Chương 3: LẬP TRÌNH GIAO DIỆN MỨC CAO

3.1. Lập trình giao diện mức cao

3.1.1. View và ViewGroup

3.1.2. Các loại Layout

3.1.3. Các đối tượng view và widget cơ bản

3.1.4. Xử lý sự kiện người dùng tương tác

3.2. Lập trình giao diện mức thấp

3.2.1. Vẽ một đối tượng đơn giản

3.2.2. Làm cho đối tượng di chuyển

3.2.3. Xử lý sự kiện trong UI mức thấp

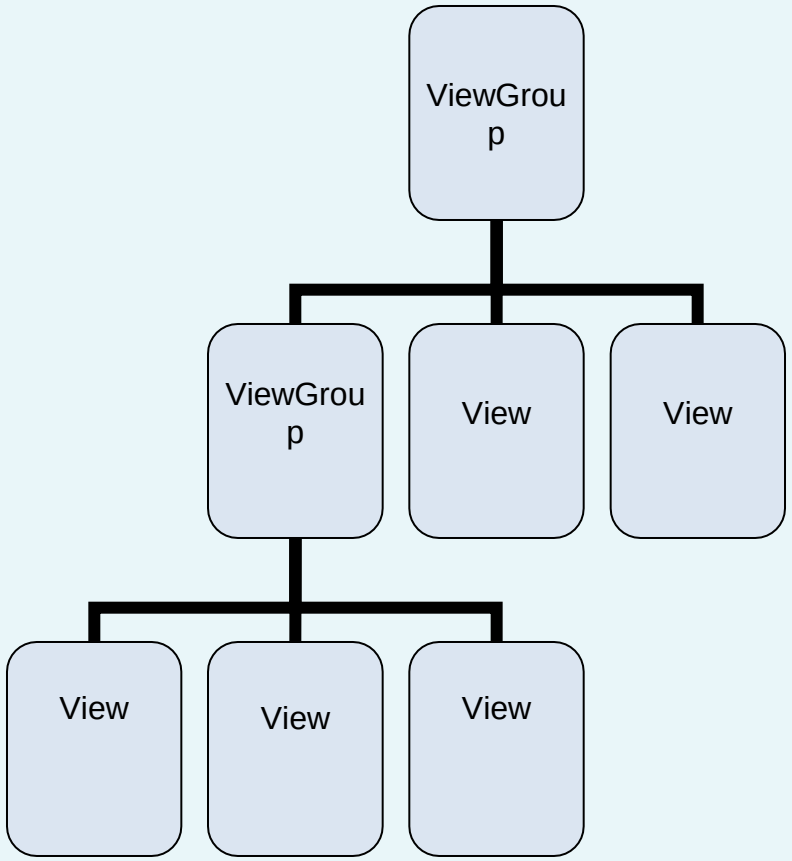
3.2.4. Lập trình luồng trong giao diện mức thấp

3.2.5. Lập trình với SurfaceView

3.1. Lập trình giao diện mức cao
3.1.1. View và ViewGroup

3.1.1. View và ViewGroup

- Giao diện người dùng mức cao được xây dựng từ các đối tượng View (widget) và ViewGroup (Layout), các lớp này nằm trong gói android.view và android.widget.
- Mỗi View và ViewGroup được định nghĩa bởi tập tin **XML** trong **thư mục layout** của project với các **thuộc tính** cần thiết.
- Để hiện một UI thì trong hàm onCreate của mỗi Activity cần phải được gọi hàm setContentView(R.layout.main); hàm này sẽ load giao diện từ file XML lên để phân tích thành mã bytecode.



Một ViewGroup có thể chứa nhiều View và một ViewGroup có thể lồng vào trong một ViewGroup khác.

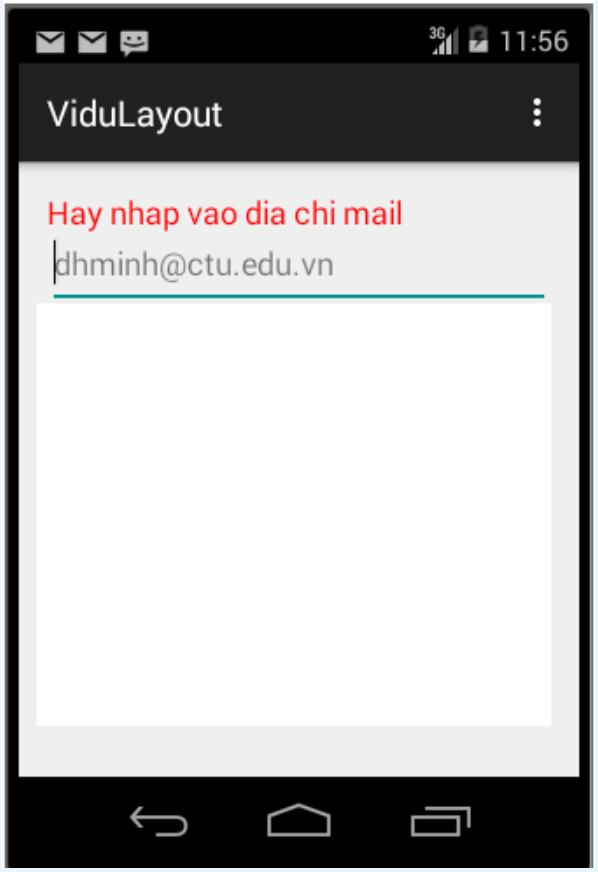
3.1. Lập trình giao diện mức cao
3.1.1. View và ViewGroup

- Khi xây dựng một UI, ta cần phải xác định vị trí và kích thước của các view (hay widget). Sau đây là các **đơn vị dùng để xác định kích thước** các widget.

- **px** (pixel): điểm ảnh cơ bản, nhỏ nhất, trên màn hình.
- **pt** (point): bằng 1/72 inch.
- **dp (**density - independent pixel): với màn hình có độ phân giải 160 px/inch thì 1dp = 1px, đây là đơn vị được khuyên dùng khi xác định kích thước của view trên layout, vì khí đó view có kích thước thay đổ tỉ lệ với kích thước màn hình, vị trí của nó không bị lệch khi hiển thị lên các màn hình có độ phân giải khác nhau.
- **sp** (Scale – Independent pixel): gần giống dp, nhưng được khuyên dùng cho text size.

