# Chương 8

# Xây dựng ứng dụng trên Android

8.0 Dẫn nhập
8.1 Tổng quát về lập trình di động
8.2 Các tính chất cơ bản của 1 ứng dụng Android
8.3 So sánh J2ME và Android
8.4 File Manifest and APK cho ứng dụng Android
8.5 Một số khả năng lập trình ứng dụng Android
8.6 Giao diện với người dùng
8.7 Lập trình loại giao diện API cấp cao
8.8 Lập trình loại giao diện Canvas
8.9 Kết chương

# 8.0 Dẫn nhập

- Chương này sẽ trình bày các tính chất cơ bản về lập trình ứng dụng di động trên Android, các tính chất về việc lập trình, đóng gói ứng dụng và đặc tả ứng dụng Android.
- Chương này cũng sẽ giới thiệu các khả năng lập trình giao diện ứng dụng Android khác nhau, cách tạo các cửa sổ giao diện ứng dụng thuộc 1 trong 2 loại : dùng các đối tượng API cấp cao hay lập trình dùng API cấp thấp cùng các thí dụ cụ thể.

# 8.1 Tổng quát về lập trình trên Android

- Platform Android do Google phát triển và phát hành gồm 3 phần chính :
  - Hệ điều hành Android quản lý các thiết bị di động.
  - Thư viện middleware gồm các package và các class java để hỗ trợ lập trình ứng dụng chạy trên hệ thống Android.
  - Tập các ứng dụng Android cơ bản luôn cài sẵn trên bất kỳ hệ thống Android nào.
- Chương này giới thiệu 1 số tính chất và khả năng lập trình cơ bản trên Android thông qua thư viện middleware của Android.

# 8.2 Các tính chất cơ bản của 1 ứng dụng Android

- Chương trình Android có cấu trúc hướng đối tượng : gồm từ 1 đến n class đối tượng hợp thành, các class được viết bằng ngôn ngữ Java.
- Một ứng dụng Android có thể chứa các components sau :
  - Activity : mỗi activity là 1 cửa sổ giao diện của chương trình.
  - Service : mỗi Service thực hiện 1 số chức năng nhưng không có giao diện.
  - Content Provider : cung cấp phương tiện dùng chung dữ liệu giữa các ứng dụng.
  - Broadcast Receiver : đáp ứng với các sự kiện bên ngoài chương trình, như có tin nhắn, có điện thoại tới,…
- Chương trình phải có ít nhất 1 Activity, các components khác có thể có hoặc không tùy nhu cầu của từng phần mềm.

# 8.3 So sánh J2ME và Android

- J2ME (Java 2 Micro Edition) là một thư viện nền Java để phát triển các ứng dụng trên thiết bị di động nay đã lỗi thời.
- Đều cung cấp khả năng lập trình cửa sổ giao diện theo 2 mức độ : dùng các đối tượng giao diện cấp cao hay tự vẽ lấy giao diện từ các tác vụ vẽ cơ bản.
- Cung cấp các package chứa các class dịch vụ để thực hiện các công việc phổ biến như các hoạt động I/O, các tác vụ hệ thống, các hoạt động multimedia, xác định vị trí, gởi/nhận tin nhắn SMS, các dịch vụ điện thoại,….
- Nhìn chung các package thư viện của Android phong phú hơn J2ME, chứa nhiều đối tượng hơn, các đối tượng mạnh hơn so với đối tượng cùng chủng loại của J2ME.
- Đặc biệt là Android cung cấp thư viện SQLite để truy xuất database dễ dàng.

# 8.4 File Manifest and APK cho ứng dụng Android

- Mỗi chương trình Android gồm nhiều class chức năng và các file tài nguyên hình ảnh, âm thanh,… Tất cả các class này, sau khi được dịch ra mã bytecode, sẽ được tích hợp lại thành 1 file duy nhất *.apk. Ta sẽ copy file này và cài đặt vào thiết bị android trước khi có thể dùng chương trình ứng dụng tương ứng.
- Trong quá trình viết chương trình android, ngoài việc viết mã nguồn Java để đặc tả các class chức năng cấu thành phần mềm, ta phải viết 1 file XML có tên là AndroidManifest.xml để đặc tả các thông tin cấu hình của phần mềm Android.
- Các môi trường lập trình như Eclipse, NetBeans... đều tạo tự động file AndroidManifest.xml từ project phần mềm Android, ta chỉ cần thêm các tag lệnh đặc tả mới cho các thông tin mà ta muốn, thí dụ như xin đọc/ghi file trên thẻ nhớ ngoài, xử lý tin nhắn SMS...

