# ĐỀ CƯƠNG BÀI GIẢNG

## BÀI 3. XỬ LÝ SỰ KIỆN TRÊN GIAO DIỆN ỨNG DỤNG

**Nội dung bài học trước khi lên lớp (**trang 2 đến 15)

- Thông báo và hội thoại
- Ý nghĩa findViewByID
- Xử lý tương tác với các thành phần cơ bản
- Các kiểu lập trình sự kiện
- ListView cơ bản

**Nội dung bài học thực hiện lên lớp** (Trang 16 đến hết)**:**

- Tùy biến ListView.
- Spinner, TimePicker và DatePicker.

**Nội dung bài học sau khi lên lớp:** Phiếu bài tập bài 1 đến bài 6

### NỘI DUNG BÀI HỌC

## 1. Thông báo và hội thoại

Hộp thoại là một cửa sổ nhỏ nhắc người dùng đưa ra quyết định hoặc nhập thông tin bổ sung.

### 1.1. Toast

Toast là thông báo nhỏ không cho phép người dùng tương tác và xuất hiện trong khoảng thời gian ngắn, sau đó đóng lại. Toast có hai mô tả dạng đầy đủ và dạng ngắn gọn.

- Dạng đầy đủ:

```
Toast t=Toast.makeText(Context, text, duration);
t.setGravity(Gravity.CENTER, 0, 0);
t.show();
```

- Dạng ngắn gọn: Toast.makeText(Context, text, duration).show();

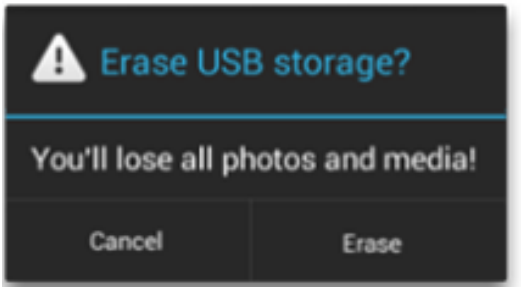
Trong đó: Context là ngữ cảnh hiển thị ; Text là thông báo xuất hiện; Duration là thời gian tồn tại thông báo (Toast.LENGTH_SHORT khoảng 2 giây và Toast.LENGTH_LONG khoảng 3.5 giây)

- Nên sử dụng các giá trị hiển thị theo thời gian như trên, không nên gõ các con số cụ thể

### 1.2. AlertDialog

AlertDialog là cửa sổ hiển thị và cho phép người dùng tương tác với hệ thống. Đây là hộp thoại thường dùng để nhận các quyết định từ phía người dùng.

Giả sử cửa sổ hỏi đáp như sau:




**Tiêu đề**: Erase USB storage?

**Nội dung thông báo**: You''ll lose all photos and media!

**2 nút lệnh xác nhận**: Cancel và Erase.

Tạo thông báo như sau

Bước 1: tạo đối tượng Builder

```
AlertDialog.Builder b=new AlertDialog.Builder(
     YourActivity.this);
```

Bước 2: Thiết lập tiêu đề, nội dung cho Dialog thông qua các phương thức như sau:

```
//đặt tiêu đề
b.setTitle("Erase USB storage");
//đặt nội dung thông báo
b.setMessage("You ''ll lose all photos and media !");
//đặt tên nút lệnh và các xử lý cho nút lệnh Cancel
b.setPositiveButton("Cancel", newDialogInterface.
     OnClickListener() {
     @Override
     Public void onClick(DialogInterface dialog, int which){
          finish();
     }});
//đặttên nút lệnh và các xử lý cho nút Erase.
b.setNegativeButton("Erase", new
     DialogInterface.OnClickListener() {
     @Override
     public void onClick(DialogInterface dialog, int which){
          dialog.cancel();
     }
     });
//gọi hội thoại hiển thị
b.create().show();
```

## 2. Ý nghĩa của findByViewId

Android chia màn hình (activity) thành 2 phần, phần thiết kế giao diện (.xml) và phần xử lý nghiệp vụ (.java). Để truy xuất tới các view trong thiết kế giao diện trong phần nghiệp vụ Android cung cấp hàm truy xuất **findByViewID**.

Mỗi khi đặt id cho bất kỳ 1 view nào thì id là duy nhất và sẽ được tự động phát sinh thành 1 thuộc tính quản lý trong lớp R của Android. Vì vậy khi truy xuất tới View ta dùng R.id.idView.

```
Phần giao diện:
<Button
     android:id="@+id/btnTong2So"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:text="Tổng 2 số" />`
```

```
Phần xử lý nghiệp vụ
Button btnTong2So=(Button)findVeiwByID(R.Id.btnTong2So);
```

## 3. Xử lý tương tác với các thành phần giao diện cơ bản

### 3.1. Xử lý tương tác với TextView

TextView dùng **hiển thị dữ liệu**. Để truy xuất TextView ta cần đặt id trong thanh công cụ Properties hoặc trong XML. Hàm findByViewId đc giới thiệu để truy xuất TextView trong phần nghiệm vụ.

```
TextView txtMess=(TextView)
 findViewByID(R.id.txtMessage).
```

**Thiết lập nội dung hiển thị có 2 cách:**

- Cách 1: trong java code (Activity): **txtMess.setText**(“xin chao”);
- Cách 2: trong XML: `android:text=”xin chao”`

**Để lấy thông tin bên trong control TextView ta dùng lệnh dưới đây**

- `String mess=txtMess.getText().toString();`

Ngoài id, text thì TextView còn có các thuộc tính quan trọng khác và các thao tác cũng được thực hiện tương tự thông qua phương thức **set** của đối tượng

### 3.2. Xử lý tương tác với EditText

EditText kế thừa từ TextView, được dùng thay **đổi nội dung text**, chứa tất cả các thuộc tính của TextView. Các thao tác với id, text, color và các thuộc tính khác thực hiện tương tự.

