http://ebooks-ict.blogspot.com

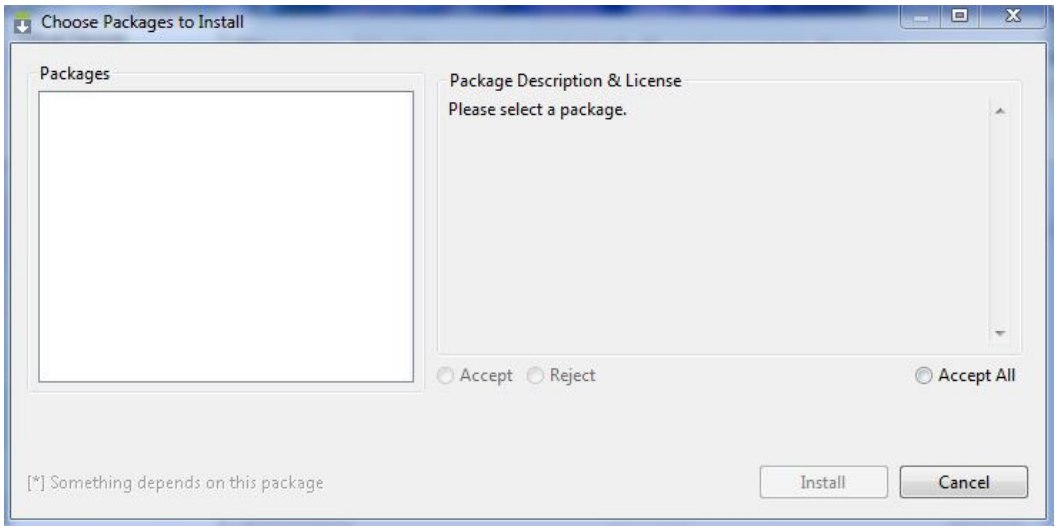
BÀI 1: HƯỚNG DẪN CÀI ĐẶT ANDROID

1.Download Android SDK:
Android SDK thực chất là tập hợp các công cụ và thư viện để phát triển các ứng dụng trên nền tảng hệ điều hành Android.

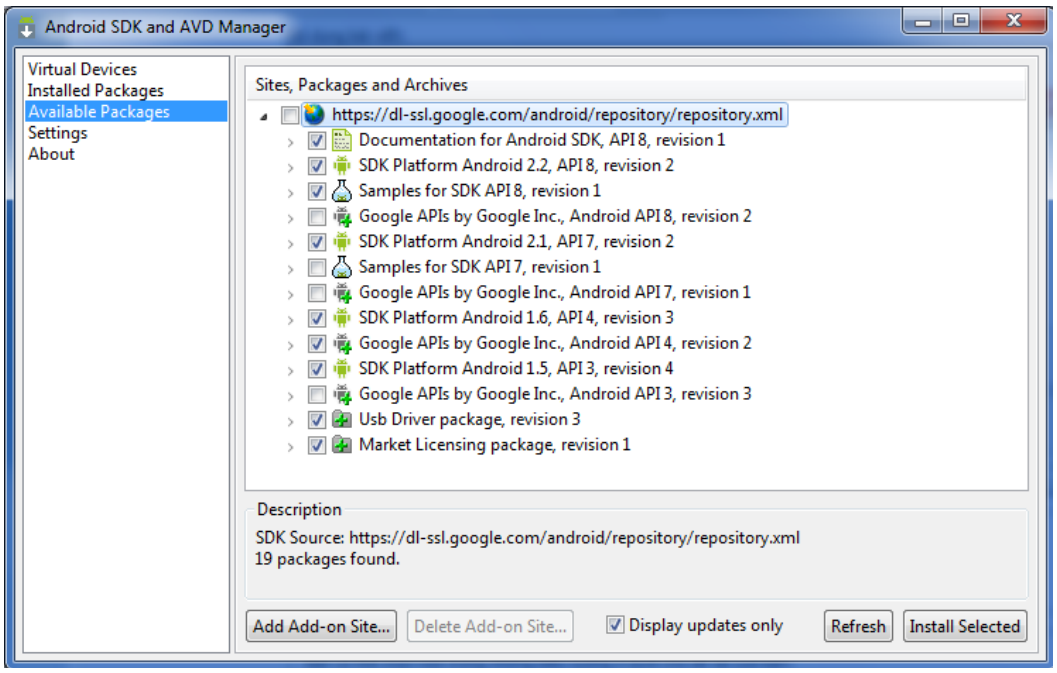
B1: Vào trang http://developer.android.com/sdk/index.html để tải Android SDK Starter. Tùy thuộc vào hệ điều hành mà bạn chọn bản Mac, Linux hay Window. Ở đây mình chọn tải bản cho Window.

Platform	Package	Size	MD5 Checksum
Windows	android-sdk_r06-windows.zip	23293160 bytes	7c7fcec3c6b5c7c3df6ae654b27effb5
Mac OS X (intel)	android-sdk_r06-mac_86.zip	19108077 bytes	c92abf66a82c7a3f2b8493ebe025dd22
Linux (i386)	android-sdk_r06-linux_86.tgz	16971139 bytes	848371e4bf068dbb582b709f4e56d903

B2: Giải nén file zip bạn vừa tải về. Chạy SDK Setup.exe. Bạn có thể gặp thông báo lỗi Fetching https://dl-sl>... Failed to fetch... Close thông báo này lại. Tiếp theo cửa sổ Choose Packages to Install xuất hiện. Nếu cửa sổ này trống rỗng -> Cancel.
-> Quay về cửa sổ Android SDK and AVD manager -> Chọn Setting, đánh dấu vào ô Force https://...
-> Chọn Available Packages



B3: Đánh dấu các Packages bạn muốn tải: Documents chính là phần Javadoc mô tả hoạt động của các phương thức và các lớp (phần này chắc chắn không thể thiếu rồi), Sample là các đoạn code mẫu, SDK Platform ứng với các phiên bản hệ điều hành (2.2 - API level 8, 2.1 - API level 7,...), và Google API để phát triển các phần mềm liên quan đến dịch vụ của Google (như Google Map nếu bạn muốn lập trình liên quan đến GPS).
Các bạn có thể tải hết nếu thích, còn muốn tối ưu thì có thể đánh dấu như mình (lưu ý USB drivers chỉ dành cho người sử dụng Windows và muốn phát triển ứng dụng test bằng điện thoại thật).
-> Install Selected
-> Install
-> Cửa sổ Install hiện ra
-> Ngồi chờ (>_<)



2.Tích hợp Android SDK vào Eclipse:

B1: Tải Eclipse nếu bạn chưa có. Mọi người có thể phân vân không biết tải bản nào cho phù hợp, nhưng theo ý kiến của mình thì có thể dùng 1 trong 2 bản sau: Eclipse for Java Developers, hoặc Eclipse for Java and Report Developers (mình dùng bản sau).

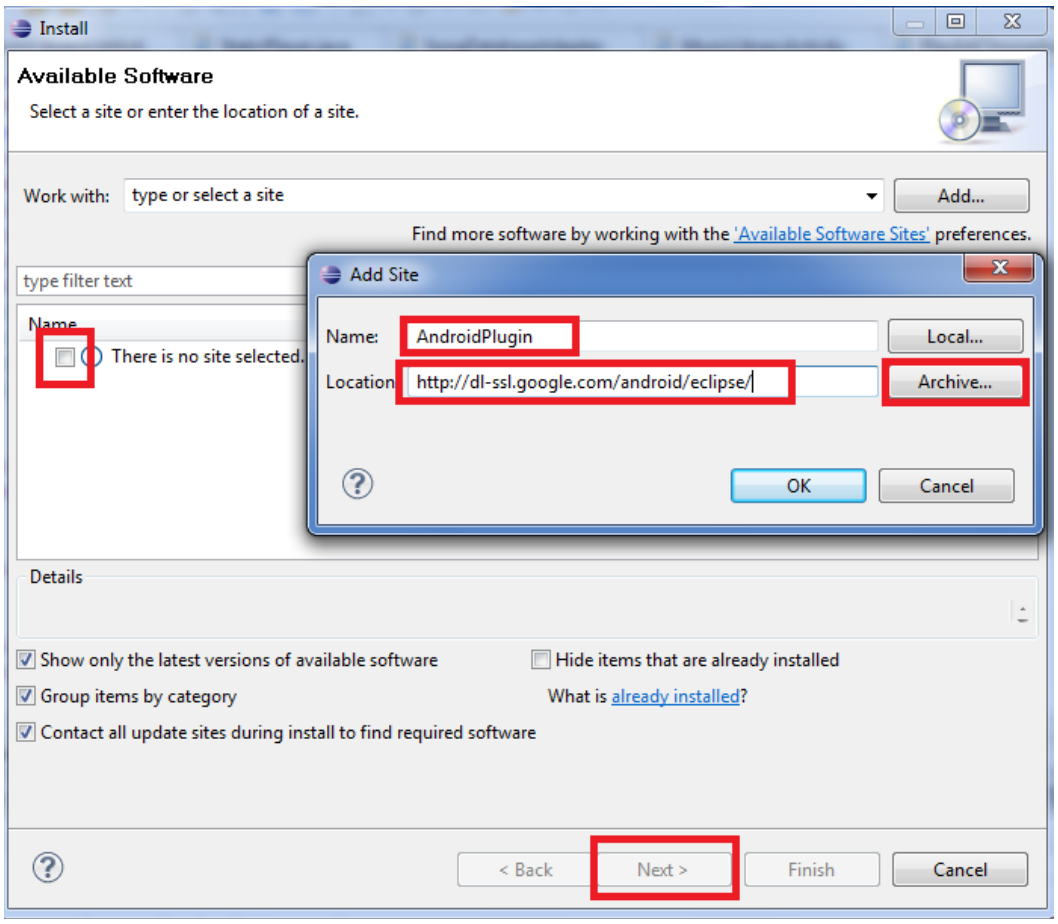
B2: Khởi chạy Eclipse, vào Help -> Install new softwares.
Chọn Add, gõ vào ô Name tên bạn muốn và Location gõ vào địa chỉ để tải về ADT:
HTML Code:

```
https://dl-ssl.google.com/android/eclipse/
```

hoặc
HTML Code:

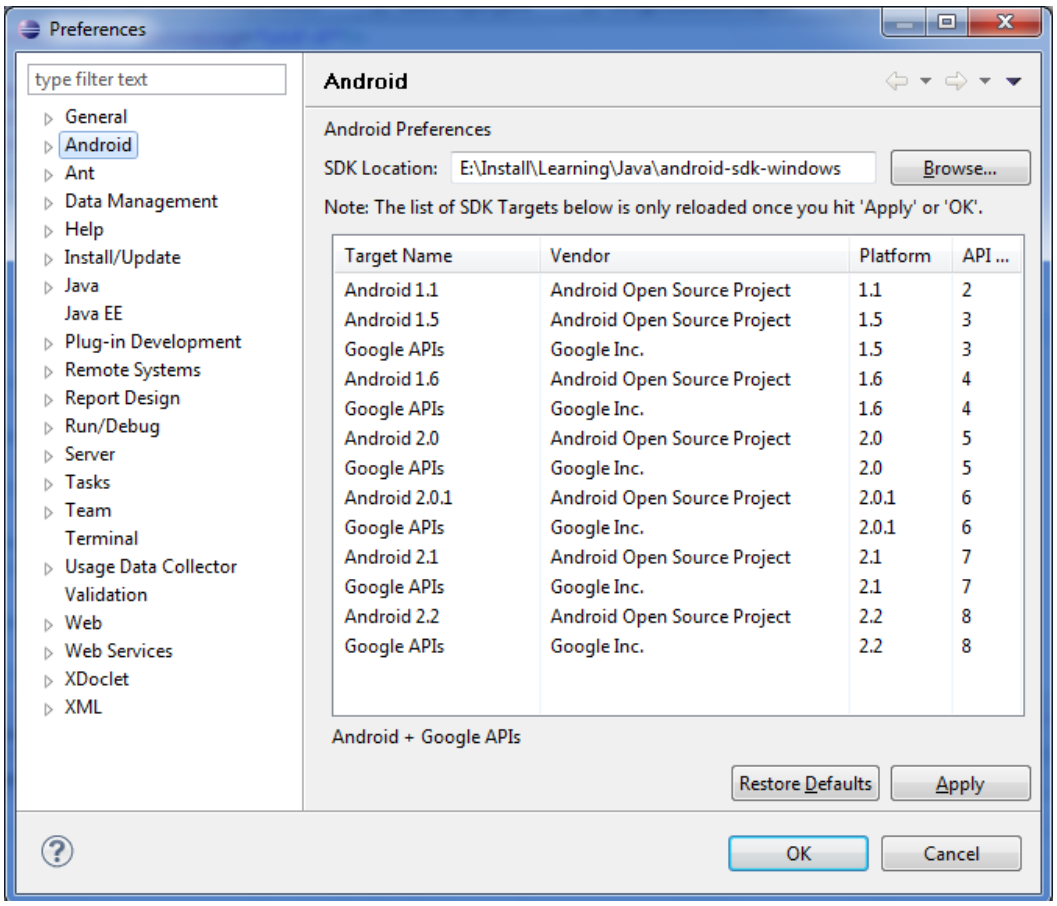
```
http://dl-ssl.google.com/android/eclipse/
```

nếu https không hoạt động. Ngoài ra bạn cũng có thể tải thẳng ADT về máy theo link http://dl.google.com/android/ADT-0.9.7.zip (bản mới nhất 0.9.7 ứng với Android 2.2), chọn Archive và browse tới file này (lưu ý không giải nén)
-> OK
-> Check vào phần dưới ô Name (sẽ hiện ra dòng Developer Tools).