# 8.4 File Manifest and APK cho ứng dụng Android

- Sau đây là nội dung cụ thể của 1 file manifest đặc tả ứng dụng Android giải phương trình bậc 2 (bài thực hành 8.1):

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.test"
    android:versionCode="1"
    android:versionName="1.0">
  <application android:label="@string/app_name" >
    <activity android:name="MainActivity"
          android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

# 8.4 File Manifest and APK cho ứng dụng Android

- Còn đây là cấu trúc của 1 file manifest đặc tả ứng dụng Android :

```
<manifest>
    <uses-permission />   <permission />   <permission-tree />   <permission-group />
    <instrumentation />   <uses-sdk />   <uses-configuration />   <uses-feature />
    <supports-screens />   <compatible-screens />   <supports-gl-texture />
    <application>
      <activity>  <intent-filter>   <action /> <category />
          <data />   </intent-filter>   <meta-data />   </activity>
      <activity-alias>   <intent-filter> . . . </intent-filter>   <meta-data />   </activity-alias>
      <service>   <intent-filter> . . . </intent-filter>   <meta-data/>   </service>
      <receiver>   <intent-filter> . . . </intent-filter>   <meta-data />   </receiver>
      <provider> <grant-uri-permission /> <meta-data />   <path-permission />   </provider>
      <uses-library />
   </application>
</manifest>
```

# 8.5 Một số khả năng lập trình ứng dụng Android

**Chu kỳ sống của Activity Android** : Chương trình android đơn giản chỉ chứa 1 Activity. Activity Android thường chứa các tác vụ chức năng sau đây :

- onCreate() : được gọi 1 lần khi Activity được tạo ra.
- onStart() : được gọi trước khi Activity hiển thị lên màn hình.
- onResume() : được gọi ngay sau onStart nếu Activity đang foreground trên màn hình.
- onPause() : được gọi khi hệ thống gọi onResume() Activity khác.
- onStop() : được gọi ngay sau Activity không được dùng nữa.
- onDestroy() : được gọi ngay trước khi Activity bị hủy.

# 8.5 Một số khả năng lập trình ứng dụng Android

**Đọc/ghi file** : dùng các class trong package java.io để truy xuất file y như trong ứng dụng Java chạy trên platform khác.

- Thí dụ về ghi file :

```
//tạo đối tượng File miêu tả file cần truy xuất
FileOutputStream fOut = new
    FileOutputStream("/storage/extSdCard/data.txt");
OutputStreamWriter myOutWriter = new
    OutputStreamWriter(fOut);
myOutWriter.append(txtData.getText());
myOutWriter.close();
fOut.close();
```

- Lưu ý là phải thêm tag lệnh XML

```
<uses-permission android:name=
"android.permission.WRITE_EXTERNAL_STORAGE" />
```

vào file manifest để đăng ký quyền ghi file trên thẻ nhớ ngoài.

# 8.5 Một số khả năng lập trình ứng dụng Android

- Thí dụ về đọc file :

```
//tạo đối tượng File miêu tả file cần truy xuất
FileInputStream fIn = new FileInputStream(/storage/extSdCard/data.txt);
BufferedReader myReader = new BufferedReader(new
    InputStreamReader(fIn));
String aDataRow = ""; String aBuffer = "";
while ((aDataRow = myReader.readLine()) != null) {
    aBuffer += aDataRow + "\n";
}
txtData.setText(aBuffer); myReader.close();
```

- Lưu ý là phải thêm tag lệnh XML

```
<uses-permission android:name=
"android.permission.READ_EXTERNAL_STORAGE" />
```

vào file manifest để đăng ký quyền đọc file trên thẻ nhớ ngoài.

# 8.5 Một số khả năng lập trình ứng dụng Android

## Gởi tin nhắn

- Để gởi 1 tin nhắn, ứng dụng gởi phải biết địa chỉ và port giao tiếp của ứng dụng nhận. Thí dụ lập trình để gởi tin nhắn :

```
SmsManager smsMgr = Smsmanager.getDefault();
smsMgr.sendTextMessage(addr, null, message,null, null);
```

- Lưu ý là phải thêm tag lệnh XML

```
<uses-permission
   android:name="android.permission.SEND_SMS"/>
<uses-permission  android:name="android.permission.WRITE_SMS"
   />
```

vào file manifest để đăng ký quyền gởi/ghi tin nhắn.

# 8.5 Một số khả năng lập trình ứng dụng Android

## Nhận tin nhắn

- Để nhận/đọc tin nhắn, trước hết ta phải thêm tag lệnh XML

```
<uses-permission android:name=
   "android.permission.RECEIVE_SMS"/>
<uses-permission android:name=
   "android.permission.READ_SMS" />
```

vào file manifest để đăng ký quyền nhận/đọc tin nhắn.

