

ĐẠI HỌC CÔNG NGHỆ TP.HCM (HUTECH)

KHOA CÔNG NGHỆ THÔNG TIN

==oOo==

BÀI THỰC HÀNH

LẬP TRÌNH DI ĐỘNG



ANDROID

Khoa CNTT – HUTECH

Bộ môn: Công Nghệ Phần Mềm

Người soạn: Nguyễn Hà Giang



Lab 1: Ứng dụng Android đầu tiên

Mục tiêu

- ✚ Làm quen với cách thức tạo ứng dụng Android cơ bản dùng Android Studio.
- ✚ Hiểu cấu trúc cơ bản của Android project.
- ✚ Dùng XML để tạo layout của Activity.
- ✚ Quen với việc sử dụng các resource trong ứng dụng Android.

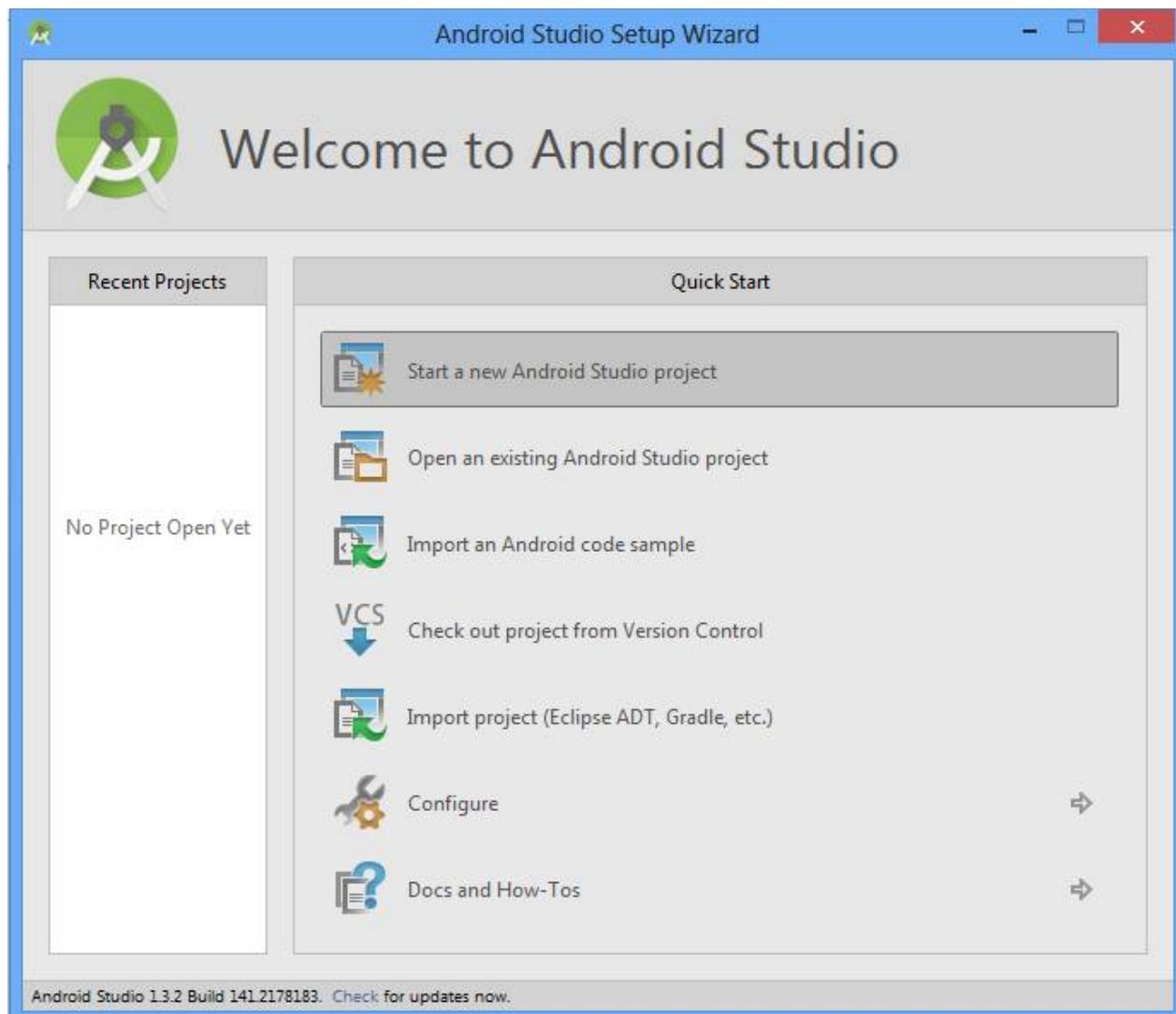
Yêu cầu

- ✚ Đã cài đặt môi trường đầy đủ để xây dựng ứng dụng Android trên Android Studio.
- ✚ Có một số kiến thức cơ bản về lập trình Android.

Hướng dẫn

1. Bước 1: Tạo ứng dụng Android từ Android Studio.

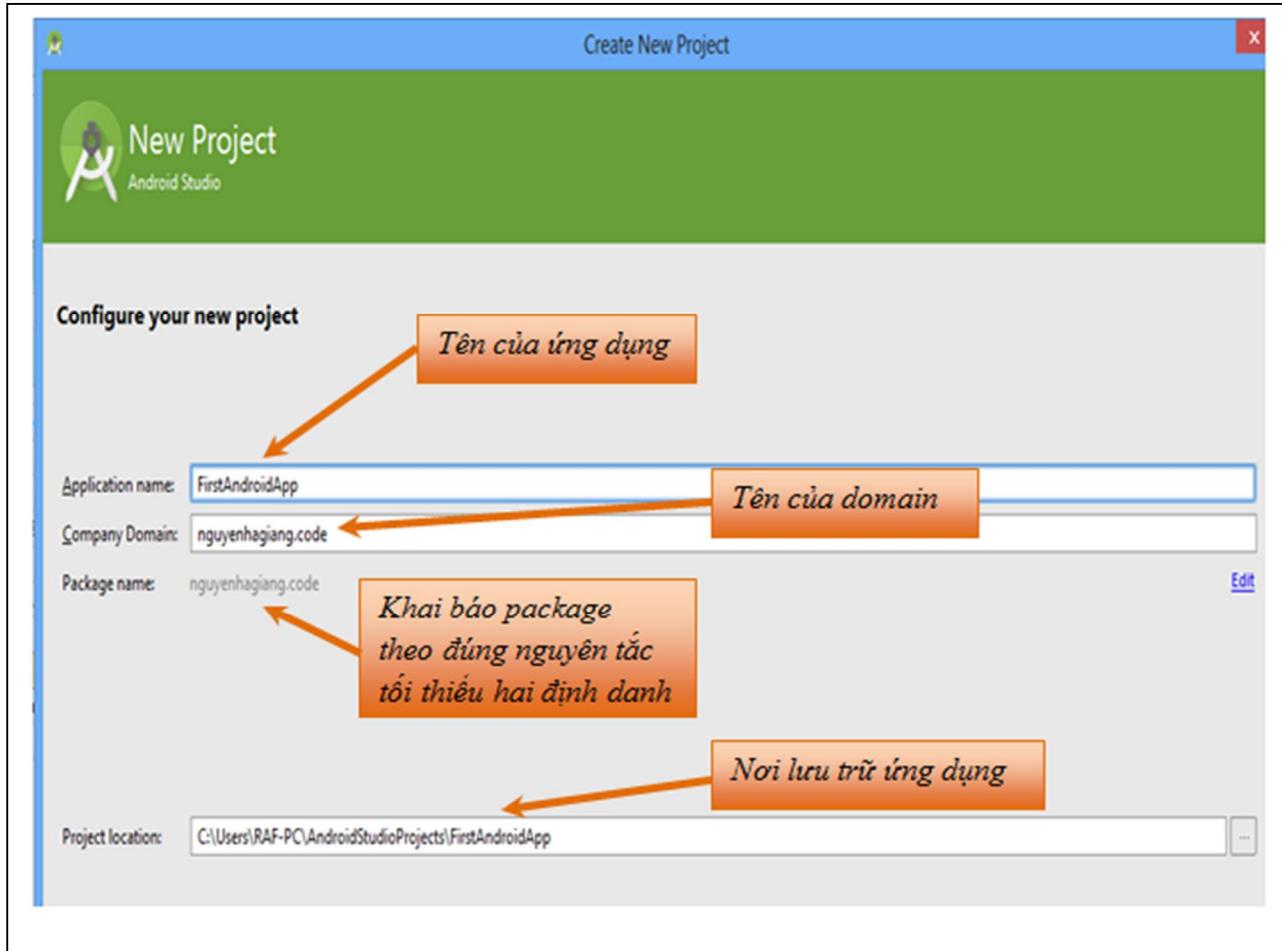
Trong hộp thoại Welcome to Android Studio, chọn Start a new Android Studio project để tạo một project mới.



Hình 1.1: Minh họa cách chọn bắt đầu Android Studio Project

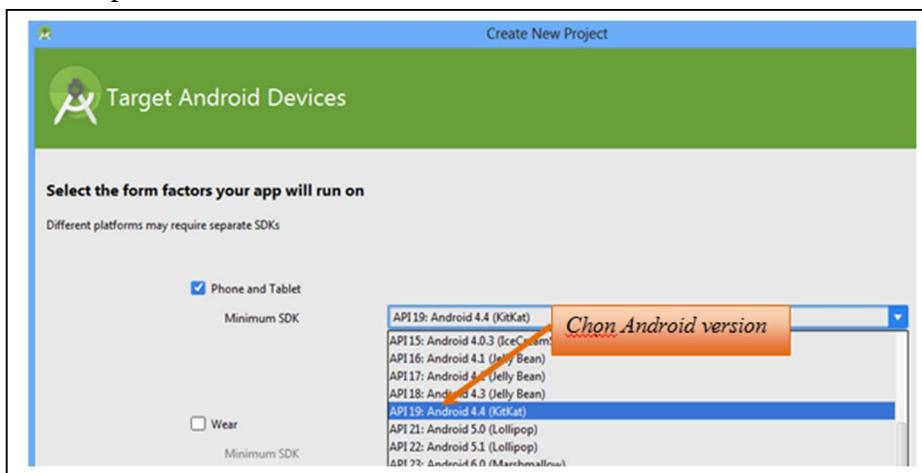
Hộp thoại đặt tên ứng dụng và domain xuất hiện.

Ngoài ra, trong Android Studio chọn File → New , chọn tiếp New Project cũng xuất hiện hộp thoại sau.



Hình 1.2: Minh họa cách tạo Android Project

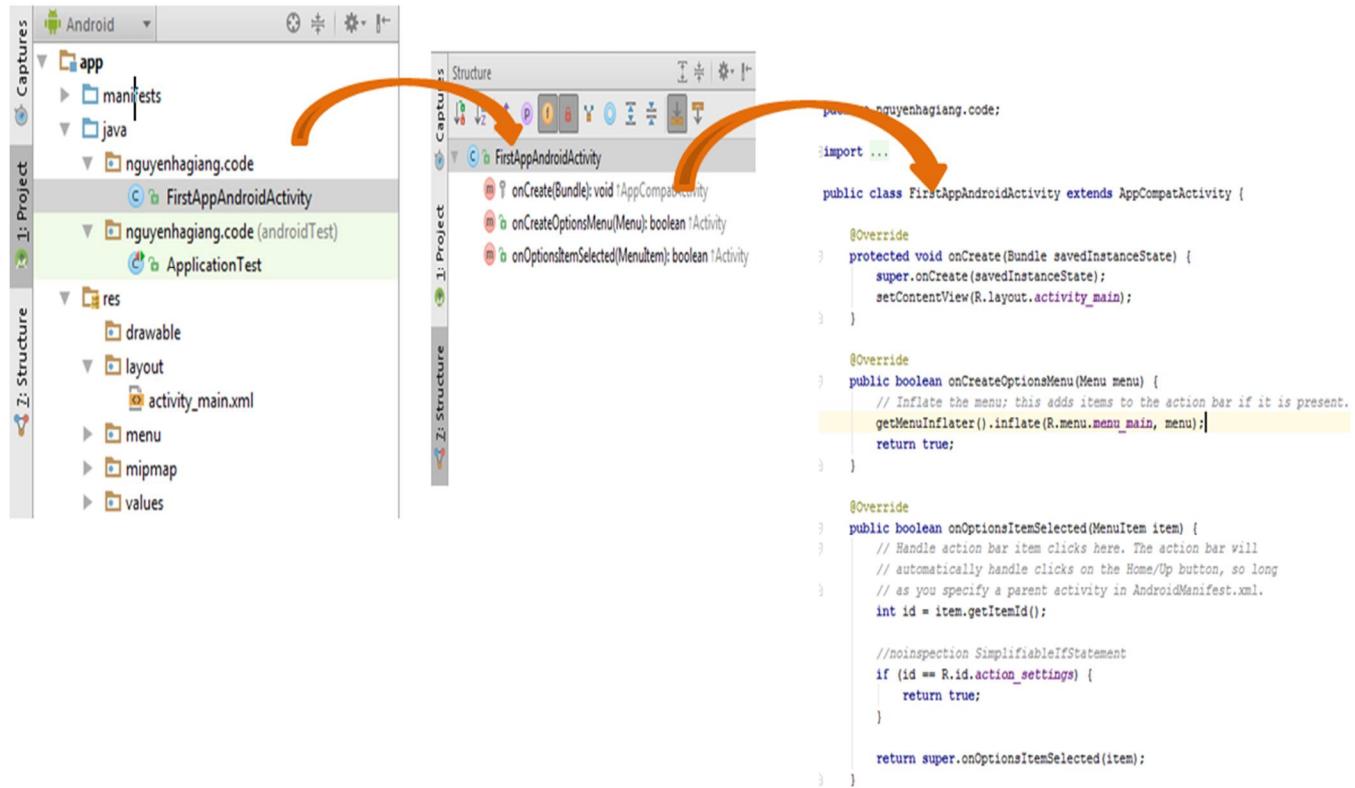
Chọn Next để chọn phiên bản Android



Hình 1.3: Minh họa cách chọn phiên bản Android

Sau khi đã khai báo các thông tin để tạo mới Android Project thì chọn Finish để hoàn tất.

Android Studio sẽ tạo một project Android có cấu trúc như sau:



Hình 1.4: Toàn bộ Android project ban đầu được Android Studio phát sinh

Ứng dụng này chỉ có duy nhất một thành phần gọi là Activity có tên là FirstAppAndroidActivity, trong ứng dụng Android, activity là thành phần GUI chứa các widget (tương tự như control trong windows form). Nói một cách tổng quát ứng dụng nếu có tương tác với người dùng thông qua UI thì phải có activity, trong ứng dụng Android có thể tạo ra nhiều Activity (giống như tạo nhiều form trong lập trình desktop).

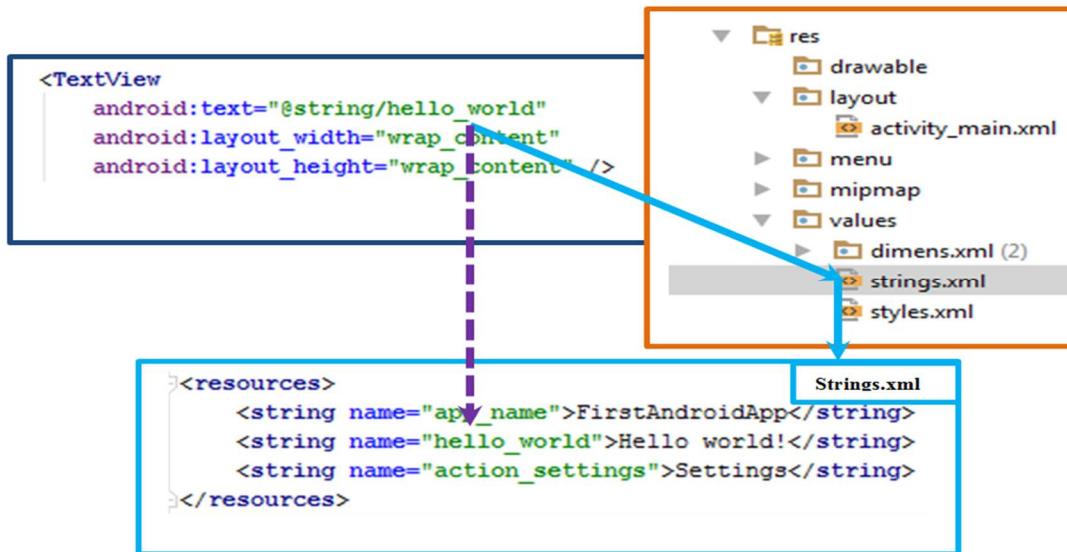
Trong Activity FirstAppAndroidActivity trên thì có phương thức override phương thức này **dùng để khởi tạo thành phần UI và các xử lý của activity**. Trong phương thức này có gọi hàm **setContentView** và truyền vào là id của layout được khai báo trong thư mục res/layout



Hình 1.5: File XML Layout chứa mô tả giao diện của activity

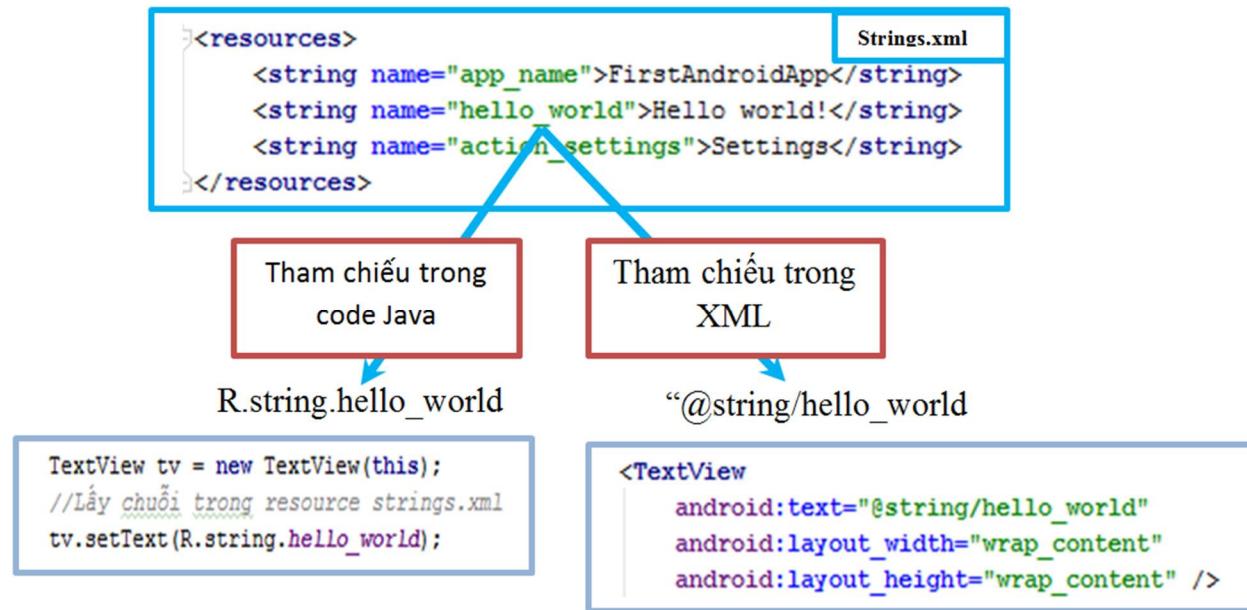
Giải thích file mô tả layout activity_main.xml của activity:

- + Bao gồm một LinearLayout, đây là dạng ViewGroup cho phép chứa các View bên trong và được sắp xếp theo hai dạng: “vertical” hay “horizontal”. Trong layout này LinearLayout được thiết lập theo phương dọc, giá trị match_parent cho biết layout sẽ chiếm hết kích thước của thành phần bao chứa nó (full kích thước).
- + Một TextView là một dạng tương tự như Label trong Windows Form, cho phép hiển thị nội dung thông tin nào đó, TextView này được thiết lập có kích thước ngang là kích thước của thành phần bao chứa, và kích thước dài là wrap, vừa đủ hiển thị nội dung. Thuộc tính android:text thiết lập chuỗi cần hiển thị trên TextView, trong phần này khai báo chuỗi là @string/hello có ý nghĩa là lấy chuỗi tên hello được khai báo trong phần resource là file strings.xml, khi đó nội dung (giá trị) của chuỗi hello sẽ hiển thị lên trên TextView.



