

BUỔI 2:

KỸ THUẬT LẬP TRÌNH SỰ KIỆN TRÊN VIEW VÀ CỬA SỔ THÔNG BÁO NGƯỜI DÙNG

I. Mục tiêu buổi thực hành

- Ôn tập cách sử dụng các Layout, các View cơ bản.
- Hiểu và thực hành 6 kiểu xử lý sự kiện thường dùng:
 - + OnClickListener.
 - + Anonymous Listener.
 - + Variable as Listener.
 - + Activity as Listener.
 - + Explicit class Listener.
 - + View Subclassing.
- Biết và thực hành các kiểu cửa sổ thông báo người dùng:
 - + Toast.
 - + AlertDialog.
 - + Custom Dialog.
 - + Notification.

II. Triển khai buổi thực hành

Nội dung 1: Kỹ thuật lập trình sự kiện trên View

1. Sự kiện onClick XML

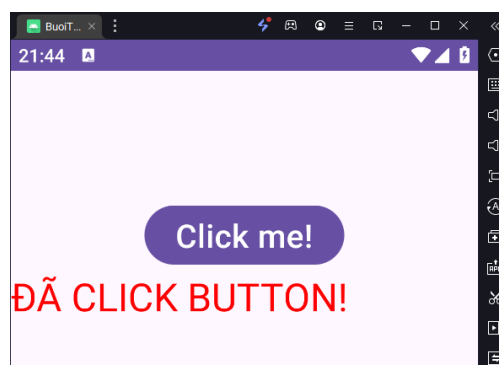
OnClick là sự kiện được kích hoạt khi người dùng nhấn vào một View nào đó, thường là Button. Một View bất kỳ đều có sự kiện OnClick. Tuy nhiên sự kiện này được gán cho View khi thiết kế giao diện, chỉ được áp dụng cho những View được kéo thả sẵn trong màn hình. Bởi vậy, nếu View được sinh ra trong quá trình thực thi ứng dụng thì không sử dụng được.

Trong file XML, ta thêm thuộc tính “android:onClick” để gán sự kiện click cho View:

```
<Button
    android:onClick="testOnClick"
    android:id="@+id/btnClickMe"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click me!"
    android:textSize="25dp"
    android:layout_gravity="center"
    android:layout_marginTop="100dp" />
```

Trong file xử lý nghiệp vụ java, ta thêm hàm tương ứng để thực hiện lập trình xử lý nghiệp vụ:

```
public void testOnClick(View view) {
    txtResult.setText("Đã Click button!");
    txtResult.setAllCaps(true);
}
```



2. Sự kiện anomous Listener

Sự kiện anomous Listener là một sự kiện dạng nặc danh. Trong file nghiệp vụ java, ta gán sự kiện như sau:

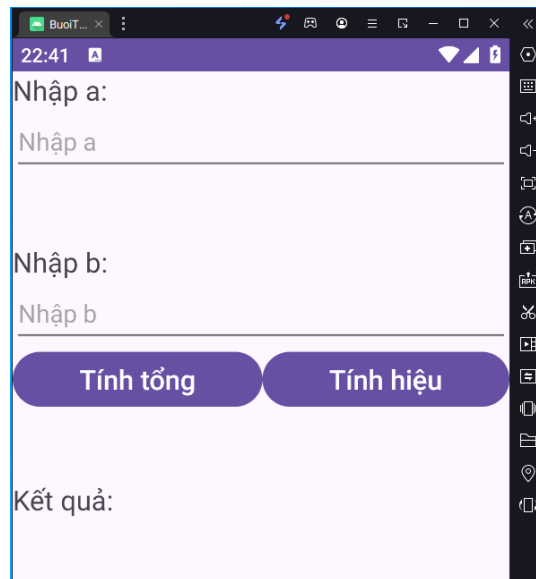
```
btnClickMe = (Button) findViewById(R.id.btnClickMe);
btnClickMe.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // xử lý nghiệp vụ ở đây
    }
});
```

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

3. Sự kiện variable as listener

Đây là loại sự kiện được gán cho View thông qua biến số, ta khai báo một biến có khả năng sinh sự kiện sau đó gán biến này cho View bất kỳ.