- Để nhận 1 tin nhắn, ta sẽ viết 1 class SmsReceiver thừa kế class BroadcastReceiver theo template sau :

```
public class SmsReceiver extends BroadcastReceiver {
   //override tác vụ onReceive để nhận và xử lý tin nhắn
   public void onReceive(Context context, Intent intent) {
      Bundle bundle = intent.getExtras();
      SmsMessage[] msgs = null;      String str = "";
```

# 8.5 Một số khả năng lập trình ứng dụng Android

```
        if (bundle != null) {
            //xử lý SMS message vừa nhận
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];
            for (int i=0; i<msgs.length; i++) {
                msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                str += "SMS from " + msgs[i].getOriginatingAddress() +
                        " :“ + msgs[i].getMessageBody().toString() + "\n";
            }
            //hiển thị nội dung thông báo SMS
            Toast.makeText(context, str,Toast.LENGTH_SHORT).show();
        }
    }
}
```

# 8.5 Một số khả năng lập trình ứng dụng Android

## Gọi điện thoại :

- Ta có thể dùng đoạn code sau để gọi 1 cuộc điện thoại :

```
private void phoneCall() {
    String phoneCallUri = "tel:911";     //số điện thoại đối tác
    Intent phoneCallIntent = new Intent(Intent.ACTION_CALL);
    phoneCallIntent.setData(Uri.parse(phoneCallUri));
    //bắt đầu gọi điện thoại cho đối tác
    startActivity(phoneCallIntent);
}
```

- Lưu ý là phải thêm tag lệnh XML sau vào file manifest để đăng ký quyền gọi điện thoại :

```
<uses-permission
    android:name="android.permission.CALL_PHONE" />
```

# 8.5 Một số khả năng lập trình ứng dụng Android

**Location** : cách dễ dàng nhất để xử lý các dịch vụ xác định vị trí GPS là thực hiện 3 bước sau :

1. định nghĩa class Activity của ứng dụng sẽ hiện thực interface LocationListener:

   public class MyProgram extends Activity implements LocationListener {…}

2. đăng ký class ứng dụng xử lý các sự kiện GPS :

```
//xác định đối tượng quản lý Location
LocationManager lm = (LocationManager) this.getSystemService
        (Context.LOCATION_SERVICE);
//đăng ký hàm xử lý sự kiện onLocationChanged
lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,
 3000,1, this);
```

# 8.5 Một số khả năng lập trình ứng dụng Android

3. hiện thực các tác vụ xử lý sự kiện GPS như :

```
//định nghĩa hàm xử lý sự kiện onLocationChanged
public void onLocationChanged(Location location) {
    //hiển thị cao độ, vĩ độ và kinh độ của vị trí hiện hành
    tvOuput.setText("Bạn đang ở vị trí : "+
    "\nVĩ độ : " + location.getLatitude() +
    "\nKinh độ : " + location.getLongitude() +
    "\nCao độ : " + location.getAltitude());
}
//định nghĩa hàm xử lý sự kiện onProviderDisabled
public void onProviderDisabled(String provider) {…}
//định nghĩa hàm xử lý sự kiện onProviderEnabled
public void onProviderEnabled(String provider) {…}
```

# 8.5 Một số khả năng lập trình ứng dụng Android

- Lưu ý, để ứng dụng được phép dùng dịch vụ GPS, trước hết ta phải thêm các tag lệnh XML sau vào file manifest của ứng dụng :

```
<uses-permission android:name=
   "android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name=
   "android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name=
   "android.permission.INTERNET"/>
```

# 8.5 Một số khả năng lập trình ứng dụng Android

## Đọc/ghi database dùng thư viện SQLite

- Thí dụ về đọc/ghi database dùng thư viện SQLite : tạo và truy xuất 1 database có tên là qlsinhvien.db chứa 2 bảng dữ liệu :
  - Bảng tbllop có 3 field nội dung : malop (text), tenlop (text), siso (integer).
  - Bảng tblsinhvien có 3 field nội dung : masv (text), tensv (text), malop (text).

```
private static String DB_PATH =
    "/data/data/YOUR_PACKAGE/databases/";
private static String DB_NAME = "qlsinhvien.db";
SQLiteDatabase db;
//tạo mới hay mở file database
Db = SQLiteDatabase.openDatabase (DB_PATH + DB_NAME, null,
    SQLiteDatabase.CREATE_IF_NECESSARY);
```

# 8.5 Một số khả năng lập trình ứng dụng Android

## Đọc/ghi database dùng thư viện SQLite

```
//tạo mới 1 bảng dữ liệu
String sql = "CREATE TABLE tbllop (malop TEXT primary key, tenlop TEXT,
    siso INTEGER);
Database.execSQL(sql);
//thêm/sửa/xóa 1 record trong 1 bảng dữ liệu
ContentValues val = new ContentValues();
val.put("malop","DHTH7C");
val.put("tenlop","Đại học 7C");
val.put("siso", 30);
If (database.insert("tbllop", null, val)==-1)
    msg = "Lỗi khi thêm record vào bảng tbllop.";
else msg = "Thêm record vào bảng tbllop thành công.";
Toast.makeText(this, msg, Toast.LENGTH_LONG).show();
```