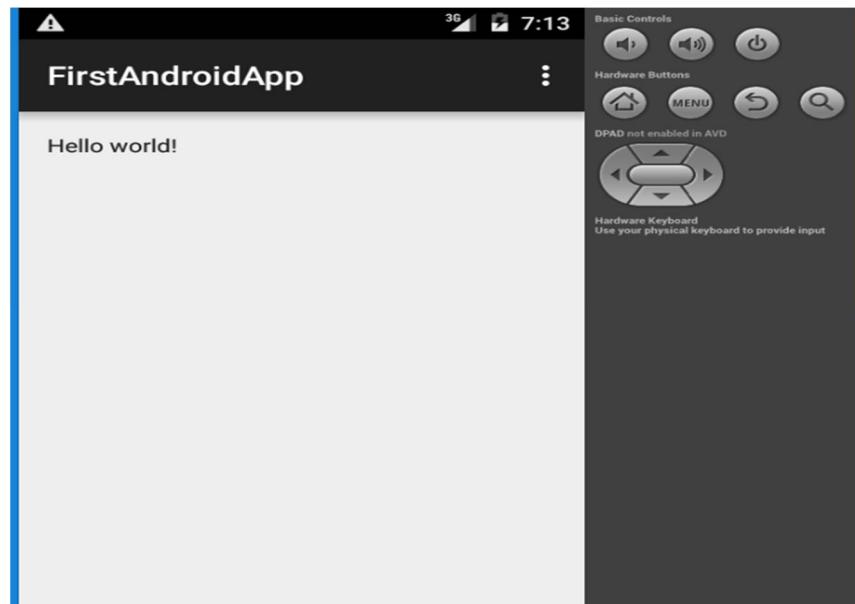
Hình 1.6: File strings.xml chứa định nghĩa các chuỗi

File strings.xml chứa các định nghĩa liên quan đến chuỗi, khi lập trình trên Android nên sử dụng file này để định nghĩa các chuỗi và trong chương trình Java hay phần layout sẽ tham chiếu đến các chuỗi này. Cách truy xuất chuỗi khai báo trong strings.xml được mô tả như hình dưới



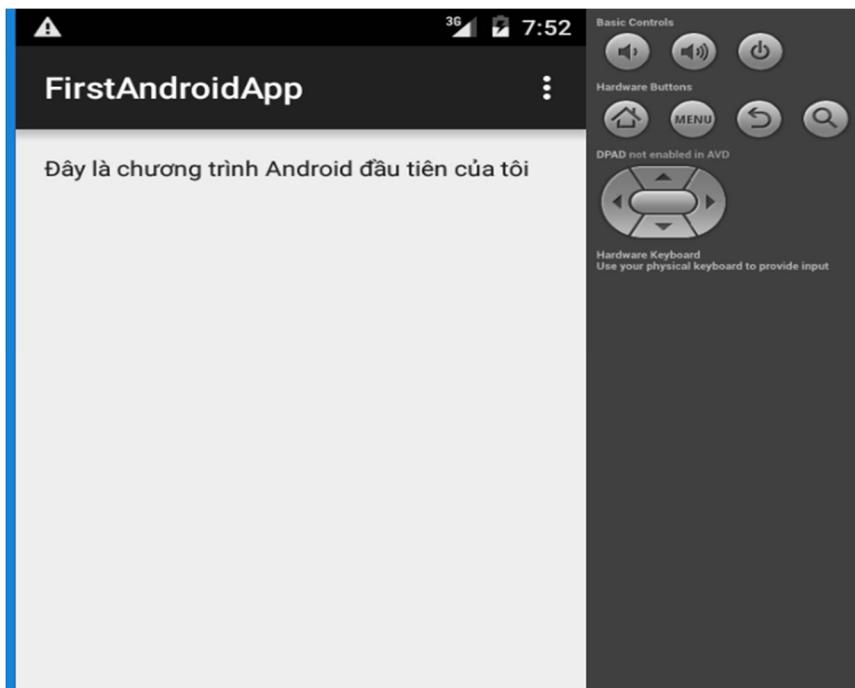
Hình 1.7: Mô tả cách thức tham chiếu đến chuỗi trong java code và XML layout.

2. Bước 2: Biên dịch và chạy ứng dụng đầu tiên ta được kết quả trên emulator như sau:



Hình 1.8: Ứng dụng khi chạy trên emulator

3. Bước 3: Modify lại chương trình để hiển thị thông báo sau: “**Đây là chương trình Android đầu tiên của tôi**”.



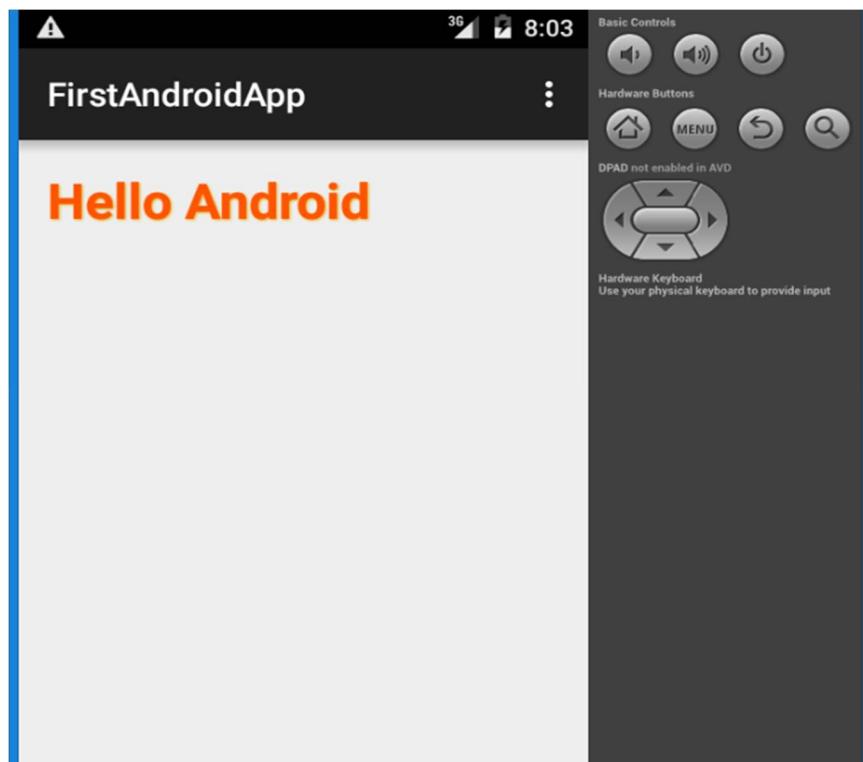
Hình 1.9: Ứng dụng sau khi modify lại chuỗi

4. Bước 4: làm quen với các thuộc tính của TextView, thiết lập các thuộc tính cho TextView theo bảng sau (thiết lập trong file layout xml).

Thiết lập thuộc tính cho TextView trong file layout XML

textSize	<i>30dp</i>
textColor	<i>#ff5500</i>
textStyle	<i>bold</i>
gravity	<i>center</i>
shadowColor	<i>#e6b121</i>
shadowRadius	<i>1.5</i>
shadowDx	<i>1</i>
shadowDy	<i>1</i>

Kết quả được activity như sau (trong demo này đã thay đổi text của TextView là “Hello Android”):



Hình 1.10: Kết quả sau khi thiết lập các thuộc tính của TextView

Trong phần khai báo màu của textColor và shadowColor ta dùng hằng số màu, việc dùng trực tiếp như vậy đôi khi khó hiểu (khi nhìn vào mã hexa không biết màu gì), ta có thể làm cách khác dễ hiểu hơn bằng cách tạo file resource định nghĩa bảng màu. Trong Android cho phép làm điều này bằng cách khai báo file colors.xml như hình minh họa sau:

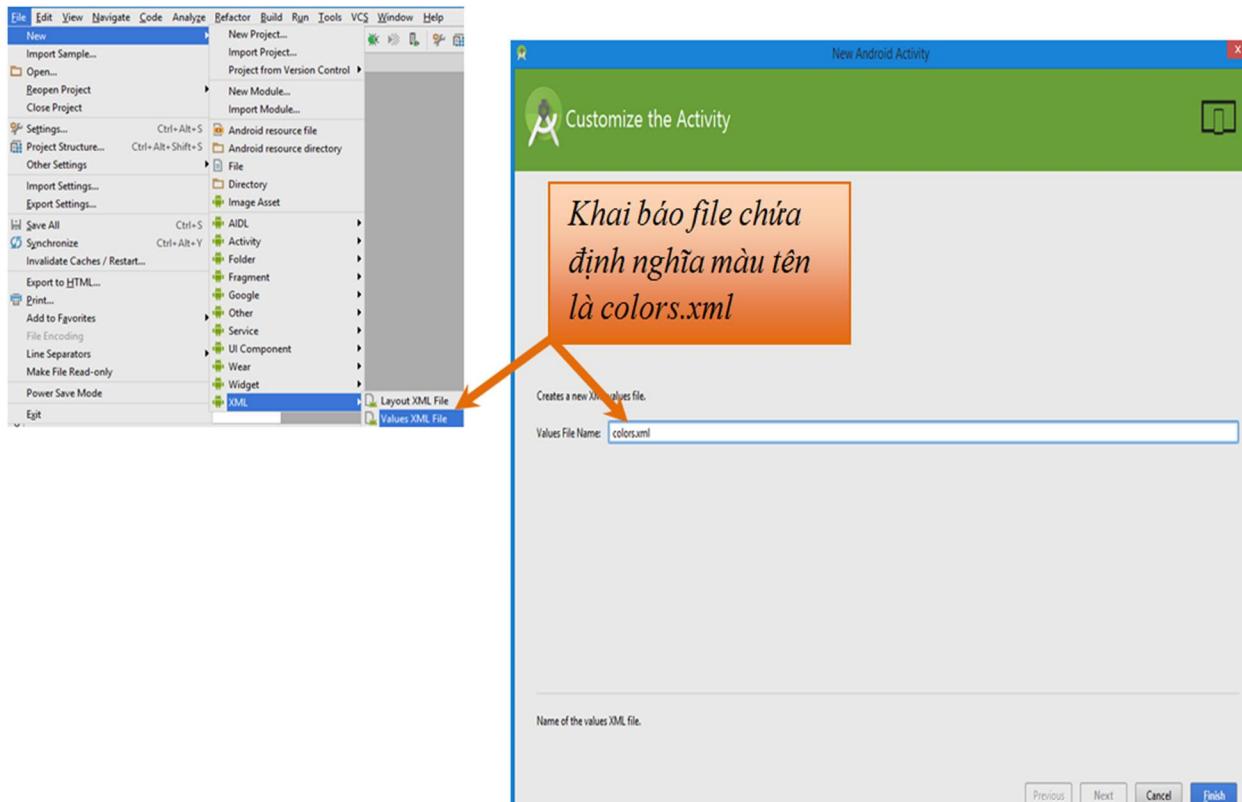
Trong file này ta định nghĩa hai màu như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="orange">#ff5500</color>
    <color name="gold">#e6b121</color>
</resources>
```

Khi tham chiếu trong layout thì dùng cú pháp sau

Thiết lập thuộc tính cho TextView trong file layout XML

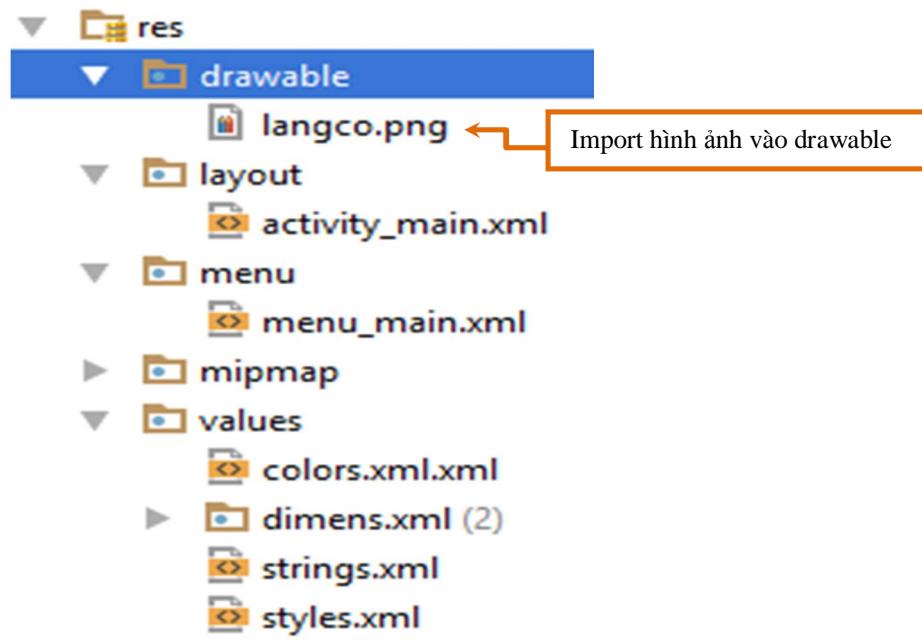
textColor	<code>@color/orange</code>
shadowColor	<code>@color/gold</code>



Hình 1.11: Màn hình bổ sung file định nghĩa hằng số màu trong resource

5. Bước 5: thêm hình nền vào trong linearlayout

Import một hình nền nào đó vào project, (cách thức import đã hướng dẫn trong phần lab về J2ME)



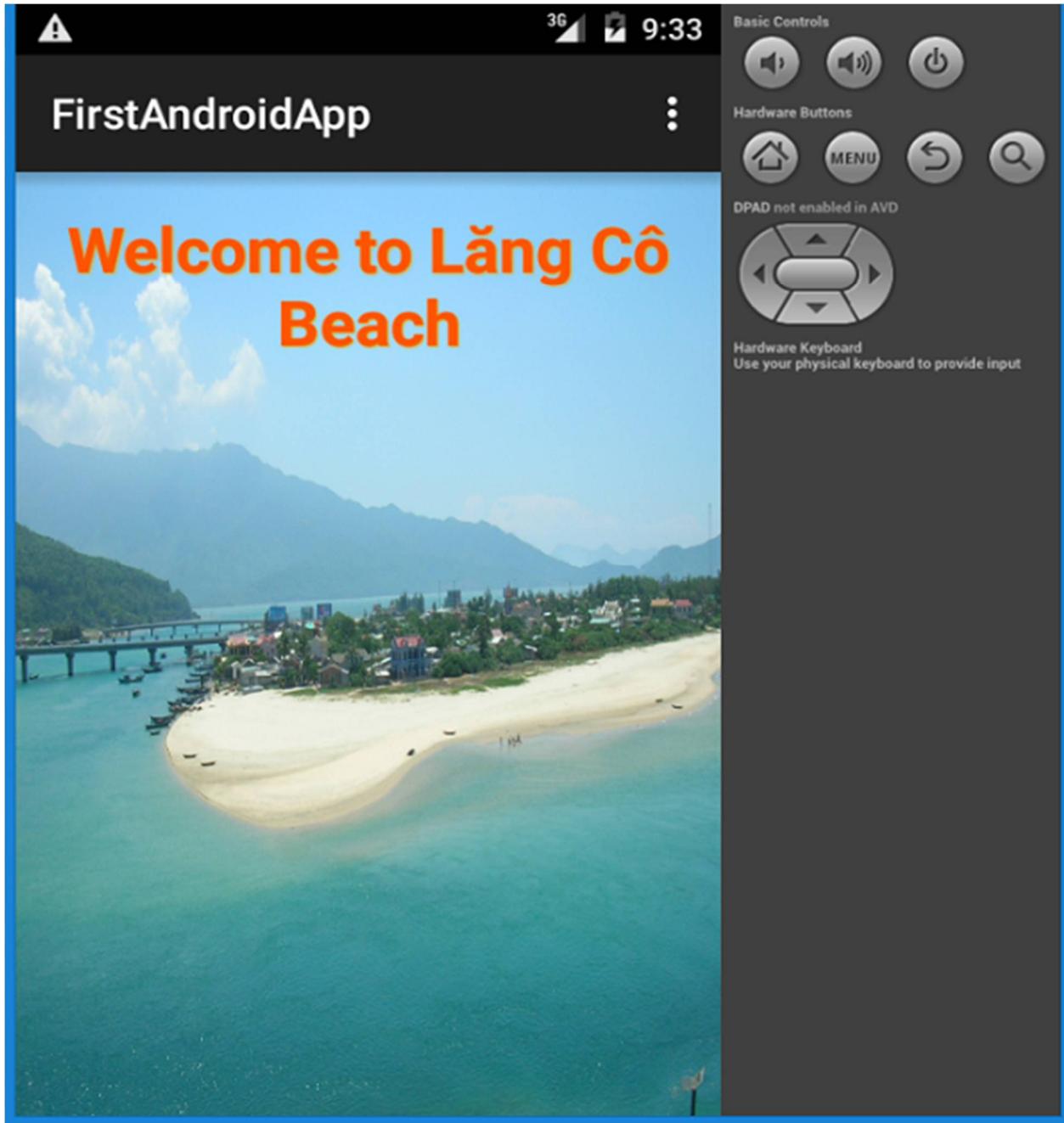
Hình 1.12: Import hình làm ảnh nền vào project

Khai báo hình nền cho LinearLayout như sau

<LinearLayout

```
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="@drawable/langco"  
    android:orientation="vertical"  
    tools:context=".FirstAppAndroidActivity">
```

Kết quả được ứng dụng như sau: (đã đổi nội dung của TextView là “Welcome to Lăng Cô Beach”



Hình 1.13: Giao diện của ứng dụng sau khi bổ sung hình nền

6. Bổ sung TextView hiển thị nội dung bên phải, dưới của layout, như hình minh họa sau
Để hiển thị được như vậy thì ở đây ta dùng dạng layout là RelativeLayout, với kiểu layout này thì các thành phần bên trong sẽ được đặt ở vị trí tương đối so với cha và các thành phần view bên trong.

Code bên dưới là phần mô tả layout trong main.xml

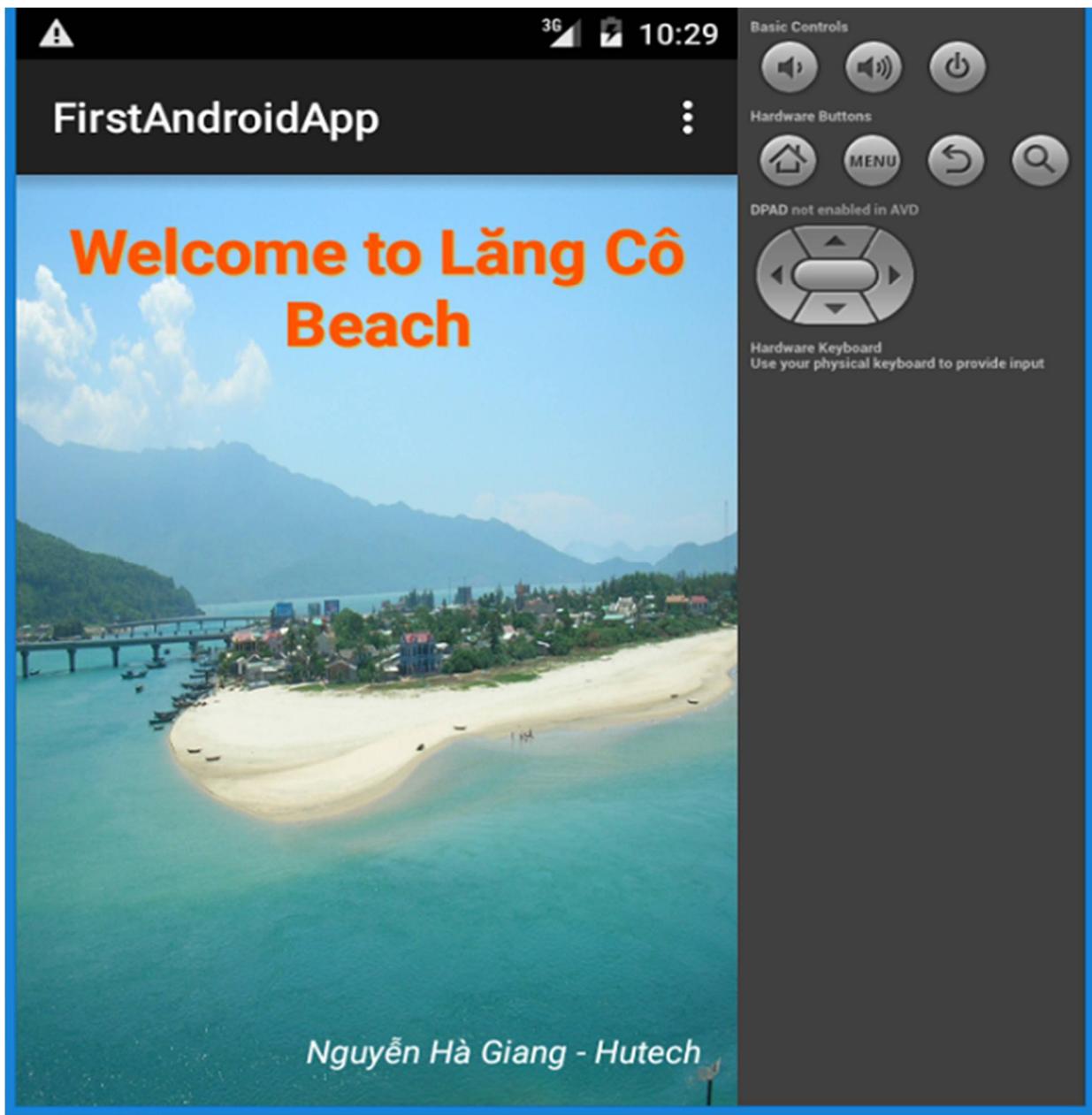
```
<RelativeLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:orientation="vertical"  
    tools:context=".FirstAppAndroidActivity"  
    android:background="@drawable/langco">
```

Sử dụng Relativelayout

```
<TextView  
    android:text="@string/hello_world"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:textSize="30sp"  
    android:textColor="@color/orange"  
    android:textStyle="bold"  
    android:gravity="center"  
    android:shadowColor="@color/gold"  
    android:shadowRadius="1.5"  
    android:shadowDx="1"  
    android:shadowDy="1"/>  
<TextView  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/copyright"  
    android:textSize="15dp"  
    android:textStyle="italic"  
    android:gravity="right"  
    android:layout_alignParentBottom="true"/>  
</RelativeLayout>
```

Canh phải và dưới so với parent

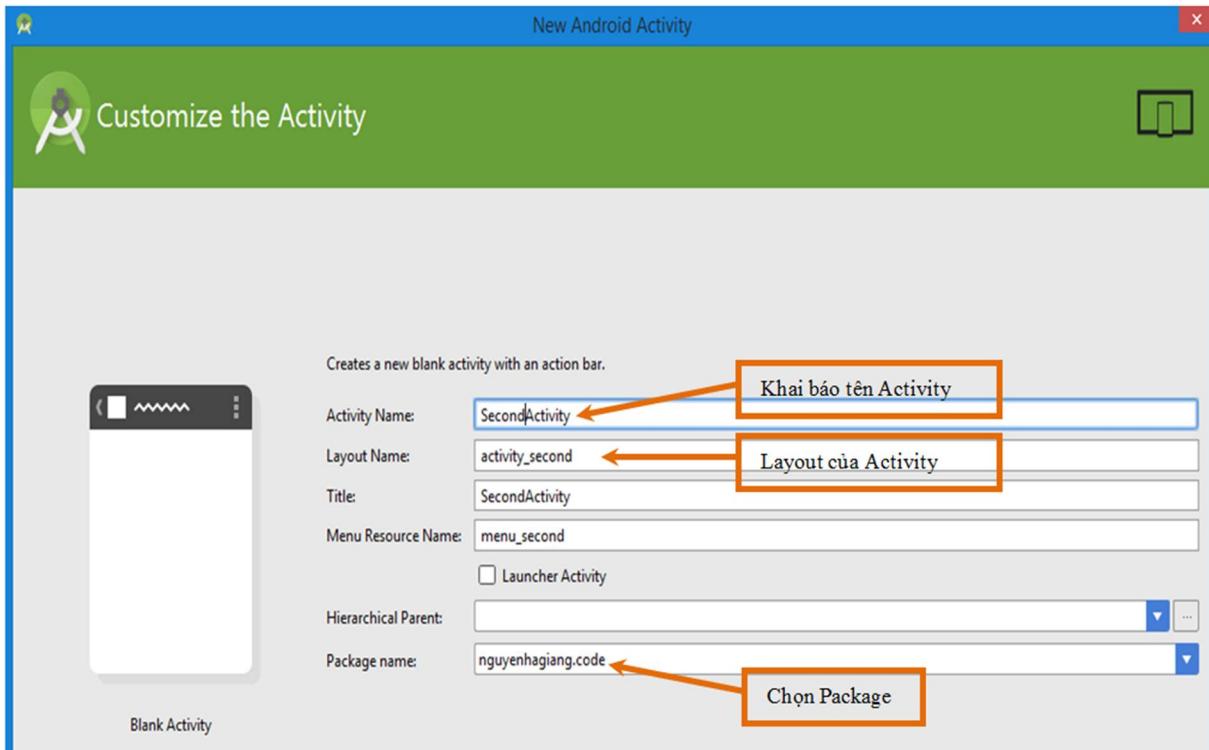
Hình 1.14: Phần layout sử dụng RelativeLayout.



Hình 1.15: Kết quả khi dùng RelativeLayout.

7. Bước 7: Minh họa tạo activity thứ 2 trong ứng dụng này, activity thứ 2 này có giao diện cho phép user nhập vào tên trong một EditText và sau đó kích vào button, ứng dụng sẽ xuất ra một cửa sổ nhỏ pop-up hiện câu chào.
 - Bước 7.1: Tạo một activity mới có tên SecondActivity: Chọn File ->New -> Activity ->Blank Activity hoặc kích chuột phải vào thư mục src của project chọn New ->Activity ->Blank Activity. Trong cửa sổ New Android Activity khai báo lớp Activity (ở đây lớp Activity mới này mặc định thừa kế Layout, Menu từ lớp Activity

trước, vì vậy bạn có thể thay đổi dạng layout là RelativeLayout hay LinearLayout trực tiếp trong mã lệnh của lớp layout đã được tạo)



Hình 1.16: Tạo mới lớp activity thứ hai trong ứng dụng

Lớp SecondActivity được phát sinh với source code như sau:

```
SecondActivity.java x activity_second.xml x
package nguyenhagiang.code;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

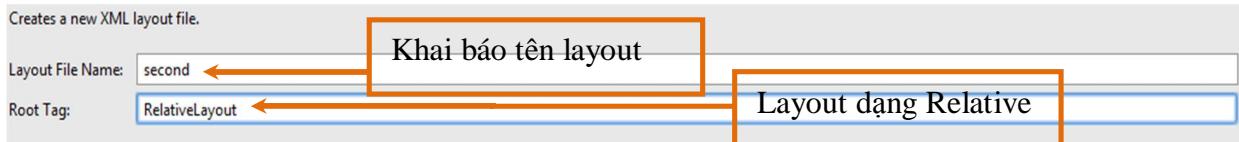
public class SecondActivity extends ActionBarActivity {

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

Hình 1.17: Source code của SecondActivity

- Ngoài ra có thể tạo file layout mới giống như đã được tạo ở hình trên chứa phần mô tả giao diện của SecondActivity: layout này là dạng Relative gồm có một EditText và một Button chứa bên trong. File layout này có tên là second.xml.

