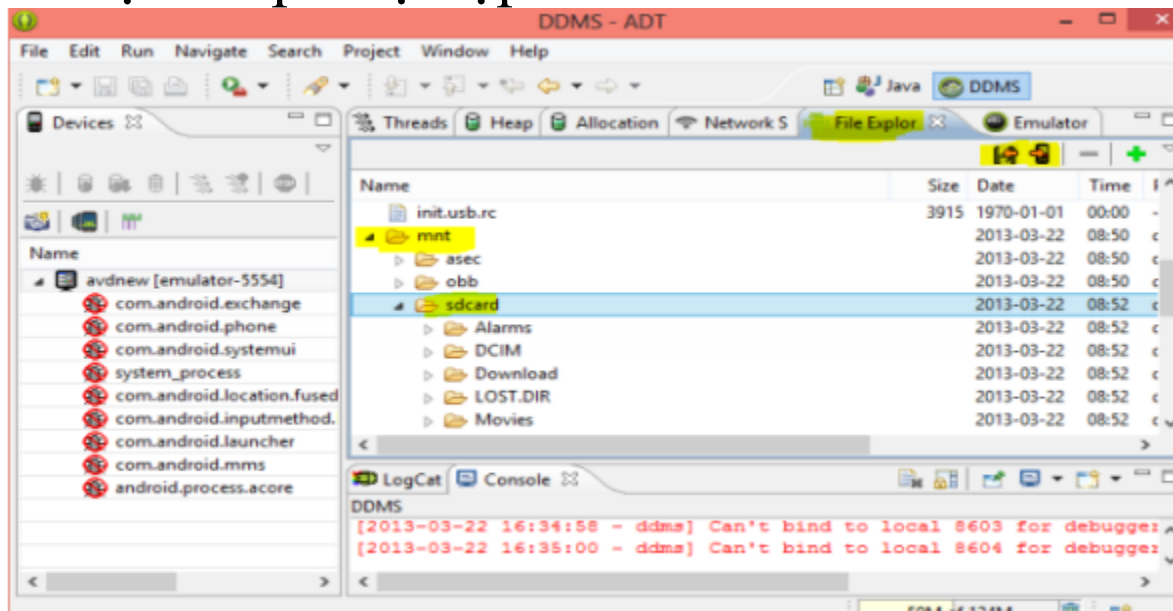
- Cách 1: Tạo thêm 1 máy ảo Android khác
- Cách 2: Sử dụng Emulator Control trong DDMS



Sử dụng Android Emulator

-- Thao tác với File --

- DDMS -> File Explorer
- Để lấy 1 tập tin từ SD Card ra: chọn tập tin đó rồi nhấn vào biểu tượng ổ đĩa mềm
- Để đưa một tập tin từ Máy tính vào SD card: nhấn vào biểu tượng điện thoại rồi chọn tập tin cần đưa, Hoặc bạn kéo thả trực tiếp một tập tin nào đó vào màn hình



➤ **Cơ bản về ngôn ngữ Java cho lập trình ứng dụng Android**

Java

- Giới thiệu ngôn ngữ Java
- Kiến trúc chương trình xây dựng trên Java
- Các kiểu dữ liệu và toán tử cơ bản trên Java
- Các cấu trúc lệnh trên Java
- Kế thừa và đa hình trên Java

Giới thiệu ngôn ngữ Java

- Ngôn ngữ lập trình Java được Sun Microsystems đưa ra giới thiệu vào tháng 6 năm 1995 và đã nhanh chóng trở thành một ngôn ngữ lập trình của các lập trình viên chuyên nghiệp.
- Java được xây dựng dựa trên nền tảng của C và C++ nghĩa là Java sử dụng cú pháp của C và đặc trưng hướng đối tượng của C++. Java là ngôn ngữ vừa biên dịch vừa thông dịch.
- Mục tiêu của các nhà thiết kế Java là cho phép người lập trình viết chương trình một lần nhưng có thể chạy trên các nền phần cứng khác nhau.

Đặc trưng của ngôn ngữ Java

- Đơn giản
- Hướng đối tượng
- Độc lập phần cứng và hệ điều hành
- Mạnh mẽ
- Bảo mật
- Phân tán
- Đa luồng
- Linh động

Cài đặt Java

Quá trình cài đặt môi trường Java trên máy bao gồm ba bước:

1. Download bộ cài đặt:
<http://www.java.sun.com/downloads/>
2. Chạy chương trình cài đặt
3. Cập nhật biến môi trường

Các kiểu ứng dụng Java

1. Ứng dụng Applet
2. Ứng dụng dòng lệnh (console)
3. Ứng dụng đồ họa
4. JSP/Servlet
5. Ứng dụng cơ sở dữ liệu
6. Ứng dụng mạng
7. Ứng dụng nhiều tầng
8. Ứng dụng cho các thiết bị di động

Kiến trúc chương trình Java

```
package packageName; // Khai báo tên gói, nếu có
import java.awt.*; // Khai báo tên thư viện sẵn có, nếu cần dùng
class className // Khai báo tên lớp
{
    /* Đây là dòng ghi chú */
    int var; // Khai báo biến
    public void methodName() // Khai báo tên phương thức
    {
        /* Phần thân của phương thức */
        statement (s); // Lệnh thực hiện
    }
}
```

Khai báo thư viện

Java chuẩn cung cấp một số thư viện như sau:

- **java.lang:** cung cấp các hàm thao tác trên các kiểu dữ liệu cơ bản, xử lý lỗi và ngoại lệ, xử lý vào ra trên các thiết bị chuẩn như bàn phím và màn hình.
- **java.applet:** cung cấp các hàm cho xây dựng các applet
- **java.awt:** cung cấp các hàm cho xây dựng các ứng dụng đồ họa với các thành phần giao diện multimedia
- **java.io:** cung cấp các hàm xử lý vào/ra trên các thiết bị chuẩn và các thiết bị ngoại vi.
- **java.util:** cung cấp các hàm tiện ích trong xử lý liên quan đến các kiểu dữ liệu có cấu trúc như Date, Stack, Vector.

Chương trình java đầu tiên

```
package vidu;  
// Đây là chương trình "First.java"  
class First  
{  
    public static void main(String args[])  
    {  
        System.out.println("Hello World");  
    }  
}
```

>javac vidu/First.java

>java vidu.First

Kết quả sẽ hiển thị trên màn hình như sau:

Hello World

KIỂU DỮ LIỆU

- Dữ liệu : là tất cả những gì được máy tính xử lý. Dữ liệu được chứa trong bộ nhớ máy tính với 1 số lượng ô nhớ nhất định, tính theo đơn vị byte.
- Kiểu dữ liệu được định nghĩa với 2 điểm chính:
 - Một tập hợp các giá trị mà 1 biến thuộc kiểu đó có thể nhận được.
 - Trên đó xác định 1 số phép toán.
- 2 kiểu dữ liệu của Java:
 - Các kiểu dữ liệu cơ bản
 - Các kiểu dữ liệu đối tượng

Các kiểu dữ liệu cơ bản trong Java

byte	Dùng để lưu dữ liệu kiểu số nguyên có kích thước một byte (8 bit). Phạm vi biểu diễn giá trị từ -128 đến 127. Giá trị mặc định là 0.
char	Dùng để lưu dữ liệu kiểu kí tự hoặc số nguyên không âm có kích thước 2 byte (16 bit). Phạm vi biểu diễn giá trị từ 0 đến u\ffff. Giá trị mặc định là 0.
boolean	Dùng để lưu dữ liệu chỉ có hai trạng thái đúng hoặc sai (độ lớn chỉ có 1 bit). Phạm vi biểu diễn giá trị là {"True", "False"}. Giá trị mặc định là False.
short	Dùng để lưu dữ liệu có kiểu số nguyên, kích cỡ 2 byte (16 bit). Phạm vi biểu diễn giá trị từ - 32768 đến 32767. Giá trị mặc định là 0.
int	Dùng để lưu dữ liệu có kiểu số nguyên, kích cỡ 4 byte (32 bit). Phạm vi biểu diễn giá trị từ -2,147,483,648 đến 2,147,483,647. Giá trị mặc định là 0.
float	Dùng để lưu dữ liệu có kiểu số thực, kích cỡ 4 byte (32 bit). Giá trị mặc định là 0.0f.
double	Dùng để lưu dữ liệu có kiểu số thực có kích thước lên đến 8 byte. Giá trị mặc định là 0.00d
long	Dùng để lưu dữ liệu có kiểu số nguyên có kích thước lên đến 8 byte. Giá trị mặc định là 0l.