B3: Next, next, Accept, next,...Finish (như Install mọi chương trình bình thường).

B4: Eclipse -> Windows -> Preferences -> Android
Nhấn nút Browse và chỉnh đường dẫn tới thư mục của Android SDK bạn tải lúc trước.
-> Apply
-> OK



3.Android Virtual Device (Emulator):

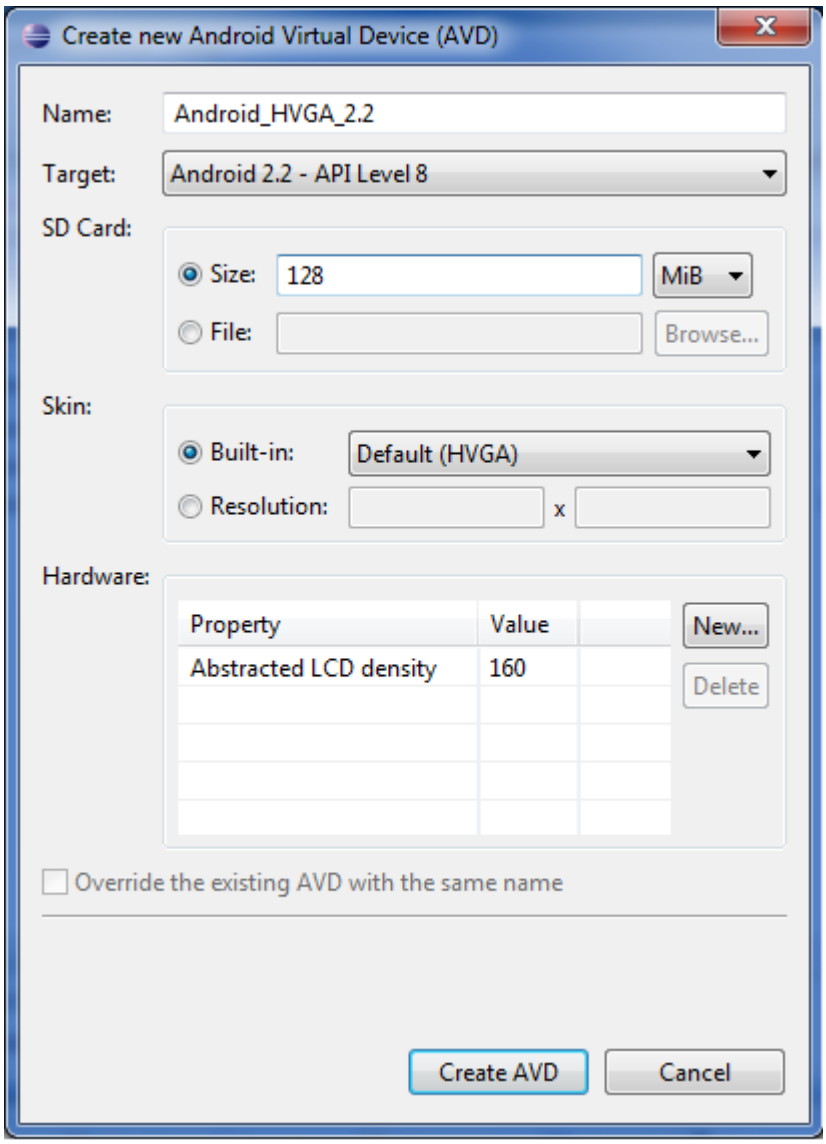
AVD là máy ảo cho developer phát triển ứng dụng. Để tạo 1 AVD bạn vào Windows -> Android SDK and AVD Manager -> Virtual Devices chọn New.
-> Cửa sổ Create new AVD hiện ra, bạn điền thông tin cho AVD bạn muốn:

Name: Tùy ý (nhưng chỉ được sử dụng các ký tự "a-z", "A-Z", ".-_", nghĩa là cả khoảng trắng cũng ko đc).

Target: Chọn phiên bản hệ điều hành bạn muốn (thường mình tạo một Android 1.6 và một Android 2.2 để test).

SD Card: gõ vào Size SD card ảo cho AVD, hoặc chỉnh tới file đã có sẵn. Nhiều AVD có thể dùng chung 1 Sdcard (chỉ cần tạo lần đầu, các lần sau chỉnh đường dẫn tới file đó).

Skin: có thể để Default (HVGA) hoặc chọn kích cỡ màn hình bạn muốn. Chỉ cần quan tâm tới 3 option: HVGA (phân giải 320-480 như G1, G2, i5700...), QVGA (240-320 như HTC Wildfire...), WVGA854 (480-854 như Milestone, NexusOne...)
-> Create AVD.


Create new Android Virtual Device (AVD)
Name:
Android_HVGA_2.2
Target:
Android 2.2 - API Level 8
SD Card:
Size:
128
MiB
File:
Browse...
Skin:
Built-in:
Default (HVGA)
Resolution:
x
Hardware:
Property
Value
New...
Abstracted LCD density
160
Delete
Override the existing AVD with the same name
Create AVD
Cancel

BÀI 2: HƯỚNG DẪN LẬP TRÌNH CƠ BẢN VỚI ANDROID

Yêu cầu kiến thức cho lập trình Android:
Để lập trình android, chỉ cần kiến thức java căn bản là hoàn toàn được. Căn bản ở đây có nghĩa là hiểu được thế nào là class, package, biết ý nghĩa của các từ khóa như public, private, protected,... thành thạo các lệnh cơ bản như if, for(), switch(), while(), ... biết sd các lệnh như Integer.parseInt() hay String.valueOf()... Nên có thêm kiến thức về gói java.util vì đây là gói hỗ trợ nhiều lớp rất mạnh được sử dụng trên mọi nền, ngoài ra các gói như java.io, java.net...

Các kiến thức về các gói lập trình cho desktop như java.awt, java.swing hoàn toàn không cần thiết (bản thân mình cũng chưa sd cái này bao giờ, nhảy vào học java là học J2ME luôn), hay các gói của J2ME cũng vậy. Lập trình Android tuy cũng là lập trình di động, nhưng các điện thoại sử dụng hđh Android có cấu hình rất mạnh (Nexus One có VXL lên tới 1Ghz), vì vậy 2 nền tảng Android và J2ME cũng rất khác nhau. Android có những gói riêng hỗ trợ lập trình cho nó và không yêu cầu khắt khe về việc tối ưu code như J2ME. Thật đáng tiếc vì J2ME mình học ko ứng dụng được mấy vào lập trình Android (tuy nhiên 1 số kỹ thuật cơ bản cho lập trình game 2D như Sprite, double buffering, Tile... thì vẫn ko hề phí phạm chút nào)

Cài đặt Android để lập trình:
Để lập trình Android thì mỗi bộ SDK của Google là không đủ, bạn còn cần tích hợp nó vào một IDE như Eclipse. Anh Giáp đã có 2 bài hướng dẫn rất chi tiết về cài đặt Android trong Eclipse cũng như Netbeans, nhưng theo mình mọi người nên sử dụng Eclipse hơn vì nó có nhiều tính năng hỗ trợ lập trình Google, còn Netbeans thì plugin cho Android vẫn chưa hoàn thiện
Eclipse
Netbeans
Tiện thể mình nói luôn, mình học Android theo 2 cuốn *Professional Android Application Development* và *Unlocking Android*. Cả 2 cuốn đều dành cho beginner nhưng cuốn đầu code nhiều, giải thích ít, cuốn thứ 2 giải thích rõ ràng hơn. Nếu có ai có ý định tham khảo thì nên đọc cuốn UA trước để hiểu rõ hơn Android, sử dụng cuốn PAAD trong việc tham khảo các đoạn code cho lập trình.
Understanding Android Application:
Việc hiểu được các thành phần (component) tạo nên một ứng dụng Android là rất cần thiết cho việc lập trình. Các thành phần này được chia làm 6 loại bao gồm:
1.Activity: hiểu một cách đơn giản thì Activity là nền của 1 ứng dụng. Khi khởi động 1 ứng dụng Android nào đó thì bao giờ cũng có 1 main Activity được gọi, hiển thị màn hình giao diện của ứng dụng cho phép người dùng tương tác.
2.Service: thành phần chạy ẩn trong Android. Service sử dụng để update dữ liệu, đưa ra các cảnh báo (Notification) và không bao giờ hiển thị cho người dùng thấy.
3.Content Provider: kho dữ liệu chia sẻ. Content Provider được sử dụng để quản lý và chia sẻ dữ liệu giữa các ứng dụng.
4.Intent: nền tảng để truyền tải các thông báo. Intent được sử dụng để gửi các thông báo đi nhằm khởi tạo 1 Activity hay Service để thực hiện công việc bạn mong muốn. VD: khi mở 1 trang web, bạn gửi 1 intent đi để tạo 1 activity mới hiển thị trang web đó.
5.Broadcast Receiver: thành phần thu nhận các Intent bên ngoài gửi tới. VD: bạn viết 1 chương trình thay thế cho phần gọi điện mặc định của Android, khi đó bạn cần 1 BR để nhận biết các Intent là các cuộc gọi tới.
6.Notification: đưa ra các cảnh báo mà không làm cho các Activity phải ngừng hoạt động.

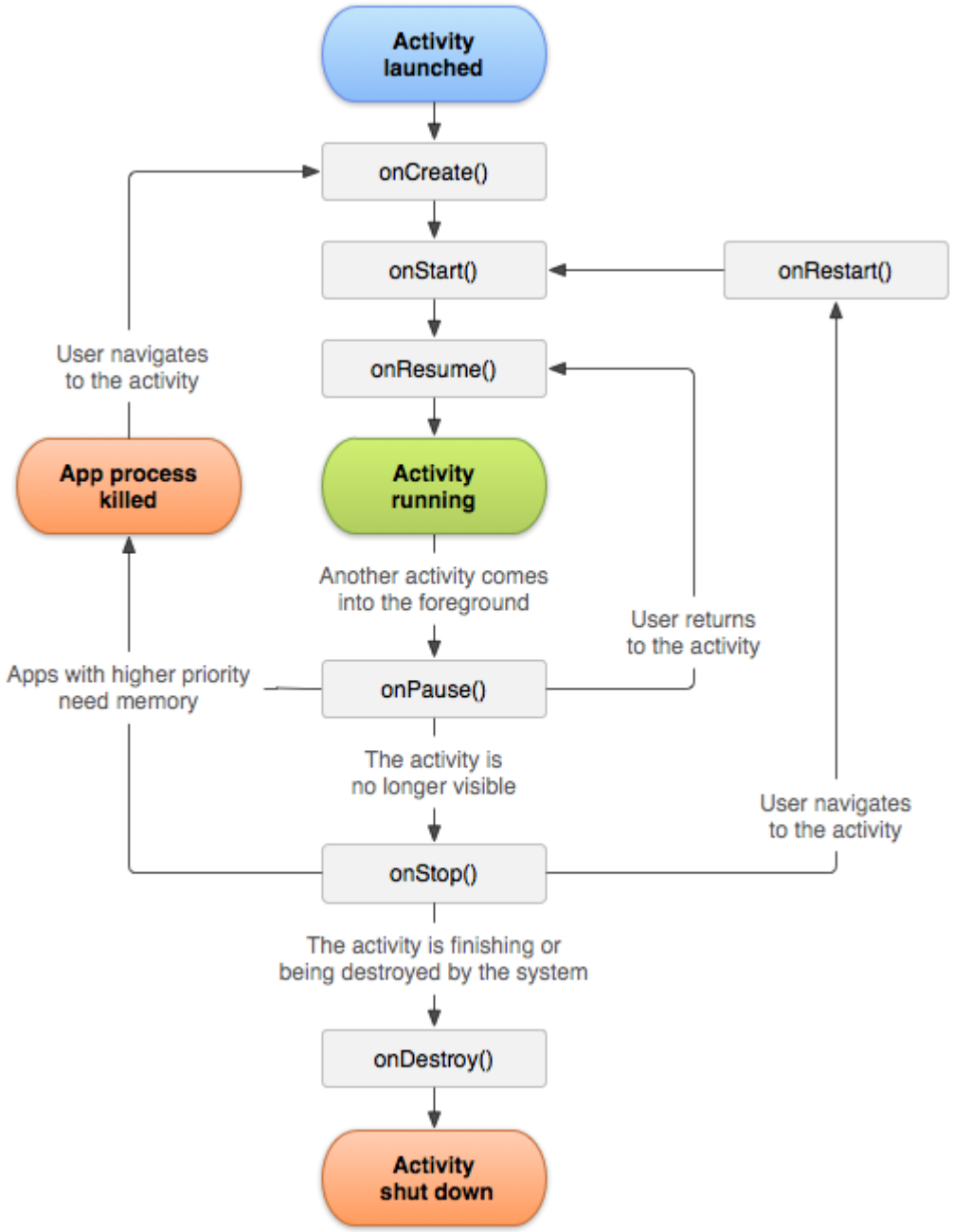
Activity, Service, Broadcast Receiver và Content Provider mới là những thành phần chính cấu thành nên ứng dụng Android, bắt buộc phải khai báo trong AndroidManifest (tham khảo bài 2 có giới thiệu đầy đủ về file này).