3.1. Lập trình giao diện mức cao
3.1.1. View và ViewGroup

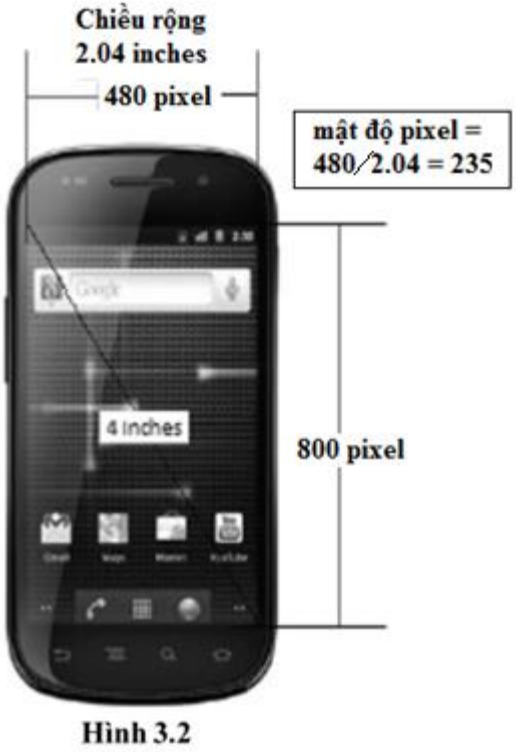
```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:layout_width="fill_parent"
       android:layout_height="wrap_content“
       android:textColor="#ff0000"
        android:text="Hãy nhập địa chỉ email "
        android:textSize="18sp"/>
 <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="dhminh@ctu.edu.vn"
        android:textSize="18sp"/>
</LinearLayout>
```



http://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html

3.1. Lập trình giao diện mức cao
3.1.1. View và ViewGroup

- Kích thước màn hình được định nghĩa bởi đơn vị inch theo đường kính hoặc độ phân giải theo chiều ngang và chiều dọc. Ta thấy mật độ pixel phụ thuộc vào kích thước màn hình và độ phân giải.
- Công thức đổi từ dp sang px:
- pixels = dp * (dpi / 160)
- Android định nghĩa và công nhận bốn mật độ màn hình:
- **Low density (*ldpi*)** — 120 dpi
- **Medium density (*mdpi*)** — 160 dpi
- **High density (*hdpi*)** — 240 dpi (HD)
- **Extra High density (*xhdpi*)** — 320 dpi (Full HD)



Hình 3.2

màn hình là 4 inches tính theo đường chéo và độ phân giải là 480 (ngang) x 800 (dọc) pixel

3.1. Lập trình giao diện mức cao
3.1.2. Các đố tượng view cơ bản

- Các đối tượng view cơ bản hỗ trợ hiển thị thông tin văn bản (text), hình ảnh và cho phép người dùng tương tác lên nó để thực hiện các chọn lựa, chúng là lớp con của lớp View. Có thể chia thành các nhòm như sau:

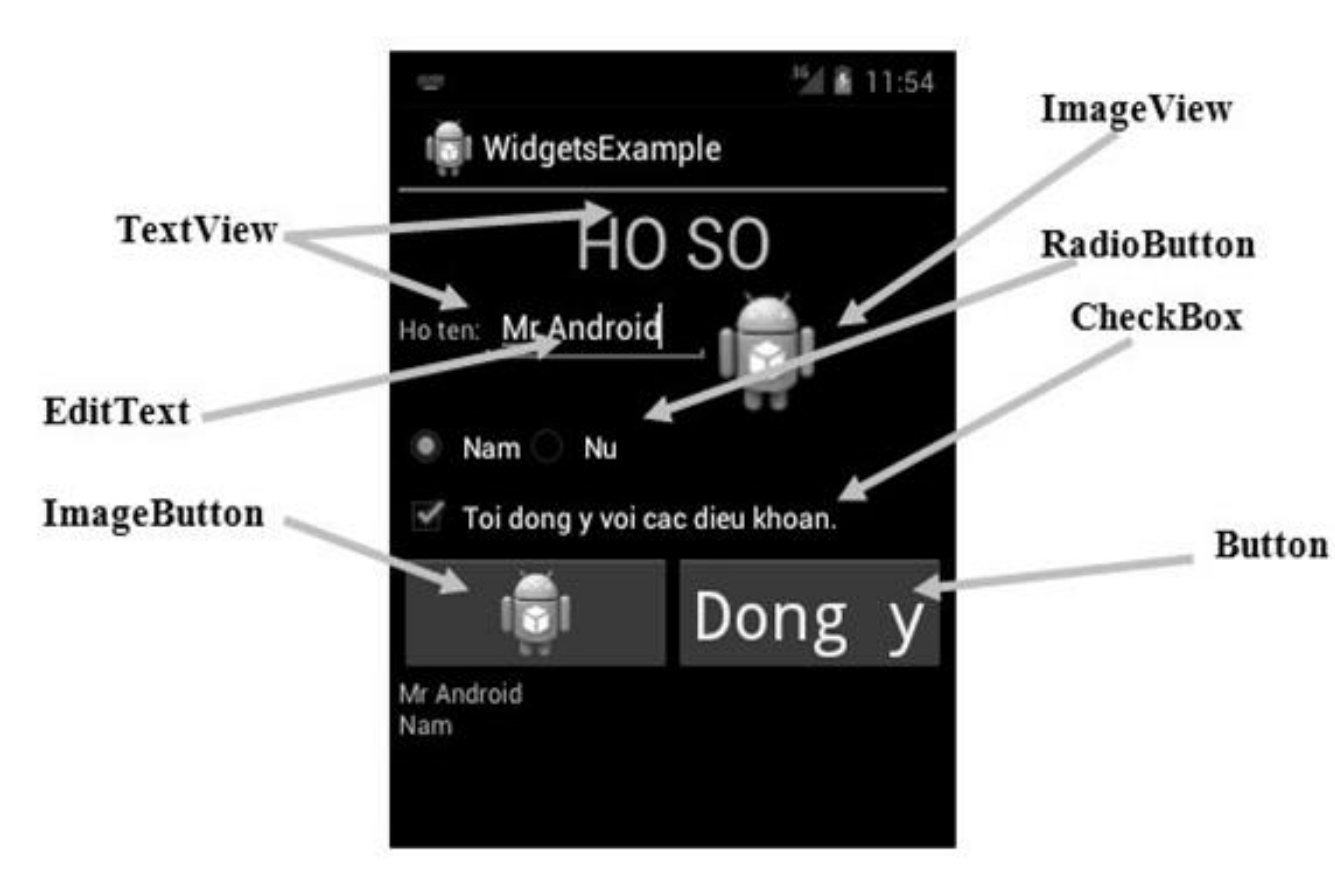
1. **Nhóm hiển thị văn bản:**
   - **TextView:** Hỗ trợ hiển thị văn bản (text) nhưng không cho phép chỉnh sửa. TextView nằm trong gói android.widget, khi sử dụng phải khai báo lệnh import android.widget.TextView.
   - **EditText:** Đối tượng này cho phép người dùng nhập vào hoặc chỉnh sửa văn bản. Trong Android đối tượng EditText được sử dụng như một TextField hoặc một TextBox, khi sử dụng phải khai báo lệnh import android.widget.EditText.
   - **AutoCompleteTextView:** là một view tương tự như EditText (nó là một lớp con của EditText), ngoại trừ việc nó hiển thị một danh sách câu gợi ý hoàn thành một cách tự động trong khi người dùng nhập văn bản.