Android tự động phân EditText thành nhiều loại khác nhau nhờ thuộc tính inputType. Để truy xuất tới điều khiển ta cũng dùng hàm findByViewID

```
EditText editSoa=(EditText) findViewByID(R.id.editSoa).
```

- Thay đổi nội dung hiển thị dùng phương thức **setText**(): editSoa.setText(5);
- Để lấy nội dung ta dùng phương thức **getText**(): editSoa.getText().toString();

### 3.3. Xử lý tương tác với Button

Button được xây dựng từ TextView cho phép thực hiện nhận phản hồi tương tác. Các thuộc tính của Button cũng tương tự TextView và EditText.

Chi tiết các lắng nghe sự kiện sẽ được trình bày trong phần sau (phần 4)

### 3.4. Xử lý tương tác với CheckBox, RadioButton

CheckBox là view cho phép chọn nhiều lựa chọn (nên đặt id bắt đầu là **chk).** **RadioButton** chỉ cho phép 1 lựa chọn tại 1 thời điểm (nên đặt tên bắt đầu bằng **rad**).

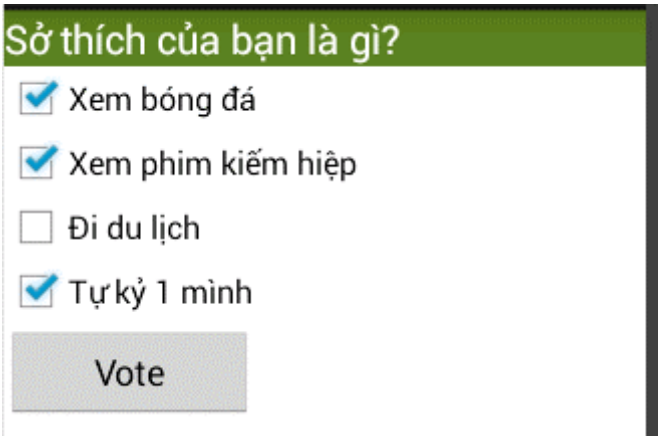
CheckBox và RadioButton đều sử dụng chung 2 phương thức :

- phương thức **setChecked**, dùng để thiết lập checked. Nếu ta gọi setChecked(true) tức là cho phép control được checked, còn gọi setChecked(false) thì control sẽ bị unchecked.

- phương thức **isChecked**, kiểm tra xem control có được checked hay không. Nếu có checked thì trả về true ngược lại trả về false

- Checkbox cho phép ta checked nhiều đối tượng, còn RadioButton thì tại một thời điểm nó chỉ cho ta checked 1 đối tượng trong cùng một group.

Ví dụ: Nếu thiết lập lựa chọn như giao diện sau




- Thiết lập cho Checkbox bất kỳ được checked mặc định trong XML:

```
<CheckBox
    android:id="@+id/checkBox1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:text="Xem bóng đá" />
```

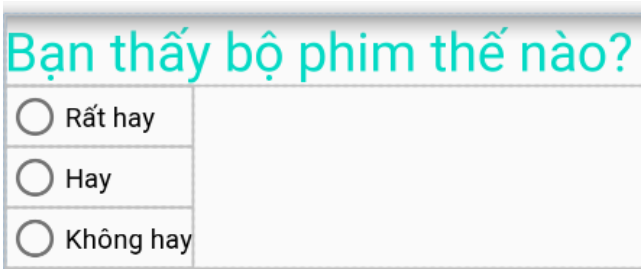
-Trong phần nghiệp vụ để kiểm tra Checkbox đó có được checked hay không xử lý như sau:

```
CheckBox chk=(CheckBox) findViewById(R.id.checkBox1);
if(chk.isChecked())
{
    //xử lý checked
}
else
{
    //xử lý unchecked
}
//Muốn thiết lập checked:
chk.setChecked(true);
//Muốn clear checked:
chk.setChecked(false);
```

Hoặc nếu muốn kiểm tra sự kiện khi người dùng chọn check này ta có thể dùng sự kiện sau

```
//kiểm tra lựa chọn
chk.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
        if (isChecked){
            //xử lý thông tin khi checkBox được chọn
        }
    }
});
```

Nếu muốn đổi lựa chọn sang RadioButton




- Tương tự như Checkbox, ta cũng có thể thiết lập checked cho RadioButton bất kỳ trong XML:

```
<RadioGroup
    android:id="@+id/radioGroup"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:visibility="visible">

    <RadioButton
        android:id="@+id/radRatHay"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Rất hay" />

    <RadioButton
        android:id="@+id/radHay"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Hay" />

    <RadioButton
        android:id="@+id/radKhongHay"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Không hay" />
</RadioGroup>
```

- RadioGroup gom nhóm các RadioButton lại theo nhóm, những RadioButton trong nhóm thì tại 1 thời điểm chỉ có 1 RadioButton được CHỌN (checked). Trong một màn hình ta có thể tạo nhiều nhóm RadioGroup khác nhau.