Understanding Android Application Life Cycle:
Android có cơ chế quản lý các process theo chế độ ưu tiên. Các process có priority thấp sẽ bị Android giải phóng mà không hề cảnh báo nhằm đảm bảo tài nguyên.
1.Foreground process: là process của ứng dụng hiện thời đang được người dùng tương tác.
2.Visible process: là process của ứng dụng mà activity đang hiển thị đối với người dùng (onPaused() của activity được gọi).
3.Service process: là Service đang running.
4.Background process: là process của ứng dụng mà các activity của nó ko hiển thị với người dùng (onStoped() của activity được gọi).
5.Empty process: process không có bất cứ 1 thành phần nào active.
Theo chế độ ưu tiên thì khi cần tài nguyên, Android sẽ tự động kill process, trước tiên là các empty process.

Android Activity Life Cycle:
Như mình đã giới thiệu ở trên , Actitvity là thành phần quan trọng nhất và đóng vai trò chính trong xây dựng ứng dụng Android. Hệ điều hành Android quản lý Activity theo dạng stack: khi một Activity mới được khởi tạo, nó sẽ được xếp lên đầu của stack và trở thành ***running activity***, các Activity trước đó sẽ bị tạm dừng và chỉ hoạt động trở lại khi Activity mới được giải phóng.

Activity bao gồm 4 state:
- ***active (running):*** Activity đang hiển thị trên màn hình (foreground).
- ***paused:*** Activity vẫn hiển thị (visible) nhưng không thể tương tác (lost focus). VD: một activity mới xuất hiện hiển thị giao diện đè lên trên activity cũ, nhưng giao diện này nhỏ hơn giao diện của activity cũ, do đó ta vẫn thấy được 1 phần giao diện của activity cũ nhưng lại không thể tương tác với nó.
- ***stop:*** Activity bị thay thế hoàn toàn bởi Activity mới sẽ tiến đến trạng thái ***stop***
- ***killed:*** Khi hệ thống bị thiếu bộ nhớ, nó sẽ giải phóng các tiến trình theo nguyên tắc ưu tiên. Các Activity ở trạng thái ***stop*** hoặc ***paused***cũng có thể bị giải phóng và khi nó được hiển thị lại thì các Activity này phải khởi động lại hoàn toàn và phục hồi lại trạng thái trước đó.

Biểu đồ miêu tả Activity state



Vòng đời của Activity:
- ***Entire lifetime:*** Từ phương thức **onCreate( )** cho tới **onDestroy( )**
- ***Visible liftetime:*** Từ phương thức **onStart( )** cho tới **onStop( )**
- ***Foreground lifetime:*** Từ phương thức **onResume( )** cho tới **onPause( )**

Khi xây dựng Actitvity cho ứng dụng cần phải viết lại phương thức **onCreate( )** để thực hiện quá trình khởi tạo. Các phương thức khác có cần viết lại hay không tùy vào yêu cầu lập trình.

XML trong Android:

Không giống như lập trình java thông thường, lập trình android ngoài các lớp được viết trong *.java còn sử dụng XML để thiết kế giao diện cho ứng dụng. Tất nhiên bạn hoàn toàn có thể thiết kế 1 giao diện như ý muốn mà không cần tới bất cứ 1 dòng XML nào, nhưng sd XML sẽ đơn giản công việc đi rất nhiều. Đồng thời sd XML sẽ giúp việc chỉnh sửa ứng dụng sau này trở nên dễ dàng.

Về nguyên tắc, khi lập trình ứng dụng ta thiết kế giao diện bằng XML và cài đặt các xử lý khi tương tác với

giao diện trong code.

1 số thành phần cơ bản trong Android:

1.Các layout:

Layout được dùng để quản lý các thành phần giao diện khác theo 1 trật tự nhất định.
- FrameLayout: Layout đơn giản nhất, thêm các thành phần con vào góc trên bên trái của màn hình.
- LinearLayout: thêm các thành phần con theo 1 chiều nhất định (ngang hoặc dọc). Đây là layout được sử dụng nhiều nhất.
- RelativeLayout: thêm các thành phần con dựa trên mối quan hệ với các thành phần khác hoặc với biên của layout.
- TableLayout: thêm các thành phần con dựa trên 1 lưới các ô ngang và dọc.
- AbsoluteLayout: thêm các thành phần con dựa theo tọa độ x, y.

Layout được sử dụng nhằm mục đích thiết kế giao diện cho nhiều độ phân giải. Thường khi lập trình nên kết hợp nhiều layout với nhau để tạo ra giao diện bạn mong muốn.

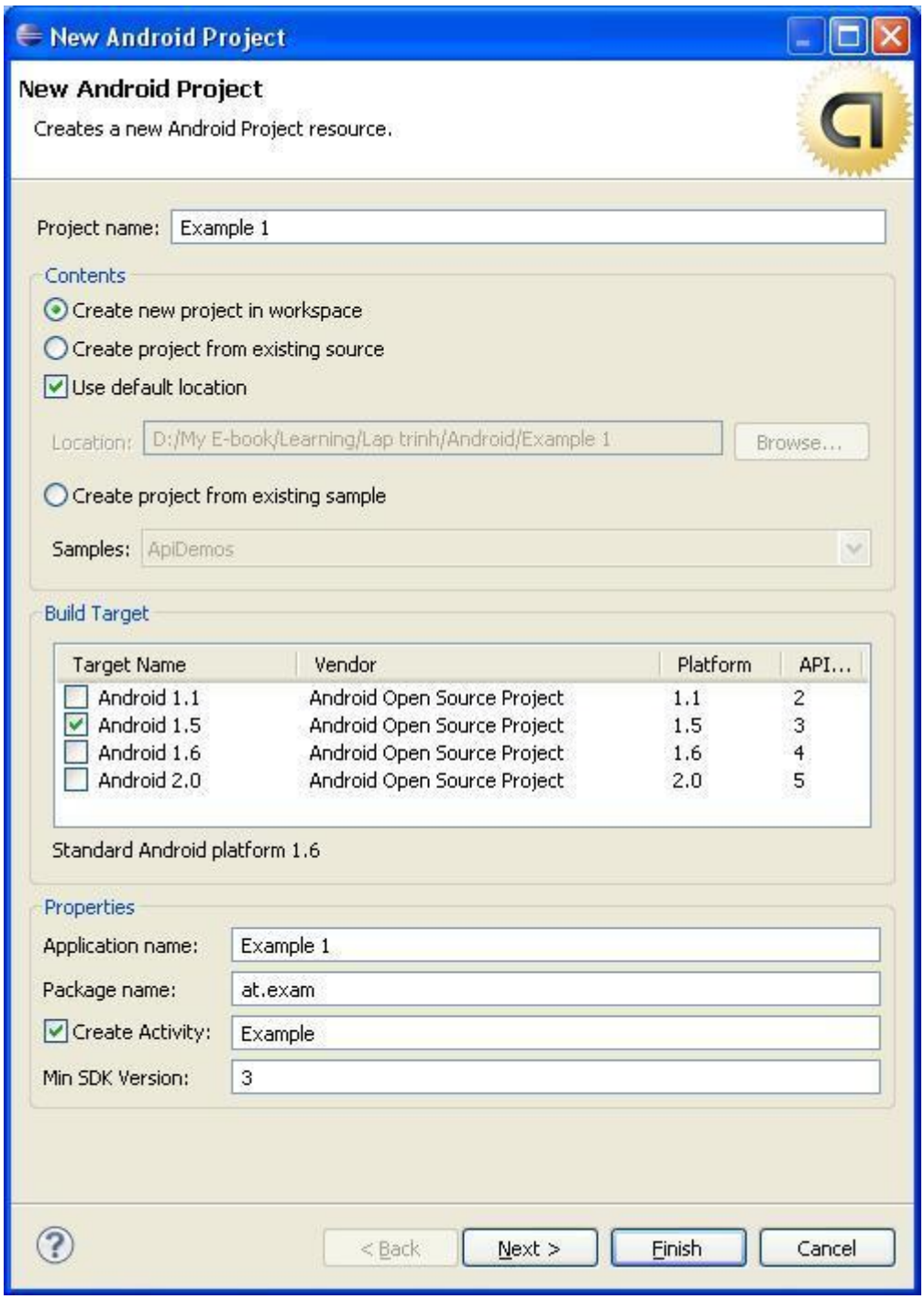
2.XML unit:

Để hiểu được các thành phần cơ bản của XML cũng như việc sử dụng XML kết hợp với code, ta sẽ đi xây dựng thử một chương trình đơn giản.

Yêu cầu: Xây dựng 1 ứng dụng cho phép gõ 1 nội dung vào rồi hiển thị ra nội dung đó ở bên dưới.

B1: Khởi tạo 1 project (ở đây sử dụng Eclipse để minh họa).
Vào thẻ File -> New -> Android Project. Nếu bạn mới lập trình Android lần đầu thì có lẽ dòng Android Project sẽ không hiện ra, khi đó xuống phía cuối chọn Other rồi vào Android -> Android Project.

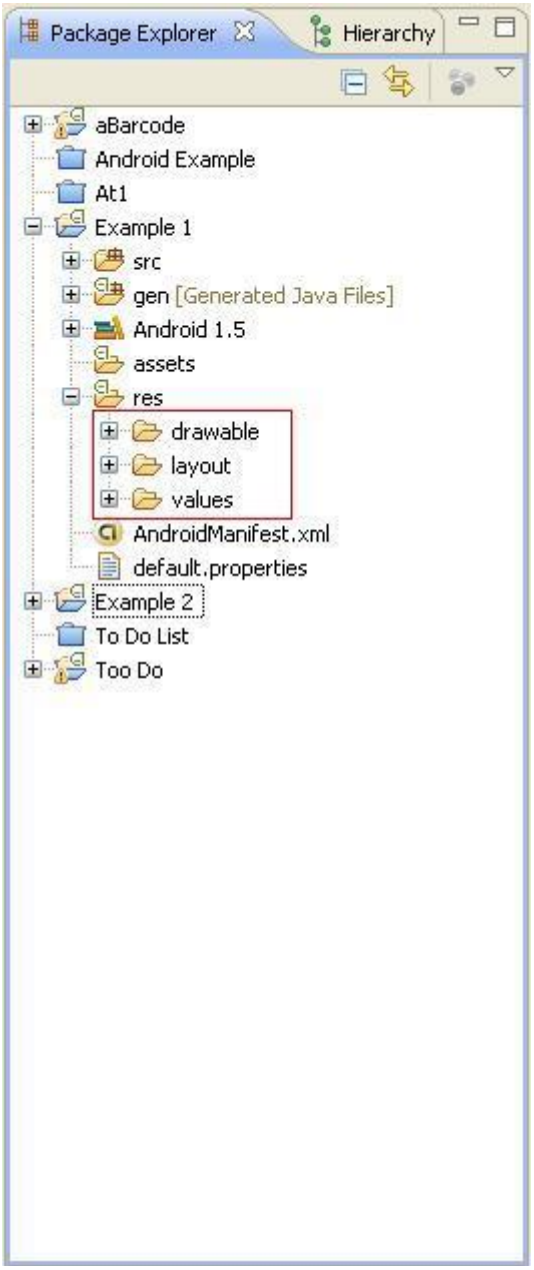
B2: Điền thông tin cho project



Project name: Example 1

Build Target: Chọn Android 1.5 (mới nhất là 2.1 nhưng hiện tại bạn chưa cần quan tâm)
Application name: Example 1
Package name: at.exam
Create Activity: Example
=> Kích nút Finish.

B3: Bên khung Package Explore bên trái đi tới thư mục res, bạn sẽ thấy có 3 thư mục con:



- drawable: thư mục chứa các hình ảnh để làm icon hoặc tài nguyên cho giao diện...
- layout: chứa các file xml để thiết kế giao diện.
- values: chứa các giá trị sử dụng trong ứng dụng được bạn định nghĩa, như các dòng ký tự (string), các màu (color), các themes...