Các kiểu dữ liệu đối tượng trong Java

Array	Một mảng của các dữ liệu cùng kiểu
class	Dữ liệu kiểu lớp đối tượng do người dùng định nghĩa. Chứa tập các thuộc tính và phương thức.
interface	Dữ liệu kiểu lớp giao tiếp do người dùng định nghĩa. Chứa các phương thức của giao tiếp.

Giá trị mặc định của kiểu dữ liệu

Kiểu dữ liệu	Giá trị mặc định
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	Null
boolean	false
Các kiểu dữ liệu hướng đối tượng	null

KIỂU DỮ LIỆU

Ép kiểu

- Khi cần thực hiện với nhiều kiểu dữ liệu khác nhau cùng lúc

float c = 35.8f;

int b = (int)c + 1;

Đầu tiên giá trị dấu phẩy động c được đổi thành giá trị nguyên 35. Sau đó nó được cộng với 1 và kết quả là giá trị 36 được lưu vào b.

- Trong Java có hai loại ép kiểu dữ liệu:
 - Nới rộng (widening): quá trình làm tròn số từ kiểu dữ liệu có kích thước nhỏ hơn sang kiểu có kích thước lớn hơn. Kiểu biến đổi này không làm mất thông tin. Ví dụ chuyển từ **int** sang **float**. Chuyển kiểu loại này có thể được thực hiện ngầm định bởi trình biên dịch.
 - Thu hẹp (narrowing): quá trình làm tròn số từ kiểu dữ liệu có kích thước lớn hơn sang kiểu có kích thước nhỏ hơn. Kiểu biến đổi này có thể làm mất thông tin như ví dụ ở trên. Chuyển kiểu loại này không thể thực hiện ngầm định bởi trình biên dịch, người dùng phải thực hiện chuyển kiểu tường minh.

Các toán tử trong Java

- Toán tử số học
- Toán tử bit
- Toán tử quan hệ
- Toán tử logic
- Toán tử điều kiện
- Toán tử gán

Toán tử số học

Phép toán	Sử dụng	Mô tả
+	$op1 + op2$	Cộng $op1$ với $op2$
-	$op1 - op2$	Trừ $op1$ cho $op2$
*	$op1 * op2$	Nhân $op1$ với $op2$
/	$op1 / op2$	chia $op1$ cho $op2$
%	$op1 \% op2$	Tính phần dư của phép chia $op1$ cho $op2$

Toán tử số học

Phép toán	Sử dụng	Mô tả
++	op++	Tăng op lên 1 đơn vị, giá trị của op được tăng lên trước khi biểu thức chứa nó được tính
++	++op	Tăng op lên 1 đơn vị, giá trị của op được tăng lên sau khi biểu thức chứa nó được tính
--	op--	Giảm op xuống 1 đơn vị, giá trị của op được giảm xuống trước khi biểu thức chứa nó được tính
--	--op	Giảm op xuống 1 đơn vị, giá trị của op được giảm xuống sau khi biểu thức chứa nó được tính

Toán tử bit

Phép toán	Sử dụng	Thực hiện
&	op1 & op2	Thực hiện phép and các bit tương ứng của op1 với op2
	op1 op2	Thực hiện phép or các bit tương ứng của op1 với op2
^	op1 ^ op2	Thực hiện phép xor các bit tương ứng của op1 với op2
~	~op2	Thực hiện phép lật các bit của op2

Toán tử bit

Phép toán	Sử dụng	Kết quả
>>	$op1 \gg op2$	Dịch chuyển $op1$ sang phải $op2$ bit, $op2$ bit phía bên phải sẽ được điền bằng các bit 0
<<	$op1 \ll op2$	Dịch chuyển $op1$ sang trái $op2$ bit(giữ nguyên dấu của $op1$), $op2$ bit nằm bên trái sẽ được điền bằng các bit 0

Toán tử quan hệ

Phép toán	Sử dụng	Nhận về giá trị true khi
>	$op1 > op2$	op1 lớn hơn op2
>=	$op1 \geq op2$	op1 lớn hơn hoặc bằng op2
<	$op1 < op2$	op1 nhỏ hơn op2
<=	$op1 \leq op2$	op1 nhỏ hơn hoặc bằng op2
==	$op1 == op2$	op1 bằng op2
!=	$op1 \neq op2$	op1 khác op2

Toán tử logic

Phép toán	Sử dụng	Nhận về giá trị true khi
&&	op1 && op2	Cả op1 và op2 đều là true, giá trị của op2 chỉ được tính khi op1 là true
	op1 op2	Hoặc op1 hoặc op2 là true, giá trị của op2 chỉ được tính khi op1 là false
!	! op	op là false
&	op1 & op2	Cả op1 và op2 đều là true, giá trị của op2 luôn được tính kể cả khi op1 là false
	op1 op2	Hoặc op1 hoặc op2 là true, giá trị của op2 luôn luôn được tính kể cả khi op1 là true
^	op1 ^ op2	Nếu op1 khác op2

Toán tử điều kiện

Gồm ba thành phần cấu thành biểu thức điều kiện.
Cú pháp:

<biểu thức 1> ? <biểu thức 2> : <biểu thức 3>;

- biểu thức 1: Biểu thức logic. Trả về giá trị True hoặc False
- biểu thức 2: Là giá trị trả về nếu <biểu thức 1> xác định là True
- biểu thức 3: Là giá trị trả về nếu <biểu thức 1> xác định là False

Toán tử gán

- Toán tử gán (=) dùng để gán một giá trị vào một biến và có thể gán nhiều giá trị cho nhiều biến cùng một lúc.
- Ví dụ :

```
int var = 20;  
int p,q,r,s;  
p=q=r=s=var;
```

Thứ tự ưu tiên của các toán tử

Thứ tự	Toán tử
1	Các toán tử đơn như +, -, ++, --
2	Các toán tử số học và các toán tử dịch như *, /, +, -, <<, >>
3	Các toán tử quan hệ như >, <, >=, <=, =, !=
4	Các toán tử logic và Bit như &&, , &, , ^
5	Các toán tử gán như =, *=, /=, +=, -=

Các lệnh vào/ra

- Java cung cấp sẵn một bộ thư viện chứa các hàm xử lý vào/ra trên các thiết bị chuẩn và các thiết bị ngoại vi là **java.io**.
- Miêu tả chi tiết về các lệnh vào ra trong Java:
<http://docs.oracle.com/javase/tutorial/essential/io/index.html>

Các lệnh điều khiển

Java cung cấp hai loại cấu trúc điều khiển:

Điều khiển rẽ nhánh

- Mệnh đề if-else
- Mệnh đề switch-case

Vòng lặp (Loops)

- Vòng lặp while
- Vòng lặp do-while
- Vòng lặp for

Điều khiển rẽ nhánh

- **Câu lệnh if-else**

```
if (conditon)
{
    action1 statements;
}
else
{
    action2 statements;
}
```

Condition: Biểu thức boolean như toán tử so sánh.

action 1: Khối lệnh được thực thi khi giá trị điều kiện là True

action 2: Khối lệnh được thực thi nếu điều kiện trả về giá trị False

Ví dụ chương trình sử dụng câu lệnh if-else

```
class CheckNumber
{
    public static void main(String args[])
    {
        int num = 10;
        if(num%5 == 0)
            System.out.println (num + " is divisible for 5!");
        else
            System.out.println (num + " is indivisible for 5!");
    }
}
```

- **Câu lệnh switch-case**

```
switch (expression)
{
    case 'value1': action 1 statement;
    break;
    case 'value2': action 2 statement;
    break;
    .....
    case 'valueN': actionN statement;
    break;
    default: default_action statement;
}
```

- Ví dụ chương trình sử dụng switch-case

```
class SwitchDemo
{
    public static void main(String args[])
    {
        int day = 2;
        switch(day)
        {
            case 0 : System.out.println("Sunday");
                      break;
            case 1 : System.out.println("Monday");
                      break;
            case 2 : System.out.println("Tuesday");
                      break;
            case 3 : System.out.println("Wednesday");
                      break;
            case 4 : System.out.println("Thursday");
                      break;
            case 5 : System.out.println("Friday");
                      break;
            case 6 : System.out.println("Saturday");
                      break;
            default: System.out.println("Invalid day of week");
        }
    }
}
```


Vòng lặp

- Vòng lặp while

```
while(condition)
{
    action statements;
}
```

condition : có giá trị bool; vòng lặp sẽ tiếp tục cho nếu điều kiện vẫn có giá trị True.

action statement : Khối lệnh được thực hiện nếu condition nhận giá trị True

Ví dụ vòng lặp while

```
class WhileDemo
{
    public static void main(String args[])
    {
        int a = 5, sum = 1;
        while (a >= 1)
        {
            sum += a;
            a--;
        }
        System.out.println("The sum is " + sum);
    }
}
```

- Vòng lặp do-while

```
do{  
    action statements;  
}while(condition);
```

condition: Biểu thức bool; vòng lặp sẽ tiếp tục khi mà điều kiện vẫn có giá trị True.

action statement: Khối lệnh luôn được thực hiện ở lần thứ nhất, từ vòng lặp thứ hai, chúng được thực hiện khi condition nhận giá trị True.