Cách tạo file second.xml như sau: kích chuột phải vào thư mục layout, chọn New -> XML -> Layout XML File



Hình 1.18: Tạo file XML layout cho SecondActivity

File second.xml ban đầu có mô tả như sau

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    </RelativeLayout>
```

- Bước 7.2: bổ sung EditText và Button vào second layout như mô tả sau

```
<EditText
    android:id="@+id/EditText01"
    android:hint="Nhập họ tên..."
    android:layout_alignParentLeft="true"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_toLeftOf="@+id/Button01" >
```

</EditText>

```
<Button
    android:id="@+id/Button01"
    android:text="Xin chào!"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```

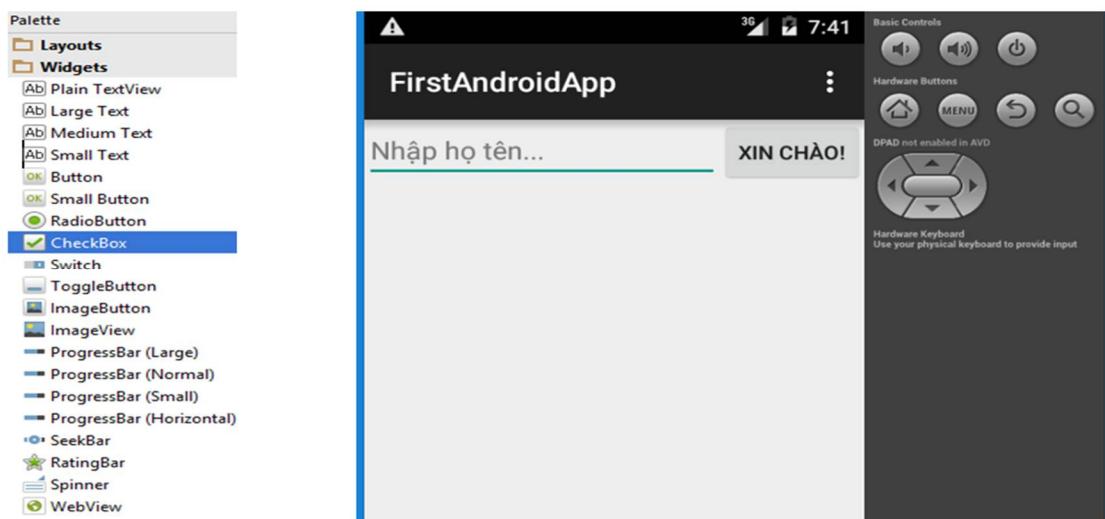
        android:layout_alignParentRight="true"
        android:onClick="showMe" >
    </Button>

```

Giải thích:

EditText	
android:id="@+id/EditText01"	Khai báo id của EditText
android:hint="Nhập họ tên..."	Xuất hiện khi nội dung empty
android:layout_alignParentLeft="true"	Canh lề trái với parent
android:layout_width="fill_parent"	Fill kích thước ngang
android:layout_height="wrap_content"	Wrap dọc
android:layout_toLeftOf="@+id/Button01"	Canh bên trái view có id là Button01
Button	
android:id="@+id/Button01"	Khai báo id của Button
android:text="Xin chào!"	Caption của Button
android:layout_width="wrap_content"	Wrap nội dung
android:layout_height="wrap_content"	Wrap nội dung
android:layout_alignParentRight="true"	Canh lề bên phải parent.
android:onClick="showMe"	Khai báo hàm xử lý sự kiện khi click

Chuyển qua Graphical layout để xem layout.



Hình 1.19: Graphical layout của activity SecondActivity

- Bước 7.3: định nghĩa hàm xử lý sự kiện click của button trong lớp activity (SecondActivity).

```
public void showMe(View v)
{
    String msg;
    // lấy tham chiếu đến EditText
    EditText et = (EditText)findViewById(R.id.EditText01);
    msg = "Xin chào " + et.getText().toString();
    // hiển thị message box
    Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_LONG).show();
}
```

- Bước 7.4: override phương thức onCreate của SecondActivity, trong phương thức onCreate load layout lên giao diện của activity.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);

    // load layout
    setContentView(R.layout.second);
}
```

- Bước 7.5: Cấu hình trong AndroidManifest.xml, khai báo activity mới và thiết lập để ứng dụng hiển thị activity thứ 2.

The diagram illustrates the movement of the intent-filter element from the first activity's declaration to the second activity's declaration in the AndroidManifest.xml file. A red arrow labeled "move" points from the original position of the intent-filter (inside the first activity's declaration) to its new position (inside the second activity's declaration). A red star marks the original location of the intent-filter, and a red box marks its new location.

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".FirstAppAndroidActivity"
        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="SecondActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

Hình 1.20: Bổ sung mô tả SecondActivity vào androidmanifest.xml

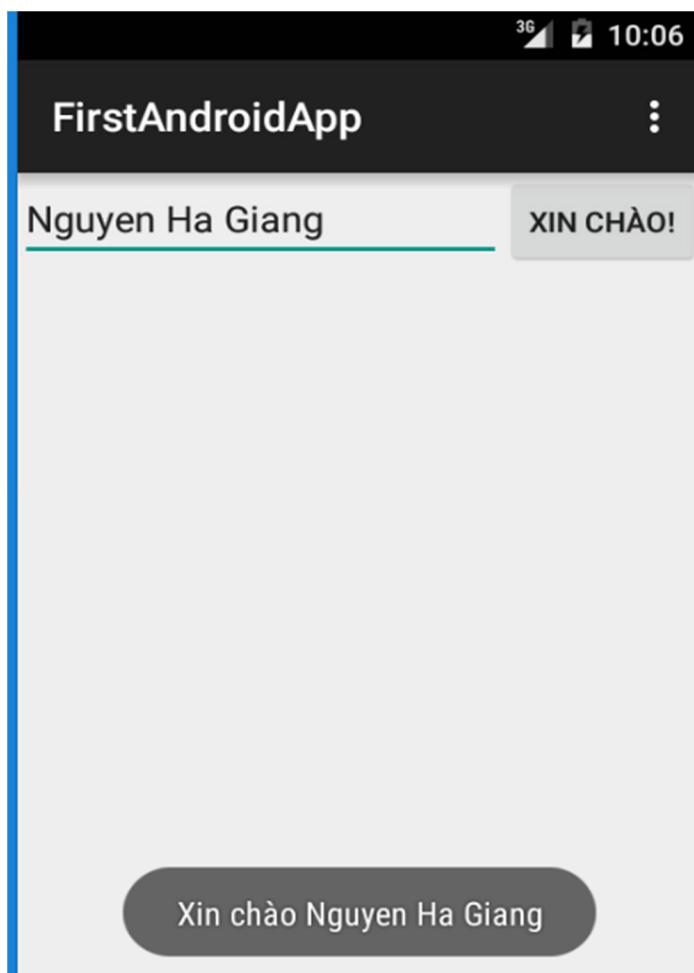
Chuyển thẻ intent-filter từ activity 1 xuống phần khai báo của activity thứ 2.

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".FirstAppAndroidActivity"
        android:label="@string/app_name">
    </activity>
    <activity android:name="SecondActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

Xem như entry point của app

Hình 1.21: Khai báo SecondActivity được hiển thị khi ứng dụng chạy

- Bước 7.6: biên dịch và chạy ứng dụng



Hình 1.22: Giao diện tương tác của ứng dụng với SecondActivity.

Mở rộng

- Viết lại ứng dụng trên không dùng XML để mô tả giao diện của các activity mà dùng code Java để thực hiện.
- Tạo một activity có giao diện như sau:



Hình 1.23: Giao diện activity

Trong đó các màu được định nghĩa như sau:

```
<resources>
    <color name="red">#f00</color>
    <color name="orange">#ffa500</color>
    <color name="yellow">#ffff00</color>
    <color name="green">#0f0</color>
    <color name="blue">#00f</color>
    <color name="indigo">#4b0082</color>
    <color name="violet">#ee82ee</color>
    <color name="back">#000</color>
    <color name="white">#fff</color>
```

</resources>

3. Viết ứng dụng đơn giản cho phép user nhập vào hai số và chọn một trong các phép toán {+,-,*,/} để thực hiện, chương trình tính kết quả và hiển thị lên màn hình.



Hình 1.24: Giao diện ứng dụng basic calc

Hướng dẫn: sử dụng widget Spinner (tương tự như thành phần combobox quen thuộc), Spinner này có thuộc tính entries lấy danh sách chuỗi để làm mục chọn, danh sách chuỗi này được định nghĩa là mảng chuỗi: <string-array> trong strings.xml.



Hình 1.25: Mô tả cách sử dụng Spinner

=oOo=



Lab 2: Sử dụng Intent

Mục tiêu

- ✚ Làm quen với cách dùng cơ chế Intent để thực hiện các yêu cầu
 - ❖ Gọi hiển thị activity từ trong activity đang làm việc
- ✚ Sử dụng AlertDialog.Builder cho phép hỏi đáp với người dùng.
- ✚ Truyền dữ liệu từ sub activity về activity cha.

Yêu cầu

- ✚ Có kiến thức cơ bản, trong việc xây dựng ứng dụng Android, tạo activity từ XML layout, khai báo và viết phần xử lý sự kiện trong code Java.
- ✚ Hiểu qua cơ chế Intent cơ bản trong lập trình Android.

Nội dung

- ✚ Tạo ứng dụng notepad đơn giản có giao diện và chức năng như hình sau:
Ứng dụng cho phép user nhập đoạn văn bản trên nhiều dòng vào một EditText ở chế độ TextMultiline. Ngoài ra ứng dụng cung cấp một menu cho phép user chọn các chức năng như sau:
 - **Clear**: xoá toàn bộ nội dung đã nhập
 - Hiển thị thông báo sẽ xoá nội dung và sau đó thực hiện việc xoá.
 - **Setting**: thiết lập màu sắc và font size
 - Hiển thị activity option để user chọn các thiết lập. sau đó các thiết lập này sẽ có hiệu lực.
 - **Exit**: thoát khỏi ứng dụng.
 - Hiển thị form xác nhận xem user có muốn thoát khỏi ứng dụng hay không.



Hình 2.1: Minh họa chức năng menu

Hướng dẫn

1. Bước 1:

- a. Tạo ứng dụng Android.
- b. Thiết kế phần layout cho activity chính của ứng dụng có mô tả như sau:

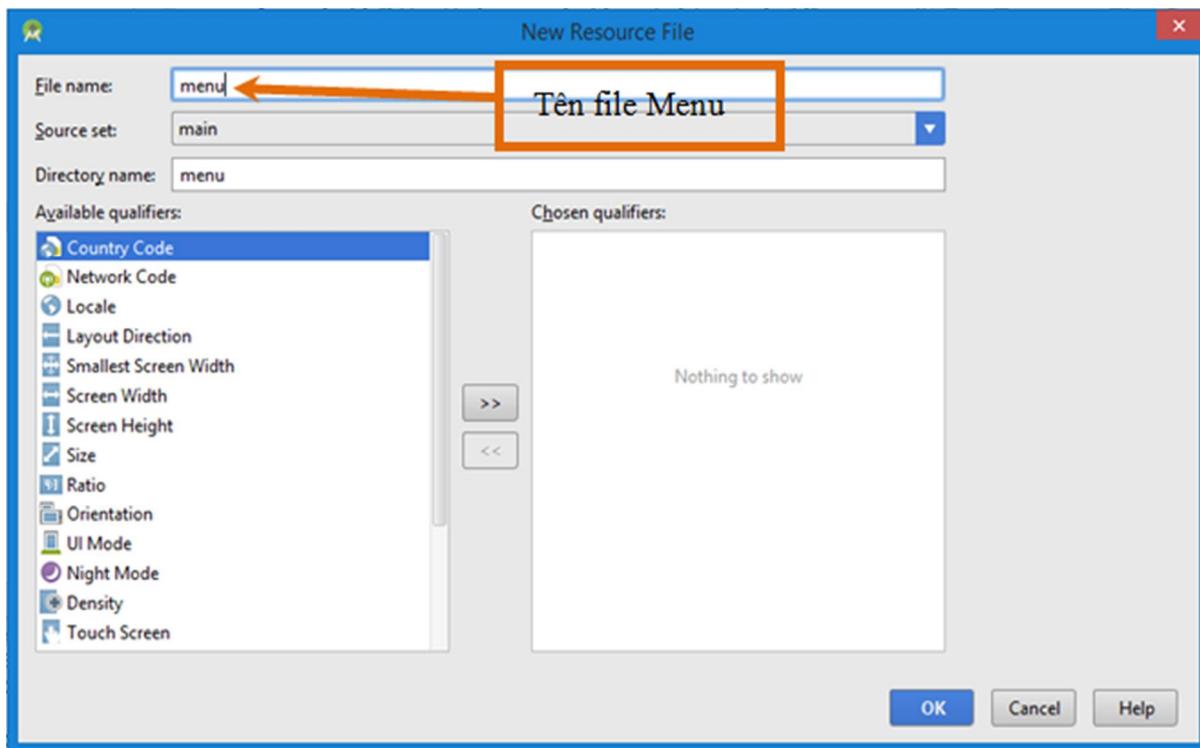
```
<LinearLayout>
    <TextView>
        <ScrollView>
            <EditText>
        </ScrollView>
    </LinearLayout>
```

Trong layout này thì LinearLayout là gốc chứa 2 thành phần bên trong là TextView chứa chuỗi “Input note here” và một scrollview (cho phép xuất hiện thanh cuộn ở thành phần bên trong)

Bên trong của ScrollView là EditText. EditText này được thiết lập ở dạng Multi Line, do chứa bên trong ScrollView nên nội dung của nó có thể vượt quá kích thước, khi đó xuất hiện thanh cuộn để cuộn lên, xuống xem dữ liệu.

2. Bước 2: tạo menu cho ứng dụng

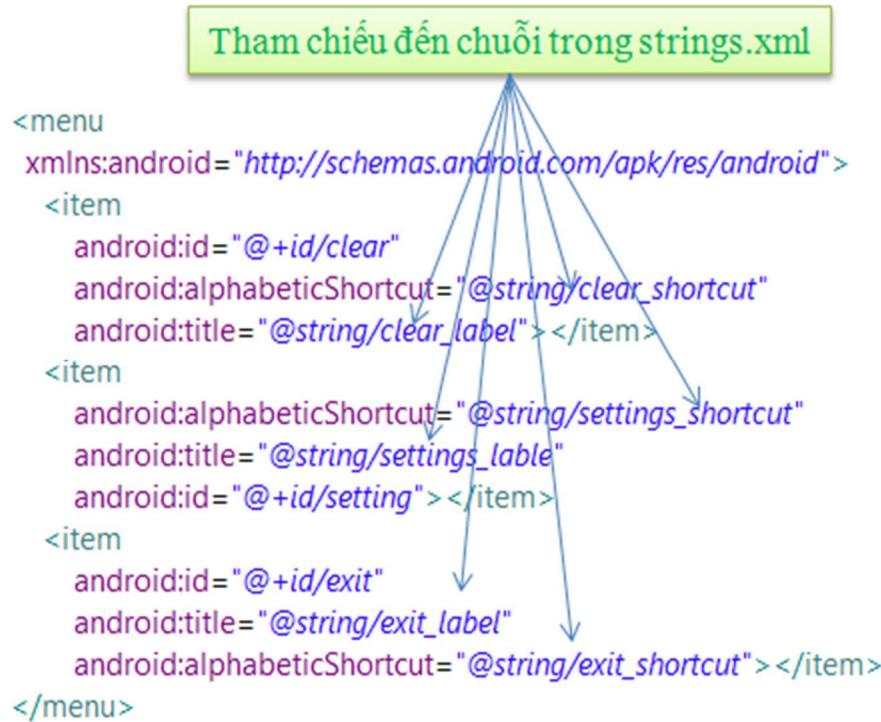
- Tạo file resource mô tả menu: chuột phải vào menu trong res -> chọn new ->Menu resource file.



Hình 2.2: Minh họa tạo XML file menu

- Tạo các item trong menu: gồm 3 item {clear, setting, exit}

Mỗi item có các thuộc tính như id (để code tham chiếu đến), title item (caption), và thuộc tính alphabeticShortcut (mô tả phím tắt).



- c. Load menu trong activity chính của ứng dụng.

Menu của ứng dụng được tạo từ phương thức onCreateOptionsMenu(Menu menu), do đó trong lớp activity ta phải override phương thức này.