Có 2 cách xử lý RadioButton nào được checked như sau:

**Cách 1:** Dựa vào RadioGroup để biết chính xác Id của RadioButton nào được checked. Dựa vào Id này ta sẽ xử lý đúng nghiệp vụ:

```
RadioGroup group =findViewById(R.id.radioGroup);
int idChecked=group.getCheckedRadioButtonId();
if (idChecked==R.id.radHay){
```

```
    //xử lý thông tin về radio được chọn
}
```

Như hình trên, bạn thấy hàm **getCheckedRadioButtonId**() : hàm này trả về Id của RadioButton nằm trong RadioGroup 1 được checked. Dựa vào Id này bạn so sánh để biết được trên giao diện người sử dụng đang checked lựa chọn nào.

**Cách 2:** Kiểm tra trực tiếp RadioButton đó có được checked hay không?

```
RadioButton radHay=findViewById(R.id.radHay);
if (radHay.isChecked()){
    //xử lý thông tin về radio được chọn
}
```

Cả 2 cách trên đều có cùng chung một mục đích và có kỹ thuật xử lý khác nhau. Tương tự với checked, để xóa bỏ checked trong group, ta dùng lệnh:

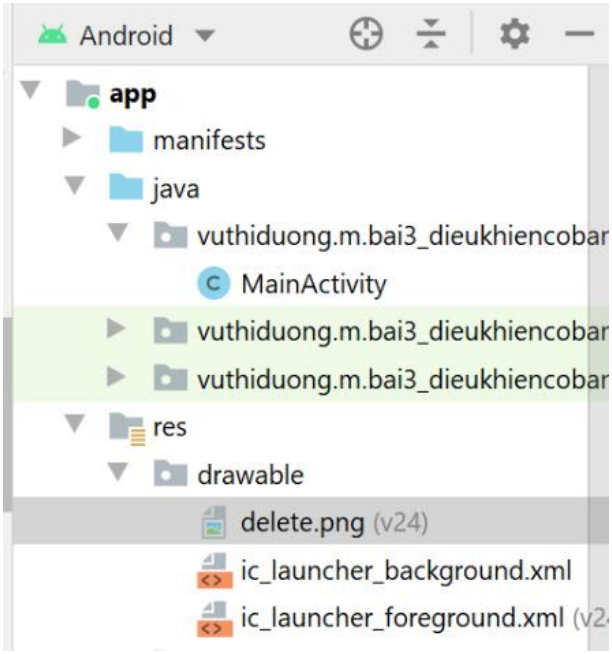
**group.clearChecked();** ở đó **group** là đối tượng RadioGroup.

**Cách 3**: Kiểm tra sự kiện khi người dùng chọn lựa vào nút lệnh

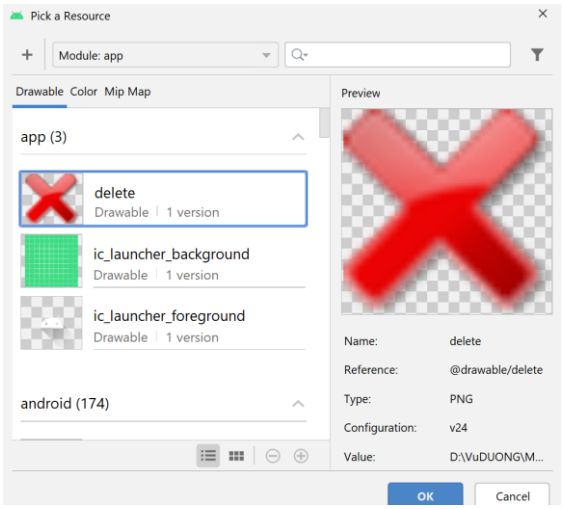
```
radHay.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
        if (isChecked){
            //xử lý thông tin nếu radio này được chọn
        }

    }
});
```

### 3.5. ImangeButton

Đây là một Button có chứa hình ảnh. Khi đặt tên có thể dùng tiếp đầu ngữ **imgBtn**. Để đặt ảnh cho ImageButton ta chọn ảnh từ hệ thống gợi ý hoặc chọn ảnh tự thiết kế đặt vào thư mục drawable trong ứng dụng. Ví dụ




Khi kéo thả Image Button vào giao diện, chương trình hiển thị cửa sổ yêu cầu chọn ảnh cho nút lệnh




Xử lý các sự kiện trên Image Button giống như Button. Để thay đổi hình ảnh bằng mã lệnh là thực hiện qua phương thức setImageResource(…). Ví dụ nếu có ảnh save trong drawable ta có thể đổi hình ảnh như sau.

```
ImangeButton imgBtnDelete=(ImangeButton)
findViewById(R.id.imgDelete);
imgBtnDelete.setImageResource(R.drawable.imgSave)
```