Ví dụ vòng lặp do-while

```
class DoWhileDemo
{
    public static void main(String args[])
    {
        int a = 1, sum = 0;
        do{
            sum += a;
            a++;
        }while (a <= 5);
        System.out.println("Sum of 1 to 5 is " + sum);
    }
}
```

- Vòng lặp for

```
for(initialization statements; condition; increment statements)
{
    action statements;
}
```

Ví dụ vòng lặp for

```
class ForDemo
{
    public static void main(String args[])
    {
        int sum = 0;
        for (int i=1; i<=5; i++)
            sum += i;
        System.out.println ("The sum is " + sum);
    }
}
```

KẾ THỪA VÀ ĐA HÌNH TRÊN JAVA

- Kế thừa đơn
- Kế thừa kép
- Lớp trừu tượng
- Đa hình

Kế thừa đơn

- Khai báo lớp kế thừa được thực hiện bởi từ khóa **extends**

```
<tên lớp con> extends <tên lớp cha>  
{  
}
```


Ví dụ kế thừa đơn

```
class Person
{
    public String name;
    public int age;
    // Phương thức khởi dựng
    public Person(String name, int age)
    {
        this.name = name;
        this.age = age;
    }
    public void show()
    {
        System.out.println(name + " is " + age + " years old!");
    }
}
```

```
class Employee extends Person
{
    public float salary;
    // Phương thức khởi dựng
    public Employee(String name, int age, float salary)
    {
        super(name, age);
        this.salary = salary;
    }
}
```

Kế thừa kép

- Cú pháp khai báo 1 giao tiếp

```
[public] interface <tên giao tiếp> [extends <danh sách giao tiếp>]  
{  
}
```

- Sử dụng giao tiếp

```
<tính chất> class <tên lớp> implements <các giao tiếp>  
{  
}
```

Ví dụ kế thừa kép

```
public interface Product
{
    public static final String MARK = "Adidas";
    public float getCost();
}
```

```
public class Shoe implements Product
{
    // Cài đặt phương thức được khai báo trong giao tiếp
    public float getCost()
    {
        return 10f;
    }
    // Phương thức truy nhập nhãn hiệu sản phẩm
    public String getMark()
    {
        return MARK;
    }
    // Phương thức main
    public static void main(String args[])
    {
        Shoe myShoe = new Shoe();
        System.out.println("This shoe is " + myShoe.getMark() +
            " having a cost of $" + myShoe.getCost());
    }
}
```

Lớp trừu tượng

- Khai báo lớp trừu tượng

```
[public] abstract class <tên lớp>  
{  
}
```

- Ví dụ khai báo và sử dụng lớp trừu tượng

```
abstract class Animal
{
    abstract String getName();
    abstract int getFeet();
}
```

```
public class Bird extends Animal
{
    // Trả về tên loài chim
    public String getName()
    {
        return "Bird";
    }

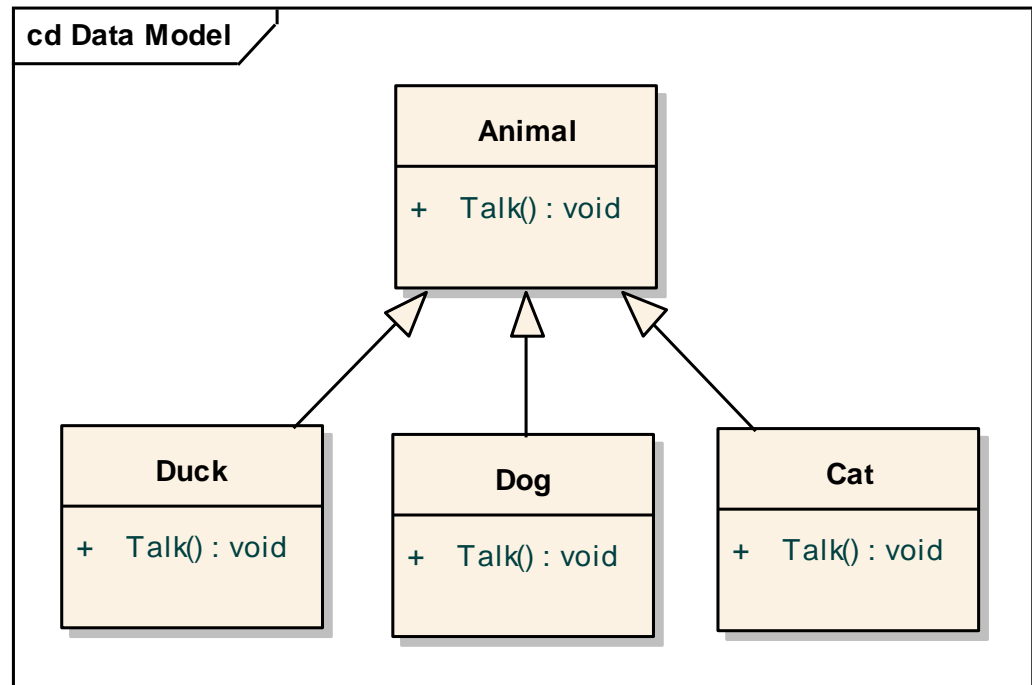
    // Trả về số chân của loài chim
    public int getFeet()
    {
        return 2;
    }
}
```

Đa hình

- **Nạp chồng**
 - Java cho phép trong cùng 1 lớp, có thể khai báo nhiều phương thức có cùng tên (danh sách các đối số khác nhau / kiểu trả về khác nhau)
- **Đa hình**

7 - Tính đa hình (polymorphism)

- Cùng 1 lệnh gửi thông điệp đến đối tượng thông qua cùng 1 tham khảo nhưng ở vị trí/thời điểm khác nhau có thể gây ra việc thực thi method khác nhau của các đối tượng khác nhau



Ví dụ tính đa hình



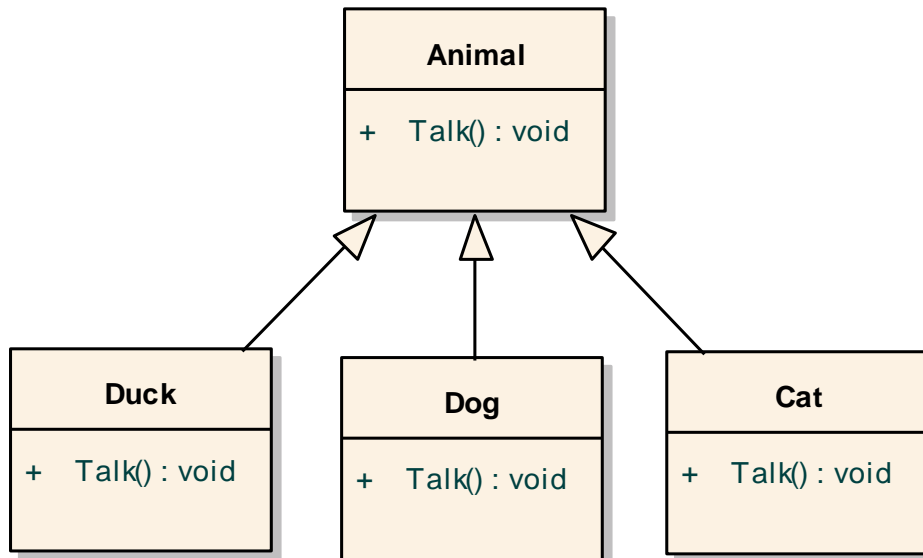
Cài đặt / Java:

```
Animal a;  
a= new Animal();  
a.Talk();
```

```
a= new Dog();  
a.Talk();
```

```
a= new Cat();  
a.Talk();
```