# 8.5 Một số khả năng lập trình ứng dụng Android

## Đọc/ghi database dùng thư viện SQLite

```
//sửa 1 record trong 1 bảng dữ liệu
ContentValues val = new ContentValues();
val.put("tenlop","Đại học 7C mới");
if (database.update("tbllop", val, "malop=?", new String[] {"DHTH7C"})==0)
    msg = "Lỗi khi sửa record trong bảng tbllop.";
else msg = "Sửa record trong bảng tbllop thành công.";
Toast.makeText(this, msg, Toast.LENGTH_LONG).show();
//xóa 1 record trong bảng dữ liệu
Database.delete("tbllop", "malop=?", new String[] {"DHTH7C"});
```

# 8.5 Một số khả năng lập trình ứng dụng Android

## Đọc/ghi database dùng thư viện SQLite

```
//xóa file database
String msg="";
if (deleteDatabase(DB_PATH + DB_NAME) == true) {
      ssg = "Xóa database thành công.";
else msg = "Lỗi khi xóa database";
Toast.makeText(this, msg, Toast.LENGTH_LONG).show();
//và các hoạt động xử lý khác : nhìn chung cũng giống như lập trình truy xuất
    database dùng ngôn ngữ truy vấn SQL.
```

# 8.6 Giao diện với người dùng

Mỗi cửa sổ ứng dụng Android là 1 Activity, nó kết hợp với 1 đối tượng View tại từng thời điểm. Đối tượng View được kết hợp có thể là 1 trong 2 dạng sau đây :

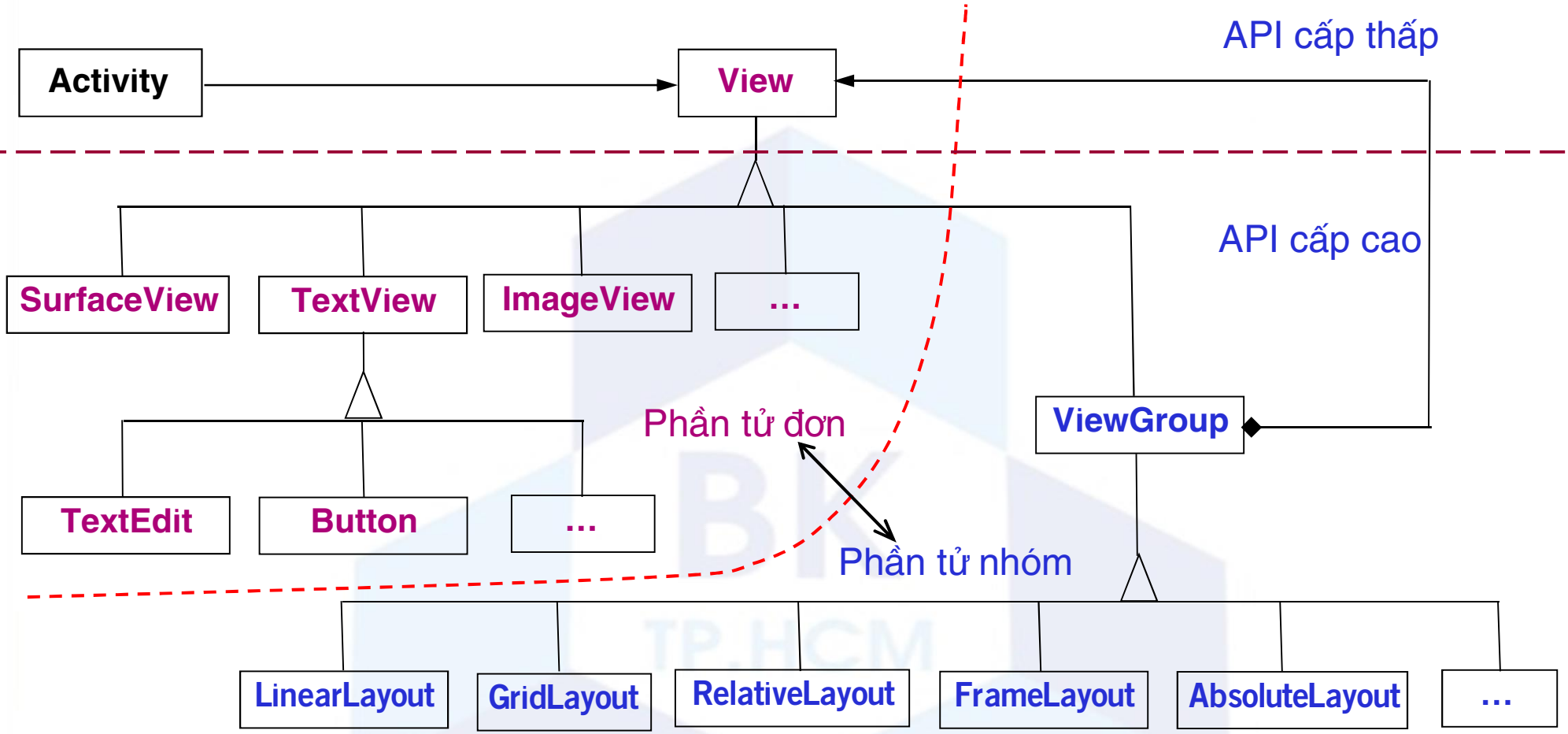
1. Đối tượng API cấp cao :
   - Là đối tượng có chức năng rõ ràng, xác định. Thí dụ như ImageView, TextView,…
   - Thường là đối tượng Layout chứa nhiều đối tượng con bên trong. Tùy theo tính chất Layout mà các đối tượng con sẽ được sắp đặt như thế nào. Đối tượng con có thể là Layout khác theo cơ chế cây phân cấp.
   - Các Layout được hỗ trợ là AbsoluteLayout, LinearLayout, GridLayout, TableLayout, RelativeLayout, FrameLayout.
   - Dễ dàng hiện thực