3.1. Lập trình giao diện mức cao
3.1.2. Các đố tượng view cơ bản

2. Nhóm nút điều khiển

- **Button:** cho phép người dùng tương tác để thực hiện các hoạt động được định nghĩa bởi người lập trình như: cancel, back, run,…. Khi sử dụng phải khai báo lệnh import android.widget.Button.
- **ImageButton:** tương tự như button nhưng có hiển thị một ảnh và không hiển thị văn bản. Khi sử dụng phải khai báo lệnh import android.widget.ImageButton.
- **CheckBox:** có hai trạng thái: Checked hoặc Unchecked, nhận hai giá trị true hoặc false. Đối tượng CheckBox cho phép chọn nhiều mục (item) cùng một lúc. Khi sử dụng phải khai báo lệnh import android.widget.CheckBox.
- **RadioGroup và RadioButton:** RadioButton cũng là loại nút nhấn có hai trạng thái: Checked và Unchecked. Một RadioButton khi đã check chọn thì không thể uncheck. RadioGroup được sử dụng để nhóm một hoặc nhiều RadioButton. → chỉ được check chọn một RadioButton trong cùng một nhóm. Khi sử dụng phải khai báo lệnh import android.widget.RadioGroup và import android.widget.RadioButton.
- **ToggleButton:** cũng là loại nút nhấn có hai trạng thái: Checked và Unchecked, nhưng có chỉ thị sáng lên và mặc định gắn với nói chữ ON hoặc OFF. Khi sử dụng phải khai báo lệnh import android.widget.ToggleButton.

3.1. Lập trình giao diện mức cao
3.1.2. Các đố tượng view cơ bản



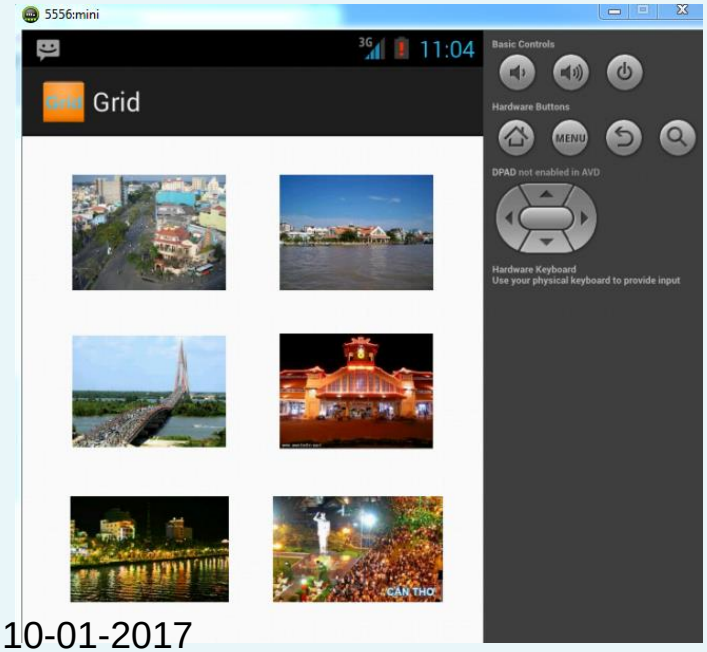
Một số view cơ bản

3.1. Lập trình giao diện mức cao
3.1.2. Các đố tượng view cơ bản

3. Nhóm hiển thị hình ảnh

- **ImageView:** dùng để hiển thị một hình ảnh, có thể tải hình ảnh từ nhiều nguồn khác nhau (chẳng hạn như resources hoặc content providers), thông thường các ảnh này được lưu vào thư mục drawable của project. Khi sử dụng phải khai báo lệnh **import android.widget.ImageView.**
- **ListView:** Không phải là một đối tượng hiển thị cụ thể, nó hiển thị một danh sách các mục (item) trên giao diện. Các mục dược duyệt bằng cách cuộn theo chiều dọc trên một danh sách. Các mục này chứa trên một lớp ListAdapter kết hợp với ListView này. ListAdapter là Adapter mở rộng, là cầu nối giữa một ListView và các dữ liệu sao lưu danh sách. ListView có thể hiển thị bất kỳ dữ liệu với điều kiện là dữ liệu đó được gói trong một ListAdapter. Khi sử dụng phải khai báo lệnh import android.widget.ListView và import android.widget.ListAdapter.
- **Gallery và ImageSwitcher** : là các view hỗ trợ hiển thị các mục (như hình ảnh) trong danh sách cuộn ngang.
- **GridView**: Tương tự như Gallery, nhưng GridView hỗ trợ hiển thị các mục tin trong một mạng lưới di chuyển hai chiều
- **WebView:** cho phép ta nhúng một trình duyệt web trong activity.
- **MapView:** Được dùng để hiển thị bản đồ.

3.1. Lập trình giao diện mức cao
3.1.2. Các đố tượng view cơ bản



Gallery hiển thị các ảnh thu nhỏ trên thanh ngang

GridView hiển thị các ảnh thu nhỏ trong không gian 2 chiều

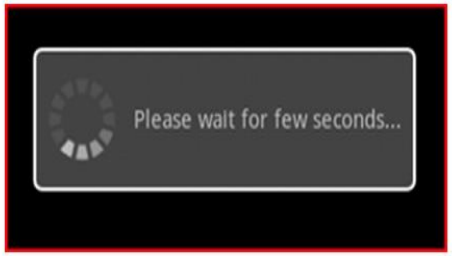
3.1. Lập trình giao diện mức cao
3.1.2. Các đố tượng view cơ bản

4. Nhóm hộp thoại

- **Dialog:** Hộp thoại Android với tùy chọn (option). Trong Android, bằng cách sử dụng hộp thoại chúng ta có thể chọn một tùy chọn từ nhiều tùy chọn bằng cách sử dụng nút radio. Khi sử dụng phải khai báo lệnh import android.app.Dialog.
- **Toast:** Về cơ bản cũng có chức năng tương tự như Dialog nhưng Toast chỉ hiển thị một cảnh báo nhỏ mà không làm ảnh hưởng tới chương trình và nó tồn tại trong một khoảng thời gian cố định (tùy chọn khi lập trình). Khi sử dụng phải khai báo lệnh import android.widget.Toast .