cd Data Model



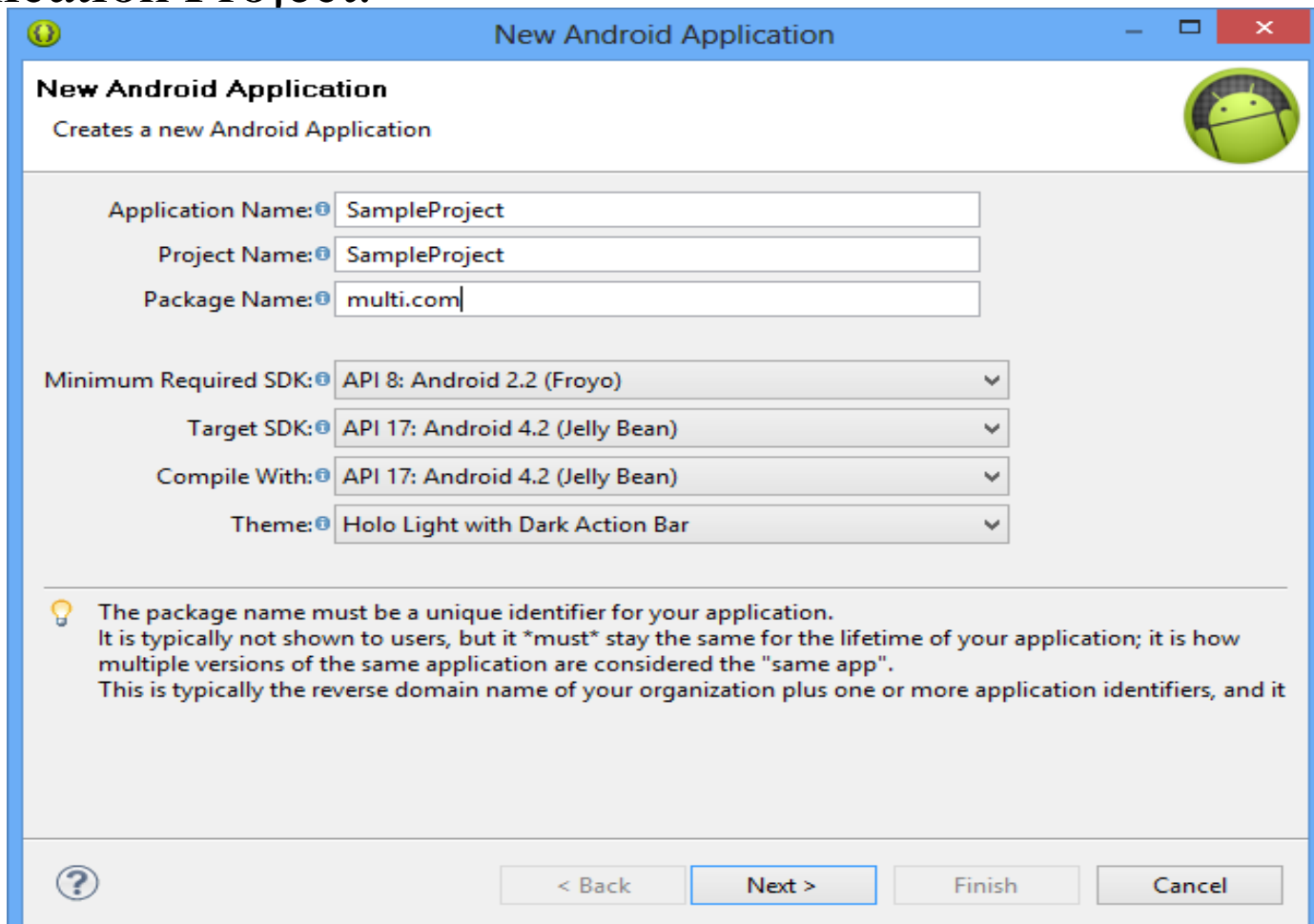
Review

1. Trình bày và nêu ví dụ về các lệnh rẽ nhánh trong Java.
2. Trình bày và nêu ví dụ về các lệnh điều khiển lặp trong Java.
3. Trình bày và lấy ví dụ về lớp và các đối tượng của lớp.
4. Trình bày và lấy ví dụ về kế thừa đơn.
5. Trình bày và lấy ví dụ về kế thừa kép.
6. Trình bày và lấy ví dụ về lớp trừu tượng.
7. Nêu ý nghĩa và cách sử dụng của nạp chồng và đa hình.

➤ **Quy trình lập trình ứng dụng Android**

KHỞI TẠO ỨNG DỤNG

- Sử dụng Eclipse trong gói ADT Bundle for windows, tạo mới Android project bằng cách chọn menu File -> New -> Android Application Project.



New Android Application
Creates a new Android Application

Application Name:

Project Name:


Package Name:


Minimum Required SDK:

Target SDK:

Compile With:

Theme:

 The package name must be a unique identifier for your application. It is typically not shown to users, but it **must** stay the same for the lifetime of your application; it is how multiple versions of the same application are considered the "same app". This is typically the reverse domain name of your organization plus one or more application identifiers, and it





Quick Access

Java

Package Explorer

- SampleProject
 - src
 - multi.com
 - MainActivity.java
 - gen [Generated Java Files]
 - android.support.v7.appcompat
 - multi.com
 - BuildConfig.java
 - R.java
 - Android 4.2.2
 - Android Private Libraries
 - Android Dependencies
 - assets
 - bin
 - libs
 - res
 - drawable-hdpi
 - drawable-ldpi
 - drawable-mdpi
 - drawable-xhdpi
 - drawable-xxhdpi
 - layout
 - activity_main.xml
 - fragment_main.xml
 - menu
 - values
 - dimens.xml
 - strings.xml
 - styles.xml
 - values-v11
 - values-v14
 - values-w820dp
 - AndroidManifest.xml
 - ic_launcher-web.png
 - proguard-project.txt
 - project.properties

Outli...

An outline is not available.

Problems @ Javadoc Declaration

2 errors, 309 warnings, 0 others (Filter matched 102 of 311 items)