Loại sự kiện này được dùng để chia sẻ cho các View trong màn hình. Ví dụ, nút Cộng và nút Trừ cùng sử dụng một sự kiện tinhToan:



```
Button btnTinhTong, btnTinhHieu;
EditText edtNhapA, edtNhapB;
TextView txvKetQua;

View.OnClickListener tinhToan = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (v.equals(btnTinhTong)) {
            // xử lý chức năng tính tổng ở đây
        } else if (v.getId() == R.id.btnTinhHieu) {
            // xử lý chức năng tính hiệu ở đây
        }
    }
};

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

    addView();

    addEvent();
}

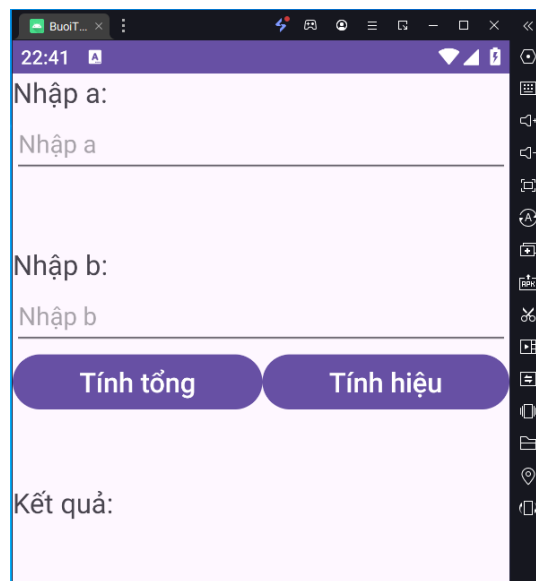
private void addView() {
    btnTinhTong = (Button) findViewById(R.id.btnTinhTong);
    btnTinhHieu = (Button) findViewById(R.id.btnTinhHieu);
    edtNhapA = (EditText) findViewById(R.id.edtNhapA);
    edtNhapB = (EditText) findViewById(R.id.edtNhapB);
    txvKetQua = (TextView) findViewById(R.id.txvKetQua);
}

private void addEvent() {
    btnTinhTong.setOnClickListener(tinhToan);
    btnTinhHieu.setOnClickListener(tinhToan);
}
```

4. Sự kiện activity as listener

Activity as Listener là một màn hình có khả năng sinh sự kiện, trường hợp sử dụng loại sự kiện này, ta cho Activity implements từ các interface sự kiện trong Android.

Trong Android, có nhiều loại interface sinh sự kiện như: sự kiện nhấn, sự kiện nhấn lâu, ... Các interface này thường có các phương thức trừu tượng (abstract method) ta bắt buộc phải override lại:



```
public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
    Button btnTinhTong, btnTinhHieu;
    EditText edtNhapA, edtNhapB;
    TextView txvKetQua;

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

        addView();

        addEvent();
    }

    private void addView() {
        btnTinhTong = (Button) findViewById(R.id.btnTinhTong);
        btnTinhHieu = (Button) findViewById(R.id.btnTinhHieu);
        edtNhapA = (EditText) findViewById(R.id.edtNhapA);
        edtNhapB = (EditText) findViewById(R.id.edtNhapB);
        txvKetQua = (TextView) findViewById(R.id.txvKetQua);
    }

    private void addEvent() {
        btnTinhTong.setOnClickListener(this);
    }
}
```

```

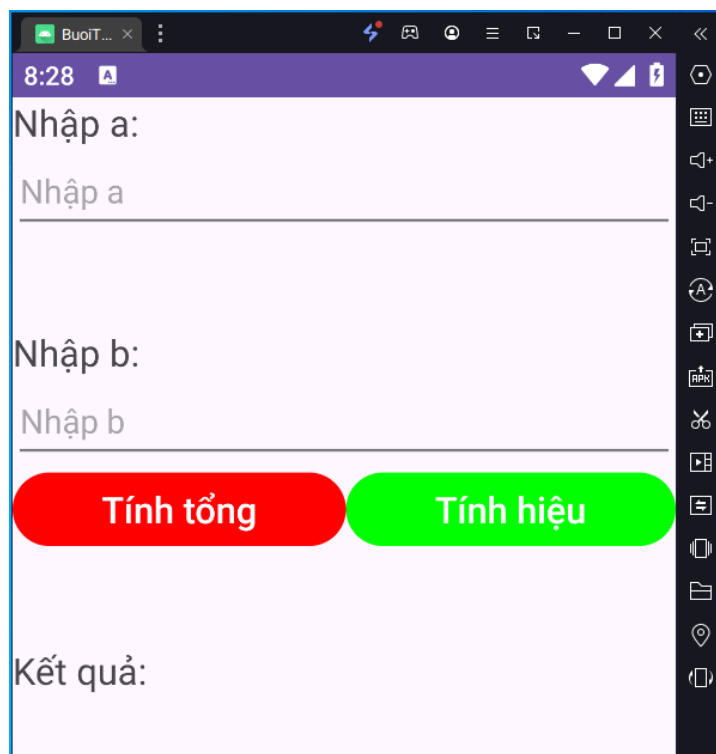
        btnTinhHieu.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        if (v.equals(btnTinhTong)) {
            // xử lý chức năng tính tổng ở đây
        } else if (v.getId() == R.id.btnTinhHieu) {
            // xử lý chức năng tính hiệu ở đây
        }
    }
}