Hình 3.7: Hộp thoại thông thường

Hình 3.8: Hộp thoại có tùy chọn



Hình 3.9: Hiển thị tiến trình trong hộp thoại

3.1. Lập trình giao diện mức cao
3.1.2. Các đố tượng view cơ bản

5. Nhóm Menu

- **Options Menu:** Hiển thị những thông tin liên quan đến Activity hiện tại. Để kích hoạt Options Menu người dùng cần nhấn nút Menu trên thiết bị. Khi sử dụng phải khai báo lệnh import android.view.Menu.
- **Context Menu:** Hiển thị những thông tin liên quan đến một đối tượng View cụ thể. Trong Android, để kích hoạt ContextMenu người dùng cần nhấn và giữ một đối tượng (Button, ImageView…). Khi sử dụng phải khai báo lệnh import android.view.MenuItem.

6. Nhóm điều chỉnh thời gian

- **DatePicker:** Cho phép người dùng chọn ngày, tháng năm. Khi sử dụng phải khai báo lệnh import java.util.Date và import android.widget.DatePicker.
- **TimePicker:** Cho phép người sử dụng chọn thời gian của một ngày, trong cả hai chế độ 24h hoặc chế độ AM, PM. Khi sử dụng phải khai báo lệnh import android.widget.TimePicker.

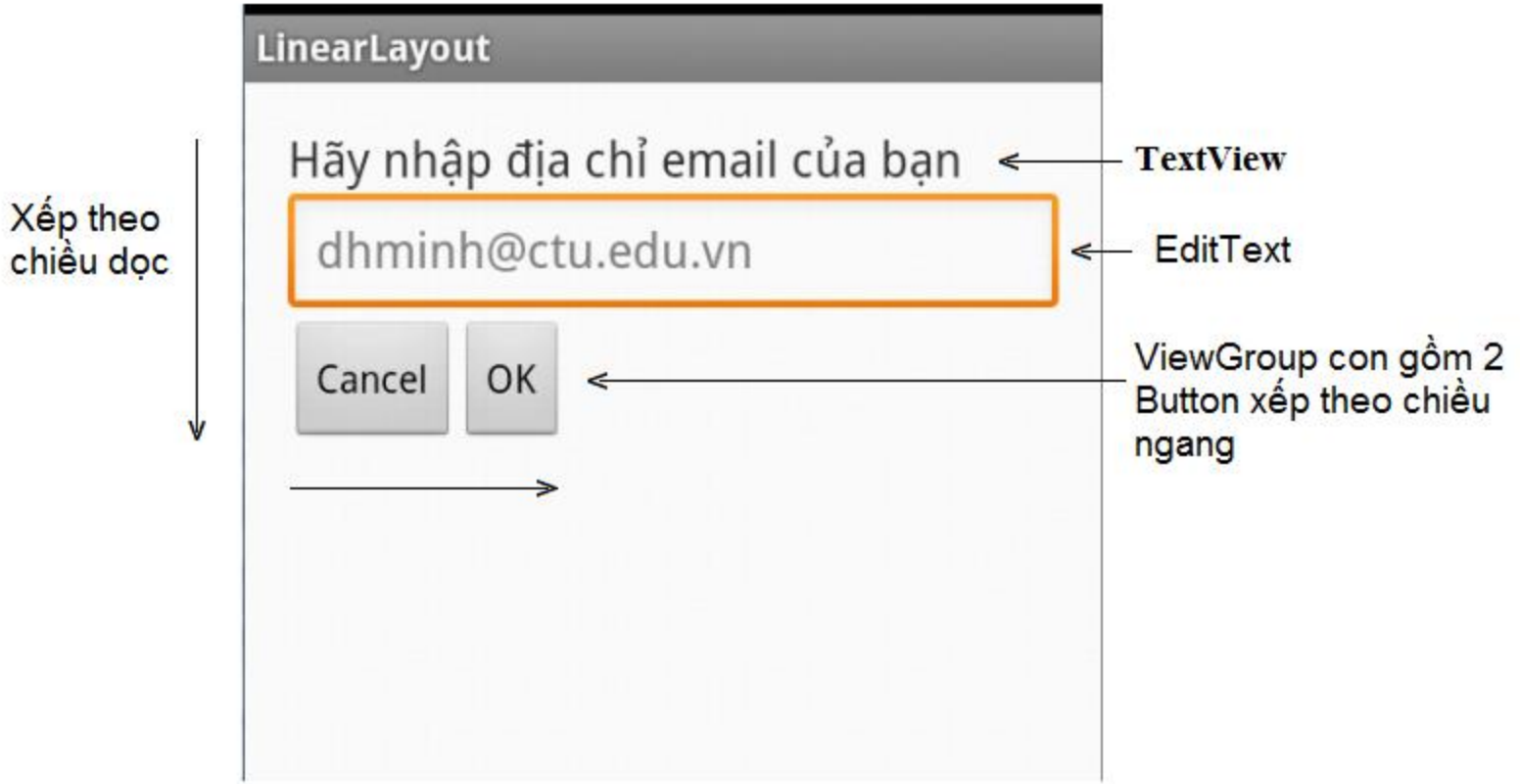
3.1. Lập trình giao diện mức cao

3.1.2. Layout

1. **LinearLayout:**
   - LinearLayout được dùng để bố trí các thành phần giao diện theo chiều ngang hoặc chiều dọc. LinearLayout làm cho các thành phần trong nó không bị phụ thuộc vào kích thước của màn hình.
   - LinearLayout có thuộc tính cần chú ý là android:orientation. Có 2 giá trị là “vertical” và “horizontal” .
2. **AbsoluteLayout:** Layout này được sử dụng để bố trí các widget vào một vị trí bất kì trong layout dựa vào 2 thuộc tính toạ độ x, y.
3. **TableLayout:** Layout này được sử dụng khi cần thiết kế một table chứa dữ liệu hoặc cần bố trí các widget theo các row và column.
4. **RelativeLayout:** Layout này cho phép bố trí các widget theo một trục đối xứng ngang hoặc dọc. Để đặt được đúng vị trí thì các widget cần được xác định một mối ràng buộc nào đó với các widget khác. Các ràng buộc này là các ràng buộc trái, phải, trên, dưới so với một widget hoặc so với layout parent.
5. **FrameLayout:** FrameLayout được dùng để bố trí các đối tượng theo lớp. Những đối tượng nào thuộc Layer bên dưới thì sẽ bị che khuất bởi các đối tượng thuộc Layer nằm trên. Có thể thêm nhiều FrameLayout chồng lên nhau và layout mới sẽ đè lên layout trước nó.