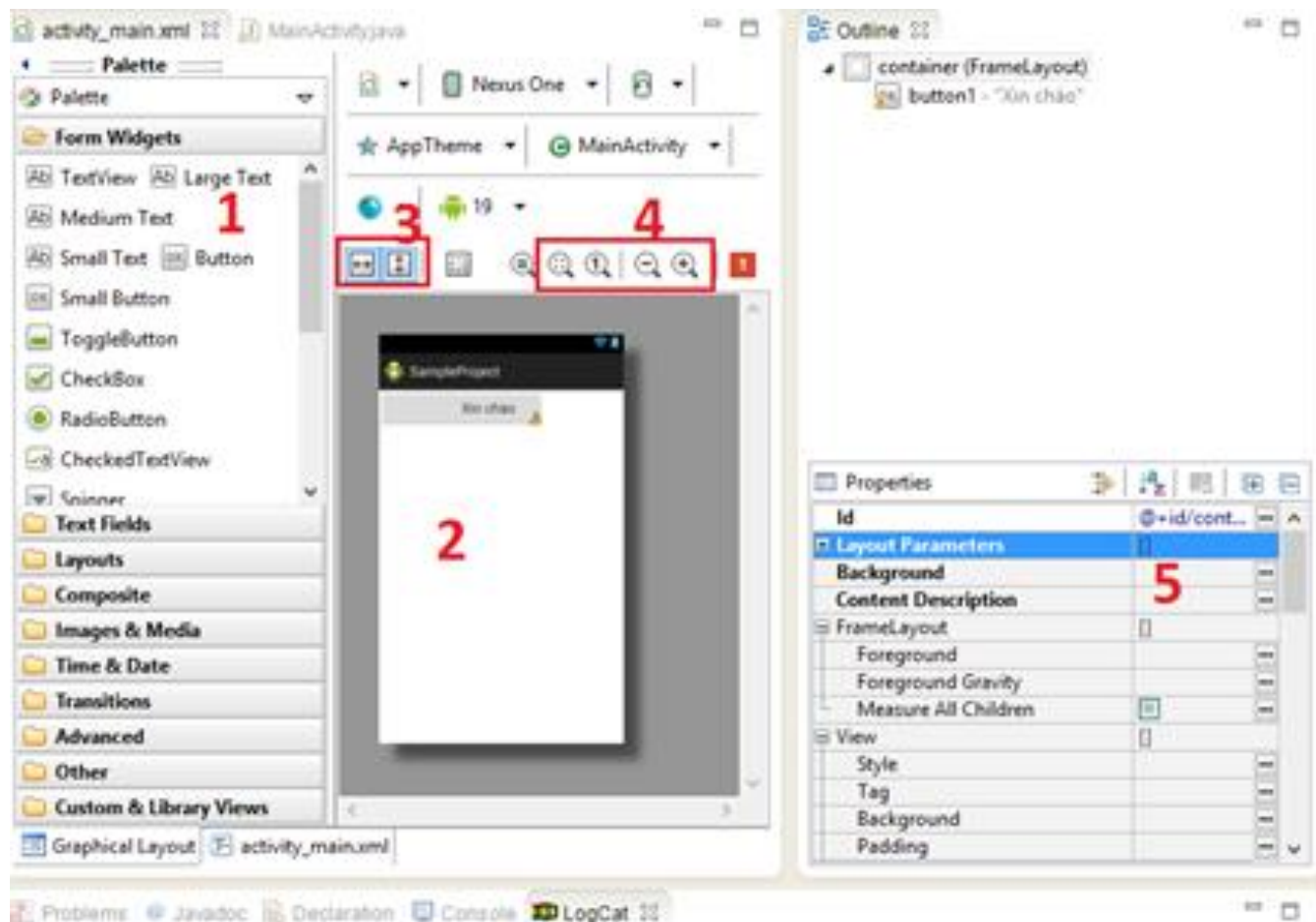
Description	Resource	Path
Errors (2 items)		
Warnings (100 of 309 items)		

File / folder	Miêu tả
src	Thư mục chứa các file mã nguồn .java cho Project
gen	Thư mục chứa file R.java – 1 file được trình biên dịch tự động sinh ra nhằm tham chiếu tới tất cả các tài nguyên trong dự án.
Android 4.2.2	Thư mục này chứa một file, android.jar, chứa tất cả các lớp thư viện Android 4.2.2 cho phát triển ứng dụng Android.
bin	Thư mục chứa file *.apk (Android Package file) được build bởi ADT để chạy chương trình.
assets	Thư mục chứa các tài nguyên của ứng dụng (ví dụ: text file, database...)
res/drawable-hdpi res/drawable-ldpi res/drawable-mdpi res/drawable-xhdpi	Thư mục chứa các đối tượng drawable được thiết kế dành cho các màn hình có độ phân giải màn hình khác nhau. Tùy thuộc vào độ phân giải màn hình, chương trình sẽ tải các tài nguyên (Resource) trong thư mục tương ứng.
res/layout	Thư mục chứa các file cho việc thiết kế giao diện.
res/values	Thư mục gồm các file XML chứa 1 tập hợp các tài nguyên. Như : định nghĩa về văn bản hiển thị trong giao diện thông qua file strings.xml; Định nghĩa màu sắc, kiểu dáng giao diện thông qua file styles.xml.
AndroidManifest.xml	File manifest mô tả các đặc điểm cơ bản của ứng dụng và định nghĩa các thành phần của ứng dụng.

Thiết kế giao diện ứng dụng

-- Kéo thả thông qua giao diện đồ họa --

- GraphicalLayout (Nhấp đúp vào activity_main.xml)



Thiết kế giao diện ứng dụng

-- Thông qua mã xml --

- activity_main.xml cho phép tạo giao diện bằng mã xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="multi.com.MainActivity"
    tools:ignore="MergeRootFrame" >

    <Button
        android:id="@+id/btnXinChao"
        android:layout_width="216dp"
        android:layout_height="wrap_content"
        android:text="Xin chào" />