```

Hiển nhiên, khi Activity có khả năng sinh sự kiện thì tất cả các View trên màn hình đều có thể chia sẻ chung sự kiện này. Ta cần gọi phương thức `setOnClickListener(this)`, cách kiểm tra này giống như sự kiện `variable as listener`.

Ngoài ra, một Activity có thể implements từ nhiều loại interface sinh sự kiện cùng một lúc, ví dụ ta có thể thêm sự kiện nhấn lâu vào màn hình, minh họa như sau:



Lập trình khi nhấn lâu vào button Tính tổng thì button đổi sang màu đỏ (Red). Khi nhấn lâu vào button Tính hiệu thì đổi sang màu xanh lá cây (Green).

```

public class MainActivity extends AppCompatActivity implements
View.OnClickListener, View.OnLongClickListener {
    Button btnTinhTong, btnTinhHieu;
    EditText edtNhapA, edtNhapB;
    TextView txvKetQua;
}

```

```

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

    addView();

    addEvent();
}

private void addView() {
    btnTinhTong = (Button) findViewById(R.id.btnTinhTong);
    btnTinhHieu = (Button) findViewById(R.id.btnTinhHieu);
    edtNhapA = (EditText) findViewById(R.id.edtNhapA);
    edtNhapB = (EditText) findViewById(R.id.edtNhapB);
    txvKetQua = (TextView) findViewById(R.id.txvKetQua);
}

private void addEvent() {
    // sự kiện nhấn 1 lần
    btnTinhTong.setOnClickListener(this);
    btnTinhHieu.setOnClickListener(this);

    // sự kiện nhấn lâu
    btnTinhTong.setOnLongClickListener(this);
    btnTinhHieu.setOnLongClickListener(this);
}

@Override
public void onClick(View v) {
    if (v.equals(btnTinhTong)) {
        // xử lý chức năng tính tổng ở đây
    } else if (v.getId() == R.id.btnTinhHieu) {
        // xử lý chức năng tính hiệu ở đây
    }
}

@Override
public boolean onLongClick(View v) {
    if (v.equals(btnTinhTong)) {
        // đổi màu button tính tổng sang màu đỏ
        btnTinhTong.setBackgroundColor(Color.RED);
    } else if (v.getId() == R.id.btnTinhHieu) {
        // đổi màu button tính hiệu sang màu xanh
        btnTinhHieu.setBackgroundColor(Color.GREEN);
    }

    return false;
}
}

```

5. Sự kiện explicit class listener

Khi sử dụng loại sự kiện explicit class listener, ta cần khai báo một lớp độc lập và có khả năng sinh sự kiện, tức là lớp mới này implements các interface sự kiện. Cách xử lý như này tương tự như Variable as listener và Activity as listener, nhưng điểm khác là ta tạo thêm một lớp khác.

```
public class MainActivity extends AppCompatActivity {
    Button btnTinhTong, btnTinhHieu;
    EditText edtNhapA, edtNhapB;
    TextView txvKetQua;

    // khai báo một lớp mới
    class MyEvent implements View.OnClickListener {
        @Override
        public void onClick(View v) {
            if (v.equals(btnTinhTong)) {
                btnTinhTong.setTextColor(Color.RED);
            } else if (v.equals(btnTinhHieu)) {
                btnTinhHieu.setTextColor(Color.GREEN);
            }
        }
    }

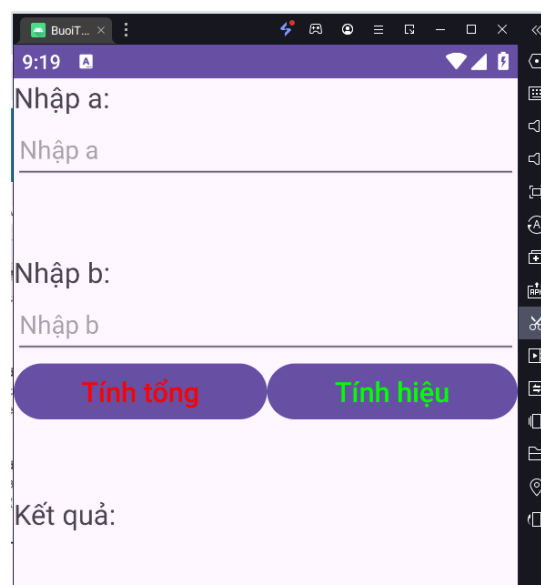
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.learn_explicit_class_listener);

        addView();

        addEvent();
    }

    private void addView() {
        btnTinhTong = (Button) findViewById(R.id.btnTinhTong);
        btnTinhHieu = (Button) findViewById(R.id.btnTinhHieu);
        edtNhapA = (EditText) findViewById(R.id.edtNhapA);
        edtNhapB = (EditText) findViewById(R.id.edtNhapB);
        txvKetQua = (TextView) findViewById(R.id.txvKetQua);
    }

    private void addEvent() {
        btnTinhTong.setOnClickListener(new MyEvent());
        btnTinhHieu.setOnClickListener(new MyEvent());
    }
}
```