# 8.6 Giao diện với người dùng

2. Đối tượng API cấp thấp :

- View chỉ là 1 màn hình trắng, chưa có bất kỳ thứ gì. Người lập trình tự do tạo các đối tượng theo nhu cầu riêng thông qua các tác vụ vẽ các đối tượng cơ bản như chuỗi, ảnh bitmap, các đối tượng đồ họa cơ bản như đoạn thẳng, hình chữ nhật, hình cong…
- Người lập trình sẽ kiểm soát đầy đủ mọi sự kiện bàn phím và mọi sự kiện touch trên màn hình cảm ứng.
- Vì phải lập trình giao tiếp chi li nên khó hiện thực hơn nhiều so với giao diện dùng các đối tượng API cấp cao.

# 8.6 Giao diện với người dùng



# 8.7 Lập trình loại giao diện API cấp cao

- Cửa sổ dùng giao diện cấp cao thường là 1 ViewGroup chứa nhiều đối tượng giao diện nhỏ theo yêu cầu của ứng dụng. Thí dụ cửa sổ ứng dụng giải phương trình bậc 2 $ax^2 + bx + c = 0$ là 1 ViewGroup gồm :
  - 3 TextView để hiển thị chuỗi yêu cầu người dùng nhập 3 tham số a, b, c.
  - 3 EditText để giúp người dùng nhập 3 tham số a, b, c
  - 1 Button để kích hoạt việc giải phương trình.
  - 3 TextView để hiển thị các kết quả.
- Tùy cách bố trí các phần tử con mà ta có thể dùng tổ hợp các Layout nào cho phù hợp.

# 8.7 Lập trình loại giao diện API cấp cao

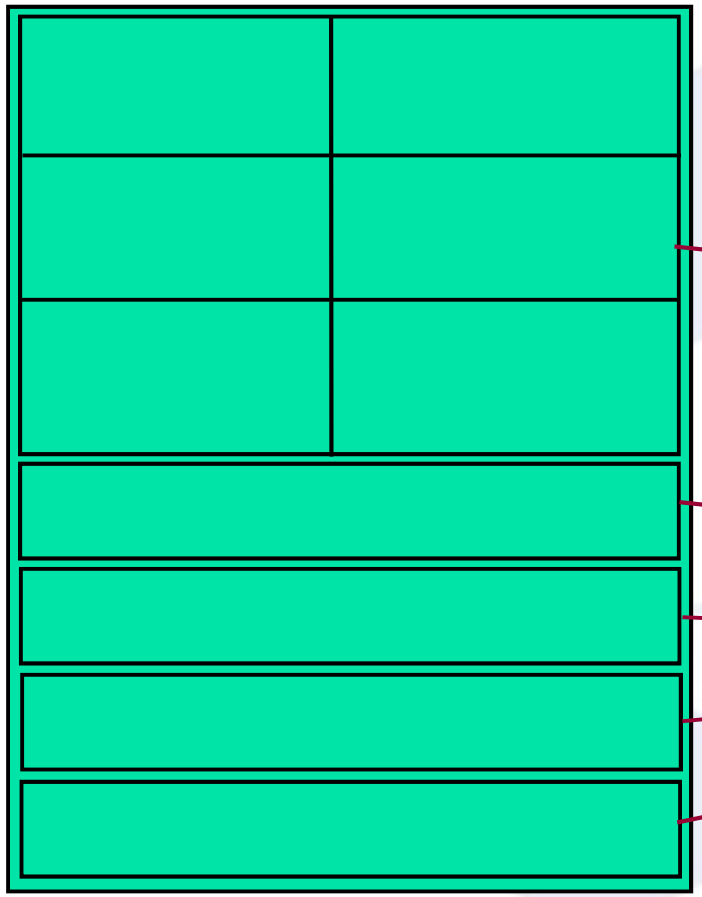
- Android cung cấp nhiều loại Layout khác nhau như :
  - LinearLayout : các phần tử con được sắp xếp tuần tự theo thứ tự được thêm vào ViewGroup (từ trên xuống, từng hàng từ trái sang phải).
  - TableLayout : các phần tử con được sắp xếp theo dạng bảng 2 chiều gồm nhiều hàng, mỗi hàng gồm nhiều cột.
  - RelativeLayout : vị trí của từng phần tử con được xác định tương đối so với phần tử đi trước nó hay so với phần tử cha.