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    // lấy menu ngữ cảnh của ứng dụng
    MenuInflater inflater = getMenuInflater();
    // thiết lập menu
    inflater.inflate(R.menu.menu, menu);
    return true;
}
  
```

- d. Khai báo phần xử lý mỗi khi user kích vào các mục chọn trên menu, phần code này chưa hoàn thiện, vì các phần xử lý của các item sẽ mô tả sau.

Phương thức onOptionsItemSelected được gọi mỗi khi có item trong menu được chọn.

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {

    super.onOptionsItemSelected(item);
    switch (item.getItemId()) {
        case R.id.clear:
            // viết phần xoá ở đây

        break;
        case R.id.setting:
            // phần setting

        break;
        case R.id.exit:
            // thoát ứng dụng
        break;
    } //end switch
    return true;
} // end onOptionsItemSelected

```

3. **Bước 3:** viết phần xử lý cho mục chọn “Clear”. Khi user chọn chức năng này thì ứng dụng sẽ hiển thị thông báo là user đã chọn chức năng xoá text, sau đó sẽ thực hiện xoá luôn.

```

case R.id.clear:
// viết phần xoá ở đây
AlertDialog.Builder message = new AlertDialog.Builder(this);
message.setTitle(R.string.message_caption);
message.setMessage(R.string.message_content);
message.setNeutralButton(R.string.close, new
    DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            EditText et = (EditText) findViewById(R.id.editText1);
            et.setText(""); // xoá nội dung edittext
        }
    }).show();
break;

```



Hình 2.3: Minh họa hiển thị thông báo xóa

4. **Bước 4:** viết chức năng xử lý cho mục chọn “exit”. Khi user chọn chức năng này thì ứng dụng sẽ xuất hiện dialog yêu cầu user xác nhận xem có muốn thoát ứng dụng hay không, nếu user chọn không thì sẽ quay lại ứng dụng, ngược lại sẽ thoát.

```
case R.id.exit:  
    // thoát ứng dụng  
    new AlertDialog.Builder(this)  
        .setTitle(R.string.exit_caption)  
        .setMessage(R.string.exit_content)  
        .setNegativeButton(R.string.yes, new DialogInterface.OnClickListener() {  
            @Override  
            public void onClick(DialogInterface dialog, int which) {  
                //thoát khỏi ứng dụng  
                finish();  
            }  
        })  
        .setPositiveButton(R.string.no, new DialogInterface.OnClickListener() {  
            @Override  
            public void onClick(DialogInterface dialog, int which) {  
                // TODO Auto-generated method stub  
                return;  
            }  
        })  
    .show();
```

```
    }  
}  
.show();  
break;
```



Hình 2.4: Minh họa hiển thị thông báo thoát

5. Bước 5: tạo activity chứa option có layout như sau:

```
<TableLayout  
    android:id="@+id/tableLayout1"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:stretchColumns="2">  
    <TableRow>  
        <TextView  
            android:layout_height="wrap_content"  
            android:id="@+id/textView1"  
            android:textAppearance="?android:attr/textAppearanceMedium"  
            android:text="@string/forecolor"  
            android:layout_width="wrap_content">  
        </TextView>
```

```
<Spinner
    android:id="@+id/spinner1"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:layout_span="2"
    android:entries="@array/color_name_array"></Spinner>
</TableRow>
<TableRow >

<TextView
    android:layout_height="wrap_content"
    android:id="@+id/textView2"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="@string/backcolor"
    android:layout_width="wrap_content">
</TextView>
<Spinner
    android:id="@+id/spinner2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_span="2"
    android:entries="@array/color_name_array">
</Spinner>
</TableRow>
<TableRow >

    android:id="@+id/tableRow3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
        <Button
            android:text="OK"
            android:layout_height="wrap_content"
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_column="2"
            android:onClick="onOK">
        </Button>
</TableRow>
```

```
</TableLayout>
```

Trong đó mảng color_name_array được định nghĩa trong strings.xml như sau

```
<string-array name="color_name_array">
    <item name="red">Red</item>
    <item name="orange">Orange</item>
    <item name="yellow">Yellow</item>
    <item name="green">Green</item>
    <item name="blue">Blue</item>
    <item name="indigo">Indigo</item>
    <item name="violet">Violet</item>
</string-array>
```

Thiết lập layout cho activity option

```
public class optionActivity extends Activity {
    // tạo 2 biến private chứa index màu mà user chọn
    private int index1=0, index2=0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);

        setContentView(R.layout.option);
        Spinner spinner1 = (Spinner)findViewById(R.id.spinner1);
        spinner1.setOnItemSelectedListener(new OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1,
                    int arg2, long arg3) {
                index1 = arg2; // lấy index user chọn
            }
            @Override
            public void onNothingSelected(AdapterView<?> arg0) {
        });

        Spinner spinner2 = (Spinner)findViewById(R.id.spinner2);
        spinner2.setOnItemSelectedListener(new OnItemSelectedListener() {
```

```

@Override
public void onItemSelected(AdapterView<?> arg0, View arg1,
    int arg2, long arg3) {
    index2 = arg2; // lấy index user chọn
}

@Override
public void onNothingSelected(AdapterView<?> arg0) { }
);
}

```

6. **Bước 6:** viết phần xử lý cho button OK trong activity option. Phần xử lý này có chức năng lấy hai index màu mà user chọn sau đó **gởi lại cho activity trước đó gọi nó.**

```

public void onOK(View view) {
    // gửi dữ liệu về activity trước
    Intent intent = new Intent();
    Bundle bundle = new Bundle();

    bundle.putInt("ForeColor", index1); // lấy giá trị màu text
    bundle.putInt("BackColor", index2); // lấy giá trị màu nền
    intent.putExtras(bundle); // gửi kèm dữ liệu
    setResult(RESULT_OK, intent); // gửi kết quả về
    finish(); // đóng activity
}

```

7. **Bước 7:** gọi hiển thị activity option khi chọn chức năng option **trong activity chính của ứng dụng.**

```

//Phần xử lý tiếp tục trong phần onOptionsItemSelected của menu chính.
case R.id.setting:
    // phần setting
    Intent intent = new Intent(this,optionActivity.class);
    final int result=1;
    // khởi tạo activity có lấy kết quả về
    startActivityForResult(intent, result);
    break;

```

8. **Bước 8:** Viết phần xử lý khi activity option trả kết quả về.

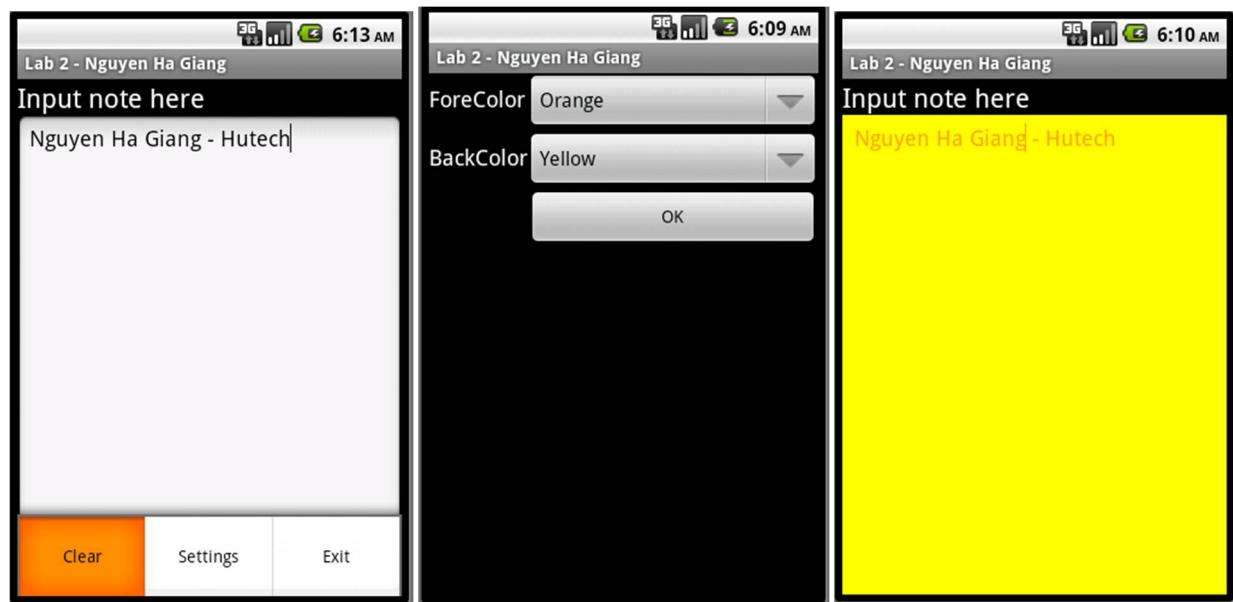
Để thực hiện điều này ta override phương thức onActivityResult trong activity chính của ứng dụng.

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
  
    super.onActivityResult(requestCode, resultCode, data);  
  
    // lấy Bundle dữ liệu  
    Bundle bundle = data.getExtras();  
    int index1 = bundle.getInt("ForeColor");  
    int index2 = bundle.getInt("BackColor");  
  
    // lấy mảng màu  
    String colorArray[] = getResources().getStringArray(R.array.color_array);  
    // tham chiếu đến editText  
    EditText et = (EditText)findViewById(R.id.editText1);  
    // thiết lập màu  
    et.setTextColor(Color.parseColor(colorArray[index1]));  
    et.setBackgroundColor(Color.parseColor(colorArray[index2]));  
}
```

Trong đó mảng màu được khai báo trong strings.xml như sau

```
<resources>  
    <string-array name="color_array">  
        <item>#f00</item>  
        <item>#ffa500</item>  
        <item>#ffff00</item>  
        <item>#0f0</item>  
        <item>#00f</item>  
        <item>#4b0082</item>  
        <item>#ee82ee</item>  
    </string-array>  
</resources>
```

9. Bước 9: Biên dịch và chạy ứng dụng trên emulator



Hình 2.5: Minh họa ứng dụng sau khi hoàn thành

Phần mở rộng:

- Bổ sung thêm các chức năng option khác như kích thước của text.
- Sinh viên lưu ý trong ứng dụng trên các chuỗi hoàn toàn được định nghĩa trước trong strings.xml. Hãy tìm hiểu tính năng Localization để có thể thay đổi nội dung các chuỗi theo ngôn ngữ được chọn.

=oOo=



Lab 3: Localization

Mục tiêu

- ✚ Tạo ứng dụng có hỗ trợ tính năng như:
 - ❖ Thay đổi ngôn ngữ hiển thị của ứng dụng theo setting language của thiết bị. Ví dụ nếu ứng dụng hỗ trợ 3 loại ngôn ngữ: Anh, Pháp, Tây Ban Nha. Khi thiết bị thiết lập ngôn ngữ nào thì ứng dụng sẽ hiển thị tương ứng theo ngôn ngữ đó.
 - ❖ Hỗ trợ hai kiểu màn hình là portrait và landscape.

Yêu cầu

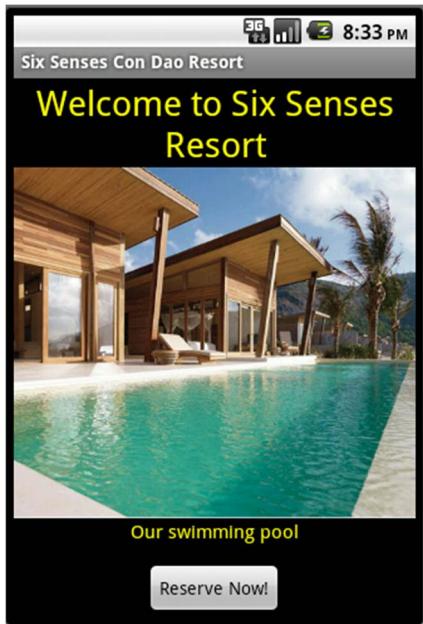
- ✚ Có kiến thức cơ bản về xây dựng ứng dụng Android.
- ✚ Sử dụng được các thành phần widget của Android.
- ✚ Quen thuộc với các dạng layout của Android: LinearLayout, RelativeLayout, TableLayout...

Nội dung

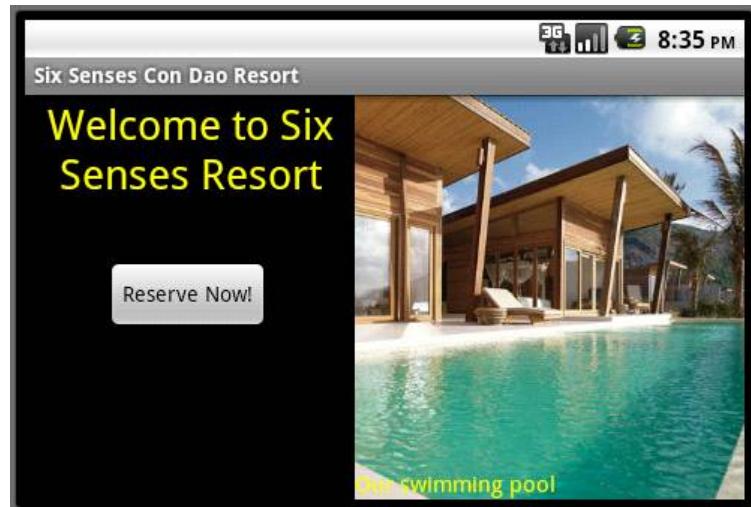
Tạo ứng dụng có màn hình giới thiệu resort, có giao diện cơ bản như hình 3.1 (ứng dụng này chỉ minh họa tính localization không có phần xử lý như book phòng...)

Hình 3.1a là giao diện của ứng dụng với orientation là portrait, hình 3.1b là giao diện của ứng dụng với orientation là landscape.

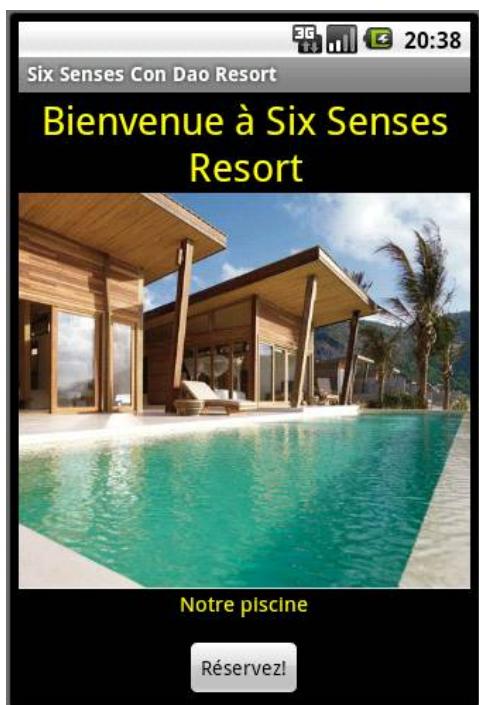
Hình 3.1c là giao diện portrait của ứng dụng khi thiết bị chuyển sang sử dụng ngôn ngữ tiếng Pháp. Để thực hiện việc thay đổi ta dùng Settings ⇒ Language & Keyboard settings ⇒ Select Language ⇒ chọn ngôn ngữ cho máy. Tương tự như vậy hình 3.1d là giao diện landscape của ứng dụng với ngôn ngữ tiếng Pháp.



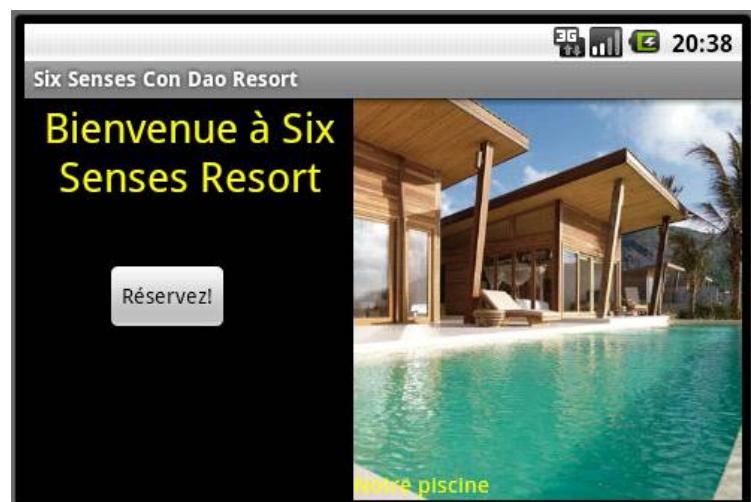
Hình 3.1a: Dạng Portrait, Tiếng Anh



Hình 3.1b: Dạng Landscape, Tiếng Anh



Hình 3.1c: Dạng Portrait, Tiếng Pháp



Hình 3.1d: Dạng Landscape, Tiếng Pháp

Hướng dẫn

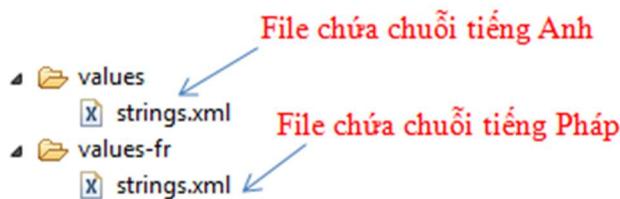
- Tạo các chuỗi trong strings.xml hỗ trợ ngôn ngữ Anh, Pháp. Do mặc định là ngôn ngữ Anh, nên file strings.xml trong thư mục res/values được dùng để thể hiện tiếng Anh và có phần khai báo các chuỗi như sau.

```
<resources>
    <string name="app_name">Six Senses Con Dao Resort</string>
    <string name="welcome">Welcome to Six Senses Resort</string>
    <string name="pool">Our swimming pool</string>
    <string name="reserve">Reserve Now!</string>
</resources>
```

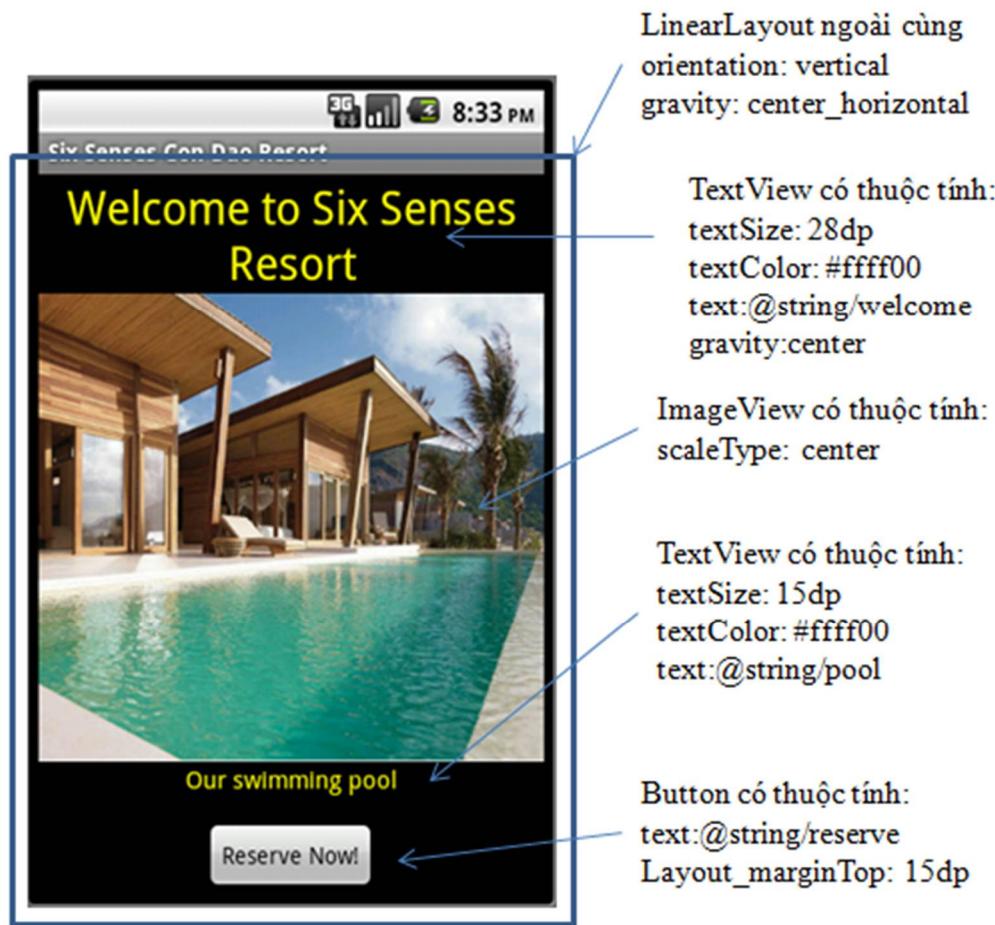
Để hỗ trợ ngôn ngữ tiếng Pháp thì ta sẽ tạo một file tương tự cũng có tên strings.xml có nội dung như sau:

```
<resources>
    <string name="app_name">Six Senses Con Dao Resort</string>
    <string name="welcome">Bienvenue à Six Senses Resort</string>
    <string name="pool">Notre piscine</string>
    <string name="reserve">Réservez!</string>
</resources>
```

Ta nhận thấy file này chỉ khác phần dữ liệu của chuỗi là chứa ngôn ngữ tiếng Pháp. **File này đặt bên trong thư mục res/values-fr** (thư mục này ta phải tự tạo ra)



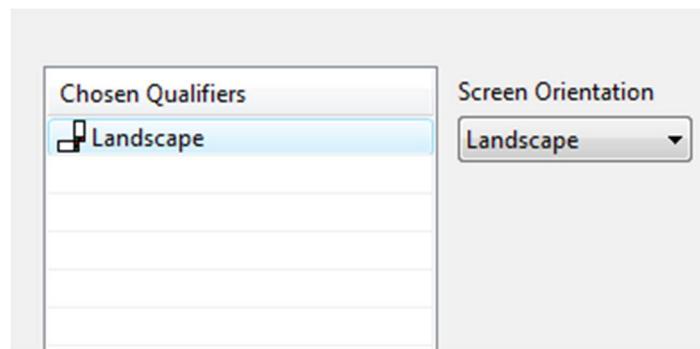
- Hỗ trợ tính năng xoay của thiết bị, từ dạng portrait sang landscape và ngược lại. Với dạng portrait thì mặc định là layout res/layout/main.xml, với layout dạng đứng này thì ta thiết kế ứng dụng có dạng như hình 3.2.



Hình 3.2: layout ở dạng portrait

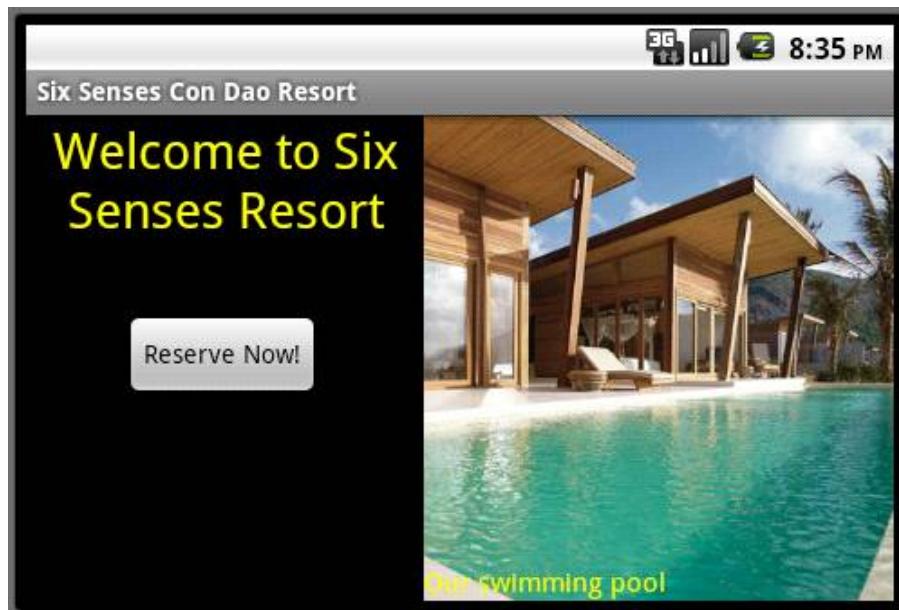
Để tạo layout ở dạng landscape thì ta làm theo các bước sau:

1. Tạo thư mục layout-land trong thư mục res.
2. Trong thư mục layout-land ta tạo file layout có tên tương ứng với file layout trong portrait là main.xml (nhớ kỹ tên phải trùng với tên layout ở dạng portrait, để khi load lên tương ứng cho activity). Để tạo dạng layout landscape thì trong phần New Android resource directory, thiết lập Orientation là landscape như hình vẽ.



Hình 3.3: Chọn Chosen Qualifiers là Landscape.

3. Thiết kế layout Landscape tương ứng như hình 3.4.



Hình 3.4: Layout dạng Landscape

Trong layout này ta có thể dùng RelativeLayout để thiết kế, phần này sinh viên tự làm.

- Biên dịch và chạy trên Emulator, chuyển đổi giữa landscape và portrait, chuyển đổi ngôn ngữ Anh, Pháp và xem kết quả.

Bài tập

- Sinh viên hỗ trợ thêm các ngôn ngữ như Đức và Tây Ban Nha. Để chuyển các chuỗi sang ngôn ngữ khác ta có thể dùng Google Translate để thực hiện ☺!

2. Mở rộng ứng dụng trên có phần xử lý Book phòng đơn giản: tạo thêm 2 activity, một là activity book phòng và 1 là activity hiển thị kết quả book phòng. Trong đó activity book phòng được gọi khi chọn chức năng Reserve Now ở activity trên, activity book phòng cho phép user nhập vào các thông tin để book (các thông tin book phòng tùy ý cho sinh viên). Sau khi nhập xong thì user chọn button Book, lúc này thông tin sẽ được chuyển sang activity hiển thị thông tin book phòng!.

=oOo=



Lab 4:

Lunch List Application Version 1 (*)

(*): Tham khảo từ *Android™ Programming Tutorials*, version 3.2 for Android 3.0, Mark L. Murphy, 2011, CommonsWare

Mục tiêu

- ✚ Xây dựng ứng dụng với các widget cơ bản trong activity
- ✚ Sử dụng giao diện Design View trong Android Studio để tạo dựng layout cho activity.
- ✚ Xây dựng lớp đối tượng chứa dữ liệu.

Yêu cầu

- ✚ Có kiến thức cơ bản về xây dựng activity và layout XML trong phần code view (markup view).
- ✚ Hiểu rõ cơ chế bind xử lý sự kiện của Button trên giao diện với phần code xử lý trong lớp activity.

Nội dung

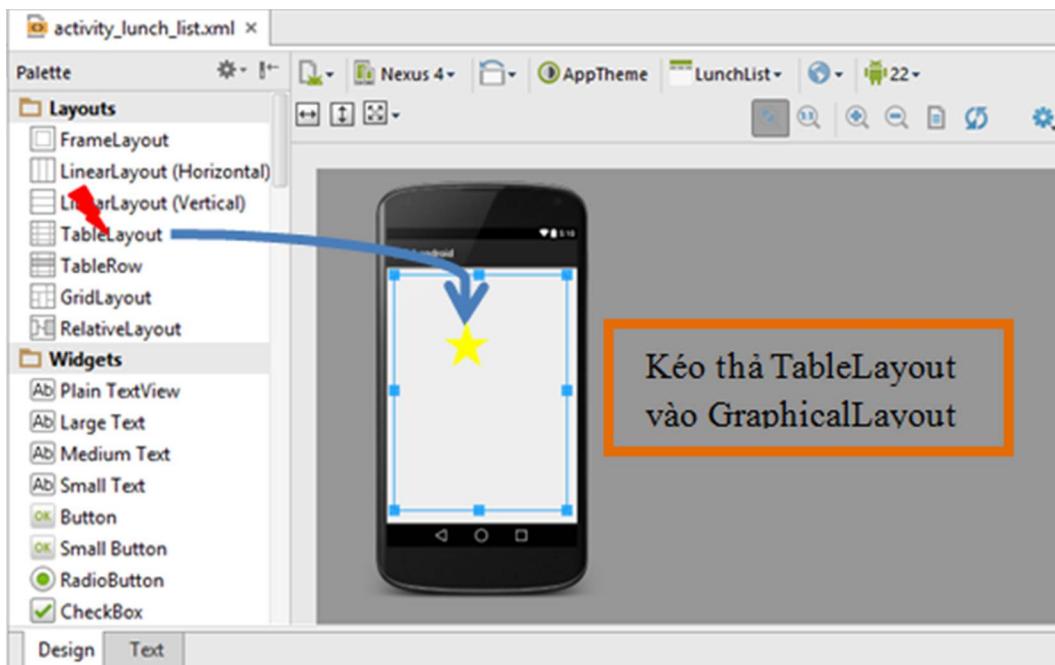
- ✚ **Bước 1:** Sử dụng Graphical Layout để xây dựng các thành phần trên layout của activity.
Trước khi thao tác thì xóa trống nội dung của file layout tương ứng.
 - ❖ Thêm TableLayout từ màn hình thiết kế vào layout. Từ folder Layouts ta kéo TableLayout thả vào vùng thiết kế đồ họa như hình 4.1.
 - ❖ Thêm các TableRow vào TableLayout: trong màn hình thiết kế chọn TableLayout vừa tạo. Sau đó chọn chức năng Add Table Row, có thể thêm phần cách viết trong file XML và dùng Graphical Layout (xem hình 4.2). Trong phần này ta sẽ thêm 2 TableRow vào TableLayout.
- Phần markup của file layout sau bước trên.

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1" >
```

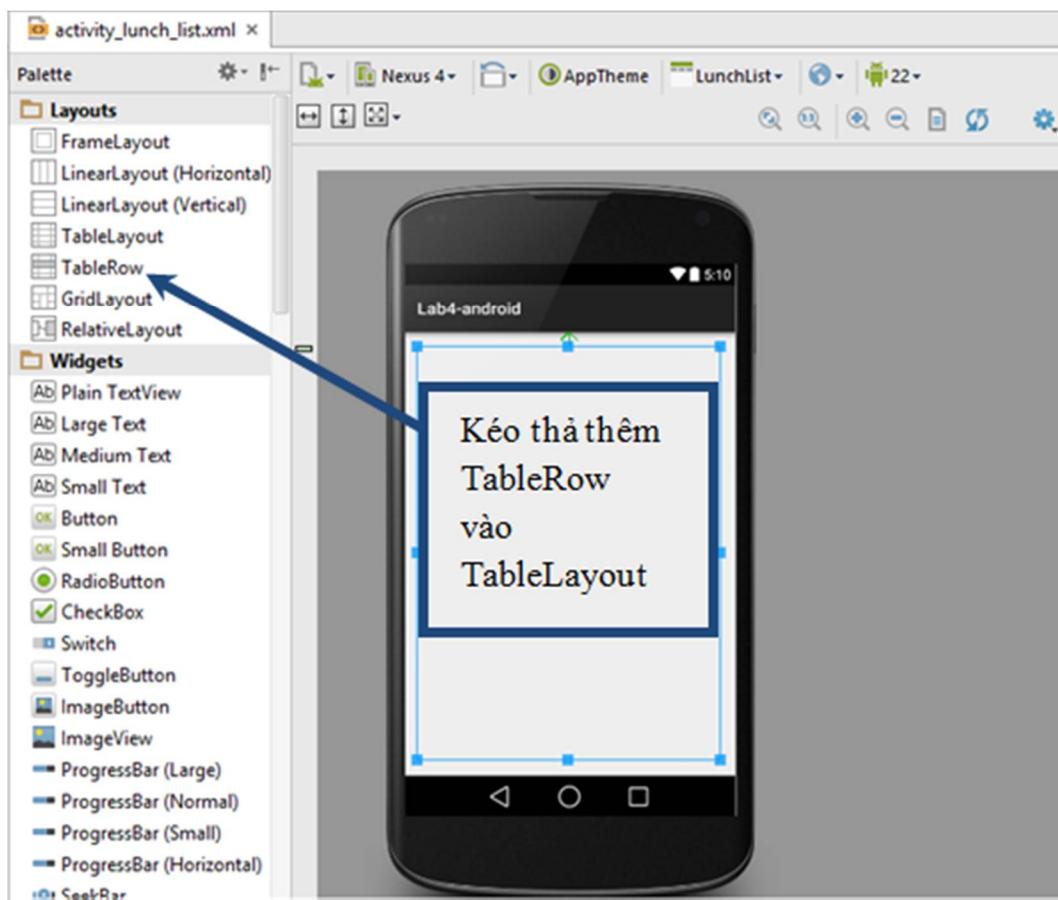
```
<TableRow
```

```
    android:id="@+id/tableRow1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content">>  
  
</TableRow>  
  
<TableRow  
    android:id="@+id/tableRow2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content">  
</TableRow>  
  
</TableLayout>
```

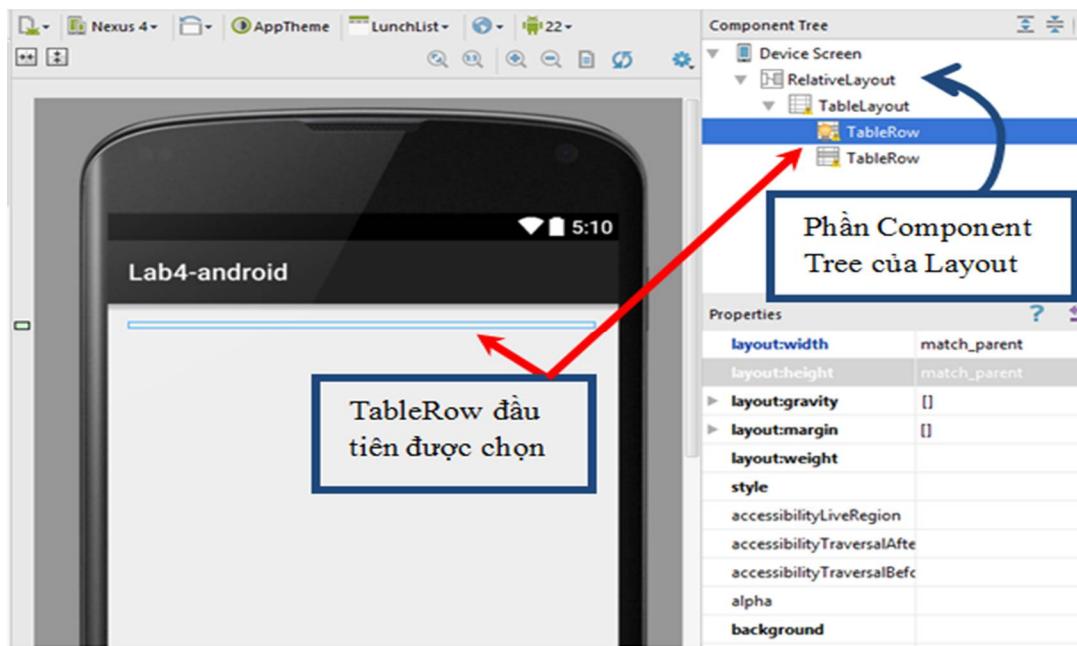
- ❖ Thêm các thành phần vào các TableRow. Trong phần này ta sẽ thêm vào một TextView và EditText cho TableRow đầu tiên. Trên màn hình Graphical Layout chọn TableRow đầu tiên trong TableLayout bằng cách dùng cửa sổ Outline để chọn, thao tác chọn như hình 4.3 minh họa.



Hình 4.1: Tạo TableLayout cho layout của activity.

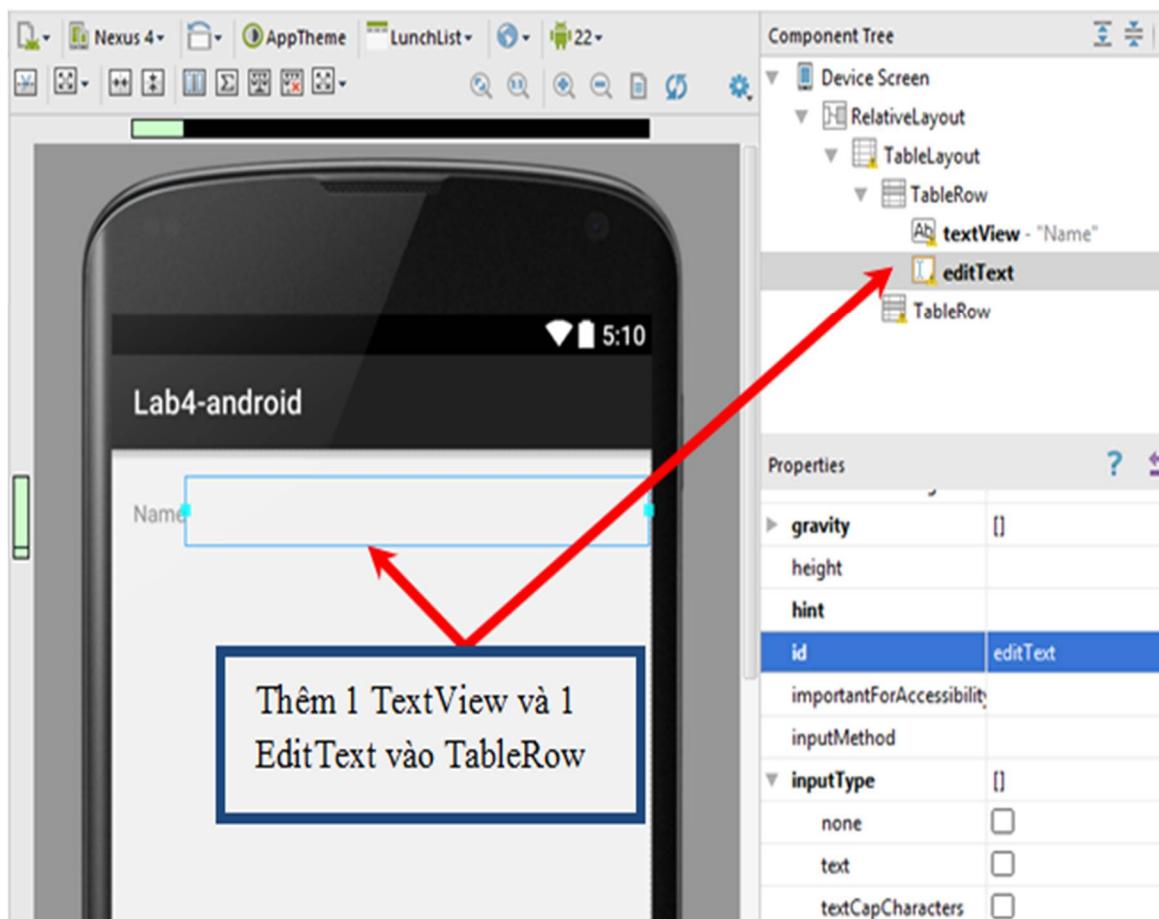


Hình 4.2: Thêm TableRow vào TableLayout.



Hình 4.3: chọn một TableRow trên TableLayout để thao tác.

Kéo một TextView và một EditText vào TableRow đầu tiên. Thay đổi thuộc tính Text của TextView là Name: và đặt id của EditText là name (Hình 4.4).

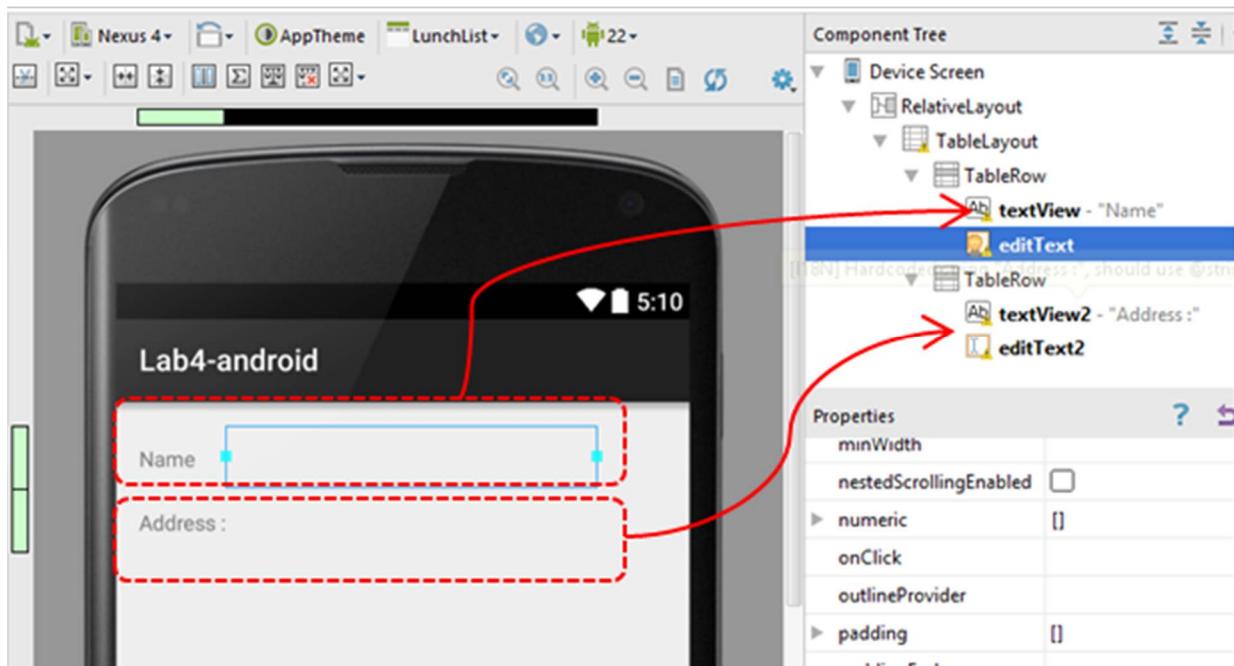


Hình 4.4: TextView và EditText trên TableRow1.

Làm tương tự với TableRow thứ 2:

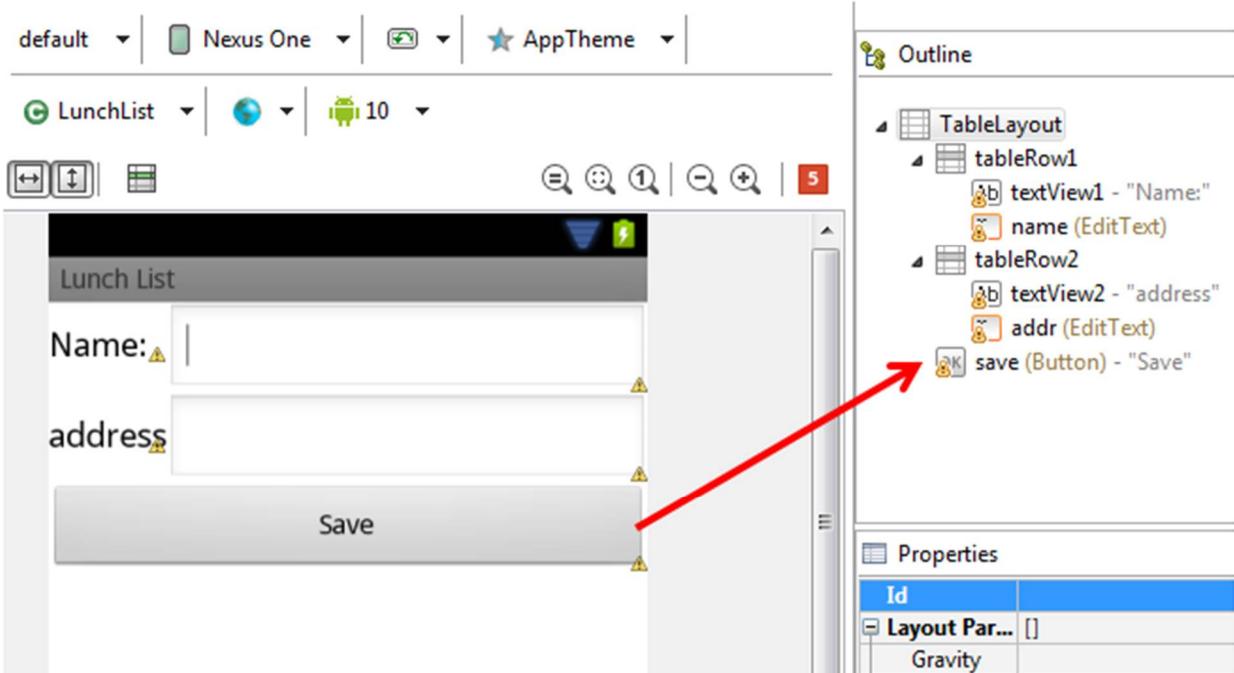
- Tạo một TextView có Text là “Address”.
- Tạo một EditText có id là addr

Xem hình 4.5.



Hình 4.5: Các thành phần trên TableLayout.

- ❖ Thêm 1 button vào TableLayout: trong Component tree chọn TableLayout và trong hộp công cụ Form Widgets kéo Button thả vào TableLayout. Khai báo thuộc tính Text của Button là “Save” và id là “save”.



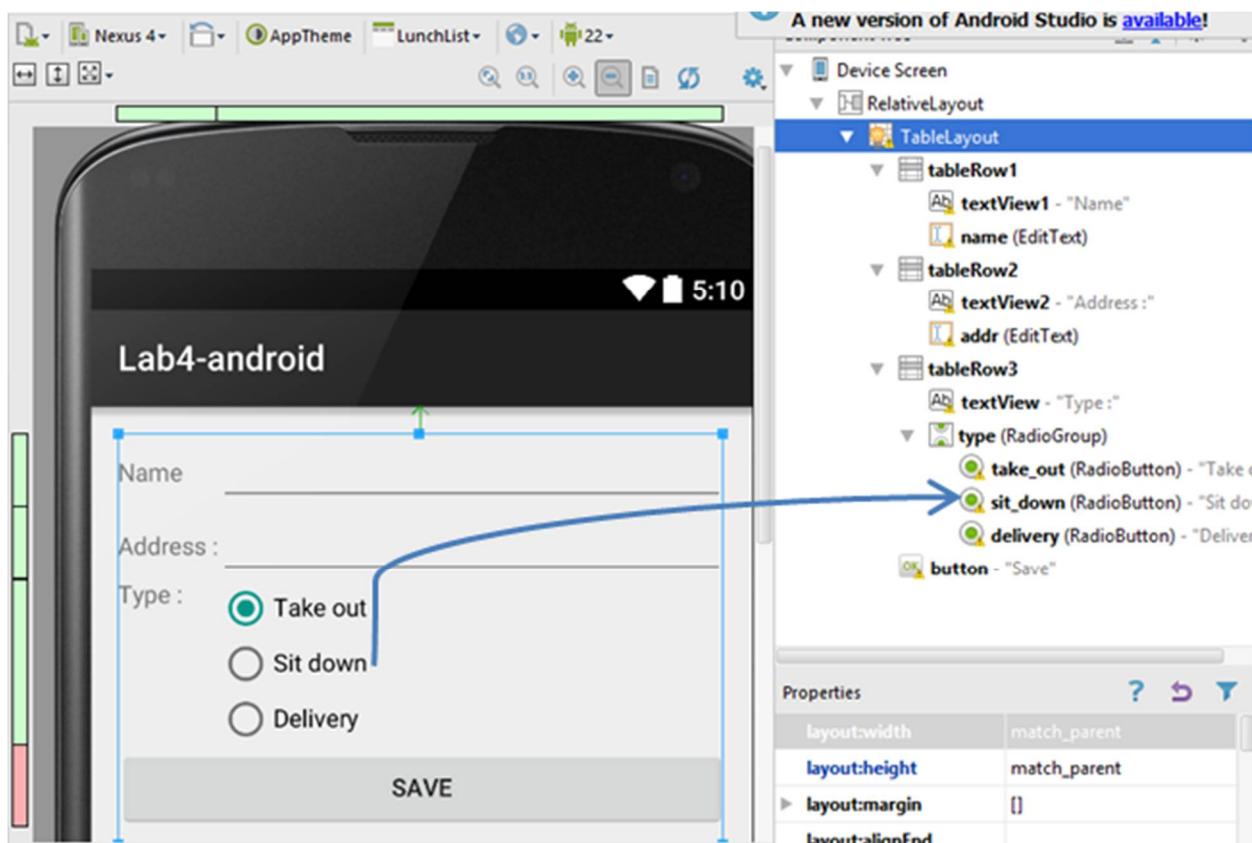
Hình 4.6: Layout sau khi thêm Button Save.