B4:Vào thư mục layout, chọn file main.xml và gõ đoạn code sau vào thay cho toàn bộ nội dung có sẵn (Eclipse hỗ trợ kéo thả cho xml nhưng theo mình không nên sử dụng):

?

code

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7     <EditText
8         android:id="@+id/edit_text"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
```

```
9           android:hint="@string/edit_hint"
10      />
11      <TextView
12          android:id="@+id/text_view"
13          android:layout_width="fill_parent"
14          android:layout_height="wrap_content"
15          android:textColor="@color/text_color"
16          android:textSize="28px"
17          android:typeface="monospace"
18      />
19  </LinearLayout>
20
21
```

Trong đoạn XML này chúng ta khai báo một Linear Layout với 2 thành phần con của nó là 1 Edit Text (dùng để gõ xâu ký tự) với 1 Text View (hiển thị xâu ký tự). Linear Layout được khai báo với từ khóa orientation nhằm chỉ ra chiều sắp xếp của 2 thành phần con là chiều dọc. Còn với layout_width, layout_height các bạn có thể cho giá trị bằng "fill_parent" hoặc "wrap_content" để thông báo thành phần này sẽ có chiều rộng (dài) phủ đầy thành phần cha hoặc chỉ vừa bao đủ nội dung.
Trong Edit Text và Text View các bạn có thể thấy có từ khóa id, từ khóa này cho phép khai báo id của các thành phần để lấy về trong code (sẽ đề cập sau).
Ngoài ra từ khóa hint trong Edit Text cho phép hiện ra phần nội dung mờ khi Edit Text vẫn chưa có ký tự nào. "@string/edit_hint" thông báo lấy trong file strings.xml xâu có tên là edit_hint.
Còn textColor của Text View thì thông báo đoạn ký tự sẽ được hiển thị với màu lấy trong file colors.xml, textSize chỉ ra cỡ chữ bằng 28 pixel và typeface chỉ ra kiểu chữ là monospace

B5:Vẫn trong thư mục res, vào values và chọn file strings.xml. Bố sung thêm dòng định nghĩa cho edit_hint như sau:

?

code

```
1<?xml version="1.0" encoding="utf-8"?>
2<resources>
3    <string name="hello">Hello World, Example!</string>
4    <string name="app_name">Example 1</string>
5    <string name="edit_hint">Enter the work here</string>
6</resources>
```

B6:Trong thư mục values, tạo file colors.xml (chuột phải vào thư mục, chọn New -> Android XML File, và lưu ý chữ s, không phải là color.xml). Gõ nội dung cho file như sau:

?

code

```
1<?xml version="1.0" encoding="utf-8"?>
2<resources>
3    <color name="text_color">#ff3300</color>
4</resources>
```

OK, vậy là bạn đã tạo một màu mới cho dòng chữ sẽ được hiển thị trong Text View (ff3300 là mã hexa của màu đỏ). Thực chất bạn hoàn toàn có thể gõ thẳng

?

code

```
1android:textColor="#ff3300"
```

trong file main.xml mà không cần tạo mới file colors.xml, nhưng mục đích của XML trong Android chính là để hỗ trợ nâng cấp chỉnh sửa dễ dàng. Nếu sau này bạn muốn sửa màu của dòng text thì chỉ cần vào

colors.xml thay đổi thay vì mò mẫm trong main.xml (có thể rất dài nếu giao diện phức tạp).

Các thành phần trên mới chỉ là các phần cơ bản của XML. Ngoài ra các bạn có thể khai báo thêm về Animation, Style và Theme (phức tạp hơn nhiều nên mình không giới thiệu trong phần cơ bản này).
BƯỚC 7: Vậy là chúng ta đã hoàn thiện phần giao diện với XML, giờ đến viết code để xử lý các sự kiện cho các thành phần:
=> vào thư mục src (source code của project) => at.exam => Example.java, gõ nội dung code sau vào:

```
1  package at.exam;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.view.KeyEvent;
6  import android.view.View;
7  import android.view.View.OnKeyListener;
8  import android.widget.EditText;
9  import android.widget.TextView;
10
11 public class Example extends Activity {
12     /** Called when the activity is first
   created. */
13     @Override
14     public void onCreate(Bundle
   savedInstanceState) {
15         super.onCreate(savedInstanceState
   );
16
17         //Thiết lập giao diện lấy từ file
   main.xml
18         setContentView(R.layout.main);
19
20         //Lấy về các thành phần trong
   main.xml thông qua id
21         final EditText edit = (EditText)
   findViewById(R.id.edit_text);
22
23
```

```
        final TextView text = (TextView)
findViewById(R.id.text_view);



        //Thiết lập xử lý cho sự kiện
nhấn nút giữa của điện thoại

        edit.setOnKeyListener(new OnKeyLis
tener() {

            @Override

            public boolean onKey(View
v, int keyCode, KeyEvent event) {

                if (event.getAction() ==
KeyEvent.ACTION_DOWN

                        && keyCode ==
KeyEvent.KEYCODE_DPAD_CENTER) {

                    text.setText(edit.get
Text().toString());

                    edit.setText("");

                    return true;


                }

                else {

                    return false;

                }

            }



        });

    }

}
```

Dạo qua một chút kiến thức cơ bản: Trong Android, các lớp sử dụng để tạo giao diện (Edit Text, Text View...) đều là lớp con của lớp View. Một số lớp thường xuyên được sử dụng để tạo giao diện:

- TextView
- EditText
- ListView
- Spinner
- CheckBox

- Button
- RadioButton

Ngoài ra bạn còn có thể tạo 1 View riêng của mình bằng cách kế thừa View có sẵn.

Các Listener được sử dụng để bắt 1 sự kiện nào đó. Ở đây mình sử dụng OnKeyListener dùng để bắt sự kiện khi nhấn 1 phím của điện thoại. Ngoài ra thường sử dụng OnClickListener để bắt sự kiện chạm vào 1 View đang hiển thị trên màn hình. Mỗi View đều phải set Listener riêng để xử lý cho sự kiện tương tác với nó, và mỗi loại View cũng lại có những Listener dành riêng cho nó (VD: CheckBox có OnCheckChangedListener)
Ở đây mình sử dụng hàm dạng inner để định nghĩa xử lý cho OnKeyListener nên có thể mọi người không quen lắm, nhưng nó cũng nằm trong phần cơ bản của Java đấy nhé.

Đề nghị lưu ý thêm phần R.id.edit_text. Để lấy hoặc truy nhập các thành phần ta đã định nghĩa trong XML ta phải sử dụng R.* như R.layout.main, R.id.edit_text. Lệnh findViewById sẽ trả về 1 View có Id thiết lập trong phần XML. Do View là lớp cha của EditText với TextView nên ở đây ta phải ép kiểu.
Ngoài ra các string hay color cũng có thể lấy về bằng lệnh getResource() . Vd: getResource().getColor(R.color.text_color)

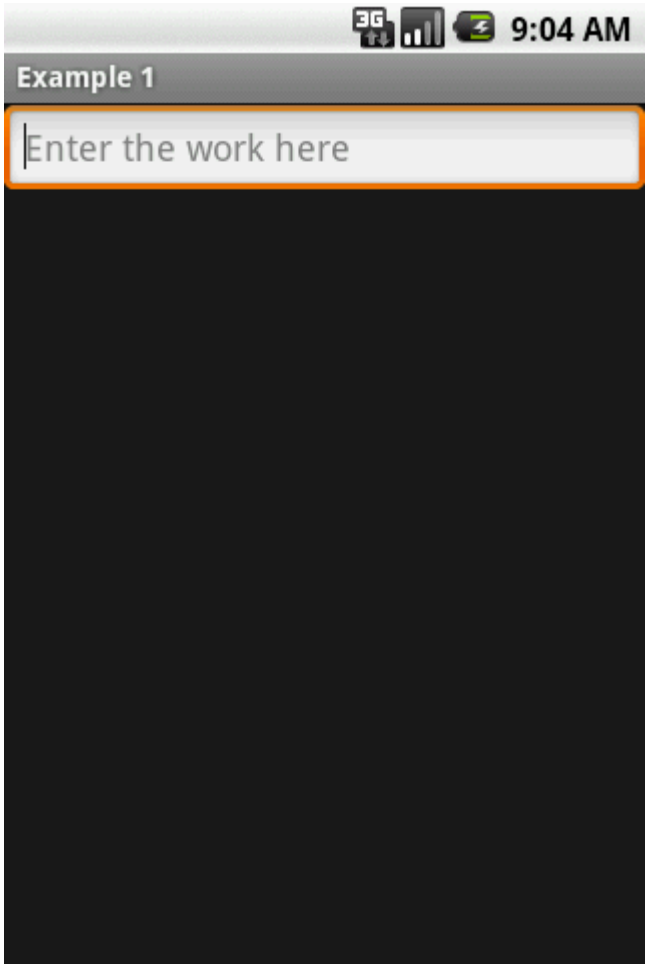
B8: Chạy chương trình. Chọn Run => Android Application và chờ cho emulator khởi động nhé. Ai có 1 Android thật có thể kết nối qua USB và thử nghiệm luôn. Tự chỉnh sửa trong code và trong XML để hiểu thêm về lập trình Android.
VD:
?
code

```
1 edit.setOnClickListener(new OnClickListener() {
2                         @Override
3             public void onClick(View v) {
4                 // TODO Auto-generated method stub
5
6             }
7
8         });
```

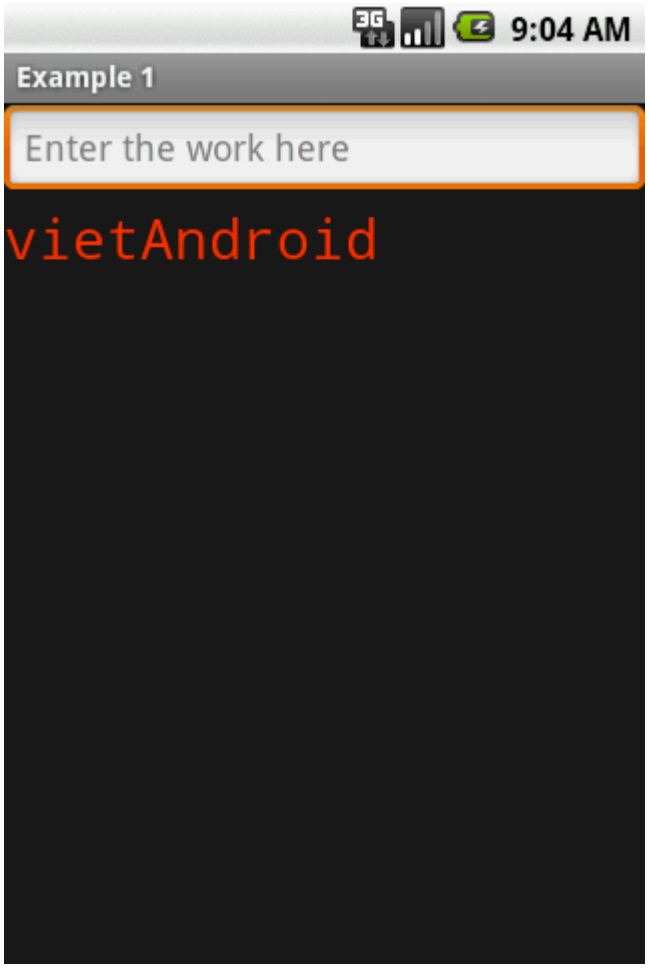
hoặc trong XML thêm vào phần Text View
?
code

```
1 android:textSize="50px"
```

để xem chương trình thay đổi như thế nào nhé ^_^


9:04 AM
Example 1
Enter the work here


9:04 AM
Example 1
vietAndroid


9:04 AM
Example 1
Enter the work here
vietAndroid

BÀI 3: HƯỚNG DẪN LẬP TRÌNH CƠ BẢN VỚI ANDROID

Trong bài trước đã giới thiệu sơ lược về các thành phần cơ bản của Android cũng như việc sử dụng XML để lập trình ứng dụng Android. Trong bài này sẽ giới thiệu thêm về Android Manifest và đi sâu hơn về vấn đề làm việc với View.

Android Manifest
Trong khung Package Explorer, ở phía dưới thư mục res, bạn sẽ thấy 1 file có tên là AndroidManifest.xml. Mỗi ứng dụng đều cần có AndroidManifest.xml để mô tả những thông tin quan trọng của nó cho hệ thống Android biết. Let's look closer:

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
03       package="at.exam"
04       android:versionCode="1"
05       android:versionName="1.0">
06     <application
   android:icon="@drawable/icon"android:label="@string/app_name">
07         <activity android:name=".Example"
08                   android:label="@string/app_name">
09             <intent-filter>
10                 <action android:name="android.intent.action.MAIN" />
11                 <category android:name="android.intent.category.LAUNCHER" />
12             </intent-filter>
13         </activity>
14     </application>
15     <uses-sdk android:minSdkVersion="3" />
16 </manifest>
```

Cụ thể những công việc mà AndroidManifest.xml thực hiện:
- Đặt tên cho Java package của ứng dụng.
- Mô tả các thành phần (component) của ứng dụng: activity, service, broadcast receiver hoặc content provider.
- Thông báo những permission mà ứng dụng cần có để truy nhập các protected API và tương tác với các ứng dụng khác.
- Thông báo những permission mà các ứng dụng khác cần có để tương tác với ứng dụng hiện thời.
- Thông báo level thấp nhất của Android API mà ứng dụng cần để chạy. (Android 1.0 là level 1, 1.1 là level 2, 1.5 level 3, 1.6 level 4 và 2.0 là level 5).

...
Hãy xem thử file AndroidManifest.xml của chương trình TooDo mình đang xây dựng:
Mã:

```
http://vietandroid.com/images/gradients/content3image-alpha.png); background-color:
rgb(240, 242, 250); color: rgb(51, 51, 51); overflow: auto; direction: ltr; max-width:
530px !important; background-repeat: repeat no-repeat;"><?xml version="1.0"
encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="android.at"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity
                android:name=".TooDo"
```

```
                android:screenOrientation="landscape"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
                android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>         <activity android:name=".WorkEnter">
        </activity>
        <receiver android:name=".AlarmReceiver">         </receiver>
    </application>
    <uses-sdk android:minSdkVersion="3" />
    <uses-permission android:name="android.permission.VIBRATE"/>
</manifest>
```

Main Activity của chương trình Too Do này là activity TooDo. Ngoài ra mình còn có 1 Activity khác có tên là WorkEnter để cho phép nhập vào thời gian và nội dung công việc. 1 Broadcast Receiver có tên là AlarmReceiver để nhận alarm gửi tới trong intent. Khi alarm được nhận sẽ có âm thanh và rung (vibration). Tất cả công việc sẽ được viết trong code, nhưng bắt buộc bạn phải khai báo các thành phần có trong ứng dụng vào AndroidManifest nếu muốn chương trình hoạt động. Tương tự, set permission để truy nhập camera, internet, đọc contact... cũng đều phải khai báo trong AM. Từ khóa screenOrientation cho phép thiết lập giao diện khi vào ứng dụng theo chiều dọc (portrait - mặc định) hay ngang (landscape), theme cho phép sử dụng style có sẵn của android là full-screen (ko có thanh status bar nữa).
Intent filter là bộ lọc dùng để giới hạn các intent được sử dụng trong activity hay receiver...
Mã:

```
http://vietandroid.com/images/gradients/content3image-alpha.png); background-color:
rgb(240, 242, 250); color: rgb(51, 51, 51); overflow: auto; direction: ltr; max-width:
530px !important; background-repeat: repeat no-repeat;"><intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="http" android:host="www.google.com"
android:path="/m/products/scan"/>
      </intent-filter>
```

Bộ lọc trên chỉ cho phép intent mở internet với đường dẫn định nghĩa sẵn (http://www.google.com/m/products/scan)

Working with View

Trong bài 1 mình đã giới thiệu qua cách sử dụng Edit Text và Text View. Thực chất các View còn lại cũng có cách sử dụng tương tự, bạn sẽ kết hợp nhiều View khác nhau để cho ra giao diện mình mong muốn. Ở đây mình sẽ đề cập nhiều tới List View (theo ý kiến mình là View khó sử dụng nhất).