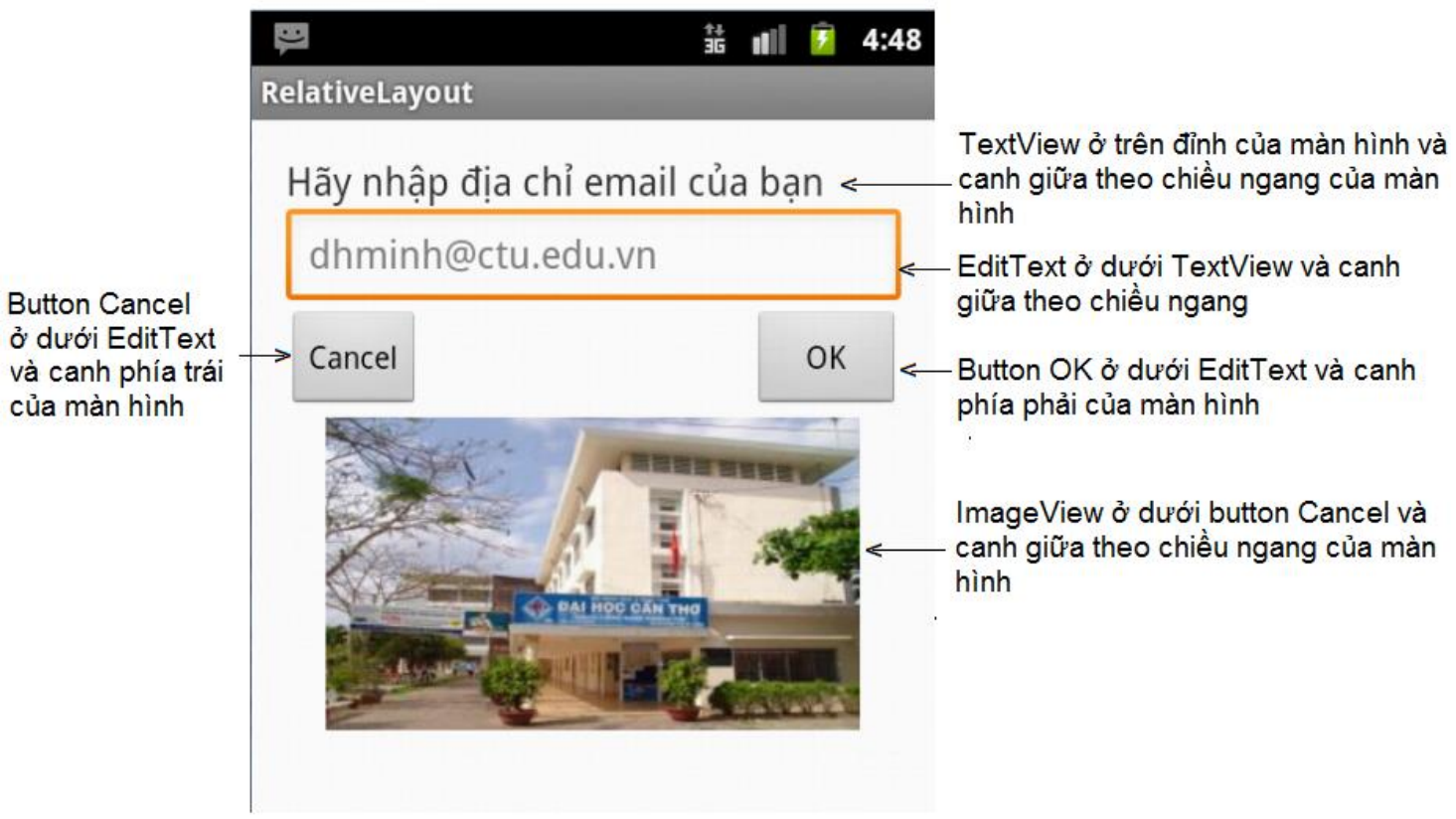
C3.1. Lập trình giao diện mức cao
3.1.2. Layout

LinearLayout



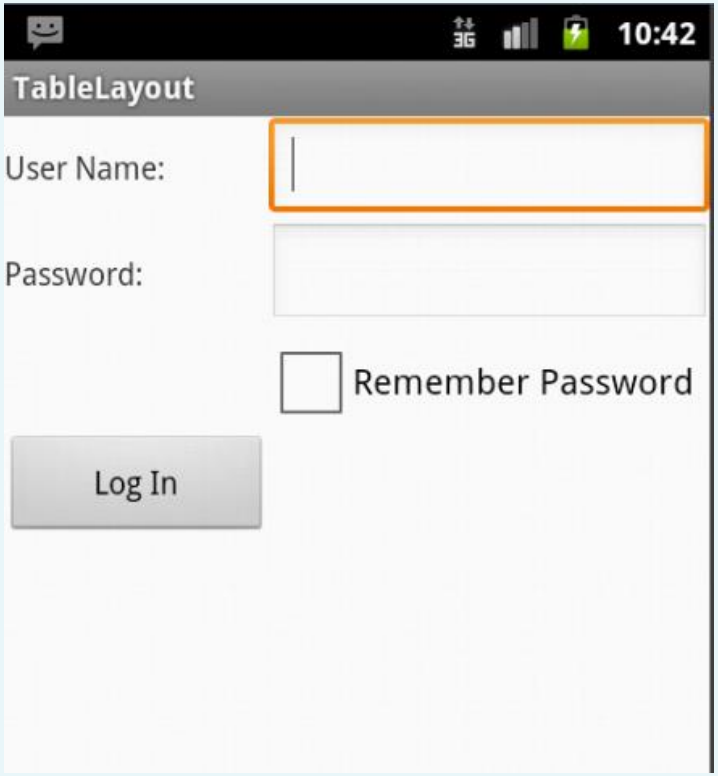
3.1. Lập trình giao diện mức cao
3.1.2. Layout