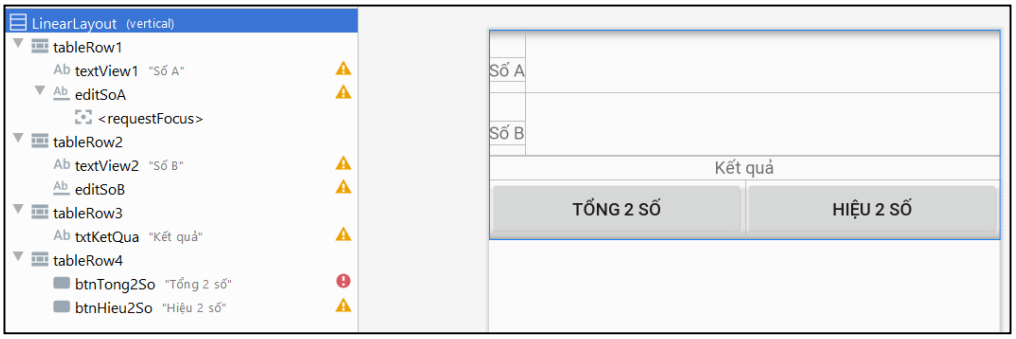
## 4. Các kiểu lập trình sự kiện

Người dùng tương tác với giao diện đồ họa của ứng dụng Android dưới 2 mức độ: mức Activity và mức View. Ở mức Activity, lớp Activity cần nạp chồng các phương thức tương

ứng được định nghĩa sẵn. Các sự kiện trên view thường dùng lắng nghe các hành vi của người dùng trên phần mềm.

Tổng quan có 5 kiểu xử lý tương tác qua sự kiện click trên giao diện

Bài toán minh họa. Giả sử có giao diện được thiết kế và đặt tên như hình sau:




### 4.1. Onclick XML

Đây là sự kiện được gắng cho các View lúc thiết kế giao diện theo cách áp dụng kéo thả. Sự kiện được xác lập qua thuộc tính onClick.

Trong giao diện (**activity_main.xml**) dùng thuộc tính **android:onClick** để gán sự kiện

```
<Button
     android:id="@+id/btnTong2So"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:onClick="tong2so"
     android:text="Tổng 2 số" />
```

Sau đó vào **Activity_main.java** khai báo hàm sự kiện có tên trung với chuỗi gán cho thuộc tính **onClick:**

```
btn_tong2so = (Button) findByViewID(R.id.btnTong2So);
public void tong2so(View v)     {
     a=Integer.parseInt(editSoA.getText()+"");
     b=Integer.parseInt(editSoB.getText()+"");
      txtKetQua.setText((a + b) + "");
   }
```

### 4.2. Sử dụng sự kiện ngầm định Anonymous Listener.

Đâu là loại gán sự kiện dạng vô danh, loại này ưa chuộng nhất trong lập trình sự kiện. Loại lắng nghe sự kiện được xử lý thuần túy bằng mã lệnh trong **Activity_main.java**. Các view sẽ được gán sự kiện tương tự như sau:

```
btn_tong2so = (Button) findByViewID(R.id.btnTong2So);
btn_tong2so.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            a=Integer.parseInt(editSoA.getText()+"");
            b=Integer.parseInt(editSoB.getText()+"");
            txtKetQua.setText((a + b) + "");
        }  });
```

Trong mã lệnh ta goi hàm setOnClickListener cho view, bên trong hàm ta gõ new sau đó nhấn tổ hợp phím **control + space**. Android studio sẽ tự động phát sinh sự kiện bên trong hàm.

Chú ý loại sự kiện này không tái sử dụng được, mỗi View sẽ sở hữu sự kiện anomous của riêng mình.

### 4.3. Cụ thể hóa phương thức onClick trong giao tiếp OnclickListener- Activity as Listener.

Activity as Listener là cách nàn hình có khả năng sinh sự kiện. Trong trường hợp này ta cho Activity triển khai từ interface sự kiện trong android.

```
public class MainActivity extends AppCompatActivity
implement View.OnclickListener{
    //các khai báo
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //xử lý lấy các điều khiển
        btn_Tong2so.setOnClickListener(this));
        btn_Hieu2so.setOnClickListener(this));
   }
   @Override
```

```
    public void onClick(View view) {
      if (view == btn_Tong2so) {
            //xử lý nút lệnh tính tổng 2 số
      }
      else if (view == btn_Hieu2so) {
              //xử lý nút lệnh tính hiệu 2 số
      }
    }
  }
```

Một Activity có thể lắng nghe nhiều sự kiện cùng 1 lúc với cách thức thực hiện tương tự như hướng dẫn trên. Ví dụ nếu Activiy implement View.OnLongClickListener thì View cũng sẽ lắng nghe bằng cấu trúc tương tự sau:

```
btn_Tong2so.setOnLongClickListener(this));
```

### 4.4. Biến nhận sự kiện – variable as listener.

Variable as listener là kỹ thuật gán sự kiện cho View thông qua tên biến. Điều đó nghĩa là ta có thể khai báo 1 biến có khả năng sinh sự kiện sau đó gán biến này cho view bất kỳ.