❖ Thêm RadioGroup: Trong phần này ta sẽ bổ sung thêm các RadioButton để cho biết loại nhà hàng: {take out, sit down, delivery}.

Tạo thêm một TableRow vào TableLayout, trong TableRow thứ 3 này ta bổ sung một TextView {Text: "Type:"} và một RadioGroup {id: @+id/types}.

Trong RadioGroup này gồm có 3 RadioButton con.

- RadioButton thứ nhất: {id:@+id/take_out; Text: "Take out"}
- RadioButton thứ hai: {id:@+id/sit_down; Text: "Sit down"}
- RadioButton thứ ba: {id:@+id/delivery; Text: "Delivery"}



Hình 4.7: Layout sau khi thêm RadioGroup.

Phần nội dung của file XML mô tả layout.

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1" >
    <TableRow
        android:id="@+id/tableRow1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Name:"
<EditText
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10" >
</EditText>
</TableRow>
<TableRow
    android:id="@+id/tableRow2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Address"
<EditText
    android:id="@+id/addr"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10" />
</TableRow>
<TableRow
    android:id="@+id/tableRow3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Type:"
<RadioGroup
    android:id="@+id/type"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
<RadioButton
    android:id="@+id/take_out"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="Take out"/>
<RadioButton
    android:id="@+id/sit_down"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sit down"/>
<RadioButton
    android:id="@+id/delivery"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="delivery"/>
</RadioGroup>
</TableRow>
<Button
    android:id="@+id/save"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Save"/>
</TableLayout>
```

- ➡ **Bước 2:** Xây dựng lớp chứa dữ liệu: Tạo file Java có tên Restaurant.java trong file này khai báo lớp public là Restaurant.

```
public class Restaurant {

    private String name="";
    private String address="";
    private String type="";

    public void setName(String name)
    {
        this.name = name;
    }
    public String getName()
    {
        return (name);
    }

    public void setAddress(String address)
    {
        this.address = address;
    }
```

```

    }
public String getAddress()
{
    return (address);
}

public void setType(String type)
{
    this.type = type;
}
public String getType()
{
    return (type);
}
}

```

- ➡ **Bước 3:** Trong lớp Activity khai báo biến thành viên là r có kiểu là Restaurant.

```

public class LunchList extends Activity {

    // bien thanh vien
    private Restaurant r = new Restaurant();
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_lunch_list);
}

```

Khai báo biến chưa dữ
liệu trong activity

Hình 4.8: Khai báo biến chưa dữ liệu trong activity chính.

- ➡ **Bước 4:** Khai báo phần xử lý cho button Save: chức năng xử lý sẽ lấy các thông tin từ người dùng trên giao diện và thiết lập cho đối tượng Restaurant là r. Trong phần này có khai báo một đối tượng onSave có kiểu là View.OnClickListener và tạo một Anonymous Inner Class định nghĩa phần xử lý onClick trên Button Save.

@Override

```

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_lunch_list); // load layout XML
}

```

Button save = (Button)findViewById(R.id.save); // tham chiếu đến Button

```
        save.setOnClickListener(onSave); // khai báo listener cho Button
    }

private View.OnClickListener onSave = new View.OnClickListener() {

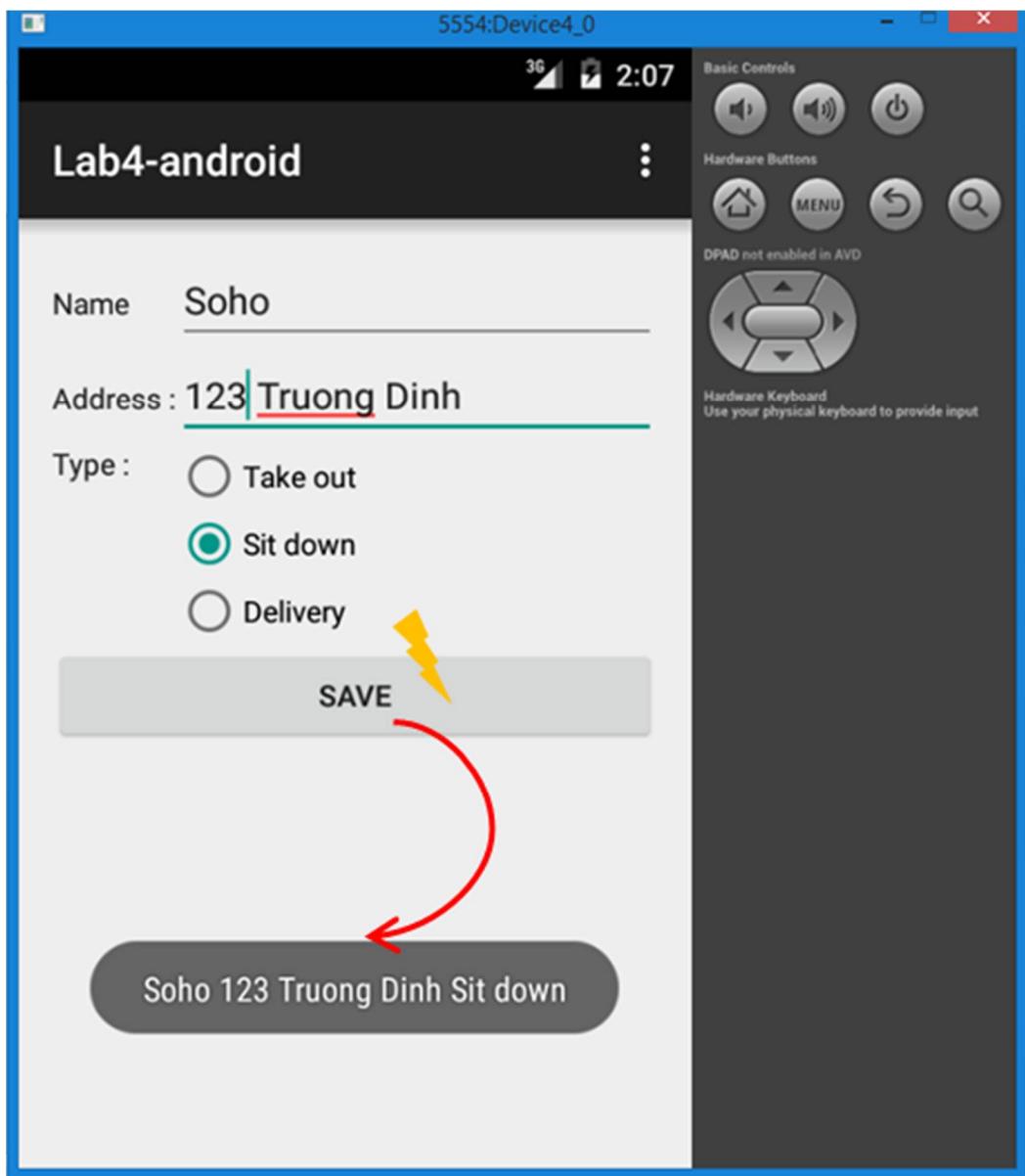
    public void onClick(View v) {
        EditText name = (EditText)findViewById(R.id.name);
        EditText address = (EditText)findViewById(R.id.addr);

        r.setName(name.getText().toString());
        r.setAddress(address.getText().toString());

        RadioGroup type = (RadioGroup)findViewById(R.id.type);
        switch (type.getCheckedRadioButtonId())
        {
            case R.id.take_out:
                r.setType("Take out");
                break;
            case R.id.sit_down:
                r.setType("Sit down");
                break;
            case R.id.delivery:
                r.setType("Delivery");
                break;
        }
    }
};
```

- ➡ **Bước 5:** (phần test) sau khi add thông tin liên quan đến một Restaurant thì hiển thị trên một cửa sổ popup. Phần này bổ sung trong hàm onClick của lớp anonymous thực thi giao diện View.OnClickListener.

Hình 4.9 minh họa phần test của ứng dụng.



Hình 4.9: Phần test ứng dụng.

=oOo=



Lab 5:

Lunch List Application Version 2 (*)

(*): Tham khảo từ *Android™ Programming Tutorials*, version 3.2 for Android 3.0, Mark L. Murphy, 2011, CommonsWare

Mục tiêu

- ✚ Bổ sung phiên bản v1 với danh sách hiển thị những nhà hàng đã thêm vào.
- ✚ Sử dụng ListView và cơ chế Adapter để gắn dữ liệu từ ArrayList lên ListView.

Yêu cầu

- ✚ Có kiến thức cơ bản về xây dựng activity và layout XML trong phần code view (markup view).
- ✚ Hiểu rõ cơ chế bind xử lý sự kiện của Button trên giao diện với phần code xử lý trong lớp activity.
- ✚ Hoàn thành phần lab 4, Lunch List Application version 1.

Nội dung

- ✚ **Bước 1:** Tạo một danh sách Restaurant, bằng cách thay biến Restaurant r trong activity bằng biến restaurantList có kiểu là ArrayList<Restaurant>

```
public class LunchList extends Activity {  
  
    // biến thanh vien danh sách nhà hàng  
    private List<Restaurant> restaurantList = new ArrayList<Restaurant>();
```

- ✚ **Bước 2:** Chuẩn bị cho việc lưu dữ liệu restaurant vào List.

Tại thời điểm này phần xử lý trong onClick không hợp lệ do tham chiếu đến biến thành viên r không còn tồn tại. Do đó ta khai báo một biến r cục bộ trong phương thức này. Biến r cục bộ này dùng để lưu dữ liệu khi người dùng nhập thông tin một nhà hàng. Biến r này trong các bước tiếp sau sẽ được dùng để add vào List.

```

private View.OnClickListener onClickListener {
    public void onClick(View v) {
        // biến Restaurant tạm để lưu dữ liệu nhập.
        Restaurant r = new Restaurant();

        EditText name = (EditText) findViewById(R.id.name);
        EditText address = (EditText) findViewById(R.id.addr);

        r.setName(name.getText().toString());
        r.setAddress(address.getText().toString());
    }
}

```

Khai báo biến tạm là r trong onClick

- ➡ **Bước 3:** Để hiển thị nội dung của một dữ liệu Restaurant thì ta phải override phương thức `toString()` của lớp `Restaurant`, phương thức này trả về thông tin `name` của nhà hàng.

```

@Override
public String toString() {
    // TODO Auto-generated method stub

    return (getName()); // trả về tên nhà hàng
}

```

- ➡ **Bước 4:** Thêm một thành phần widget là `ListView` vào trong layout chính của ứng dụng. Do màn hình của thiết bị di động khá hạn chế nên, việc tiết kiệm không gian trong một màn hình là rất quan trọng. Do thành phần `ListView` này chúng ta chưa biết chính xác số lượng item (thông tin nhà hàng) nên ta không thể tính trước kích thước dài của `ListView`. Một giải pháp là dùng `RelativeLayout` để tổ chức các thành phần. Phần nội dung của form sẽ neo với biên dưới màn hình, còn danh sách sẽ được trải từ biên trên đến phần detail bên dưới.

- ❖ **Bước 4.1:** Chúng ta tạo layout của activity có thành phần `RelativeLayout` ở ngoài cùng, chứa `TableLayout` đã thiết kế của version 1 vào bên trong.
- ❖ **Bước 4.2:** Bổ sung thêm `ListView` vào layout. Bổ sung các thuộc tính cho `TableLayout` và `ListView` như sau:

- **TableLayout:**

- `id = "@+id/details" // khai báo id cho TableLayout.`
- `layout_alignParentBottom="true"`

- **ListView:**

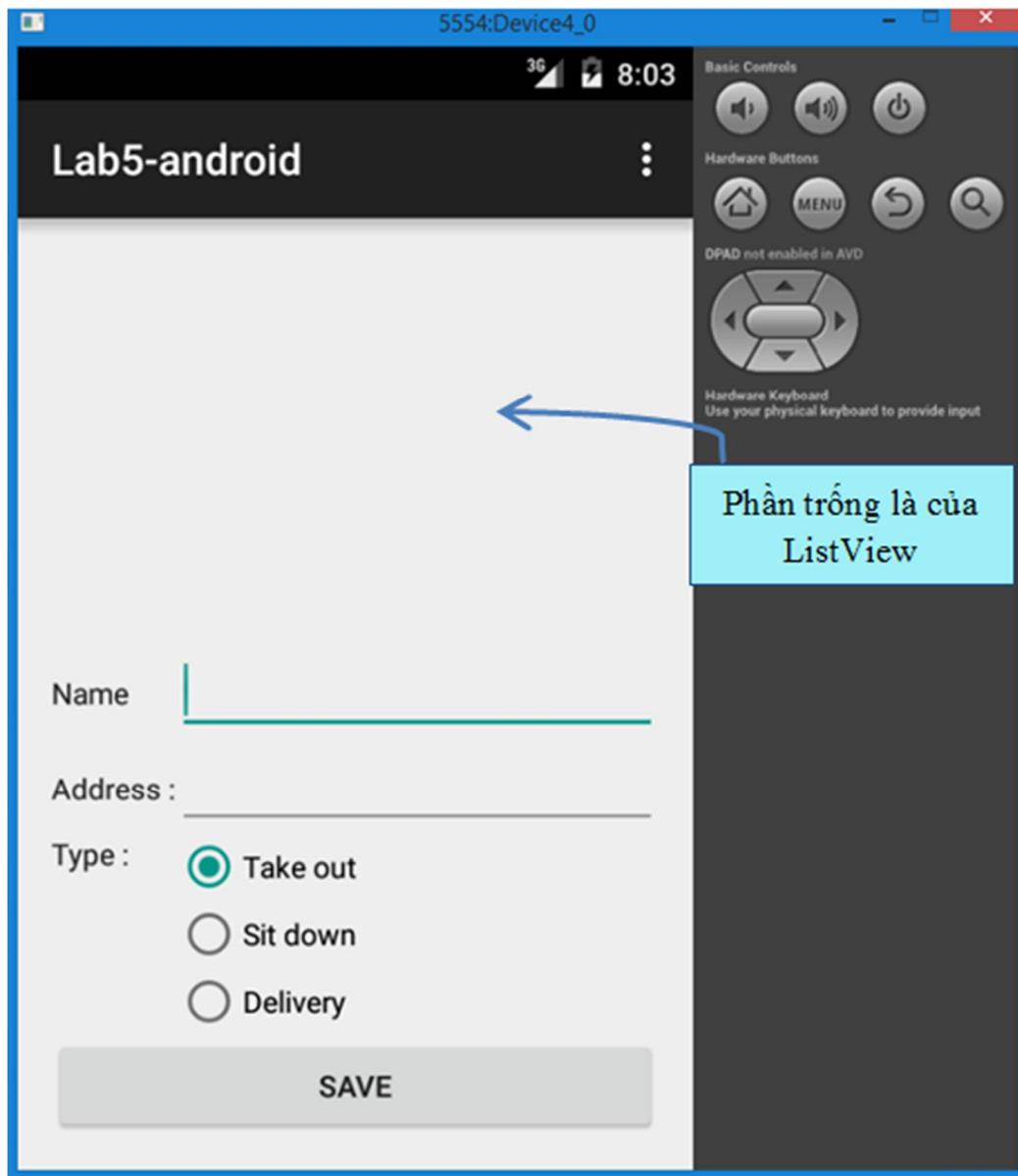
- layout_alignParentTop="true"
- layout_above="@+id/details" (đặt bên trên TableLayout details)
- id = "@+id/restaurants"

Phần XML của file layout, lưu ý dòng in đậm là nơi cần bổ sung.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/wrapper"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent">  
    <TableLayout  
        android:id="@+id/details"  
        android:layout_width="fill_parent"  
        android:layout_height="wrap_content"  
        android:layout_alignParentBottom="true"  
        android:stretchColumns="1">  
        <TableRow  
            android:id="@+id/tableRow1"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content">  
            <TextView  
                android:id="@+id/textView1"  
                android:layout_width="wrap_content"  
                android:layout_height="wrap_content"  
                android:text="Name:" />  
            <EditText  
                android:id="@+id/name"  
                android:layout_width="wrap_content"  
                android:layout_height="wrap_content"  
                android:ems="10">  
            </EditText>  
        </TableRow>  
        <TableRow  
            android:id="@+id/tableRow2"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content">  
            <TextView  
                android:id="@+id/textView2"  
                android:layout_width="wrap_content"  
                android:layout_height="wrap_content"  
                android:text="Address" />  
            <EditText
```

```
        android:id="@+id/addr"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10" />
    </TableRow>
    <TableRow
        android:id="@+id/tableRow3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <TextView
            android:id="@+id/textView3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Type:"
            android:textAppearance="?android:attr/textAppearanceMedium" />
        <RadioGroup
            android:id="@+id/type"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
            <RadioButton
                android:id="@+id/take_out"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:checked="true"
                android:text="Take out" />
            <RadioButton
                android:id="@+id/sit_down"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Sit down" />
            <RadioButton
                android:id="@+id/delivery"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="delivery" />
        </RadioGroup>
    </TableRow>
    <Button
        android:id="@+id/save"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Save" />
</TableLayout>
<ListView
```

```
    android:id="@+id/restaurants"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_above="@+id/details"
    android:layout_alignParentTop="true">
</ListView>
</RelativeLayout>
```



Hình 5.3: Giao diện khi bổ sung thêm ListView.

- ➡ **Bước 5:** Tạo một Adapter và gắn với ListView

ListView cho đến lúc này vẫn còn trống chưa có dữ liệu. Trong phần này sẽ bổ sung code để thêm các đối tượng vào ArrayList và dùng cơ chế binding của Adapter để đưa dữ liệu đó lên trên ListView.

Cách thực hiện:

- ❖ Tạo một đối tượng tên adapter có kiểu là ArrayAdapter<Restaurant>
- ❖ Khai báo adapter tham chiếu đến restaurantList.
- ❖ Thiết lập adapter cho ListView thông qua phương thức setAdapter.

Khai báo adapter trong lớp activity LunchList như hình 5.4.

```
public class LunchList extends Activity {  
  
    // biến thành viên danh sách nhà hàng  
    private List<Restaurant> restaurantList = new ArrayList<Restaurant>();  
    private ArrayAdapter<Restaurant> adapter = null;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState)  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_lunchlist);
```

Khai báo thành phần
Adapter trong lớp
Activity LunchList

Hình 5.4: Khai báo Adapter

Định nghĩa adapter và thiết lập cho ListView trong phần khởi tạo của Activity LunchList như hình 5.5.

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_lunch_list);  
  
    Button save = (Button)findViewById(R.id.save);  
    save.setOnClickListener(onSave);  
  
    ListView list = (ListView)findViewById(R.id.restaurants);  
  
    adapter = new ArrayAdapter<Restaurant>(this,  
        android.R.layout.simple_list_item_1,  
        restaurantList);  
    list.setAdapter(adapter);  
}  
}
```

Tạo adapter tham chiếu đến
restaurantList và mỗi item có kiểu
mặc định do Android định nghĩa là
simple_list_item_1

Hình 5.5: Khai báo Adapter

Bổ sung phần thêm dữ liệu Restaurant vào ArrayList. Sau khi button Save được chọn, chương trình cập nhật thông tin của r và sau đó thêm vào restaurantList. Dữ liệu của restaurantList được cập nhật và thể hiện trong ListView.