</FrameLayout>
```

AndroidManifest.xml

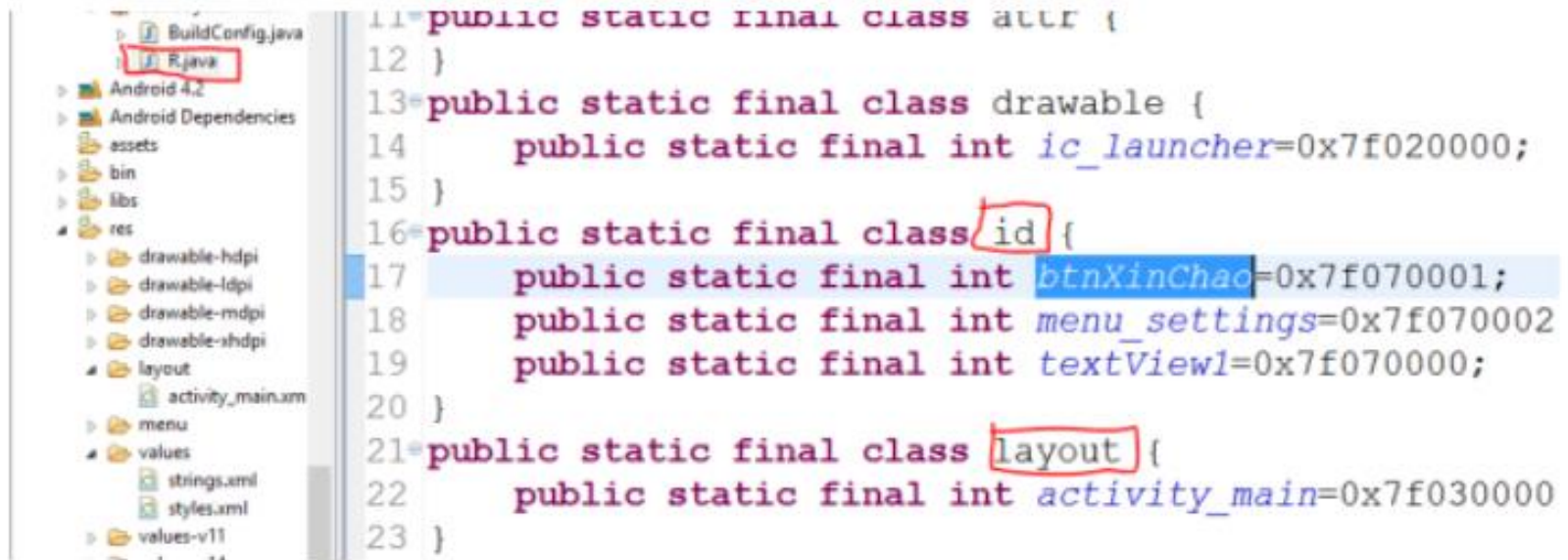
- Bất kỳ một Activity nào muốn được triệu gọi thành công trong Android Project thì bắt buộc nó phải được khai báo trong tập tin AndroidManifest.xml. Phần tử activity chứa phần tử con <intentfilter> cho biết activity này được thực thi đầu tiên khi chạy ứng dụng

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

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

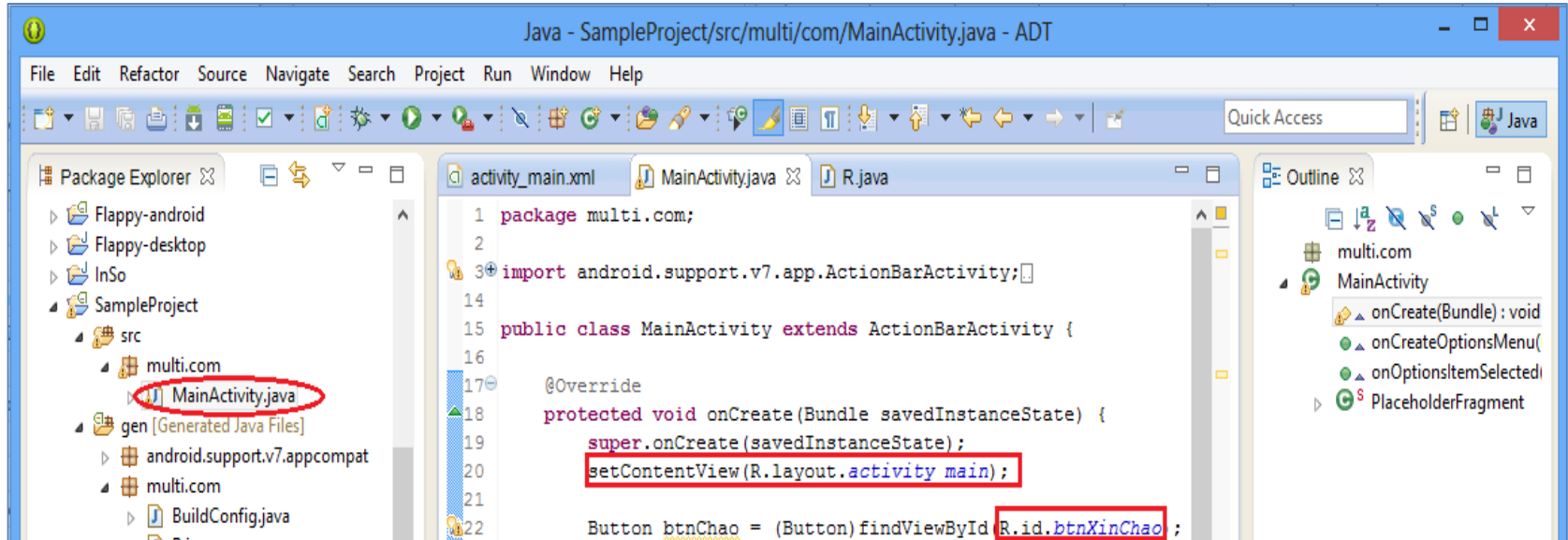

Gen/R.java

- Thư mục **gen** là thư mục do Android tự động tạo ra. Các resource của ứng dụng sẽ được sinh ra bên trong **R.java**. Dựa vào đây ta có thể truy suất các đối tượng khi coding.



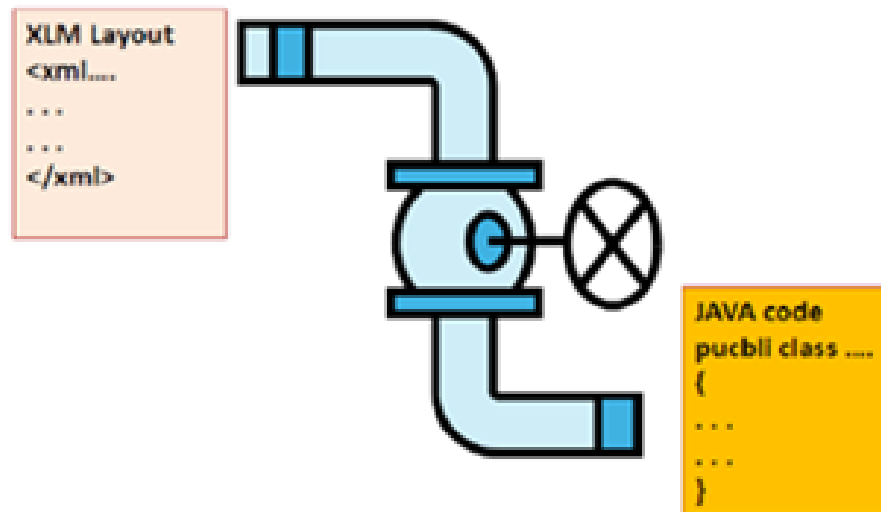
Viết code

- Nhấp đúp chuột vào MainActivity.java, viết code xử lý cho ứng dụng như sau:



Biên dịch chương trình

- Bấm chuột phải vào ứng dụng trong Package Explorer, chọn **Run As**, chọn Android Application. Khi đó Android Project sẽ được tự động biên dịch và chuyển qua Android Executable (.dex).
- Tại thời điểm biên dịch ứng dụng, Android sẽ dịch mã XML thành lớp giao diện Android dưới dạng mã Java (Java bytecode).

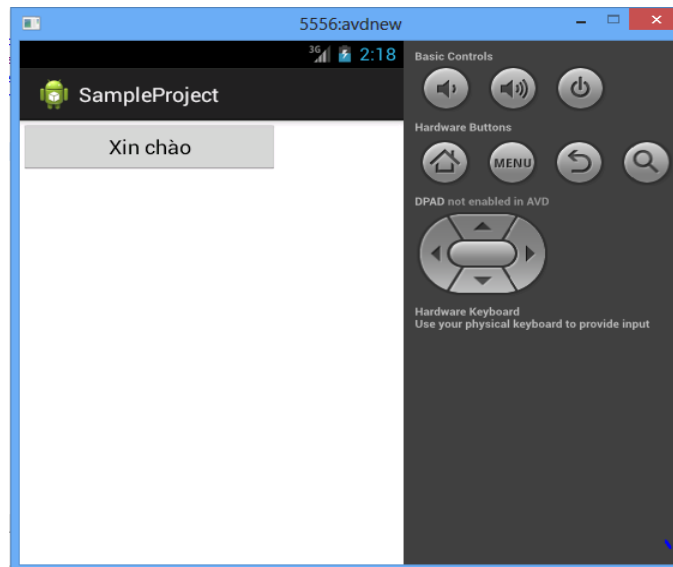


Hình 2.29. Cơ chế dịch XML thành Java code

CHẠY VÀ GỠ LỖI CHƯƠNG TRÌNH

Chạy chương trình

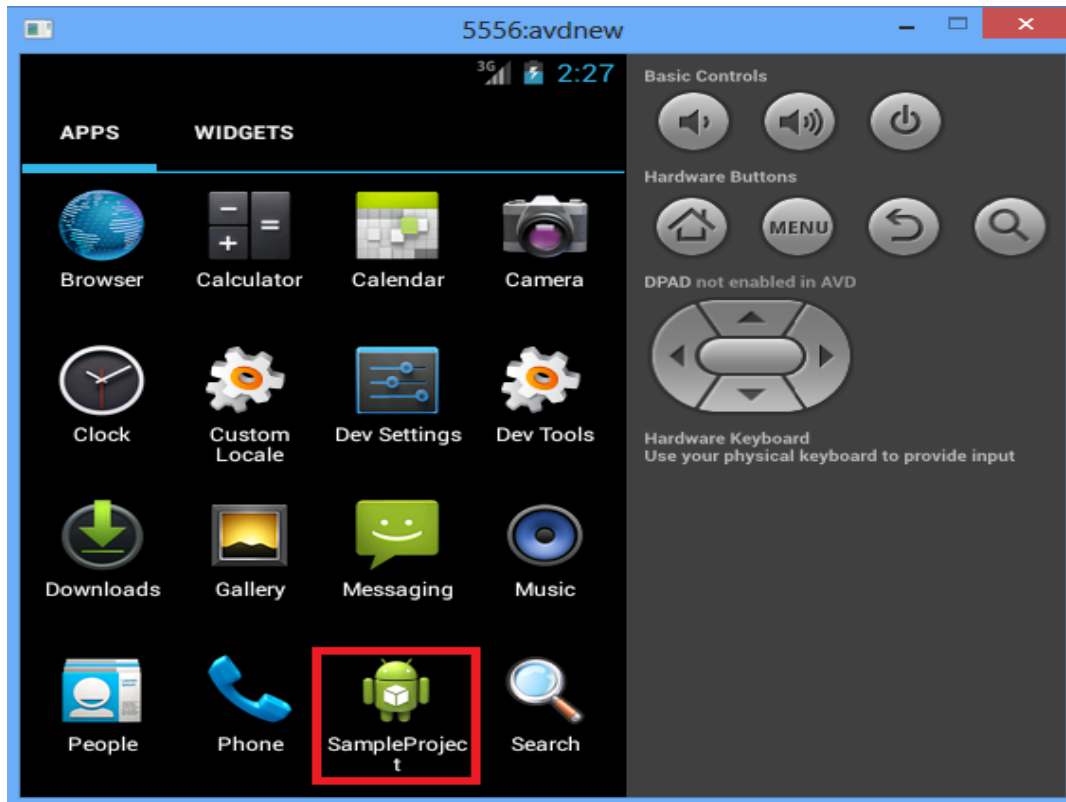
- Sau khi biên dịch, ứng dụng sẽ được đóng gói thành tập tin .apk. Đồng thời, dựa vào loại thiết bị/ emulator, file .apk sẽ được tải và cài đặt vào thiết bị android được chọn.