Trong ví dụ minh họa trên, ta tạo ra một lớp độc lập là MyEvent có khả năng sinh sự kiện, lớp mới này được implements từ interface OnClickListener. Khi đó, các View khi được chỉ định có sự kiện onClick thì ta cần gọi phương thức “setOnClickListener(new MyEvent())”.

6. Sự kiện view subclassing

Sự kiện view subclassing là loại sự kiện được gán khi runtime. Ta xét ví dụ sau: ban đầu màn hình có một button Show, khi nhấn vào button này thì có một layout khác hiện thị. Trên layout mới có thêm một TextView và một Button. Nhấn vào Button này sẽ về màn hình ban đầu.

Code XML cho layout ban đầu:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/btnShow"
        android:onClick="showInfor"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Show"
        android:textSize="30dp" />
</LinearLayout>
```

Code trong file xử lý nghiệp vụ:

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.learn_view_subclassing);
    }

    public void showInfor(View view) {
        LinearLayout.LayoutParams layoutParams= new
        LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
        LinearLayout.LayoutParams.WRAP_CONTENT);
        LinearLayout linearLayout=new LinearLayout(this);
        linearLayout.setLayoutParams(layoutParams);
        linearLayout.setOrientation(LinearLayout.VERTICAL);

        TextView txvInfor=new TextView(this);
        txvInfor.setText("Hello");
        txvInfor.setLayoutParams(layoutParams);
        linearLayout.addView(txvInfor);

        Button btnBack=new androidx.appcompat.widget.AppCompatButton(this) {
            @Override
            public boolean performClick() {
                // trở về màn hình gốc trước đó
            }
        }
```



```

        setContentView(R.layout.learn_view_subclassing);
        return super.performClick();
    }

};
btnBack.setText("Back");
btnBack.setLayoutParams(layoutParams);
linearLayout.addView(btnBack);

setContentView(linearLayout);
}
}

```

Nội dung 2: Các kiểu cửa sổ thông báo người dùng

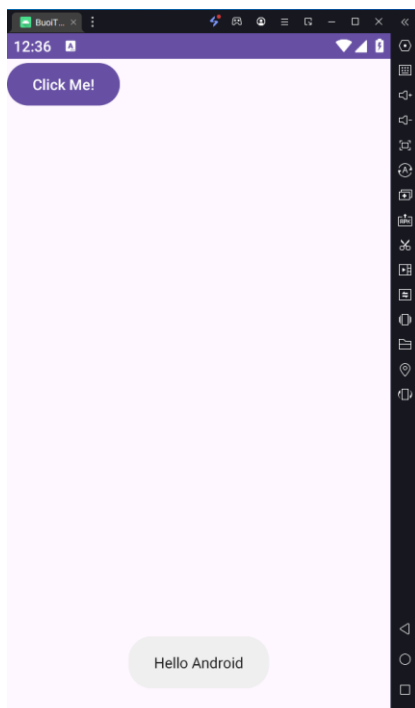
1. Loại cửa sổ Toast

Cửa sổ Toast được dùng để hiển thị trong Activity hoặc trong Service. Đặc điểm của Toast là không cho phép người dùng tương tác, nó hiển thị và tự động đóng lại sau một khoảng thời gian.