```
public void onClick(View v) {
    // bien Restautant tam de luu du lieu nhap.
    Restaurant r = new Restaurant();

    EditText name = (EditText)findViewById(R.id.name);
    EditText address = (EditText)findViewById(R.id.addr);

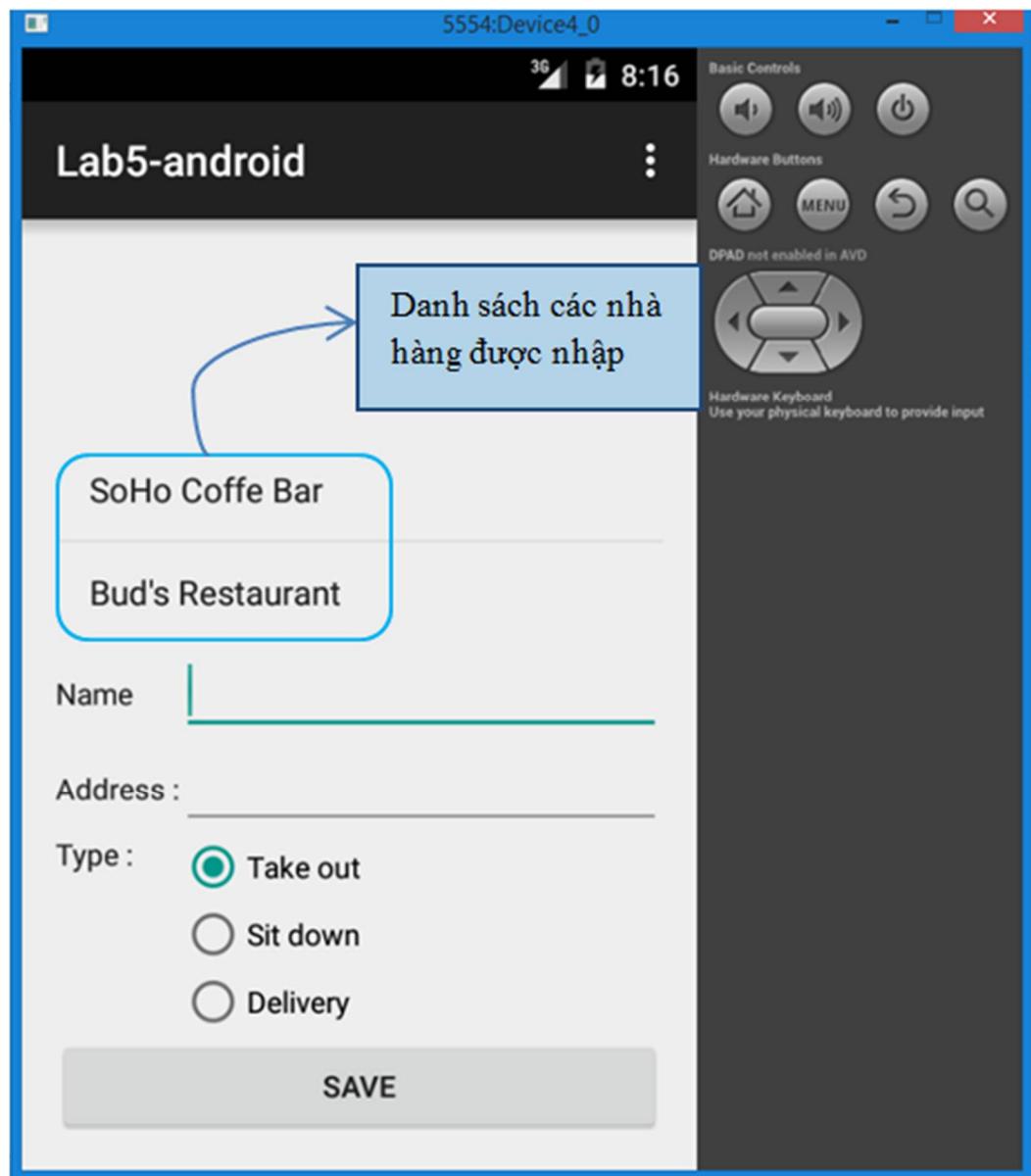
    r.setName(name.getText().toString());
    r.setAddress(address.getText().toString());

    RadioGroup type = (RadioGroup)findViewById(R.id.type);
    switch (type.getCheckedRadioButtonId())
    {
        case R.id.take_out:
            r.setType("Take out");
            break;
        case R.id.sit_down:
            r.setType("Sit down");
            break;
        case R.id.delivery:
            r.setType("Delivery");
            break;
    }
    // them vao restaurantList
    restaurantList.add(r);
}
```

Thêm một item Restaurant vào
ArrayList **restaurantList**

Hình 5.6: Phần code hoàn chỉnh của hàm onClick.

Kết quả khi chạy ứng dụng.



Hình 5.7: Giao diện của ứng dụng LunchList version 2.

=oOo=



Lab 6:

Lunch List Application Version 3 (*)

(*): Tham khảo từ *Android™ Programming Tutorials*, version 3.2 for Android 3.0, Mark L. Murphy, 2011, CommonsWare

Mục tiêu

- ✚ Cập nhật lại layout của ListView, cụ thể mỗi dòng sẽ hiển thị 3 thông tin:
 - ❖ Hình icon minh họa cho loại nhà hàng.
 - ❖ Tên nhà hàng
 - ❖ Địa chỉ nhà hàng
- ✚ Xây dựng riêng lớp ListAdapter để xử lý phần hiển thị trên từng dòng của ListView
- ✚ Tạo lớp RestaurantHolder để đưa nội dung của một nhà hàng trong danh sách lên dòng.

Yêu cầu

- ✚ Đã hoàn thành lab 5 Lunch List Application Version 2.

Nội dung

- ✚ **Bước 1:** Tạo một lớp Custom Adapter, lớp có tên **RestaurantAdapter** lớp kế thừa từ `ArrayAdapter<Restaurant>`. Lớp này sẽ tạo ra layout riêng cho từng row trong ListView. Lưu ý lớp RestaurantAdapter là dạng inner class trong activity LunchList.

```

class RestaurantAdapter extends ArrayAdapter<Restaurant>
{
    public RestaurantAdapter(Context context, int textViewResourceId) {
        super(context, textViewResourceId);
        // TODO Auto-generated constructor
    }

    public RestaurantAdapter()
    {
        super(LunchList.this,
              android.R.layout.simple_list_item_1,
              listRestaurant);
    }
}

```

Lớp Custom Adapter

Lấy dữ liệu từ listRestaurant

Hình 6.1: Tạo lớp custom adapter

Cập nhật lại biến thành viên adapter trong activity có kiểu là RestaurantAdapter và phần code khởi tạo trong hàm onCreate().

```

public class LunchList extends Activity {

    // khai báo danh sách restaurant
    List<Restaurant> listRestaurant = new ArrayList<Restaurant>();

    RestaurantAdapter adapter = null;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lunch_list);

        Button save = (Button) findViewById(R.id.save);
        save.setOnClickListener(onSave);

        ListView list = (ListView) findViewById(R.id.restaurants);
        adapter = new RestaurantAdapter();
        list.setAdapter(adapter);
    }
}

```

Cập nhật
RestaurantAdapter

Hình 6.2: Cập nhật lại kiểu cho biến adapter trong activity.

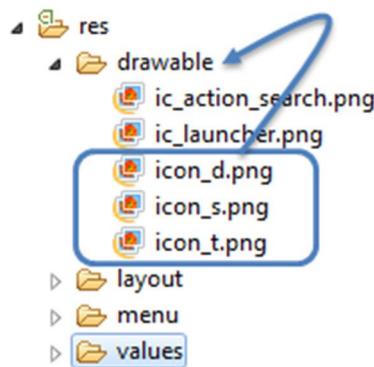
- Bước 2:** Thiết kế layout cho từng dòng trong ListView. Trong phiên bản trước phần ListView chỉ hiển thị tên của nhà hàng. Ta sẽ bổ sung thêm đầy đủ thông tin bao gồm tên, địa chỉ và kiểu. Riêng đối với trường hợp là kiểu thì dùng icon để thể hiện như bảng 6.1. Sinh viên có thể dùng bất cứ icon nào tùy ý miễn thể hiện được các loại khác nhau của nhà hàng.

Bảng 6.1: Các icon minh họa kiểu nhà hàng.

Type	Icon	File name
Take out		type_t.png

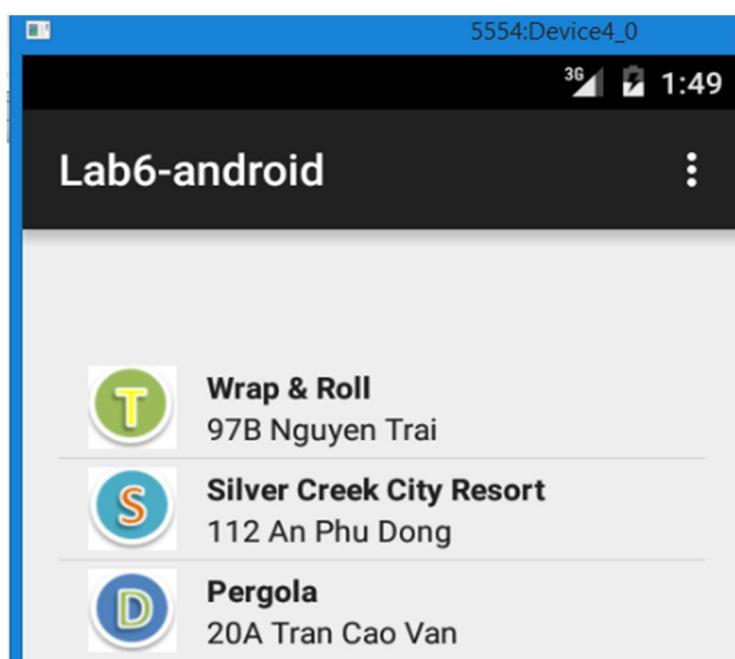


Import tất cả 3 file trên {type_t.png, type_s.png, type_d.png} vào thư mục **res/drawable**



Hình 6.3: Import các icon vào resource.

Phần ListView có dạng hiển thị như sau:



Hình 6.4: Phần thể hiện của ListView

Để xây dựng được layout như hình minh họa trên, thì phải dùng hai LinearLayout phối hợp với nhau. Khai báo file **row.xml** trong thư mục **res/layout**, file này có nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="4dip">
    <ImageView android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_alignParentTop="true"
        android:layout_alignParentBottom="true"
        android:layout_marginRight="4dip"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView android:id="@+id/title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center_vertical"
            android:textStyle="bold"
            android:singleLine="true"
            android:ellipsize="end" />
        <TextView android:id="@+id/address"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center_vertical"
            android:singleLine="true"
            android:ellipsize="end"/>
    </LinearLayout>
</LinearLayout>
```

Một số các thuộc tính không thường dùng được bổ sung cho layout trên.

- android:padding: bổ sung thêm không gian trống (whitespace) bên ngoài nội dung của widgets. Phần không gian này vẫn tính cho kích thước thật của widget.
- android:textStyle: xác định kiểu chữ đậm hay in nghiêng.

- android:ellipsize: xác định vị trí bị cắt khi nội dung của chuỗi vượt quá không gian cho phép.

 **Bước 3:** Thiết lập layout row.xml vừa xây dựng trong bước 2 cho RestaurantAdapter. Thực hiện bằng cách override phương thức getView() và inflate layout cho từng dòng. Phương thức getView sẽ duyệt qua từng dòng và lấy thông tin của một nhà hàng trong List<Restaurant> listRestaurant để cập nhật cho dòng đó. Thông tin bao gồm tên nhà hàng, địa chỉ nhà hàng được cập nhật lên các TextView tương ứng trong layout row.xml. Thông tin về kiểu nhà hàng sẽ được thể hiện với các icon tương ứng trong thư mục Drawable. Các icon được load lên ImageView thông qua phương thức setImageResource và truyền các id tương ứng của các icon.

```

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    // TODO Auto-generated method stub
    View row = convertView;
    if (row == null) {
        LayoutInflator inflater = getLayoutInflater();
        row = inflater.inflate(R.layout.row, null);
    }
    // cap nhat du lieu cho tung dong
    Restaurant r = listRestaurant.get(position);

    ((TextView)row.findViewById(R.id.title)).setText(r.getName());
    ((TextView)row.findViewById(R.id.address)).setText(r.getAddress());
    ImageView icon = (ImageView)row.findViewById(R.id.icon);

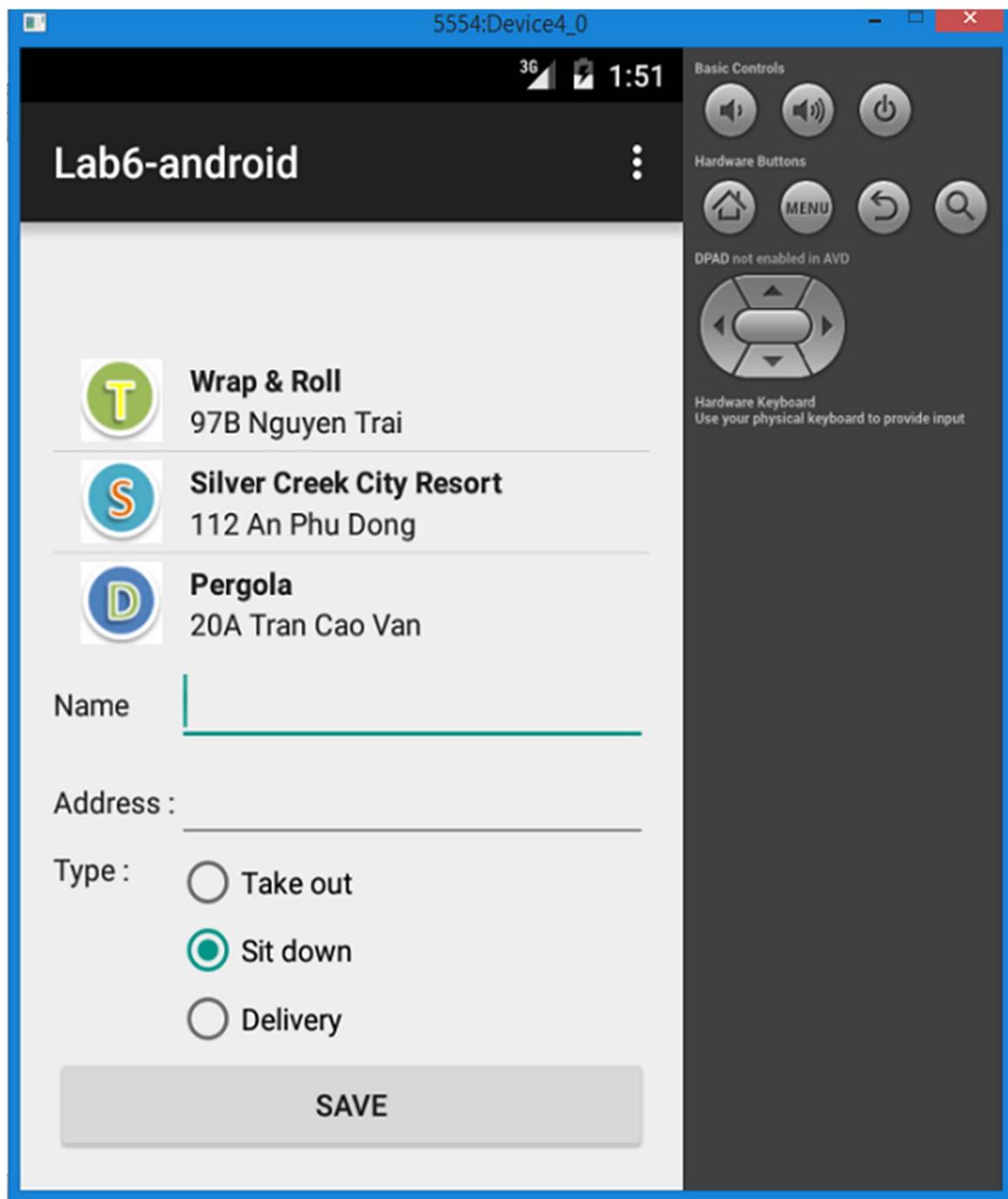
    String type = r.getType();
    if (type.equals("Take out"))
        icon.setImageResource(R.drawable.icon_t);
    else if (type.equals("Sit down"))
        icon.setImageResource(R.drawable.icon_s);
    else
        icon.setImageResource(R.drawable.icon_d);
    return row;
}

```

Lớp RestaurantAdapter

Hình 6.5: Override getView() trong lớp RestaurantAdapter.

 **Bước 4:** Biên dịch và chạy chương trình



Hình 6.6: Demo ứng dụng LunchList.

=oOo=



Lab 7:

Lunch List Application Version 4 (*)

(*): Tham khảo từ *Android™ Programming Tutorials*, version 3.2 for Android 3.0, Mark L. Murphy, 2011, CommonsWare

Mục tiêu

- ✚ Tổ chức lại ứng dụng Lunch List bằng cách đưa ListView vào một tab và form nhập liệu vào một tab khác của thành phần TabView.
- ✚ Sử dụng lớp TabActivity làm lớp cơ sở cho Activity của ứng dụng
- ✚ Tìm hiểu cách tổ chức phân chia các view trong layout vào trong các tab con khác nhau.

Yêu cầu

- ✚ Hoàn thành phần code của bài lab 6.

Nội dung

- ✚ **Bước 1:** Thay đổi layout của version 3, chuyển layout sang dạng tab và chia những thành phần widget trên giao diện ra thành 2 tab có tên là list tab và detail tab. Cách thực hiện
 - ❖ Xóa RelativeLayout khỏi layout, do giao diện ở phiên bản này chia ListView và form nhập liệu ở hai tab khác nhau.
 - ❖ Bổ sung thêm TabHost, TabWidget và FrameLayout, trong đó thành phần FrameLayout sẽ chứa nội dung list và detail (form nhập liệu).

Phần code của layout sau khi thay đổi:

```
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tabhost"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TabWidget android:id="@+id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
        <FrameLayout android:id="@+id/tabcontent"
            android:layout_width="match_parent"
```

```
        android:layout_height="match_parent">
    <ListView
        android:id="@+id/restaurants"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
    <TableLayout
        android:id="@+id/details"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:stretchColumns="1"
        android:paddingTop="4dip">
        <TableRow>
            <TextView android:text="Name:"/>
            <EditText android:id="@+id/name"/>
        </TableRow>
        <TableRow>
            <TextView android:text="Address:"/>
            <EditText android:id="@+id/addr"/>
        </TableRow>
        <TableRow>
            <TextView android:text="Type: "/>
            <RadioGroup android:id="@+id/types">
                <RadioButton android:id="@+id/take_out"
                    android:checked="true"
                    android:text="Take-out" />
                <RadioButton android:id="@+id/sit_down"
                    android:text="Sit-down" />
                <RadioButton android:id="@+id/delivery"
                    android:text="Delivery" />
            </RadioGroup>
        </TableRow>
        <Button android:id="@+id/save"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Save" />
    </TableLayout>
</FrameLayout>
</LinearLayout>
</TabHost>
```

- ➡ **Bước 2:** Thiết kế phần thẻ hiện theo dạng tab cho activity chính của ứng dụng bằng cách thay đổi lớp LunchList. Lớp này sẽ chuyển sang kế thừa từ lớp TabActivity thay cho

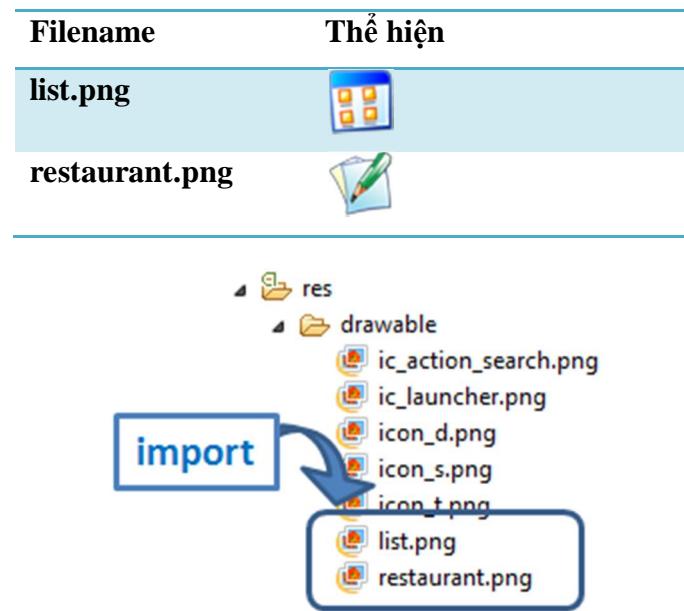
Activity tổng quát. Bổ sung phần mã lệnh để TabHost sử dụng nội dung của FrameLayout và chia thành những tab khác nhau. Các bước thực hiện bao gồm.

- ❖ Thêm phần import android.app.TabActivity và android.widget.TabHost.
- ❖ Thay đổi phần kế thừa của LunchList từ Activity sang TabActivity.



Hình 7.1: Bổ sung cho lớp LunchList.

- ❖ Sử dụng 2 icon có kích thước 32x32 làm icon cho các tab.