Yêu cầu: Xây dựng một chương trình cho phép nhập nội dung công việc và thời gian rồi list ra

B1: Vẫn bắt đầu bằng cách khởi tạo một Project mới: File -> New -> Android Project.

Project name: Example 2

Build Target: Chọn Android 1.5

Application name: Example 2

Package name: at.exam

Create Activity: Example

=> Kích nút Finish.

B2: Đi tới res/main.xml để xây dựng giao diện cho chương trình:

Mã:

```
http://vietandroid.com/images/gradients/content3image-alpha.png); background-color:
rgb(240, 242, 250); color: rgb(51, 51, 51); overflow: auto; direction: ltr; max-width:
530px !important; background-repeat: repeat no-repeat;"><?xml version="1.0"
encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <EditText
        android:id="@+id/work_enter"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/work_hint"
        android:lines="1"
        android:textSize="24px"
```

```
    />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        >
        <TextView
            android:layout_width="50px"
            android:layout_height="wrap_content"
            android:text="@string/hour_edit"
            android:typeface="normal"
            android:textSize="15px"
            android:textStyle="bold"
            android:padding="5px"
        />
        <EditText
            android:id="@+id/hour_edit"
            android:layout_width="45px"
            android:layout_height="wrap_content"
            android:hint="12"
            android:textColorHint="@color/hint_color"
            android:textSize="20px"
            android:gravity="center"
            android:padding="5px"
            android:numeric="integer"
            android:maxLength="2"
        />
        <TextView
            android:layout_width="65px"
            android:layout_height="wrap_content"
            android:text="@string/minute_edit"
            android:typeface="normal"
            android:textSize="15px"
            android:textStyle="bold"
            android:padding="5px"
        />
        <EditText
            android:id="@+id/minute_edit"
            android:layout_width="45px"
            android:layout_height="wrap_content"
            android:hint="00"
            android:textColorHint="@color/hint_color"
            android:textSize="20px"
            android:gravity="center"
            android:padding="5px"
            android:numeric="integer"
            android:maxLength="2"
        />
    </LinearLayout>
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/button_content"
        />
    <ListView
        android:id="@+id/list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

Giao diện ta thiết kế ở đây có 1 Linear Layout làm thành phần chính, các thành phần con của nó gồm 1 Edit Text (dùng để nhập nội dung công việc), 1 Linear Layout (lại gồm các thành phần con để nhập giờ và phút thực hiện công việc), 1 Button (để thêm nội dung công việc vào List View) và 1 List View dùng để list các công việc bạn đã nhập.
Từ khóa lines được dùng để cố định số dòng và nên sử dụng với Edit Text thay vì dùng mỗi wrap_content vì nếu sd wrap_content thì Edit Text sẽ tự giãn ra nếu dòng nhập vào vượt giới hạn đường bao (làm hỏng giao diện bạn thiết kế).

Từ khóa gravity thông báo các thành phần con sẽ được sắp xếp ntn ở thành phần cha. Ở đây mình dùng "center" nghĩa là thành phần con nằm ở trung tâm. Hãy thử thêm vào 1 Edit Text:
Mã:

```
http://vietandroid.com/images/gradients/content3image-alpha.png); background-color: rgb(240, 242, 250); color: rgb(51, 51, 51); overflow: auto; direction: ltr; max-width: 530px !important; background-repeat: repeat no-repeat;">android:gravity="center"
```

Bạn sẽ thấy dòng chữ nhập vào sẽ bắt đầu từ giữa của Edit Text chứ không bắt đầu từ bên trái như trước nữa.
Từ khóa padding dùng để cách 1 khoảng cách cho thành phần. Nếu không có padding thì 2 thành phần con thuộc cùng 1 LinearLayout sẽ được xếp sát nhau, nhưng nếu 1 thành phần con sử dụng padding thì sẽ tạo được khoảng cách với thành phần còn lại theo mong muốn. Ngoài ra còn có paddingLeft, paddingRight, paddingTop, paddingBottom.
Từ khóa numeric dùng để giới hạn dạng ký tự nhập vào. Ở đây mình muốn chỉ nhập vào chữ số nên dùng "integer"
Từ khóa maxLength dùng để giới hạn số ký tự nhập vào. Do Edit Text này dùng để nhập giờ nên maxLength="2".
Ok, giờ đến 1 chút kiến thức về các đơn vị của dimenson:
- px (pixel): điểm chấm trên màn hình.
- in (inch)
- mm (milimet)
- pt (point) = 1/72 m
- dp (density - independent pixel): cái này hơi khó giải thích. Nói chung dp được sử dụng cho nhiều độ phân giải, và với độ phân giải 160 px/inch thì 1 dp = 1 px.
- sp: gần giống dp, nên sử dụng cho text size.
Nói chung nên sử dụng dp và sp để định nghĩa size cho các thành phần, vì nó có tỉ lệ cố định với độ phân giải của màn hình. Còn nếu bạn chủ tâm xây dựng cho 1 độ phân giải nhất định thì dùng px cho chính xác và chắc chắn.
B3: Tới values/strings.xml chỉnh sửa như sau:
?

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Example 2</string>
    <string name="work_hint">Enter the work here</string>
    <string name="hour_edit">Hour</string>
    <string name="minute_edit">Minute</string>
    <string name="button_content">Add work</string>
</resources>
```

B4: Tạo mới colors.xml trong values với nội dung:
?

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="hint_color">#cccccc</color>
</resources>
```

OK, vậy là đã hoàn thiện phần giao diện. Các bạn có thể cho chạy thử ngay để kiểm tra xem giao diện đã như ý muốn chưa chứ không cần đợi hoàn thành cả code (Run as -> Android Application).

B5: Time to coding. Tới thư mục src/Example.java và thay đổi nội dung file như sau:
?

```
1  package at.exam;
2
3  import java.util.ArrayList;
4
5  import android.app.Activity;
6  import android.app.AlertDialog;
7  import android.content.DialogInterface;
8  import android.os.Bundle;
9  import android.view.View;
10 import android.view.View.OnClickListener;
11 import android.widget.ArrayAdapter;
12 import android.widget.Button;
13 import android.widget.EditText;
14 import android.widget.ListView;
15
16 public class Example extends Activity {
17     /** Called when the activity is first created. */
18     @Override
19     public void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.main);
22
23         //Tạo mảng để chứa String nội dung công việc và giờ
24         final ArrayList<String> arrayWork = new ArrayList<String>();
25         //Adapter dùng để kết nối mảng với List View
26         final ArrayAdapter<String> arrayAdapter
   = new ArrayAdapter<String>(this,
27                 android.R.layout.simple_list_item_1, arrayWork);
```