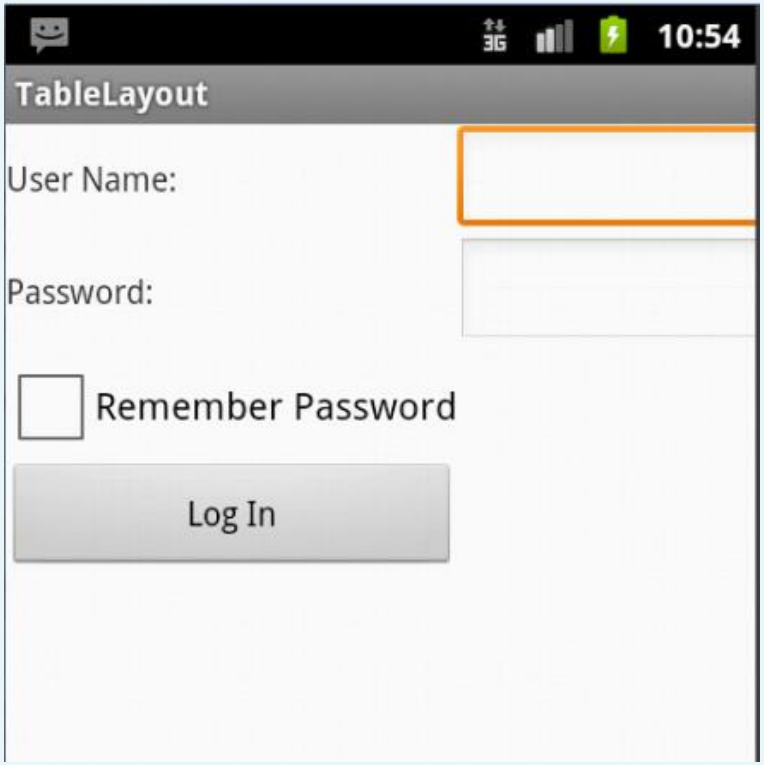
RelativeLayout



3.1. Lập trình giao diện mức cao
3.1.2. Layout

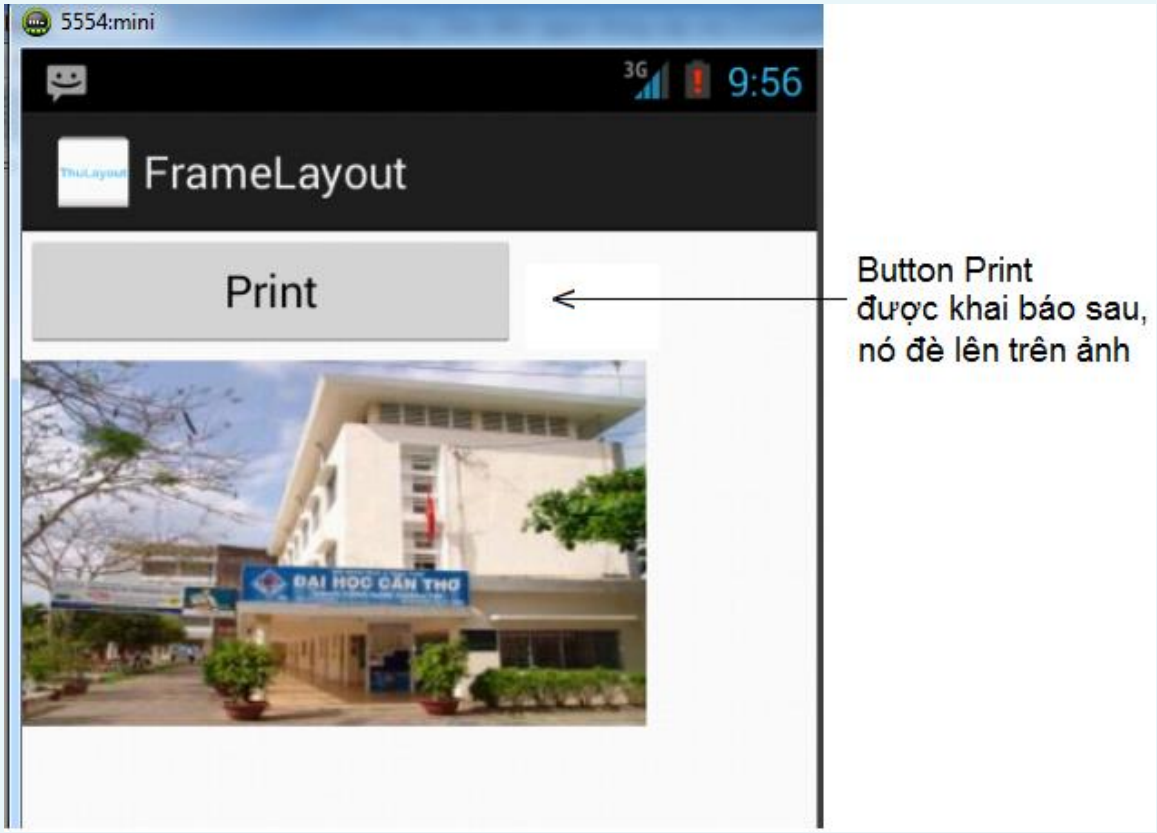
Tablelayout





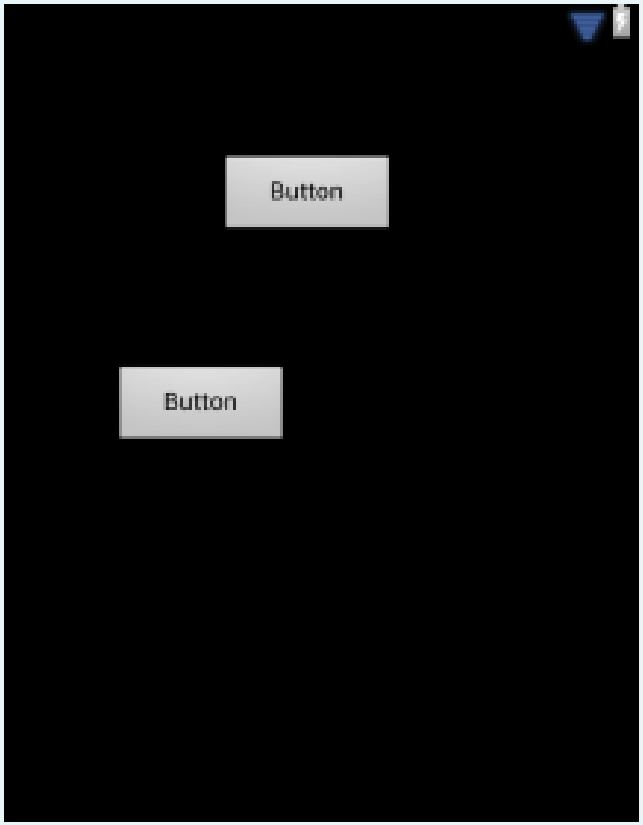
3.1. Lập trình giao diện mức cao
3.1.2. Layout

FrameLayout



3.1. Lập trình giao diện mức cao
3.1.2. Layout

AbsoluteLayout



3.1. Lập trình giao diện mức cao
3.1.4. Xử lý sự kiện người dùng tương tác

- **Có hai dạng xử lý sự kiện:**
  - sự kiện khi người dùng tương tác trực tiếp lên đối tượng thông qua màn hình cảm ứng
  - sự kiện khi sử dụng phím của thiết bị
- **Xử lý sự kiện tương tác trực tiếp**
  - Các đối tượng tương tác: button, text view, edit text, image, list item,…
  - Để bắt sự kiện này cần sử dụng phương thức:

  public void onClick(View v) kế thừa từ lớp android.view.View.

```
public class Vidu extends Activity {
   /** Called when the activity is first created. */
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   Button button = (Button)this.findViewById(R.id.btn);
   button.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {
// TODO Auto-generated method stub
   EditText edit=(EditText) findViewById(R.id.editText);
   String  message=edit.getText().toSyring();
   TextView tx = (TextView)findViewByid(R.id.tx);
   textview.setText(message);
                              }
                    });
      }
   }
```