# 8.7 Lập trình loại giao diện API cấp cao

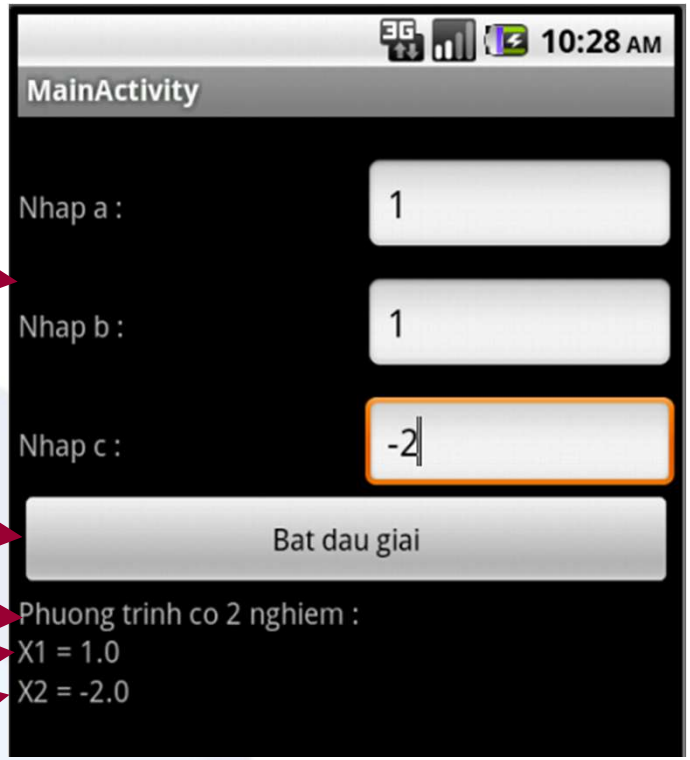
- FrameLayout : các phần tử con được sắp xếp tuần tự theo thứ tự được thêm vào ViewGroup nhưng ở cùng 1 vị trí trên trái của ViewGroup, do đó chúng sẽ chồng lên nhau và ta chỉ thấy 1 trong chúng tại từng thời điểm.
- GridLayout : các phần tử con được sắp xếp theo dạng bảng 2 chiều gồm nhiều hàng, mỗi hàng gồm nhiều cột (gần giống với TableLayout).
- AbsoluteLayout : vị trí của từng phần tử con được xác định độc lập gồm tọa độ đỉnh trên trái, độ rộng, độ cao của nó.

# 8.7 Lập trình loại giao diện API cấp cao

- Thí dụ, các phần tử trong cửa sổ ứng dụng giải phương trình bậc 2 có thể được tổ chức theo hình sau :



TableLayout




LinearLayout

# 8.7 Lập trình loại giao diện API cấp cao

Thí dụ về đoạn code tạo cửa sổ giải phương trình bậc 2 :

```
//tác vụ onCreate của class Activity miêu tả ứng dụng
public void onCreate(Bundle savedInstanceState) {
    //gọi tác vụ của cha thực hiện trước
    super.onCreate(savedInstanceState);
    //tạo Layout LinearLayout miêu tả toàn bộ cửa sổ
    LinearLayout linearLayout = new LinearLayout(this);
    params = new LayoutParams(FILL_PARENT,
     WRAP_CONTENT);
    linearLayout.setLayoutParams(params);
    linearLayout.setOrientation(VERTICAL);
    //tạo 1 TableLayout chứa 3 hàng 2 cột
    TableLayout tl = new TableLayout(this);
```

# 8.7 Lập trình loại giao diện API cấp cao

```
tl.setPadding(0, 10, 0, 0);
tl.setStretchAllColumns(true);
//tạo hàng đầu để nhập tham số a
TableRow tableRow = new TableRow(this);
tableRow.setPadding(0, 10, 0, 0);
//tạo 1 TextView để hiển thị thông báo yêu cầu nhập tham số a
lblA = new TextView(this); lblA.setText("Nhập a : ");
tableRow.addView(lblA);
//tạo 1 ExitText để người dùng nhập tham số a
txtA = new EditText(this);
tableRow.addView(txtA);
//add hàng nhập tham số a vào tableLayout
tl.addView(tableRow);
```