```
28
29          //Các EditText để vào nội dung công việc được lấy về từ XML
30          final EditText workEnter = (EditText) findViewById(R.id.work_enter);
31          final EditText hourEdit = (EditText) findViewById(R.id.hour_edit);
32          final EditText minuteEdit = (EditText) findViewById(R.id.minute_edit);
33
34          //Button khi nhấn sẽ thêm công việc vào ListView
35          final Button button = (Button) findViewById(R.id.button);
36
37          //ListView chứa danh sách công việc
38          final ListView list = (ListView) findViewById(R.id.list);
39          //Cần set Adapter cho list để biết sẽ lấy nội dung từ mảng arrayWork
40          list.setAdapter(arrayAdapter);
41
42          //Định nghĩa Listener xử lý sự kiện nhấn vào button
43          OnClickListener add = new OnClickListener() {
44              @Override
45              public void onClick(View v) {
46                  //Nếu 1 trong 3 Edit Text không có nội dung thì hiện lên cảnh
báo
47                  if (workEnter.getText().toString().equals("") ||
48                          hourEdit.getText().toString().equals("") ||
49                          minuteEdit.getText().toString().equals(""))
{
50                      AlertDialog.Builder builder
= newAlertDialog.Builder(Example.this);
51                      builder.setTitle("Info missing");
52                      builder.setMessage("Please enter all information of the
work");
53                      builder.setPositiveButton("Continue", newDialogInterface.O
54nClickListener() {
                            public void onClick(DialogInterface dialog, int which)
```

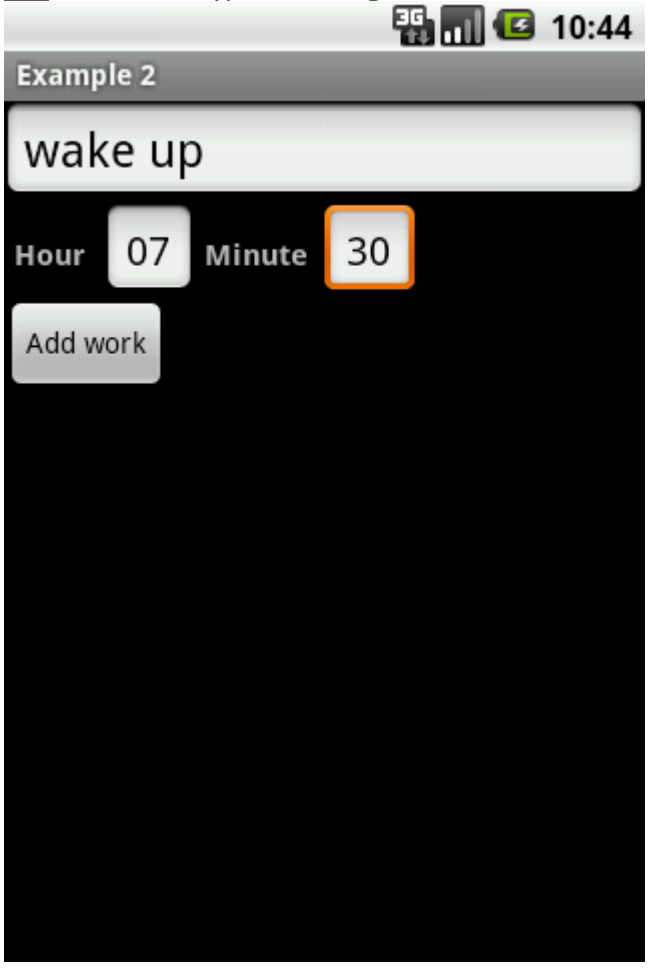
```
55 {
56                             // TODO Auto-generated method
stub
57                         }
58                     });
59                     builder.show();
60                 }
61                 //Lấy nội dung công việc và thời gian ra từ Edit Text và đưa
vào list
62                 else {
63                     String str = workEnter.getText().toString() + " - "
64                                     + hourEdit.getText().toString() + ":"
65                                     + minuteEdit.getText().toString();
66                     arrayWork.add(0,str);
67                     arrayAdapter.notifyDataSetChanged();
68                     workEnter.setText("");
69                     hourEdit.setText("");
70                     minuteEdit.setText("");
71                 }
72             }
73         };
74
75         //set Listener cho button
76         button.setOnClickListener(add);
77     }
78 }
```

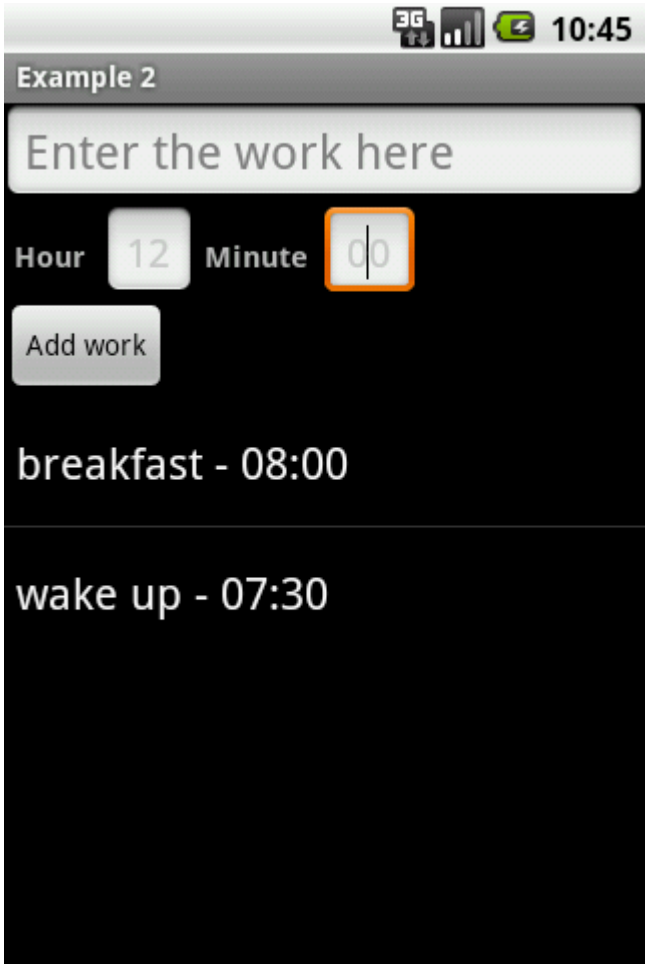
Mình đã chú thích đầy đủ và đoạn code cũng khá dễ hiểu. Tuy nhiên cần lưu ý 2 vấn đề ở đây.
- Khởi tạo đối tượng ArrayAdapter: Các bạn thấy đối số truyền vào là ***(this, android.R.layout.simple_list_item_1, arrayWork)***. This là đối số của lớp Context (ở đây chính là activity Example). Bạn sẽ gặp Context trong rất nhiều khởi tạo các lớp và nên hiểu Context có ý nghĩa gì. Mình xin đưa ra giải thích của anh Giáp (thank mr giaplv):
http://vietandroid.com/images/misc/quote-left.png); background-color: transparent; width: 9px; height: 13px; position: absolute; left: -9px; max-width: 540px !important; background-position: 0% 50%; background-repeat: no-repeat no-repeat;">
*Context thuộc **android.content** (android.content.Context).*

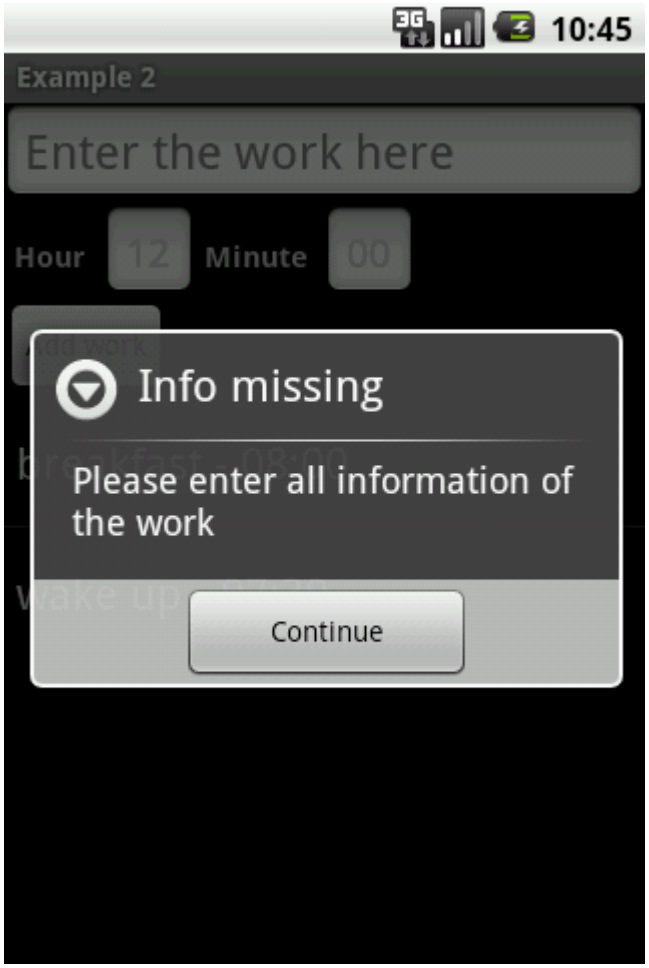
Là một Interface (lớp giao tiếp) chứa hầu hết thông tin về môi trường ứng dụng của android, có nghĩa là mọi thao tác, tương tác với hệ điều hành điều phải qua lớp này.
Nó là một lớp abstract (trừu tượng) cung cấp cho những lớp khác các phương thức để tương tác với hệ thống Android. Nó cho phép truy cập tới các nguồn tài nguyên (resources) đã được định nghĩa và các lớp khác. Ví dụ như nó có thể khởi tạo và chạy các activities, các broadcast và các intents,... Chúng ta coi như Contex là một lớp ở mức ứng dụng (Application level- liên quan tới hệ thống).
Tóm lại context giúp chúng ta dễ dàng truy cập và tương tác tới các tài nguyên của hệ thống, các thông tin, các dịch vụ (services), các thông số cấu hình, database, wallpaper, danh bạ, cuộc gọi, kết nối, chế độ rung (vibrator),...
****sở dĩ hầu hết các lớp có liên quan tới UI (layout, button, textview, imageview, listview,...) đều pải super tới Context vì bản thân nó đảm nhiệm việc truy cập resource (R.id, R.layout,....). Nếu chúng ta không tham chiếu tới Context class thì đương nhiên không thể dùng tới các resources mà chúng ta đã tạo ra.*
Tiếp theo là android.R.layout.simple_list_item_1, đối này định nghĩa cách thể hiện item (ở đây là String) trong List View. Các bạn hãy ghi nhớ android.R.* là các tài nguyên (resource) có sẵn của Android cho phép bạn truy cập và sử dụng. Sau này khi hướng dẫn tạo custom View cho List View mình sẽ đề cập lại vấn đề này.
Cuối cùng arrayWork chính là mảng cần được bind của adapter.

- AlertDialog là lớp cho phép đưa ra 1 hộp thoại, thường dùng để đưa ra thông tin hoặc cảnh báo đơn giản. Trong code mình tạo 1 builder, tạo tiêu đề (title) cho nó, đưa ra thông báo (message) và cuối cùng là tạo 1 positive button (nhưng không định nghĩa xử lý khi nhấn nút này, vì vậy nếu bạn nhấn nút thì dialog sẽ chỉ đơn giản thực hiện việc đóng lại).

B6: Tiến hành chạy thử chương trình. Run as -> Android Application. Enjoy yourself




10:45
Example 2
Enter the work here
Hour
12
Minute
00
Add work
breakfast - 08:00
wake up - 07:30


10:45
Example 2
Enter the work here
Hour
12
Minute
00
Info missing
Please enter all information of the work
Continue

BÀI 4: HƯỚNG DẪN LẬP TRÌNH CƠ BẢN VỚI ANDROID

Trong bài này sẽ hướng dẫn cách tạo 1 custom ViewGroup, sử dụng ViewGroup này vào ListView, và cuối cùng là tạo 1 Option Menu. Đây cũng sẽ là bài cuối cùng mình viết về làm việc với View, các bài sau sẽ chuyển qua Intent và BroadCast Receiver.

Custom ViewGroup

ViewGroup thông thường chúng ta hay gặp là LinearLayout, Relative Layout. Xây dựng custom ViewGroup cho phép chúng ta tạo 1 tập các widget được sắp xếp theo ý muốn rồi đưa vào sử dụng.

Yêu cầu: Xây dựng ứng dụng dạng To Do List: Cho phép nhập vào nội dung công việc và thời gian thực hiện công việc rồi đưa vào list công việc. Cho phép xóa các công việc khỏi list.

B1: Khởi tạo project: File -> New -> Android Project
Project name: Example 3
Build Target: Chọn Android 1.5
Application name: Example 3
Package name: at.exam
Create Activity: Example
=> Kích nút Finish.

B2: Xây dựng custom view group trong XML. Đi tới res\layout tạo 1 file XML mới là list.xml. Gõ nội dung sau vào:
Mã:

```
<?xml version="1.0" encoding="utf-8"?> <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="horizontal">     <CheckBox
android:id="@+id/check_work"
android:layout_width="wrap_content"
android:layout_height="wrap_content"           android:text=""
android:paddingTop="45px"           android:paddingRight="10px"
/>       <LinearLayout
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="vertical"                  >
<TextView               android:id="@+id/work_content"
android:textSize="24px"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:lines="1"
android:textColor="@color/work_color"            />
<TextView               android:id="@+id/time_content"
android:textSize="16px"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:lines="1"
```

```
android:textColor="@color/time_color"           />
</LinearLayout> </LinearLayout>
```

Custom ViewGroup của chúng ta ở đây khá đơn giản, đó là 1 LinearLayout chứa 2 thành phần: 1 CheckBox và 1 LinearLayout khác gồm 2 TextView để hiển thị nội dung công việc và thời gian.

B3: Đã xong giao diện cho custom ViewGroup, chúng ta sẽ thiết kế giao diện cho chương trình trong main.xml. Ở đây mình dùng lại giao diện của Example 2 trong bài 2. Mã:

```
<?xml version="1.0" encoding="utf-8"?> <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"     >     <EditText
android:id="@+id/work_enter"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:hint="@string/work_hint"         android:lines="1"
android:textSize="24px"     />     <LinearLayout
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="horizontal"         >         <TextView
android:layout_width="50px"
android:layout_height="wrap_content"
android:text="@string/hour_edit"
android:typeface="normal"             android:textSize="15px"
android:textStyle="bold"             android:padding="5px"
/>         <EditText             android:id="@+id/hour_edit"
android:layout_width="45px"
android:layout_height="wrap_content"             android:hint="12"
android:textColorHint="@color/hint_color"
android:textSize="20px"             android:gravity="center"
android:padding="5px"             android:numeric="integer"
android:maxLength="2"         />         <TextView
android:layout_width="65px"
android:layout_height="wrap_content"
android:text="@string/minute_edit"
android:typeface="normal"             android:textSize="15px"
android:textStyle="bold"             android:padding="5px"
/>         <EditText             android:id="@+id/minute_edit"
android:layout_width="45px"
android:layout_height="wrap_content"             android:hint="00"
android:textColorHint="@color/hint_color"
android:textSize="20px"             android:gravity="center"
android:padding="5px"             android:numeric="integer"
android:maxLength="2"         />             </LinearLayout>
<Button         android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:gravity="center"
android:text="@string/button_content"         />     <ListView
```

```
android:id="@+id/list"          android:layout_width="fill_parent"
android:layout_height="wrap_content"     /> </LinearLayout>
```

B4: Tạo file colors.xml trong res\value:

Mã:

```
<?xml version="1.0" encoding="utf-8"?> <resources>     <color
name="work_color">#ffffff</color>     <color
name="time_color">#cccccc</color>         <color
name="hint_color">#cccccc</color> </resources>
```

work_color là màu của nội dung công việc trong list. time_color màu của thời gian công việc. hint_color màu của text hint (dòng hướng dẫn) các EditText.

B5: Chỉnh sửa file strings.xml trong res\value:

Mã:

```
<?xml version="1.0" encoding="utf-8"?> <resources>     <string
name="app_name">Example 3</string>     <string
name="work_hint">Enter the work here</string>     <string
name="hour_edit">Hour</string>     <string
name="minute_edit">Minute</string>     <string
name="button_content">Add work</string> </resources>
```

B6: Time to coding. Đi tới src\at.exam tạo một class mới là CustomViewGroup với nội dung sau:

Mã:

```
package at.exam;  import android.content.Context; import
android.view.LayoutInflater; import android.widget.CheckBox;
import android.widget.LinearLayout; import
android.widget.TextView;  public class CustomViewGroup extends
LinearLayout {     public CheckBox cb;     public TextView
workContent;     public TextView timeContent;     public
CustomViewGroup(Context context) {         super(context);
//Sử dụng LayoutInflater để gán giao diện trong list.xml cho class
này          LayoutInflater li = (LayoutInflater)
this.getContext()
.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
li.inflate(R.layout.list, this, true);                  //Lấy về
các View qua Id          cb = (CheckBox)
findViewById(R.id.check_work);          workContent = (TextView)
findViewById(R.id.work_content);          timeContent = (TextView)
findViewById(R.id.time_content);     } }
```

Đoạn code trên giúp ta định nghĩa giao diện của custom ViewGroup mới dựa trên file list.xml. Mọi người cũng có thể tạo giao diện bằng code, ko cần sử dụng XML nhưng sẽ phức tạp hơn và mình cũng ko giới thiệu ở đây.

B7: Tạo 1 class Work cũng trong at.exam để thể hiện công việc:

Mã:

```
package at.exam;  public class Work {     private String
workContent;     private String timeContent;     private boolean
isChecked;          public Work(String workContent, String
timeContent) {          this.workContent = workContent;
this.timeContent = timeContent;          isChecked = false;     }
public String getContent() {          return workContent;     }
```

```
public String getTime() {          return timeContent;      }
public void setChecked(boolean isChecked) {          this.isChecked
= isChecked;      }           public boolean isChecked() {
return isChecked;      } }
```

Code rất đơn giản nên mình sẽ không chú thích nữa.

B8: Chúng ta đã tạo xong custem ViewGroup, bây giờ chính là lúc sử dụng. Tạo 1 class mới tên là ListWorkApdapter trong at.exam:

Mã:

```
package at.exam;  import java.util.ArrayList; import
android.content.Context; import android.view.LayoutInflater;
import android.view.View; import android.view.ViewGroup; import
android.widget.ArrayAdapter; import android.widget.CheckBox;
import android.widget.CompoundButton; import
android.widget.TextView; import
android.widget.CompoundButton.OnCheckedChangeListener;  public
class ListWorkAdapter extends ArrayAdapter<Work>{
ArrayList<Work> array;      int resource;      Context context;
public ListWorkAdapter(Context context, int textViewResourceId,
ArrayList<Work> objects) {          super(context,
textViewResourceId, objects);          this.context = context;
resource = textViewResourceId;          array = objects;
}          //Phương thức xác định View mà Adapter hiển thị, ở đây
chính là CustomViewGroup      //Bắt buộc phải Override khi kế thừa
từ ArrayAdapter      @Override      public View getView(int
position, View convertView, ViewGroup parent) {          View
workView = convertView;                    if (workView == null) {
workView = new CustomViewGroup(getContext());          }
//Lấy về đối tượng Work hiện tại          final Work work =
array.get(position);           if (work != null) {
TextView workContent = ((CustomViewGroup) workView).workContent;
TextView timeContent = ((CustomViewGroup) workView).timeContent;
CheckBox checkWork = ((CustomViewGroup) workView).cb;
//Set sự kiện khi đánh dấu vào checkbox trên list
checkWork.setOnCheckedChangeListener(new OnCheckedChangeListener()
{                  @Override                  public void
onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
work.setChecked(isChecked);                                        }
});                           //Lấy về nội dung cho TextView và
CheckBox dựa vào đối tượng Work hiện tại
workContent.setText(work.getContent());
timeContent.setText(work.getTime());
checkWork.setChecked(work.isChecked());          }
return workView;      }      }
```

ListWorkAdapter sẽ được sử dụng thay thế cho ArrayAdapter được bind với ListView. Thông thường ArrayAdapter chỉ cho hiển thị String bằng TextView, nhưng với việc kế thừa và override phương thức getView, ta có thể định nghĩa lại hiển thị cho các thành phần của ListView.

B9: Việc cuối cùng cần làm là viết lại Activity. Tới Example.java và chỉnh sửa theo nội dung sau:

Mã:

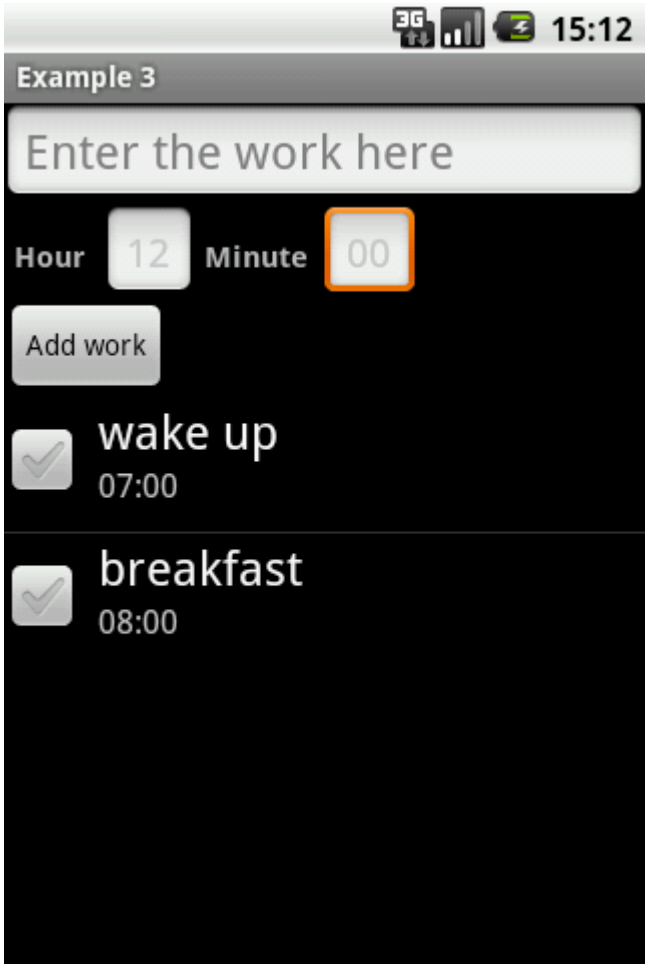
```
package at.exam;  import java.util.ArrayList;  import
android.app.Activity; import android.app.AlertDialog; import
android.content.DialogInterface; import android.os.Bundle; import
android.view.Menu; import android.view.MenuItem; import
android.view.View; import android.view.View.OnClickListener;
import android.widget.ArrayAdapter; import android.widget.Button;
import android.widget.EditText; import android.widget.ListView;
public class Example extends Activity {         //Các hằng dùng
cho tạo Option Menu     private static final int DELETE_WORK =
Menu.FIRST;     private static final int ABOUT = Menu.FIRST + 2;
ArrayList<Work> array;     ListWorkAdapter arrayAdapter;
@Override     public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);                  array = new
ArrayList<Work>();         arrayAdapter = new
ListWorkAdapter(this,                 R.layout.list, array);
final EditText workEnter = (EditText)
findViewById(R.id.work_enter);         final EditText hourEdit =
(EditText) findViewById(R.id.hour_edit);         final EditText
minuteEdit = (EditText) findViewById(R.id.minute_edit);
final Button button = (Button) findViewById(R.id.button);
//Tạo list view cho danh sách công việc         final ListView
list = (ListView) findViewById(R.id.list);
list.setAdapter(arrayAdapter);                  OnClickListener
add = new OnClickListener() {             @Override
public void onClick(View v) {                 if
(workEnter.getText().toString().equals("") ||
hourEdit.getText().toString().equals("") ||
minuteEdit.getText().toString().equals("")) {
AlertDialog.Builder builder = new
AlertDialog.Builder(Example.this);
builder.setTitle("Info missing");
builder.setMessage("Please enter all information of the work");
builder.setPositiveButton("Continue", new
DialogInterface.OnClickListener() {                         public
void onClick(DialogInterface dialog, int which) {
// TODO Auto-generated method stub
}                                       });
builder.show();                 }                 else {
String workContent = workEnter.getText().toString();
String timeContent = hourEdit.getText().toString() + ":"
+ minuteEdit.getText().toString();                     Work work =
new Work(workContent, timeContent);
array.add(0, work);
arrayAdapter.notifyDataSetChanged();
workEnter.setText("");                    hourEdit.setText("");
minuteEdit.setText("");                 }             }
};                  button.setOnClickListener(add);             }
//Tạo Option Menu     public boolean onCreateOptionsMenu(Menu
menu) {         super.onCreateOptionsMenu(menu);
menu.add(0, DELETE_WORK, 0,"Delete"
```

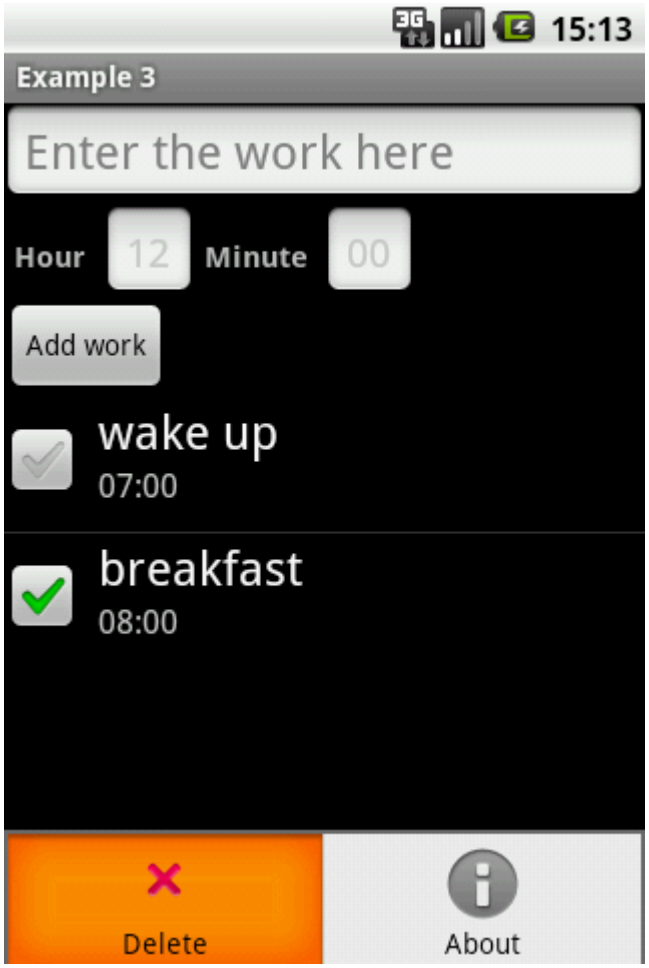
```
).setIcon(android.R.drawable.ic_delete);
menu.add(0, ABOUT, 0,"About"