- Tại thời điểm chạy ứng dụng: đầu tiên phương thức onCreate() của activity được thực thi cho phép giao diện ứng dụng được kết nối vào activity qua hàm setContentView().



CHẠY VÀ GỠ LỖI CHƯƠNG TRÌNH

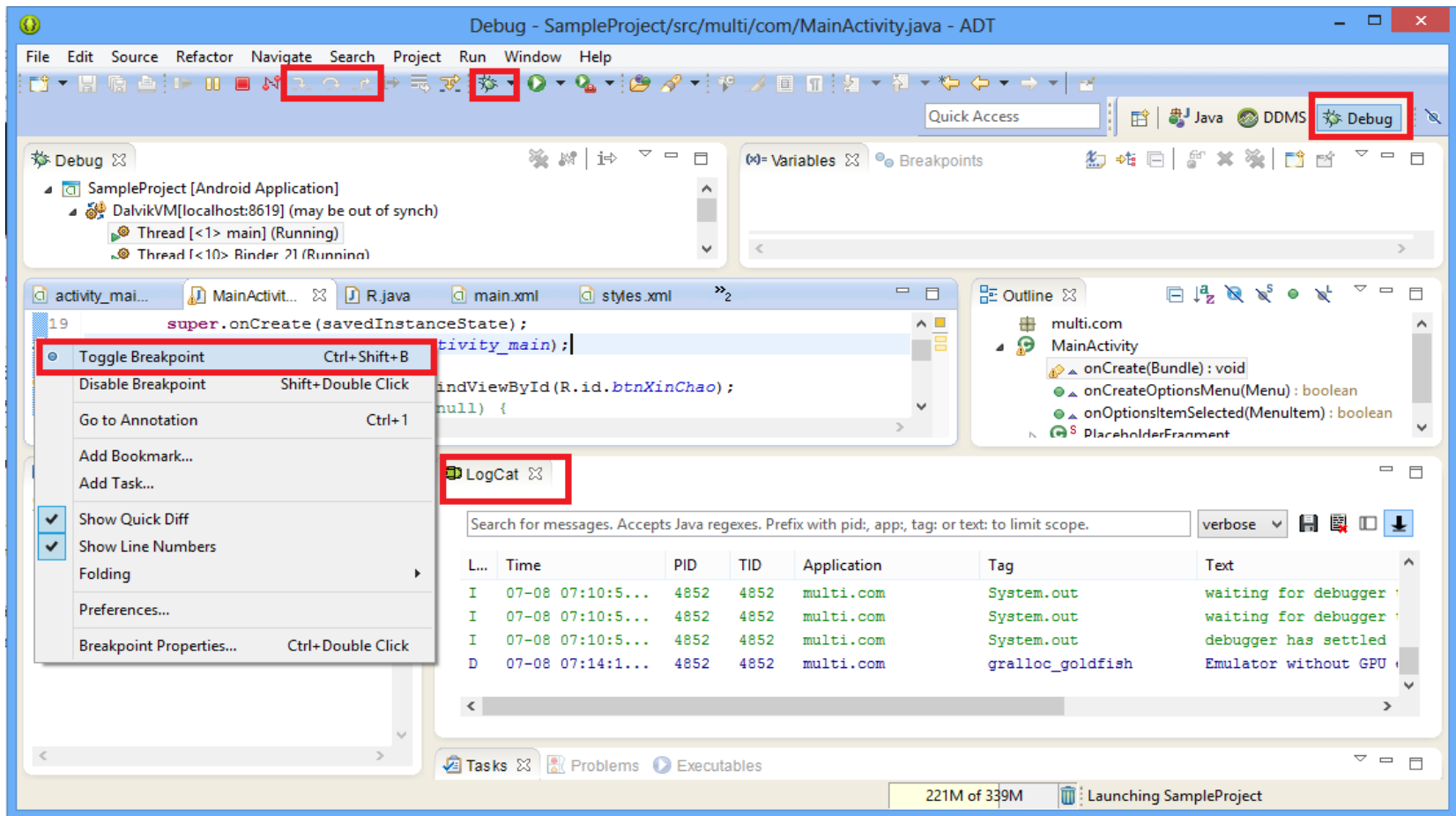
Chạy chương trình

Chọn Home → launcher → SampleProject



CHẠY VÀ GỠ LỖ CHƯƠNG TRÌNH

Gỡ lỗi : Windows → Open Perspective → Debug



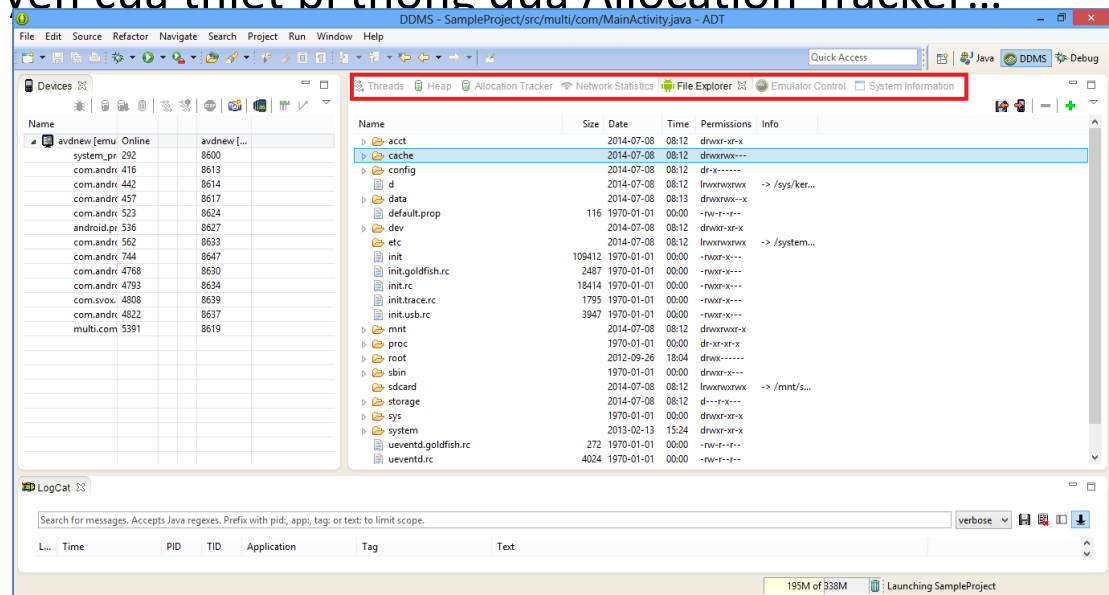
CHẠY VÀ GỠ LỖI CHƯƠNG TRÌNH

Gỡ lỗi : Windows → Open Perspective → Debug

- Quá trình debug ứng dụng chạy ứng dụng qua các breakpoint được đặt.
- Chương trình dừng lại cho ta quan sát kết quả tại các breakpoint bằng Toggle Breakpoint.
- F11 để debug
- Một số thao tác trên debug
 - Bấm (hoặc phím tắt F6) để chương trình chạy qua từng dòng lệnh.
 - Bấm (hoặc phím tắt F5) để chương trình chạy sâu vào thực hiện từng lệnh được đặt điểm neo.
 - Bấm (hoặc phím tắt F8) để chương trình chạy qua hết các điểm đặt điểm neo.
 - Bấm (hoặc phím tắt ctr+F2) để dừng quá trình debug.

DDMS (Dalvik Debug Monitor Server)

- Công cụ cho phép quản lý thiết bị :
 - Thiết lập số nguồn đích nhằm test chức năng thoại/ nhắn tin thông qua Emulator Control
 - Quản lý file thông qua File Explorer
 - Theo dõi mạng thông qua Network Statistics
 - Theo dõi bộ nhớ cung cấp cho thiết bị thông qua Heap
 - Theo dõi thông tin hệ thống thông qua System Information
 - Quản lý sự di chuyển của thiết bị thông qua Allocation Tracker...



Tổng kết chương 2

- Nội dung chương 2 đã trình bày các vấn đề cơ bản liên quan đến môi trường lập trình ứng dụng di động trên nền tảng di động phổ biến nhất hiện nay là Android, đó là :