3.1. Lập trình giao diện mức cao

3.1.4. Xử lý sự kiện người dùng tương tác

Xử lý sự kiện từ phím

- Sự kiện này sử dụng phương thức:

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

kế thừa từ lớp:

android.view.KeyEvent.

```
public class Vidu extends Activity {
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
final TextView text = (TextView)Vidu.this.findViewById(R.id.tx);
text.setOnKeyListener(new View.OnKeyListener() {
public boolean onKey(View v, int keyCode, KeyEvent event){
    if (event.getAction() == KeyEvent.ACTION_DOWN
        && keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
            text.setText("Hello world!");
    return true;
                        }
   else {
            return false;
                        } } } );
   }   }
```

http://developer.android.com/reference/android/view/KeyEvent.html

3.1. Lập trình giao diện mức cao

3.1.4. Xử lý sự kiện người dùng tương tác

LẬP TRÌNH GIAO DIỆN MỨC THẤP

- Giao diện mức cao cung cấp những đối tượng được dựng sẵn
- Với những ứng dụng đòi hỏi sự mới lạ hoặc game được hỗ trợ bởi giao diện mức thấp
- Trong Android, để vẽ ta cần phải có bốn thành phần cơ bản:
  - Một đối tượng kiểu Bitmap để giữ các pixel cần vẽ.
  - Một đối tượng chứa nét vẽ cần vẽ ra (có thể là Rect, Path, Bitmap…).
  - Một đối tượng kiểu Paint dùng để định nghĩa màu sắc, style….
  - Một đối tượng Canvas dùng để thực thi lệnh vẽ.
- Khi vẽ một đối tượng cần xác định vị trí mà nó xuất hiện trên màn hình. → dùng hệ trục tọa độ với tâm là điểm trái trên của màn hình. Giá trị x tăng dần về phía phải, giá trị y tăng dần xuống phía dưới.
- Các phương thức sau đây sẽ giúp biết được chiều rộng và chiều cao của canvas (không thể chỉ định kích thước cho canvas, mà sẽ trả về diện tích lớn nhất có thể có đối với một thiết bị cho trước::
  - c.getWidth(): xác định chiều rộng của canvas c
  - c.getHeight(): xác định chiều cao của canvas c

3.2. LẬP TRÌNH GIAO DIỆN MỨC THẤP

Android cung cấp **android.graphics framework** để thực hiện việc vẽ tự do:

- đối tượng **Canvas** quyết định vẽ hình dạng gì
- đối tượng **Paint** quyết định vẽ như thế nào.
- Bước quan trọng nhất trong việc vẽ đối tượng tuỳ ý là ghi đè lên phương thức **onDraw ().**
- **Tham số của onDraw ()** là một đối tượng Canvas.
- **Lớp Canvas** cung cấp các phương thức vẽ với nhiều kiểu như: vẽ văn bản (drawText), đường thẳng (drawLines), vẽ bitmap (drawBitmap), vẽ hình chữ nhật (drawRect), v.v...

3.2.1. Lập trình với lớp View (Vẽ một đối tượng đơn giản)

1. Vẽ một đối tượng đơn giản

- Lớp Panel kế thừa của lớp View. Trong đó sẽ cài đặt lại hai phương thức khởi tạo và ghi đè phương thức onDraw().
- đối tượng Context cho phép truy xuất đến các đối tượng cũng như các dịch vụ của hệ thống. Đối tượng này cần thiết để vẽ một giao diện ra màn hình thiết bị.
- phương thức onDraw(Canvas c) chính là phương thức vẽ giao diện.

```
public class Panel extends View {
    private final Paint paint = new Paint();
    public Panel(Context context){
            super(context);
            }
@Override
protected void onDraw(Canvas c){
            super.onDraw(c);
            paint.setColor(Color.WHITE);
            c.drawPaint(paint);
            paint.setColor(Color.RED);
            c.drawCircle(150, 60, 20, paint);
            }
}
```

3.2.1. Lập trình với lớp View (Vẽ một đối tượng đơn giản)

- Đối tượng Paint được dùng định nghĩa màu sắc, style… những gì ta cần vẽ ra. Nói một cách đơn giản thì đối tượng paint đóng vai trò như một biến có thể gán và sử dụng nhiều giá trị cùng lúc, nếu muốn thay đổi giá trị ta chỉ việc gán lại chứ không cần phải khai báo mới.
- Để chương trình được thực thi, trong tập tin Activity cần phải khai báo và gọi lớp Panel

```
paint.setColor(Color.WHITE);
//Gán màu sắc cho đối tượng paint
c.drawPaint(paint);
//drawPaint vẽ nền cho chương trình
paint.setColor(Color.RED);
//Gán lại màu sắc cho đối tượng paint
c.drawCircle(150, 60, 20, paint);
//Vẽ hình tròn với các thuộc tính của nó
```

```
package vidu.view;
import android.app.Activity;
import android.os.Bundle;
 public class ViduView extends Activity {
Panel vidu;
@Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      vidu = new Panel(this);
      setContentView(vidu);
   }
}
```

3.2.2. Làm cho đối tượng di chuyển

- Ví dụ: làm cho đối tượng di chuyển với tốc độ cho sẵn, khi chạm biên sẽ quay lại.
- Khai báo thêm các thuộc tính:
  - public int x=0;
  - public int y=0;
  - private int Vx=1,Vy=1;
  - private int duongkinh = 30;
- Để xác định được giá trị này trong phương thức onDraw() ta bổ sung thêm lệnh:

  duongkinh = bitmap.getWidth();
- Sau đó bổ sung thêm phương thức update():

```
private void update() {
 // Get new (x,y) position
     x = x + Vx;
     y = y + Vy;
 //Neu cham bien thi quay lai va doi Ball
  if (x<=0||(x>=getWidth()-duongkinh))
    {
       Vx = -Vx;
     }
     if (y<=0||y>=getHeight()-duongkinh)
   {
       Vy = -Vy;
     }
   invalidate();  // Ve lai
  }
```