).setIcon(android.R.drawable.ic_menu_info_details);          return
true;     }          //Xử lý sự kiện khi các option trong Option
Menu được lựa chọn     public boolean
onOptionsItemSelected(MenuItem item) {          switch
(item.getItemId()) {                          case DELETE_WORK: {
deleteCheckedWork();                    break;              }
case ABOUT: {                  AlertDialog.Builder builder = new
AlertDialog.Builder(this);
builder.setTitle("VietAndroid");
builder.setMessage("AUTHOR:" + "\n" + "  Nguyen Anh Tuan" + "\n" +
"SOURCE:" + "\n" + "  diendan.vietandroid.com");
builder.setPositiveButton("Close", new
DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int which) {
}                  });
builder.setIcon(android.R.drawable.ic_dialog_info);
builder.show();                    break;              }          }
return true;          }      private void deleteCheckedWork() {
if (array.size() > 0) {              for (int i = array.size() - 1;
i >= 0; i--) {                  if (array.get(i).isChecked()) {
array.remove(i);
arrayAdapter.notifyDataSetChanged();                        continue;
}              }          }              } }
```

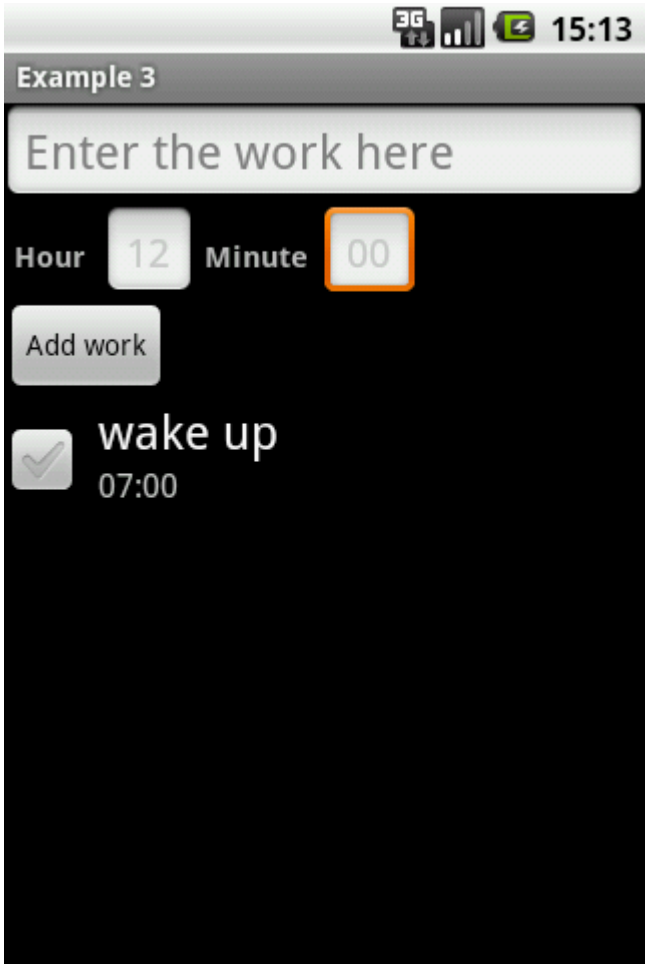
OK. Vậy là xong. Option Menu là menu ẩn chỉ hiện ra khi bạn nhấn nút Menu của điện thoại. Option Menu rất tiện trong việc đưa ra các tùy chỉnh, giống như khi bạn nhấn phím Esc khi đang chơi game trên PC vậy.

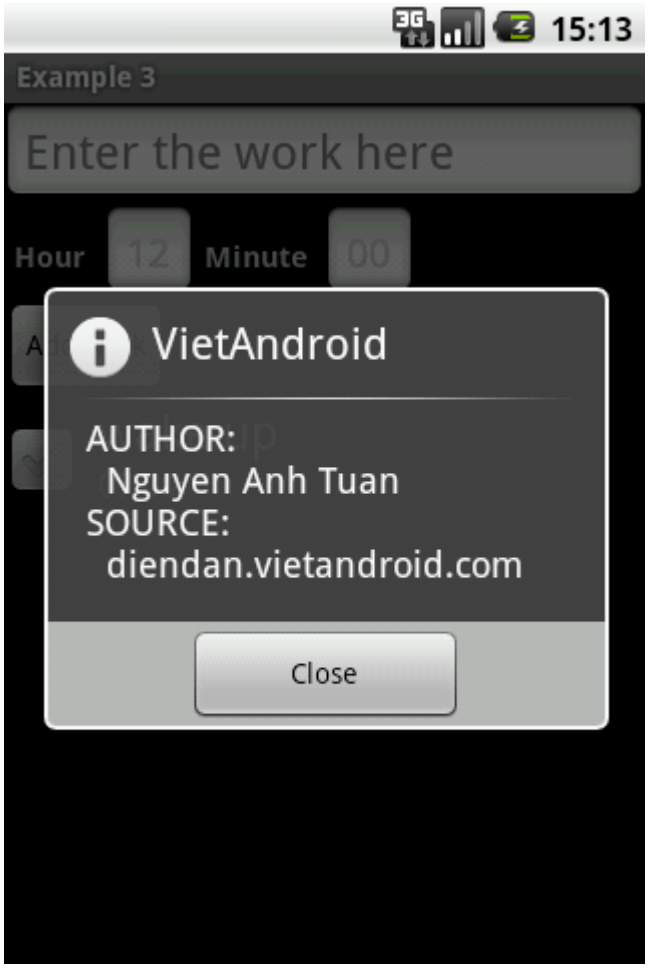
Các bạn có thể lưu ý là thay vì sử dụng ArrayList<String> như trước mình đã thay bằng ArrayList<Work> và trong khởi tạo đối tượng arrayAdapter thì đối số thứ 2 là R.layout.list thay vì android.R.layout.simple_list_item_1, nghĩa là chúng ta đã sử dụng layout do mình tự tạo thay vì layout Android cung cấp sẵn cho hiển thị các thành phần của ListView.

Nếu chạy thử, các bạn có thể thấy khi ta đánh dấu vào checkbox của 1 thành phần trong list, rồi nhấn Menu và chọn delete thì thành phần sẽ bị gỡ bỏ khỏi danh sách.


15:12
Example 3
Enter the work here
Hour
12
Minute
00
Add work
wake up
07:00
breakfast
08:00


15:13
Example 3
Enter the work here
Hour
12
Minute
00
Add work
wake up
07:00
breakfast
08:00
Delete
About


15:13
Example 3
Enter the work here
Hour
12
Minute
00
Add work
wake up
07:00



vietandroid

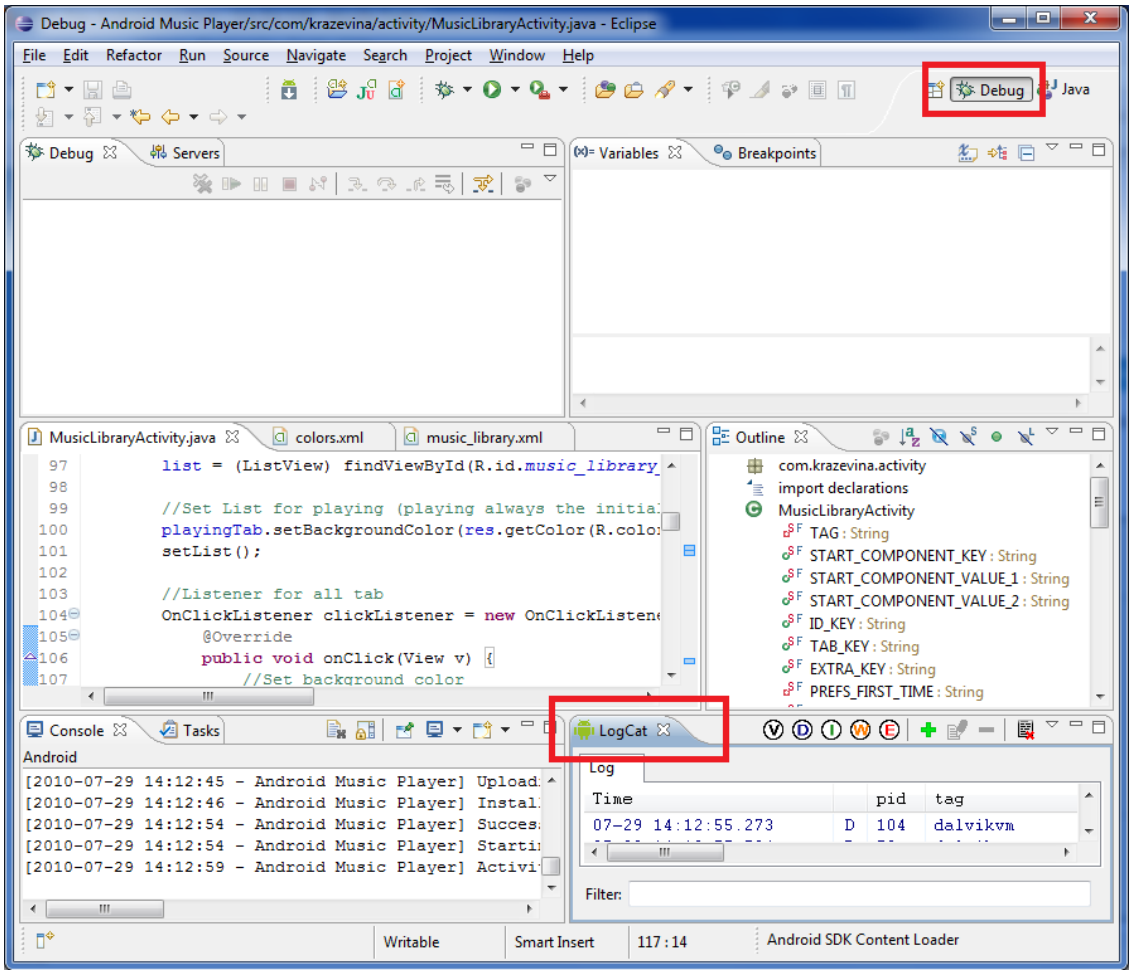
BÀI 5: MỘT SỐ CHỨC NĂNG CẦN BIẾT KHI LẬP TRÌNH ANDROID VỚI ECLIPSE

Nhiều người chuyển từ J2SE hoặc J2ME sang Android sẽ ngạc nhiên vì câu lệnh debug kinh điển System.out.println() không còn in ra trên cửa sổ Console nữa. Google đã thay thế nó bằng Logcat, một cửa sổ ghi lại toàn bộ hoạt động của hệ điều hành.

1.Debug cho ứng dụng Android:

Nhiều người chuyển từ J2SE hoặc J2ME sang Android sẽ ngạc nhiên vì câu lệnh debug kinh điển **System.out.println()** không còn in ra trên cửa sổ Console nữa. Google đã thay thế nó bằng Logcat, một cửa sổ ghi lại toàn bộ hoạt động của hệ điều hành. Để mở Logcat, trước tiên các bạn chọn Window -> Open Perspective -> Debug. Nếu ko thấy option Debug thì chọn Other và tìm Debug trong cửa sổ mới hiện ra. Sau đó chọn tab Debug mới xuất hiện ở góc trên bên phải của Eclipse (xem hình). Theo kinh nghiệm của mình thì tốt nhất là Maximize Logcat ra luôn cho dễ quan sát.

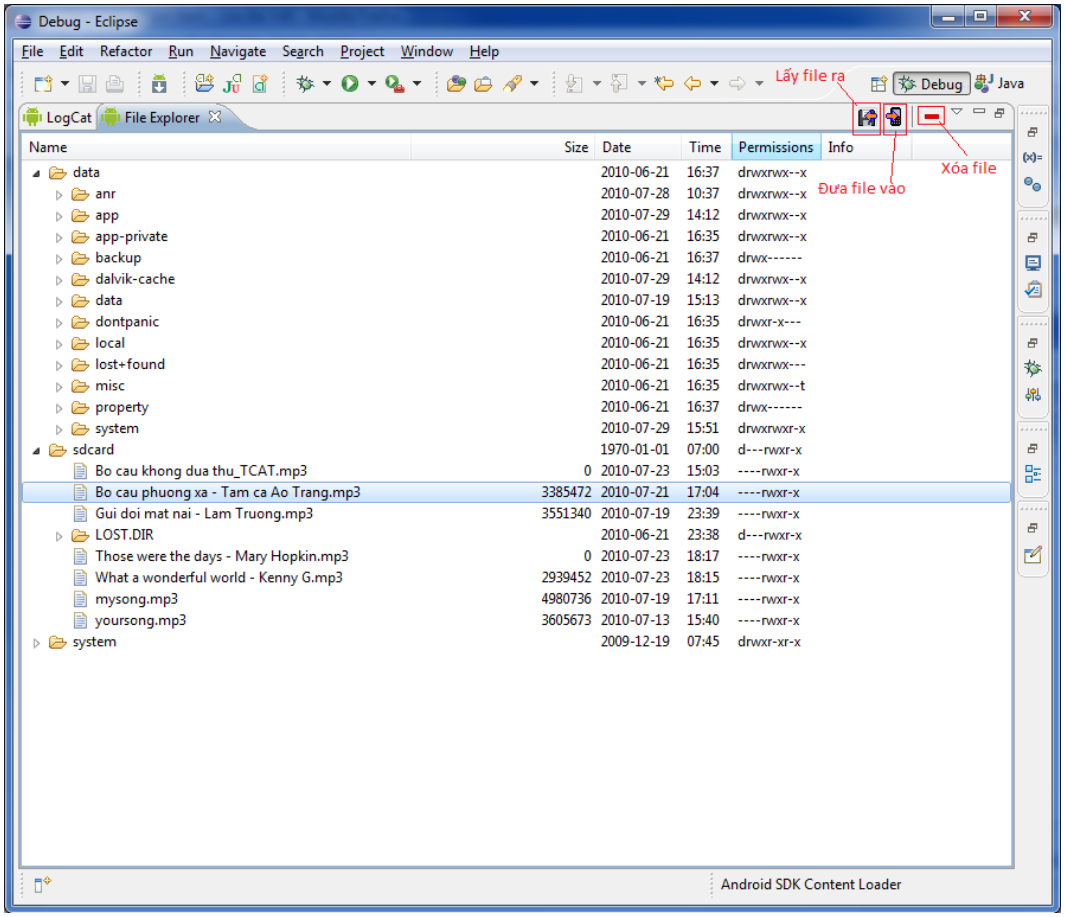
Cảm nhận ngày xưa khi mới sử dụng Logcat là rối và khó dùng. Nhưng càng về sau mình càng quen và thấy nó tiện hơn Console nhiều, vì Console chỉ đưa ra thông báo do chúng ta gọi, còn Logcat thì đưa cả luôn những thông báo của hệ điều hành, giúp chúng ta nắm được hệ điều hành đang làm gì, gọi đến cái gì, khởi chạy những gì...



2.File Explorer của Android:

File Explorer là một chức năng hữu ích Google đưa vào giúp chúng ta quản lý file trong sd card và cả file system data (chỉ quản lý được của emulator, không thể truy nhập system data của thiết bị thật). FE giúp bạn dễ dàng đưa file vào / lấy file ra trong sdcard ảo của emulator, xóa cơ sở dữ liệu của chương trình để khởi tạo lại (only emulator)...

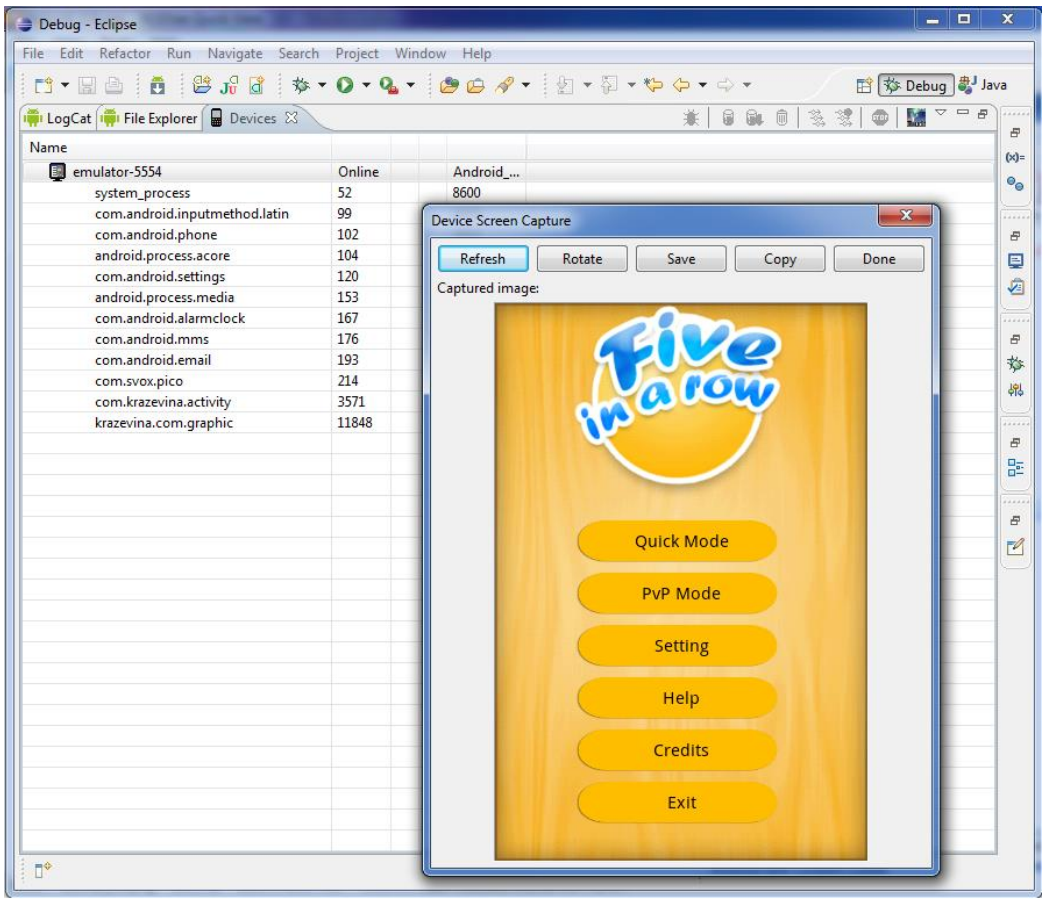
Mở FE bằng cách vào Window -> Show View -> Others -> Android -> File Explorer. Mình để FE trong cửa sổ Debug cho tiện quản lý.



3.Device của Android:

Device cũng là một chức năng hữu ích nữa trong Android giúp bạn quản lý thiết bị ảo cũng như thật của mình. Mở Device bằng cách vào Window -> Show View -> Device hoặc vào Window -> Show View -> Others -> Android -> Device.

Chức năng mình thường sử dụng nhất của device là Screen Capture, cực kỳ tiện để lấy ảnh minh họa làm thuyết trình hoặc giới thiệu trên Google Market.


Debug - Eclipse
File Edit Refactor Run Navigate Search Project Window Help
Debug
Java
LogCat
File Explorer
Devices
Name
emulator-5554
Online
Android_...
system_process
52
8600
com.android.inputmethod.latin
99
com.android.phone
102
android.process.acore
104
com.android.settings
120
android.process.media
153
com.android.alarmclock
167
com.android.mms
176
com.android.email
193
com.svox.pico
214
com.krazevina.activity
3571
krazevina.com.graphic
11848
Device Screen Capture
Refresh
Rotate
Save
Copy
Done
Captured image:
Five in a row
Quick Mode
PvP Mode
Setting
Help
Credits
Exit