Đây là loại sự kiện dùng để chi sẻ cho các View trong cùng màn hình.

Bước 1: Khai báo biến thuộc lớp OnClickListener như sau:

```
View.OnClickListener myListenner = new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (view == btn_Tong2so) {
                //xử lý nút lệnh tính tổng 2 số
            }
     Else if (view == btn_Hieu2so) {
                //xử lý nút lệnh tính hiệu 2 số
            }
        }
    };
```

Bước 2: Gán biến lắng nghe sự kiện cho các nút lệnh như sau:

```
btn_Tong2so.setOnClickListener( myListenner);
```

```
btn_Hieu2so.setOnClickListener( myListenner);
```

### 4.5. Lớp nhận sự kiện – Explicit class Listener.

Đây là loại sự kiện khi khai báo lớp độc lập có khả năng sinh sự kiện. Lớp này implements từ interface OnClickListener. Giả sử lớp phát sinh sự kiện tên là DoSomeThing, các view muốn sử dụng chỉ cần gọi: `setOnClickListener(new DoSomeThing())`.

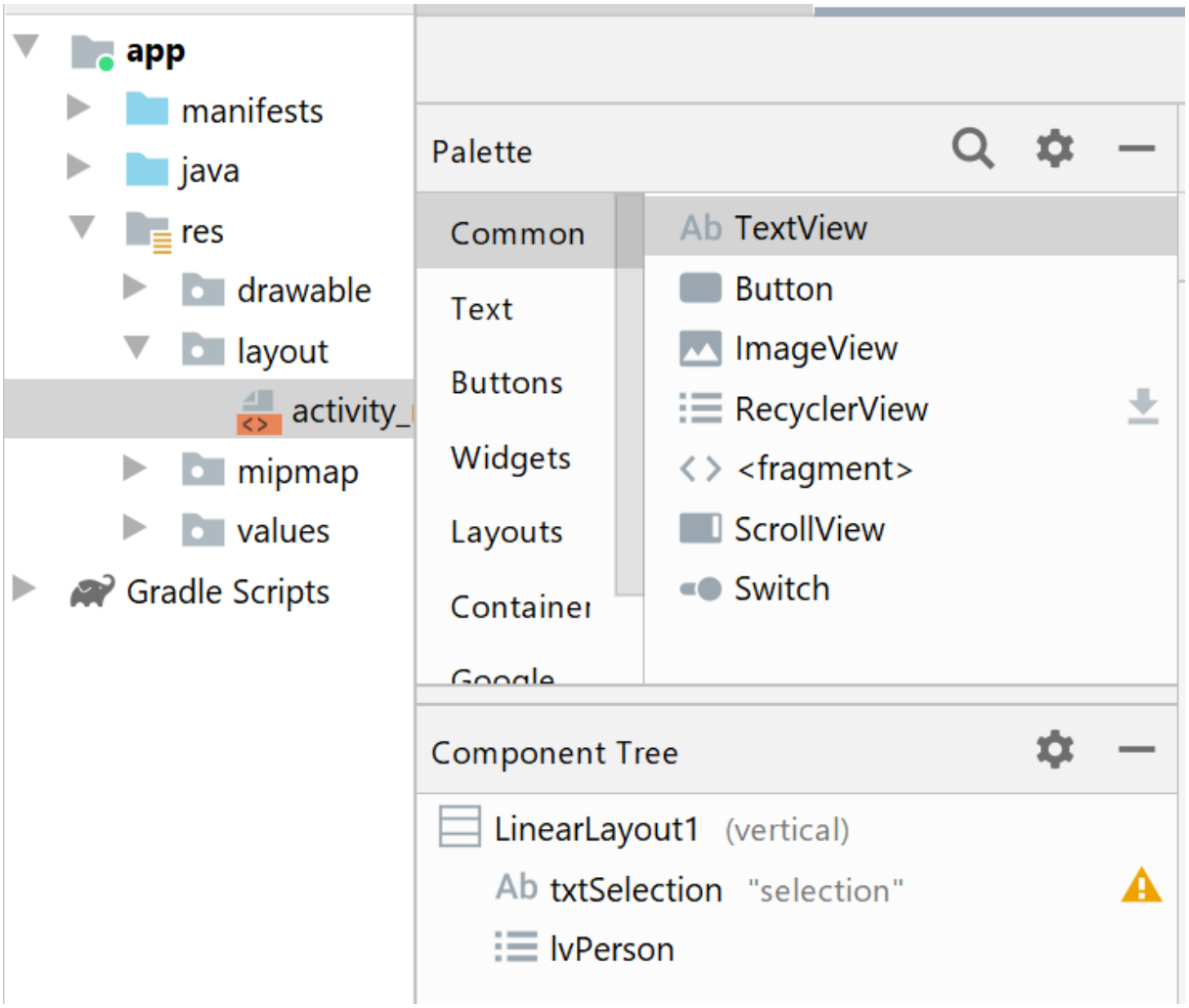
```
public class MainActivity extends AppCompatActivity {
    //các khai báo
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getwidget();
        btn_tong2so.setOnClickListener(new DoSomeThing());
    }
    //CÁCH 5- TẠO LỚP LẮNG NGHE SỰ KIỆN
    protected class DoSomeThing implements
View.OnClickListener {
        @Override
        public void onClick(View view) {
            if(view.getId()==R.id.btnTong2So){
                //xử lý nội dung nút lệnh tính tổng
            }
        }
    }
```