Hình 7.2: Import hai icon vào project

Phần code được bổ sung cho onCreate() được thay đổi như sau:

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
}

```

```
setContentView(R.layout.activity_lunch_list);

Button save = (Button) findViewById(R.id.save);
save.setOnClickListener(onSave);

ListView list = (ListView) findViewById(R.id.restaurants);
adapter = new RestaurantAdapter();
list.setAdapter(adapter);

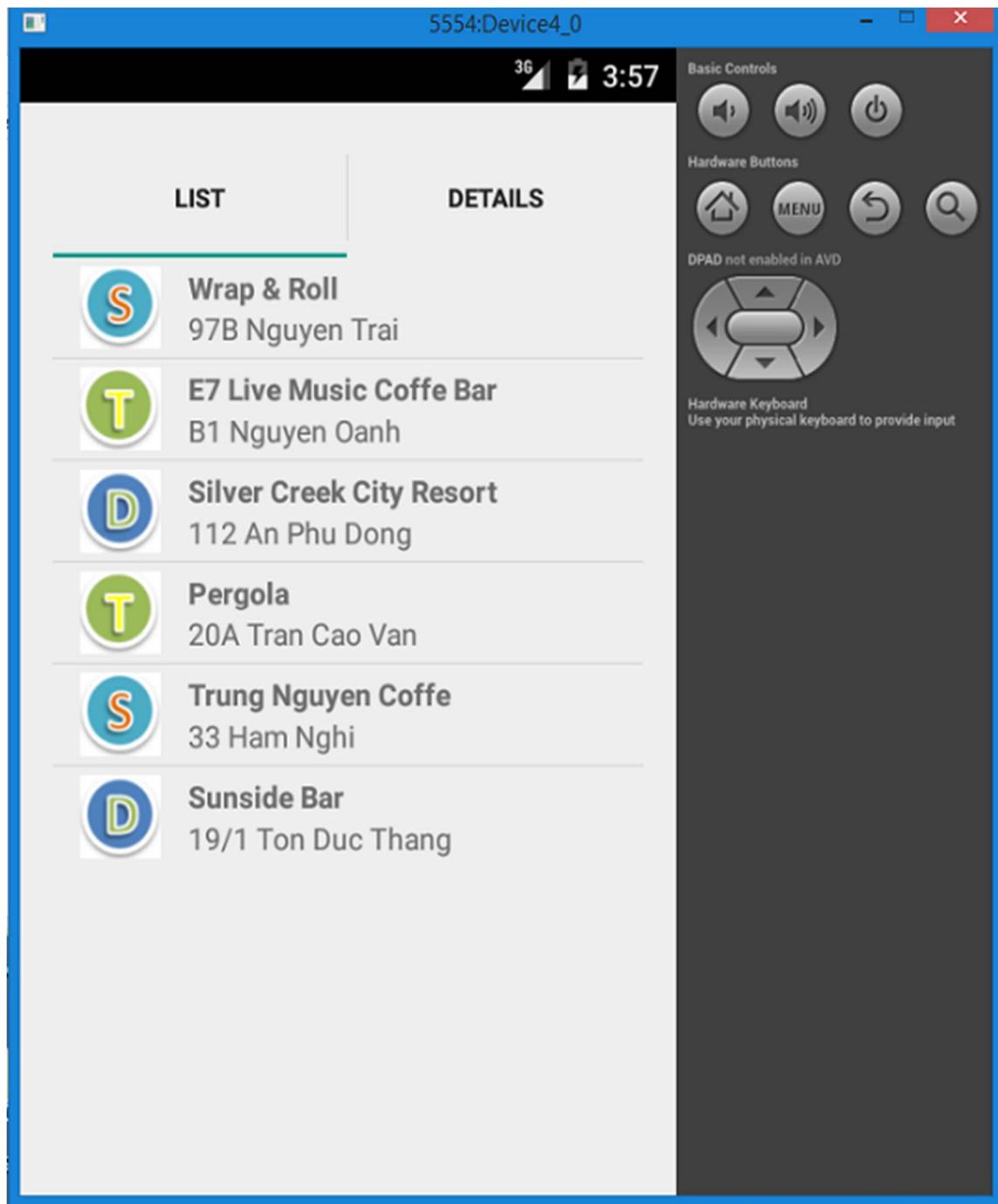
// Phân bổ sung cho Tab
TabHost.TabSpec spec = getTabHost().newTabSpec("tag1");
spec.setContent(R.id.restaurants);
spec.setIndicator("List",getResources().getDrawable(R.drawable.list));
getTabHost().addTab(spec);

spec = getTabHost().newTabSpec("tag2");
spec.setContent(R.id.details);
spec.setIndicator("Details",
    getResources().getDrawable(R.drawable.restaurant));
getTabHost().addTab(spec);

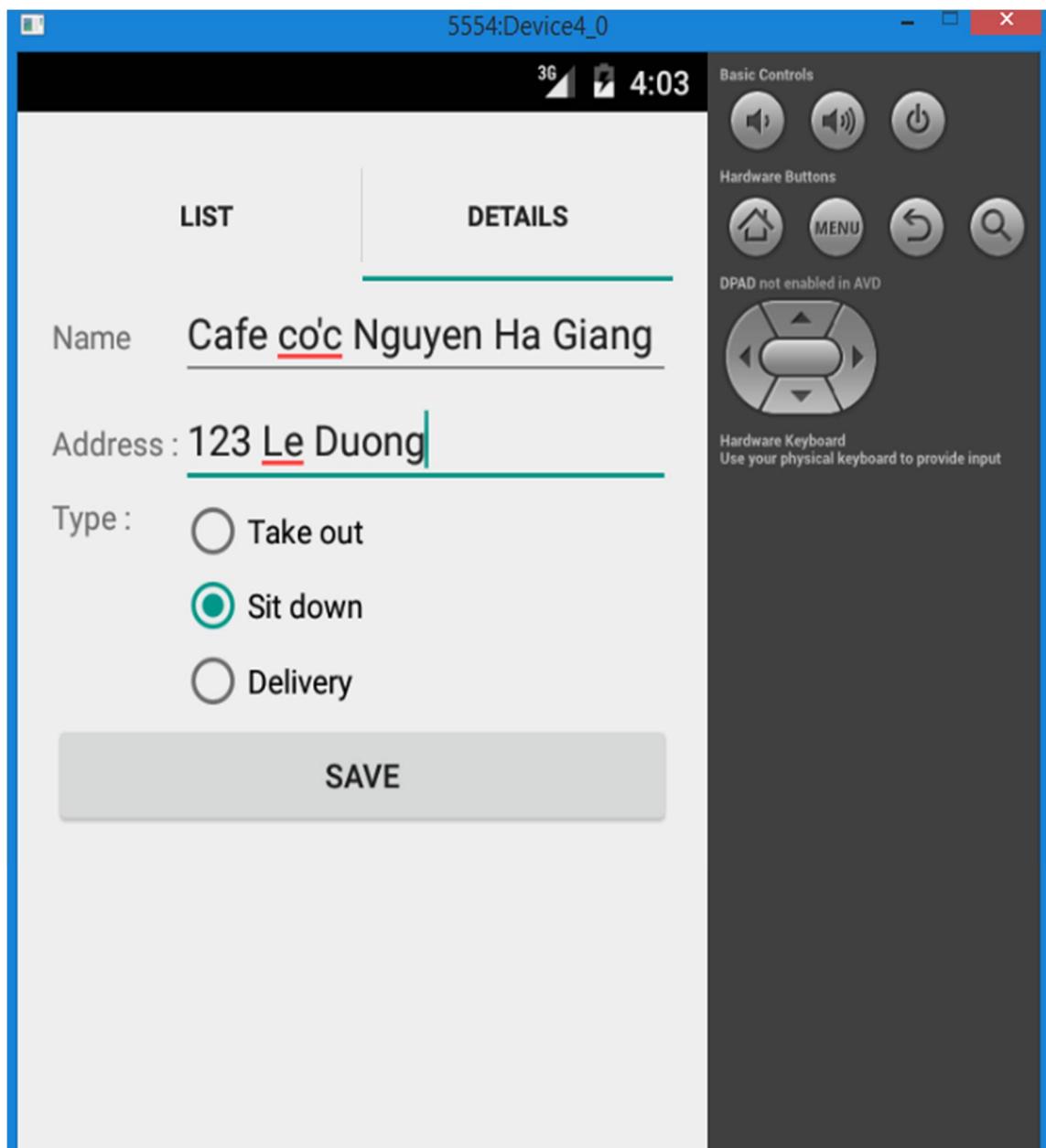
getTabHost().setCurrentTab(0);

}// end onCreate
```

Biên dịch và chạy ứng dụng!



Hình 7.3: Tab list hiển thị danh sách nhà hàng.



Hình 7.4: Tab details cho phép nhập thông tin của một nhà hàng.

- ➡ **Bước 3:** Bổ sung chức năng cho phép user chọn một nhà hàng trong tab list và thông tin của nhà hàng được chọn đó sẽ hiển thị trong tab details. Thực hiện theo các bước sau:
- ❖ Import android.widget.AdapterView vào lớp activity

```

import android.widget.TabHost;
import android.app.TabActivity;
import android.widget.AdapterView;

public class LunchList extends TabActivity {

    // khai báo danh sách restaurant
    List<Restaurant> listRestaurant = new ArrayList<Restaurant>();

    RestaurantAdapter adapter = null;
}

```

Hình 7.5: Import thư viện AdapterView

- ❖ Tạo một biến AdapterView.OnItemClickListener tên là onListClick trong lớp LunchList.

Khai báo trong activity LunchList

```

private AdapterView.OnItemClickListener onListClick = new
        AdapterView.OnItemClickListener() {

    public void onItemClick(AdapterView<?> parent, View view,
                          int position, long id) {
        // TODO Auto-generated method stub
        // Phần code sẽ viết trong bước 4.

    }
};

```

Hình 7.6: Khai báo OnItemClickListener.

- ❖ Thiết lập OnItemClickListener cho ListView trong onCreate() của activity.

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_lunch_list);

    Button save = (Button) findViewById(R.id.button_save);
    save.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(getApplicationContext(), MainActivity.class);
            startActivity(intent);
        }
    });

    ListView list = (ListView) findViewById(R.id.restaurants);
    list.setOnItemClickListener(onListClick);

    adapter = new RestaurantAdapter();
    list.setAdapter(adapter);
}

```

Thiết lập ItemClick Listener cho ListView trong onCreate()

Hình 7.7: Thiết lập OnItemClickListener cho ListView

- Bước 4:** Viết phần xử lý cho Item Click Listener của ListView, phần code lấy item được chọn (dùng position) trong ListView thông qua adapter. Thiết lập các thông tin từ item chọn bao gồm: {tên nhà hàng, địa chỉ, và kiểu} lên tab details.

```

private AdapterView.OnItemClickListener onListClick = new
    AdapterView.OnItemClickListener() {

    public void onItemClick(AdapterView<?> parent, View view, int position,
                           long id) {
        // TODO Auto-generated method stub

        Restaurant r = listRestaurant.get(position); // lấy item được chọn
        EditText name;
        EditText address;
        RadioGroup types;

        // Tham chiếu đến các view trong details
        name = (EditText) findViewById(R.id.name);
        address = (EditText) findViewById(R.id.addr);
        types = (RadioGroup) findViewById(R.id.types);

        // thiết lập thông tin tương ứng
        name.setText(r.getName());
        address.setText(r.getAddress());
        if (r.getType().equals("Sit down"))
            types.check(R.id.sit_down);
        else if (r.getType().equals("Take out"))
    }
}

```

```
        types.check(R.id.take_out);
    else
        types.check(R.id.delivery);
    // sinh viên có thể bổ sung lệnh sau để chuyển view về tab details
    getTabHost().setCurrentTab(1);
}
};
```

➡ **Bước 5:** Biên dịch và chạy ứng dụng LunchList Version 4!

=oOo=



Lab 8:

Lunch List Application Version 5 (*)

(*): Tham khảo từ *Android™ Programming Tutorials*, version 3.2 for Android 3.0, Mark L. Murphy, 2011, CommonsWare

Mục tiêu

- ✚ Nâng cấp phiên bản Lunch List App Version 4 với chức năng lưu trữ.
- ✚ Làm quen với cách thức lưu trữ dữ liệu dùng SQLite. Cách thức khai báo và tạo một database SQLite.
- ✚ Tìm hiểu cách truy vấn dữ liệu từ một table trong SQLite và trả kết quả dữ liệu về một đối tượng Cursor.
- ✚ Sử dụng Cursor và CursorAdapter để duyệt và hiển thị nội dung của từng row dữ liệu lên view của ListView.

Yêu cầu

- ✚ Hoàn thành phần code của bài lab 7.

Nội dung

- ✚ **Bước 1:** Tạo một lớp chứa code hỗ trợ để thao tác trên database SQLite, bao gồm tên database và schema cho table. Lớp có tên RestaurantHelper kế thừa từ lớp cơ sở SQLiteOpenHelper.

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;
```

```
public class RestaurantHelper extends SQLiteOpenHelper {

    // khai báo tên database và schema

    private static final String DATABASE_NAME = "lunchlist.db";
    private static final int SCHEMA_VERSION = 1;

    // Bổ sung constructor chứa một tham số kiểu Context
    public RestaurantHelper(Context context)
    {
        // gọi constructor của SQLiteOpenHelper truyền tên database và chema
```

```

        super(context, DATABASE_NAME, null, SCHEMA_VERSION);
    }

public RestaurantHelper(Context context, String name,
    CursorFactory factory, int version) {
    super(context, name, factory, version);
    // TODO Auto-generated constructor stub
}

@Override
public void onCreate(SQLiteDatabase db) {
    // TODO Auto-generated method stub

}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // TODO Auto-generated method stub

}

// end RestaurantHelper

```

- ✍ **Bước 2:** Tạo bảng dữ liệu để lưu trữ thông tin nhà hàng, phần code tạo bảng dữ liệu được thực thi trong phương thức onCreate()

```

@Override
public void onCreate(SQLiteDatabase db) {
    // TODO Auto-generated method stub
    db.execSQL("CREATE TABLE restaurants (_id INTEGER PRIMARY KEY
        AUTOINCREMENT, name TEXT, address TEXT,
        type TEXT);"

}

```

Đoạn code trên tạo bảng restaurants với cấu trúc được mô tả gồm id, name, address, và type.

- ✍ **Bước 3:** Do RestaurantHelper là cầu nối để truy cập CSDL, do đó trong activity LunchList phải có một đối tượng là RestaurantHelper để truy cập dữ liệu và thêm nội dung mới. Các thao tác bao gồm open CSDL và sau khi hoàn tất phần xử lý liên quan đến dữ liệu thì close CSDL.

- ❖ **Bước 3.1:** tạo một biến dữ liệu thành viên là helper trong LunchList.

```
public class LunchList extends TabActivity {
```

```
// khai bao danh sach restaurant
List<Restaurant> listRestaurant = new ArrayList<Restaurant>();

RestaurantAdapter adapter = null;

// khai bao bien thanh vien lien quan den truy cap du lieu
RestaurantHelper helper;
```

- ❖ **Bước 3.2:** khởi tạo đối tượng cho helper trong phương thức onCreate của LunchList, lưu ý phần khởi tạo này phải viết sau khi gọi setContentView()

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_lunch_list);

    // khai bao doi duong RestaurantHelper
    helper = new RestaurantHelper(this);
```

- ❖ **Bước 3.3:** override phương thức onDestroy trong LunchList để đóng CSDL

```
@Override
protected void onDestroy() {
    // TODO Auto-generated method stub
    super.onDestroy();

    // Đóng cơ sở dữ liệu
    helper.close();
} // end onDestroy
```

❖

-  **Bước 4:** Lưu trữ một thông tin nhà hàng vào trong CSDL. Bổ sung một số phần xử lý hỗ trợ thêm dữ liệu mới vào trong CSDL.

- ❖ **Bước 4.1:** tạo phương thức insert trong RestaurantHelper như sau:

```
// phuong thuc insert mot dong thong tin nha hang
public void insert(String name, String address, String type)
{
    // tao doi tuong du lieu ContentValues
    ContentValues cv = new ContentValues();
    // dua cac du lieu vao theo tung cap ten field va value
    cv.put("name", name);
    cv.put("address", address);
```

```
        cv.put("type", type);

        // yeu cau SQLiteDatabase insert du lieu vao database
        getWritableDatabase().insert("restaurants", "name", cv);

    }

❖ Bước 4.2: gọi phương thức insert trong chức năng onSave trên LunchList.

private View.OnClickListener onSave = new View.OnClickListener() {

    public void onClick(View v) {
        // TODO Auto-generated method stub
        Restaurant restaurant = new Restaurant();
        // lay thong tin ve editText;
        EditText name = (EditText)findViewById(R.id.name);
        EditText address = (EditText)findViewById(R.id.addr);

        restaurant.setName(name.getText().toString());
        restaurant.setAddress(address.getText().toString());

        RadioGroup types = (RadioGroup)findViewById(R.id.types);

        // kiem tra xem user chon loai nao
        switch (types.getCheckedRadioButtonId()) {
            case R.id.take_out:
                restaurant.setType("Take out");
                break;
            case R.id.sit_down:
                restaurant.setType("Sit down");
                break;

            case R.id.delivery:
                restaurant.setType("Delivery");
                break;

            default:
                break;
        }
        // them vao CSDL
        helper.insert(restaurant.getName(), restaurant.getAddress(),
                    restaurant.getType());
    }
};
```

 **Bước 5:** lấy danh sách restaurant từ CSDL

Trong bước trước ta đã thêm dữ liệu của restaurant vào trong CSDL. Tiếp theo ta sẽ viết code để lấy dữ liệu trong CSDL ra. Do đó, bổ sung thêm phần xử lý để truy vấn dữ liệu và trả về đối tượng Cursor với cấu trúc các cột của bảng restaurants. đối tượng Cursor trong Android thì tương tự với cursor trong các thư viện truy cập CSDL của ngôn ngữ khác.

Cụ thể ta bổ sung thêm phương thức getAll(), trong phương thức này gọi phương thức rawQuery() của SQLiteDatabase. Truyền câu truy vấn thích hợp vào phương thức rawQuery() để lấy dữ liệu.

```
// phuong thuc truy van toan bo du lieu
public Cursor getAll()
{
    Cursor cur;
    cur = getReadableDatabase().rawQuery("SELECT _id, name, address, type
                                            FROM restaurants ORDER BY name", null);
    return (cur);
}
```

Để thuận tiện cho việc truy cập từng trường dữ liệu trong Cursor ta bổ sung thêm các phương thức kiểu getter để lấy các dữ liệu tương ứng trong một dòng dữ liệu từ Cursor, phần khai báo các phương thức getter này ở trong lớp RestaurantHelper.

```
public String getName(Cursor c)
{
    // truy cap cot thu 2 la cot name
    return (c.getString(1));
}
public String getAddress(Cursor c)
{
    // truy cap cot thu 3 la cot address
    return (c.getString(2));
}
public String getType(Cursor c)
{
    // truy cap cot thu 4 la type
    return (c.getString(3));
}
```

 **Bước 6:** Thay đổi Adapter để sử dụng được Cursor.

Trong phiên bản trước lớp RestaurantAdapter kế thừa từ ArrayAdapter thì không thích hợp với Cursor. Do đó trong phần này ta sẽ thay đổi để RestaurantAdapter xử lý tốt với Cursor. Chính xác thay đổi lớp kế thừa từ ArrayAdapter sang CursorAdapter.

ArrayAdapter thì tạo view cho mỗi item trong array hay list. Trong khi CursorAdapter thì tạo view cho mỗi dòng trong Cursor.

- ❖ **Bước 6.1:** Tạo lớp RestaurantAdapter với những thay đổi như sau.

```
class RestaurantAdapter extends CursorAdapter
{
    public RestaurantAdapter(Cursor c)
    {
        super(LunchList.this, c);
    }
    public RestaurantAdapter(Context context, Cursor c) {
        super(context, c);
        // TODO Auto-generated constructor stub
    }

    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        // TODO Auto-generated method stub
        View row = view;
        ((TextView)row.findViewById(R.id.title)).
            setText(helper.getName(cursor));
        ((TextView)row.findViewById(R.id.address)).
            setText(helper.getAddress(cursor));
        ImageView icon = (ImageView)row.findViewById(R.id.icon);

        String type = helper.getType(cursor);
        if (type.equals("Take out"))
            icon.setImageResource(R.drawable.icon_t);
        else if (type.equals("Sit down"))
            icon.setImageResource(R.drawable.icon_s);
        else
            icon.setImageResource(R.drawable.icon_d);
    }// end bindView

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        // TODO Auto-generated method stub
        LayoutInflator inflater = getLayoutInflator();
        View row = inflater.inflate(R.layout.row, parent, false);
```

```

        return row;
    }
}// end class RestaurantAdapter

```

- ❖ **Bước 6.2:** Bổ sung lại phần khai báo trong lớp LunchList cho phù hợp với sự thay đổi. Bao gồm thay đổi kiểu dữ liệu của listRestaurant từ ArrayList sang Cursor và đổi tên từ listRestaurant sang curRestaurant (cho thích hợp)

```
public class LunchList extends TabActivity {
```

```
// khai bao danh sach restaurant dung Cursor
Cursor curRestaurant = null;
```

```
RestaurantAdapter adapter = null;
```

```
// khai bao bien thanh vien lien quan den truy cap du lieu
RestaurantHelper helper = null;
```

- ❖ **Bước 6.3:** Bổ sung phần code lấy dữ liệu từ CSDL đưa vào Cursor là tạo adapter trong phương thức onCreate() của activity LunchList.

```
@Override
```

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_lunch_list);
```

```
// khai tao doi duong RestaurantHelper
helper = new RestaurantHelper(this);
```

```
Button save = (Button) findViewById(R.id.save);
save.setOnClickListener(onSave);
```

```
ListView list = (ListView) findViewById(R.id.restaurants);
list.setOnItemClickListener(onListClick);
```

```
// lay du lieu tu CSDL
curRestaurant = helper.getAll();
startManagingCursor(curRestaurant);
adapter = new RestaurantAdapter(curRestaurant);
list.setAdapter(adapter);
```

```
TabHost.TabSpec spec = getTabHost().newTabSpec("tag1");
spec.setContent(R.id.restaurants);
spec.setIndicator("List",getResources().getDrawable(R.drawable.list));
```

```

        getTabHost().addTab(spec);

        spec = getTabHost().newTabSpec("tag2");
        spec.setContent(R.id.details);
        spec.setIndicator("Details",
                          getResources().getDrawable(R.drawable.restaurant));
        getTabHost().addTab(spec);
        getTabHost().setCurrentTab(0);
    }
}

```

 **Bước 7:** xóa những tham chiếu ArrayList trong code ở phần trước. Do những phần code của ArrayList không còn làm việc với Cursor. Cụ thể là trong đối tượng listener onClick cần phải di chuyển Cursor đến vị trí position để lấy vị trí của item được chọn.

```

private AdapterView.OnItemClickListener onListClick = new
    AdapterView.OnItemClickListener() {

    public void onItemClick(AdapterView<?> parent, View view, int position,
                           long id) {
        // di chuyển cursor đến vị trí hiện tại là position
        curRestaurant.moveToPosition(position);

        EditText name;
        EditText address;
        RadioGroup types;

        name = (EditText)findViewById(R.id.name);
        address = (EditText)findViewById(R.id.addr);
        types = (RadioGroup)findViewById(R.id.types);

        // sử dụng đối tượng helper để lấy các thông tin nhà hàng
        name.setText(helper.getName(curRestaurant));
        address.setText(helper.getAddress(curRestaurant));
        if (helper.getType(curRestaurant).equals("Sit down"))
            types.check(R.id.sit_down);
        else if (helper.getType(curRestaurant).equals("Take out"))
            types.check(R.id.take_out);
        else
            types.check(R.id.delivery);

        // chuyển qua tab detail
        getTabHost().setCurrentTab(1);
    }
};

```

✿ **Bước 8:** Refesh lại danh sách sau khi thực thi phần insert.

```
private View.OnClickListener onSave = new View.OnClickListener() {  
  
    public void onClick(View v) {  
        // TODO Auto-generated method stub  
        Restaurant restaurant = new Restaurant();  
        // tham chieu den cac view trong layout  
        EditText name = (EditText)findViewById(R.id.name);  
        EditText address = (EditText)findViewById(R.id.addr);  
        RadioGroup types = (RadioGroup)findViewById(R.id.types);  
  
        restaurant.setName(name.getText().toString());  
        restaurant.setAddress(address.getText().toString());  
  
        // kiem tra xem user chon loai nao  
        switch (types.getCheckedRadioButtonId()) {  
            case R.id.take_out:  
                restaurant.setType("Take out");  
                break;  
            case R.id.sit_down:  
                restaurant.setType("Sit down");  
                break;  
            case R.id.delivery:  
                restaurant.setType("Delivery");  
                break;  
            default:  
                break;  
        }  
        // them vao CSDL  
        helper.insert(restaurant.getName(), restaurant.getAddress(),  
                     restaurant.getType());  
        // refesh lai du lieu  
        curRestaurant.requery();  
    }// end onClick  
}; // end View.OnClickListener
```

✿ **Bước 9:** Biên dịch và chạy chương trình.

=oOo=