# 8.7 Lập trình loại giao diện API cấp cao

```
//tạo hàng thứ 2 để nhập tham số b
tableRow = new TableRow(this);
tableRow.setPadding(0, 10, 0, 0);
//tạo 1 TextView để hiển thị thông báo yêu cầu nhập tham số b
lblB = new TextView(this); lblB.setText("Nhập b : ");
tableRow.addView(lblB);
//tạo 1 ExitText để người dùng nhập tham số b
txtB = new EditText(this);
tableRow.addView(txtB);
//add hàng nhập tham số b vào tableLayout
tl.addView(tableRow);
//tạo hàng thứ 3 để nhập tham số c
tableRow = new TableRow(this);
tableRow.setPadding(0, 10, 0, 0);
```

# 8.7 Lập trình loại giao diện API cấp cao

```
//tạo 1 TextView để hiển thị thông báo yêu cầu nhập tham số c
lblC = new TextView(this); lblC.setText("Nhập c : ");
tableRow.addView(lblC);
//tạo 1 ExitText để người dùng nhập tham số c
txtC = new EditText(this);
tableRow.addView(txtC);
//add hàng nhập tham số b vào tableLayout
tl.addView(tableRow);
//add table vào Layout
linearLayout.addView(tl);
//tạo button và add vào linearLayout
button = new Button(this); button.setText("Bắt đầu giải");
linearLayout.addView(button);
```

# 8.7 Lập trình loại giao diện API cấp cao

```
    //thiết lập đối tượng xử lý sự kiện Click vào button
    button.setOnClickListener(submitListener);
    //tạo 3 TextView hiển thị kết quả và add vào linearLauout
    lblKetqua = new TextView(this);
    linearLayout.addView(lblKetqua);
    lblX1 = new TextView(this);
    linearLayout.addView(lblX1);
    lblX2 = new TextView(this);
    linearLayout.addView(lblX2);
    //kết hợp cửa sổ giao diện linearLayout cho ứng dụng
    setContentView(linearLayout);
}
```

# 8.7 Lập trình loại giao diện API cấp cao

- Để xử lý sự kiện Click (chạm tay và thả ra) trên từng đối tượng con của cửa sổ, ta hiện thực tác vụ xử lý sự kiện tương ứng của interface OnClickListener theo template như sau :

```
private View.OnClickListener submitListener =
    new View.OnClickListener() {
        public void onClick(View view) {
         //code xử lý sự kiện touch vào đối tượng
    }
    }
```

- Để xử lý các sự kiện kết hợp với cửa sổ (View), ta hiện thực các tác vụ xử lý tương ứng của View (giới thiệu chi tiết trong slide 38) :

# 8.8 Lập trình loại giao diện API cấp thấp

- Cửa sổ dùng API cấp thấp thường là đối tượng con của View. Để hiển thị nội dung của cửa sổ ứng dụng, ta sẽ override tác vụ onDraw(Canvas canvas) của class cha View.
- Phân tích nội dung cần hiển thị bất kỳ, ta thấy nội dung này là tập hợp nhiều chuỗi văn bản, nhiều phần tử ảnh bitmap, nhiều phần tử đồ họa toán học như hình chữ nhật, hình tròn,...
- Tham số canvas của tác vụ onDraw() là đối tượng cung cấp các tác vụ vẽ các loại phần tử khác nhau. Việc xuất nội dung bất kỳ lên cửa sổ ứng dụng là quá trình lặp gọi từng tác vụ của đối tượng canvas để vẽ từng phần tử cấu thành nội dung phức tạp.
- Trong các slide kế, ta sẽ giới thiệu các đoạn code mẫu thực hiện vẽ các loại nội dung cần hiển thị.

# 8.8 Lập trình loại giao diện API cấp thấp

Xuất chuỗi văn bản :

```
protected void onDraw(Canvas canvas) {
    //xác định vùng chữ nhật chứa chuỗi
    int xl = 0, yt = 20, h = 20;
    //xóa vùng hình chữ nhật chứa chuỗi
    paint.setStyle(Paint.Style.FILL); paint.setAntiAlias(true);
    paint.setColor(Color.argb(255,40,40,40));
    canvas.drawRect (xl, yt, this.getWidth(), yt+h, paint);
    //thiết lập các thuộc tính về font chữ cần dùng
    paint.setColor(Color.argb(255,255,0,255));
    paint.setTextSize(12);
    //gọi tác vụ drawText để xuất chuỗi ở vị trí mong muốn
    String s = "Đại học Bách Khoa Tp. HCM";
    canvas.drawText (s, xl, yt+h-4, paint);
}
```