## 5. Xử lý tương tác với một số điều khiển nâng cao

### 5.1. ListView đơn giản

*a. Đưa dữ liệu vào ListView với mảng dữ liệu có sẵn*

Giả sử ta có giao diện đơn giản thiết kế như sau




Kết quả thực hiện gán nội dung Listview là danh sách các thành phố lớn. Kết quả chạy chương trình:


Thực hiện: các bước khởi tạo dữ liệu  và gán cho ListView thông qua 5 bước như sau

```
//1. Khởi tạo dữ liệu cho mảng arr (còn gọi là data source)
 final String arr[]={"Huế","Sài gòn","Hà Nội","Hải phòng"};
//2. Lấy đối tượng Listview dựa vào id
ListView lv=(ListView) findViewById(R.id.lvperson);
//3. Gán Data source vào ArrayAdapter
ArrayAdapter<String>adapter= new ArrayAdapter<String>
     (this, android.R.layout.simple_list_item_1, arr);
//4. Đưa Data source vào ListView
     lv.setAdapter(adapter);
//5. Thiết lập sự kiện cho Listview, khi chọn phần tử
     lv.setOnItemClickListener( new
     AdapterView.OnItemClickListener() {
     public void onItemClick(AdapterView<?> arg0,
           View arg1,int arg2,long arg3)
       {
          //đối số arg2 là vị trí phần tử trong arr
          txt.setText("position :"+arg2+" ; value
="+arr[arg2]);
       }
      });
```

Khởi tạo danh sách lưu trữ trong ListView có thể thực hiện là ArrayList minh họa như sau. Lưu ý là adapter và ArrayList đều tham chiếu đến cùng kiểu dữ liệu

```
  //Khai báo lại các tài nguyên
            ArrayList<String> arrList = null;
            ArrayAdapter<String> adapter=null;
  lv=(ListView) findViewById(R.id.lvperson);
  //1. Tạo ArrayList object
   arrList= new ArrayList<String>();
   //2. Gán Data Source (ArrayList object) vào ArrayAdapter
   adapter=new ArrayAdapter<String>
   (this, android.R.layout.simple_list_item_1, arrList);
   //3. gán Adapter vào ListView
   lv.setAdapter(adapter);
   //4. Xử lý thêm dữ liệu cho list
    arrList.add("Hà nội");
    arrList.add("Sài gòn");
```

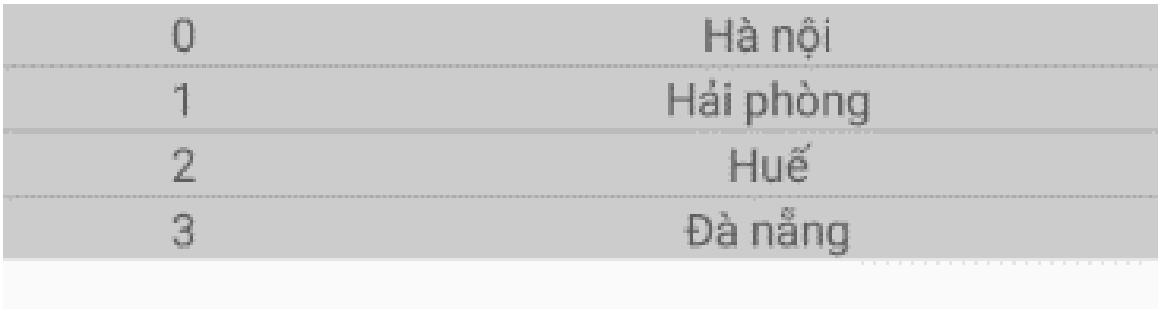
```
    arrList.add("Huế");
    arrList.add("Hải Phòng");
    //lắng nghe thay đổi dữ liệu đi kèm
    adapter.notifyDataSetChanged();
    //5. Xử lý sự kiện chọn một phần tử trong ListView
    lv.setOnItemClickListener(new
    AdapterView.OnItemClickListener() {
    public void onItemClick(
         AdapterView<?> arg0,View arg1, intarg2,longarg3) {
         txtchon.setText("position : "+ arg2+
          "; value ="+arrList.get(arg2));
  }
  });
```

- Xử lý sự kiện chọn lâu 1 dòng trong ListView **setOnItemLongClickListener.** Sự kiện **setOnItemLongClickListener**, được gắn cho ListView Item, khi nhấn lâu từ 2.5 tới 3 giây thì sự kiện này sẽ xảy ra. Tương tự như **setOnItemClickListener**, đối số có tên **arg2** được dùng để xác định được vị trí của phần tử nằm trong ArrayList.
- Lưu ý trong quá trình thay đổi các phiên bản, có thể tên các đối số khác nhau nhưng vị trí thì không thay đổi. Vậy căn cứ vào vị trí ta có thể tham chiếu đến nội dung tương ứng.

```
//6. xử lý sự kiện Long click
lv.setOnItemLongClickListener(new
    AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> arg0,
    View arg1,intarg2, longarg3) {
         arrList.remove(arg2);//xóa phần tử thứ arg2
         adapter.notifyDataSetChanged();
         return false;
    }
 });
```