 - Những khái niệm cơ bản về hệ điều hành Android, quá trình thiết lập môi trường lập trình ứng dụng di động trên nền tảng Android.
 - Các kiến thức cơ bản về ngôn ngữ Java cho lập trình ứng dụng di động Android: Tập ký tự, từ khóa, tên; Cấu trúc chung của chương trình và qui tắc viết chương trình; Khái niệm, phân loại kiểu dữ liệu; Biến, hằng và các phép toán; Các lệnh vào ra và các lệnh điều khiển; Lớp và đối tượng; Kế thừa và đa hình; Các thư viện chuẩn trong Java.
 - Quy trình lập trình ứng dụng di động Android: khởi tạo ứng dụng, viết chương trình, biên dịch, chạy chương trình và gỡ lỗi.
- Trong chương 3 tiếp theo, bài giảng sẽ đi sâu vào trình bày các kỹ thuật lập trình cốt lõi để xây dựng lên ứng dụng Android.

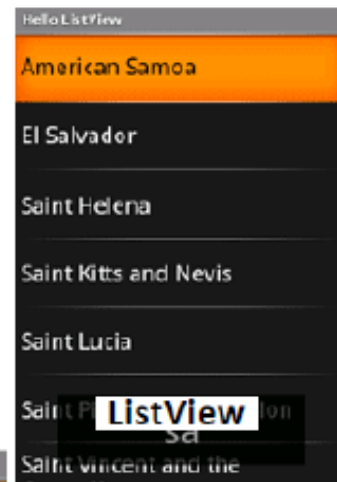
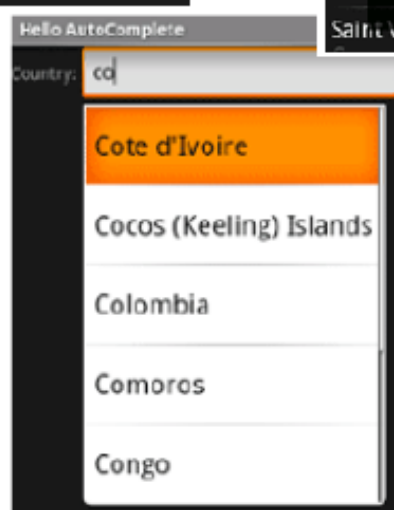
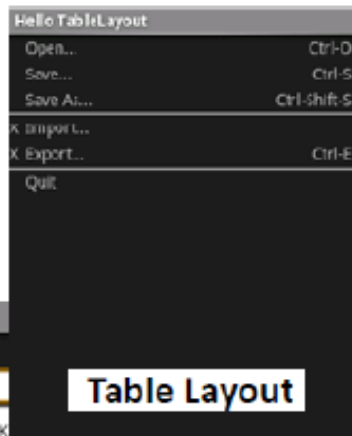
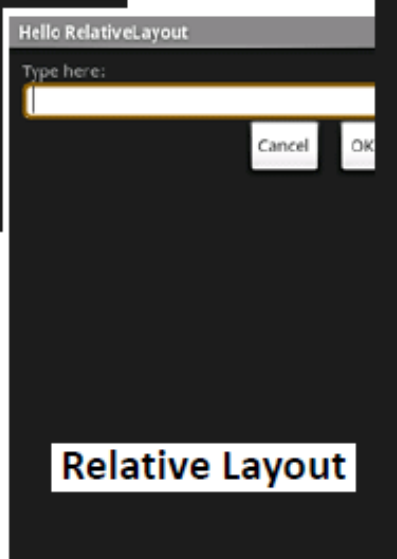
Chuẩn bị cho buổi sau

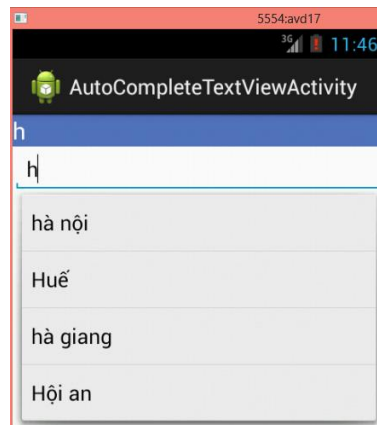
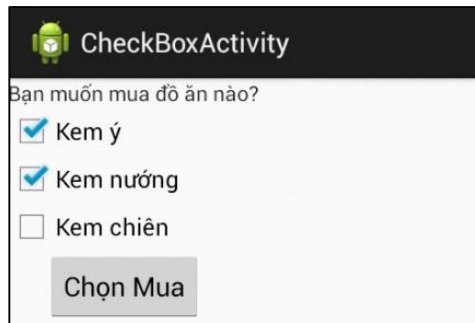
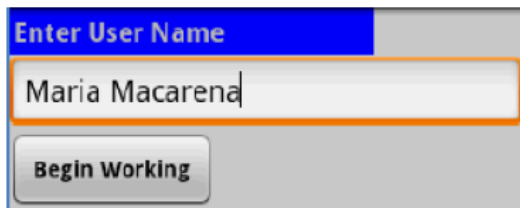
Trình bày cách thức sử dụng + ví dụ về:

1. Layout. (1)
2. Intent (1)
3. Fragment, TabSelection. (2)
4. TextView, EditText. (2)
5. Button, ImageButton. (3)
6. CheckBox, ToggleButton. (3)
7. RadioButton, RadioGroup. (4)
8. TimePicker, DatePicker. (4)
9. ScrollView, ImageView. (5)
10. ListView, Spinner. (5)
11. AutoCompleteTextView, Gallery. (6)
12. SlideDrawer, Menu. (6)

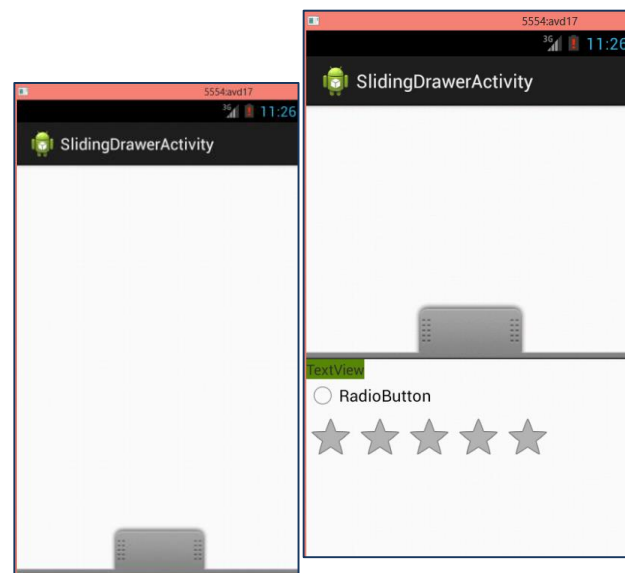
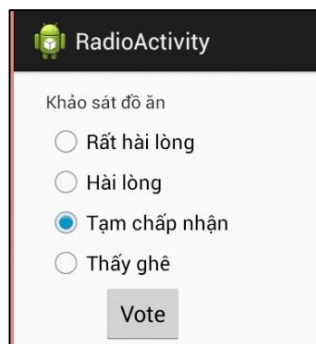
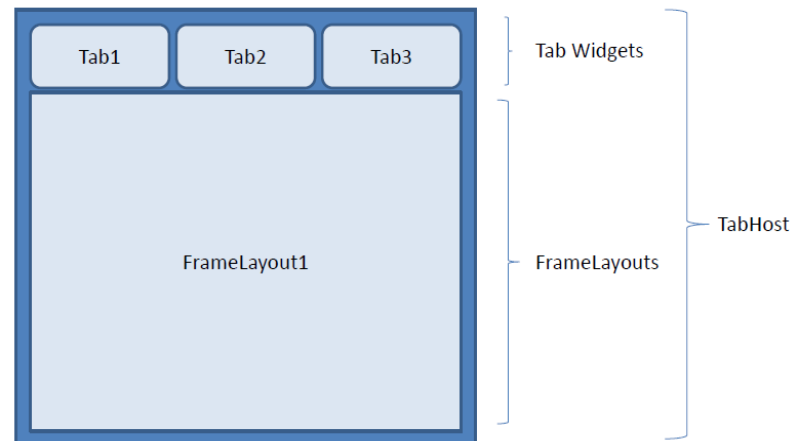


row one
row two
row three
row four





Components



BTVN

- BTVN nộp file mềm làm câu hỏi ôn tập chương (cá nhân)