# 8.8 Lập trình loại giao diện API cấp thấp

Xuất đoạn thẳng :

```
protected void onDraw(Canvas canvas) {
    //thiết lập 2 đỉnh của đường chéo trên trái -> dưới phải cần vẽ
    int xl = 0, yt = 0;
    int xr = this.getWidth(), yb = this.getHeight();
    //thiết lập màu vẽ
    paint.setColor(Color.argb(255,255,255,255));
    //vẽ đoạn thẳng qua 2 đỉnh xác định
    canvas.drawLine (xl, yt, xr, yb, paint);
    //vẽ tiếp đường chéo trên phải -> dưới trái
    canvas.drawLine(xr, yt, xl, yb,paint);
}
```

# 8.8 Lập trình loại giao diện API cấp thấp

Xuất khối hình chữ nhật :

```
protected void onDraw(Canvas canvas) {
    //xác định 2 đỉnh chéo của hình chữ nhật cần vẽ
    int xl = 100, yt = 80;
    int xr = 300, yb = 200;
    //thiết lập chế độ tô nền, màu tô nền và tô nền hình chữ nhật
    paint.setStyle(Paint.Style.FILL);
    paint.setColor(Color.argb(255,40,40,100));
    canvas.drawRect (xl, yt, xr, yb, paint);
    //thiết lập chế độ vẽ đường viền, màu vẽ và vẽ đường viền
    paint.setStyle(Paint.Style.STROKE);
    paint.setColor(Color.argb(255,255,0,100));
    canvas.drawRect (xl, yt, xr, yb, paint);
    }
```

# 8.8 Lập trình loại giao diện API cấp thấp

Vẽ vòng tròn :

```
protected void onDraw(Canvas canvas) {
    //xác định tâm và bán kính hình tròn cần vẽ
    int xc = 200, yc = 200;
    int bk = 50;
    //thiết lập chế độ tô nền và màu nền rồi tô nền vòng tròn
    paint.setStyle(Paint.Style.FILL);
    paint.setColor(Color.argb(255,0,0,255));
    canvas.drawCircle (xc, yc, bk, paint);
    //thiết lập chế độ vẽ đường viền và màu vẽ rồi vẽ đường tròn
    paint.setStyle(Paint.Style.STROKE);
    paint.setColor(Color.argb(255,0,255,255));
    canvas.drawCircle (xc, yc, bk, paint);
}
```

# 8.8 Lập trình loại giao diện API cấp thấp

Xuất hình bitmap :

```
protected void onDraw(Canvas canvas) {
    //tạo đối tượng ảnh bitmap từ file ảnh gốc
    Bitmap bm = BitmapFactory.decodeResource(getResources(),
R.drawable.giadinh);
    //xác định vùng hình chữ nhật trong ảnh gốc cần hiển thị
    Rect rs = new Rect();
    rs.left = 0; rs.top = 0;
    rs.right = bm.getWidth(); rs.bottom = bm.getHeight();
    //xác định vùng hình chữ nhật hiển thị ảnh trong View
    Rect rd = new Rect();
    rd.left = 0; rd.top = 300;
    rd.right = 150; rd.bottom = 450;
    //hiển thị vùng rs trong ảnh gốc lên vùng rd của View
    canvas.drawBitmap(bm,rs,rd, paint);
}
```

# 8.8 Lập trình loại giao diện API cấp thấp

Xử lý các sự kiện do người dùng kích hoạt trên View :

```
//override hàm xử lý sự kiện ấn phím
public boolean onKeyDown(int keyCode, KeyEvent event) {…}
//override hàm xử lý sự kiện thả phím
public boolean onKeyUp(int keyCode, KeyEvent event) {…}
//override hàm xử lý sự kiện ấn lặp phím
public boolean onKeyMultiple (int keyCode, int repeatCount,
      KeyEvent event) {…}
//override hàm xử lý các sự kiện cảm ứng
public boolean onTouchEvent(MotionEvent event) {
   if (event.getAction() == MotionEvent.ACTION_DOWN) {
       //code xử lý sự kiện chạm tay vào màn hình cảm ứng
    }
…
```

# 8.9 Kết chương

- Chương này đã trình bày các tính chất cơ bản về lập trình ứng dụng di động trên Android, các điểm giống nhau giữa Android và J2ME cùng một số tính chất về việc lập trình, đóng gói ứng dụng và đặc tả ứng dụng Android.
- Chương này cũng đã giới thiệu các khả năng lập trình giao diện ứng dụng Android khác nhau, cách tạo các cửa sổ giao diện ứng dụng thuộc 1 trong 2 loại : dùng các đối tượng API cấp cao hay lập trình dùng API cấp thấp cùng các thí dụ cụ thể.