## 5.2. Tùy biến ListView

ListView còn cho phép người dùng chỉnh sửa, tạo ra các dòng hiển thị phức tạp đáp ứng nhu cầu người dùng. Giả sử ta sẽ thiết kế lại màn hình listView đơn giản ở trên có kết quả như sau



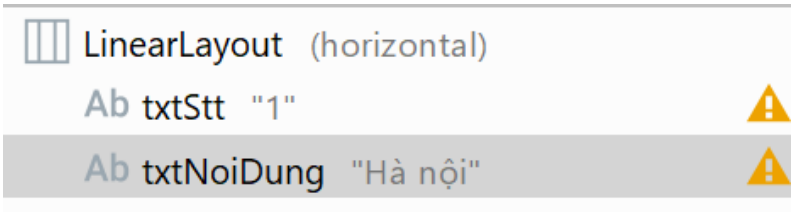

Các bước thực hiện tùy biến hiển thị listview như sau

Bước 1: mô hình hóa đối tượng nếu cần

Bước 2: thiết kế layout cho từng dòng có cấu trúc yêu cầu.

| Stt | tên |
|---|---|

Giả sử layout có tên myitemlist có cấu trúc thiết kế như sau:




Bước 3: kế thừa ArrayAdapter, hiệu chỉnh hàm getView để hiển thị dữ liệu yêu cầu

```
class MyAdapter extends ArrayAdapter{
    Activity context;
    int layoutID;
    ArrayList<String> list=null;

    public MyAdapter(@NonNull Activity context, int
resource, @NonNull List objects) {
        super(context, resource, objects);
        this.context = context;
        this.layoutID = resource;
        this.list = (ArrayList<String>) objects;
    }

    public View getView(int position, @Nullable View
convertView, @NonNull ViewGroup parent) {
        LayoutInflater inflater =
```

```
        context.getLayoutInflater();
        convertView = inflater.inflate(layoutID, null);
        if ((list.size() > 0) && (position >= 0)) {
        //lấy dòng thứ i
        final TextView txtSTT =
convertView.findViewById(R.id.txtStt);
        final TextView txtNoiDung =
convertView.findViewById(R.id.txtNoiDung);
        //lấy bản ghi thứ possition gán cho hiển thị
        txtSTT.setText(position + "");
        txtNoiDung.setText(list.get(position).toString() +
"");
          }
        return convertView;
    }
```

Bước 4: Trong phần xử lý nghiệp vụ chính, gán Adapter mới cho listview và thực hiện các công việc tương tự như listview cơ bản

```
public class MainActivity extends AppCompatActivity {
    //khai báo các control dùng trong chương trình
    Button btn;
    ListView lv;
    //khai báo arrayList
    ArrayList<String>arrList=null;
    //tùy biến theo adapter mới
    MyAdapter adapter=null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //kết nối các control với Activity
        lv=(ListView)findViewById(R.id.lvPerson);
        //1. tạo 1 arraylisst object
        arrList = new ArrayList<String>();
        arrList.add("Hà nội");
        arrList.add("Hải phòng");
        arrList.add("Huế");
        arrList.add("Đà nẵng");
```

```
        //2. gán DataSource vào ArrayAdapter
        adapter=new MyAdapter(MainActivity.this,
                R.layout.myitemlist,
                arrList);
        //3. Gán adapter vào listView
        lv.setAdapter(adapter);
        btn=(Button)findViewById(R.id.btnNhap);
        //5. Xử lý sự kiện chọn một phần tử trong listView
        lv.setOnItemClickListener(new
    AdapterView.OnItemClickListener() {
        @Override
public void onItemClick(AdapterView<?> arg0, View arg1, int
arg2,long arg3) {
    txtSelection.setText("vị trí:"+arg2+" giá
trị:"+arrList.get(arg2));
    txtTen.setText(arrList.get(arg2)+"");
    }
});
//6 Xử lý sự kiện LongClick
lv.setOnItemLongClickListener(new
AdapterView.OnItemLongClickListener() {
@Override
    public boolean onItemLongClick(AdapterView<?> arg0,
View arg1,int arg2, long arg3) {
    arrList.remove(arg2);
    adapter.notifyDataSetChanged();
return false;
            }
        });
    }
```

### 5.3. Spinner.

**Spinner là điều khiển hoạt động** tương tự như ComboBox trong C#, tương tự như JComboBox trong Java. Để đổ dữ liệu lên **Spinner** tương tự ListView và bổ sung phương thức **setDropDownViewResource.**

*a. Các bước thực hiện hiển thị dữ liệu lên Spinner.*

Bước 1: đưa dữ liệu cần hiển thị vào mảng hay danh sách (arraylist, mảng thông thường, mảng đối tượng)