3.2.2. Làm cho đối tượng di chuyển

- Cuối cùng thay đổi trong phương thức onDraw() như kề bên:
  - Với phương thức onDraw() này thì giao diện sẽ được vẽ ra màn hình đối tượng tại vị trí (x,y) với giá trị mặc định (0,0).
  - Sau đó gọi phương thức update() để tính toán lại tọa độ (x,y) mới.
  - Trong phương thức update() ta gọi phương thức invalidate(), đây là một cách gián tiếp để gọi đệ quy lại onDraw().
  - Bằng cách này giao diện sẽ liên tục được vẽ với giá trị (x,y) luôn thay đổi tạo cảm giác đối tượng đang di chuyển trên nền cố định.

```
@Override
protected void onDraw(Canvas c){
   super.onDraw(c);
   paint.setColor(Color.WHITE);
   c.drawPaint(paint);
   Bitmap bitmap =
BitmapFactory.decodeResource(get
Resources(),R.drawable.ball);
    c.drawBitmap(bitmap, x, y, paint);
    duongkinh = bitmap.getWidth();
     update();
    }
```

3.2.3. Xử lý sự kiện trong giao diện mức thấp

- Giống như lập trình giao diện mức cao, các đối tượng giao diện mức thấp cũng có thể được bắt và xử lý sự kiện.

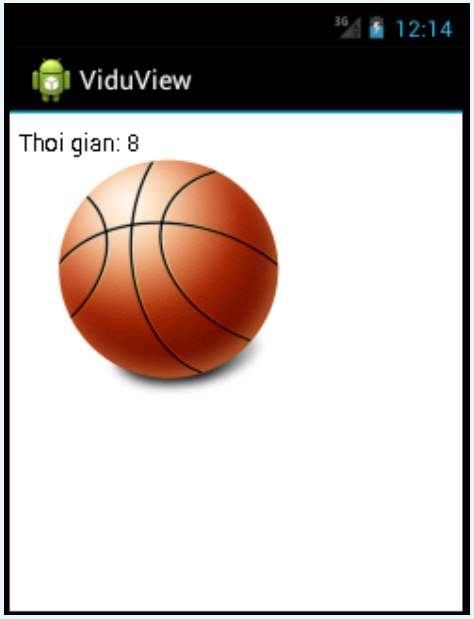
```
public boolean onTouchEvent(MotionEvent event) {
int eventaction = event.getAction();
switch (eventaction) {
        case MotionEvent.ACTION_DOWN:
 // hanh dong se thuc thi khi cham vao man hinh
            break;
        case MotionEvent.ACTION_MOVE:
  // hanh dong xay ra khi ngon tay di chuyen tren man hinh
            break;
        case MotionEvent.ACTION_UP:
  // hanh dong xay ra khi ngon tay roi khoi man hinh
            break;
    }
    return true;
}
```

3.2.4. Lập trình luồng trong giao diện mức thấp

- **Luồng** là giao một công việc cần thực hiện cho một đối tượng. Sau khi giao công việc cho đối tượng đó thì nó sẽ được thực thi đồng thời với các công việc chính đang được thực hiện. Bằng cách này, một công việc có thể được **thực thi đồng thời với các công việc** khác mà không phải chờ đợi để được thực thi tuần tự.
- Từ project trên, ta ví dụ về luồng như sau: Bổ sung thêm một đồng hồ đếm giờ cho Panel. Tiến hành đếm ngược từ 30 giây trở về 0 và in ra "Thoi gian: số giây". Khi đếm đến 0 thì ngừng in đối tượng ra màn hình và thay vào đó là câu "Game over!". Ở đây ta thấy có hai công việc đang được thực hiện đồng thời:
  - **Một là:** vẽ ra màn hình đoạn text: "Thời gian: số giây" và vẽ ra đối tượng di chuyển.
  - **Hai là:** tuần tự giảm giá trị của biến đếm mỗi lần một đơn vị. Sau khi giảm giá trị thì ngừng lại 1 giây và thực hiện lại công việc.
- **Lớp Panel được sửa lại**

3.2.4. Lập trình luồng trong giao diện mức thấp

- Lớp **TimeCount** kế thừa lớp Thread dùng để khai báo một luồng cho java.
  - Lớp này chỉ có một phương thức chính là run(). Phương thức này sẽ được thực thi khi luồng được khởi động.
  - Luồng được tạo ra từ lớp TimeCount làm nhiệm vụ đếm ngược biến count, mỗi lần đếm xong sẽ ngừng lại 1000 ms và thực hiện lại công việc cho đến khi count<=0. Rồi sau đó bổ sung thêm thuộc tính TimeCount time = new TimeCount();
- Sau khi thực hiện xong các bước trên ta đã được một luồng tên là time. Tuy nhiên đến đây thì luồng time này vẫn chưa được khởi động. Để khởi động luồng trên ta **bổ sung lệnh time.start()** vào phương khởi tạo của Panel để sau khi khởi tạo đối tượng xong sẽ khởi động luồng time.



3.2.5. Lập trình với SurfaceView

- SurfaceView là một bề mặt chuyên sử dụng để vẽ. Nó là một lớp kế thừa từ View và bổ sung thêm một số phương thức cũng như tính năng hỗ trợ cho việc vẽ liên tục lên bề mặt.
- Để hiểu rõ hơn ta sẽ tiến hành tạo một poject mới sử dụng SurfaceView để vẽ ra màn hình một đối tượng và cho nó di chuyển như đã làm với View.
- Ta định nghĩa một lớp (VD: tên là **GamePanel)** kế thừa **SurfaceView** và cài đặt lại **SurfaceHolder** bởi lệnh **interface SurfaceHolder.Callback.** Trong đó có một phương thức khởi tạo và ba phương thức xử lý các sự kiện: surface được tạo ra, surface thay đổi các thông số và khi surface bị hủy.

3.2.5. Lập trình với SurfaceView

```
import android.content.Context; import android.view.SurfaceHolder;
import android.view.SurfaceView;
public class GamePanel extends SurfaceView implements
SurfaceHolder.Callback {
public GamePanel(Context context) {
      super(context);
    }
    public void surfaceCreated(SurfaceHolder holder) {
    }
    public void surfaceChanged(SurfaceHolder holder, int
format, int width, int height) {
    }
    public void surfaceDestroyed(SurfaceHolder holder) {
    }
}
```

3.2.5. Lập trình với SurfaceView

- Qua ví dụ về lớp **View** và lớp **SurfaceView** ta có thể thấy đối với **View** tất cả các đối tượng được vẽ trên cùng một luồng giao diện và luồng này cũng được sử dụng cho tất cả tương tác người dùng.
- Đối với **SurfaceView**, thì có thể vẽ trên các luồng riêng biệt, do đó nếu ứng dụng cần cập nhật giao diện nhanh chóng thì nên sử dụng **SurfaceView**. Lớp này còn có ưu điểm là xử lý chuyển động nhanh và có thể dùng nhiều tài nguyên.