Bước 2: tạo đối tượng Spinner trên giao diện

Bước 3: Tạo đối tượng ArrayAdapter nhận liên kết giữa Spinner và mảng hoặc danh sách dữ liệu

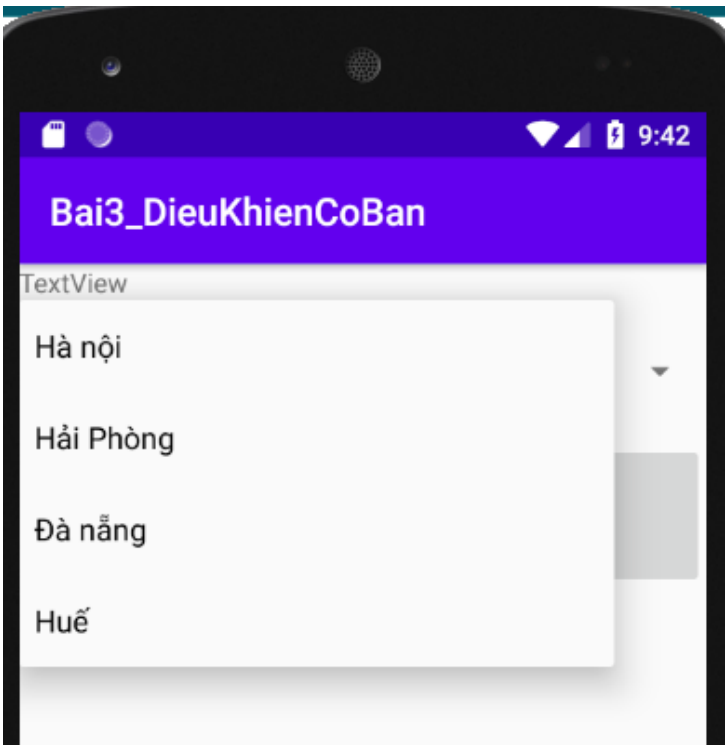
Bước 4: Gán ArrayAdapter cho Spinner.

*b. Sử dụng Spinner*

Bước 1: Vào nhóm Container của thanh công cụ Palette, tìm và kéo điều khiển ra giao diện. Một số thuộc tính và sự kiện quan trọng

| Thuộc tính/ sự kiện | Mô tả |
|---|---|
| Id | Đặt id cho điều khiển |
| Layout_width | Độ rộng |
| Layout_height | Độ cao |
| setAdapter | Gán Adapter cho ListView |
| setOnItemSelectedListener | Gán sự kiện chọn 1 dòng |

Bước 2: chuẩn bị dữ liệu. Giả sử có kết quả như sau




Chuẩn bị dữ liệu và gán Adapter:

```
Spinner spinThuDo;
TextView txtHienThi;
ArrayList<String> listThuDo=new ArrayList<String>();
ArrayAdapter<String> adapter=null;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //chuẩn bị dữ liệu
    listThuDo.add("Hà nội");
    listThuDo.add("Hải Phòng");
    listThuDo.add("Đà nẵng");
    listThuDo.add("Huế");
    //lấy điều khiển
    txtHienThi=findViewById(R.id.txtHienThi);
    spinThuDo=findViewById(R.id.spinThuDo);
    //set adapter
    adapter=new ArrayAdapter<String>(
            this,
            R.layout.support_simple_spinner_dropdown_item,
            listThuDo);

adapter.setDropDownViewResource(R.layout.support_simple_spin
ner_dropdown_item);
    spinThuDo.setAdapter(adapter);

}
```

Chú ý thuộc tính `R.layout.support_simple_spinner_dropdown_item` hỗ trợ hiển thị dữ liệu. Giao diện của Spinner có thể thay bằng giao diện khác, cách làm tương tự như tùy biến ListView

Cách gán sự kiện cho Spinner khác với ListView, cách thực hiện như sau:

```
//gán sự kiện cho spiner
spinThuDo.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View
view, int position, long id) {
        txtHienThi.setText(listThuDo.get(position));
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
```

### 5.4. Time selection

Android hỗ trợ chọn ngày giờ thông qua 2 điều khiển: DatePicker, TimePicker Và các hộp thoại hỗ trợ hiển thị DatePickerDialog, TimePickerDialog.

Điều khiển DatePicker và DatePickerDialog cho phép đặt thời gian bắt đầu hiển thị thông qua các thuộc tính: year, month, và day.

Trong đó: month nhận giá trị từ 0 = tháng 1 đến tháng 11 = tháng 12. Cả 2 điều khiển đều hỗ trợ đối tượng nhận dữ liệu mới trong sự kiện OnDateChangedListener hoặc OnDateSetListener

Điều khiển TimePicker và TimePickerDialog hỗ trợ đặt thời gian bắt đầu người dùng nhìn thấy trên hội thoại với giờ trong khoảng(0 - 23) và phút (0 - 59) và xác định cách đặt giờ cho 1 ngày là 24 hay 12 có kèm theo AM/PM. Hội thoại cũng trợ giúp lưu và đặt giờ vào hội thoại qua sự kiện:OnTimeChangedListener hoặc OnTimeSetListener.