Thuộc tính chỉ định thời gian hiển thị:

- Nếu là Toast.LENGTH_LONG thì thời gian hiển thị là khoảng 3.5 giây.
- Nếu là Toast.LENGTH_SHORT thì thời gian hiển thị là khoảng 1.5 giây.

Ví dụ minh họa cho cửa sổ Toast:



– Code XML:

```

<Button
    android:onClick="clickMe"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click Me!" />

```

– Code java:

```

public void clickMe(View view) {
    Toast.makeText(MainActivity.this, "Hello
    Android", Toast.LENGTH_SHORT).show();
}

```

2. Loại cửa sổ AlertDialog

Khác với Toast, AlertDialog là cửa sổ hiển thị cho phép người dùng tương tác với hệ thống. Thường loại cửa sổ này được dùng để nhận quyết định từ phía người dùng.

Giả sử ta có một button Thoát trên giao diện, khi nhấn Thoát thì ứng dụng hỏi người dùng để xác nhận có muốn thoát hay không?

Code XML:

```
<Button
    android:onClick="thoatApp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="right"
    android:text="Thoát" />
```

Code java:

```
private void thoatUngDung() {
    AlertDialog.Builder builder = new
    AlertDialog.Builder(MainActivity.this);
    // thiết lập tiêu đề
    builder.setTitle("Hỏi thoát");
    // thiết lập icon
    builder.setIcon(android.R.drawable.ic_dialog_alert);
    // thiết lập nội dung thông báo
    builder.setMessage("Bạn muốn thoát ứng dụng?");
    // thiết lập nút đồng ý
    builder.setPositiveButton("Có", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            finish();
        }
    });
    // thiết lập nút không đồng ý
    builder.setNegativeButton("Không", new
    DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
        }
    });
    // tạo cửa sổ dialog
    AlertDialog alertDialog=builder.create();

    // khi người dùng nhấn bên ngoài dialog thì dialog không biến mất
    alertDialog.setCanceledOnTouchOutside(false);

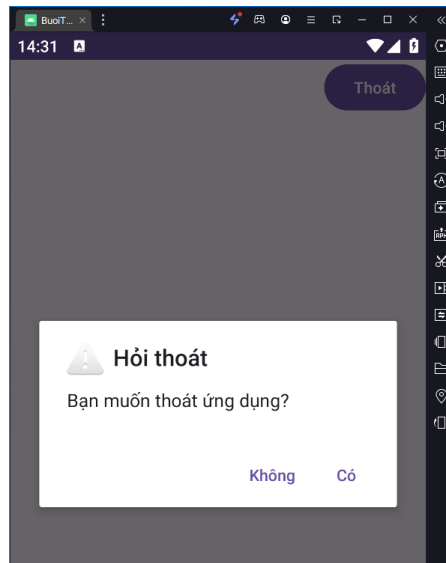
    // hiển thị cửa sổ dialog
    alertDialog.show();
}

public void thoatApp(View view) {
```

```

    thoatUngDung();
}

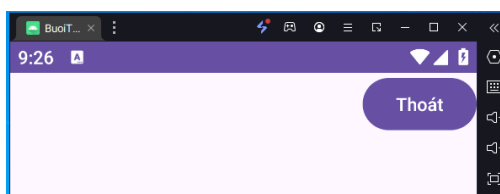
```



3. Loại cửa sổ CustomDialog

CustomDialog là loại cửa sổ do người lập trình tự thiết kế nhằm đáp ứng mục đích của khách hàng, thao tác giống như AlertDialog.

Ví dụ, ta có màn hình gốc có button Thoát như bên dưới, khi nhấn button Thoát thì xuất hiện một dialog xác nhận thoát hay không. Nhưng lập trình tối ưu tức là tạo ra một lớp mới để xử lý dialog. Như vậy, ở màn hình nào cần xuất hiện dialog thì ta gọi đến lớp mới đó để tương tác.

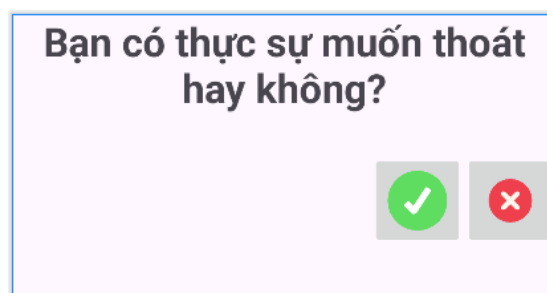


```

<Button
    android:onClick="thoat"
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="right"
    android:text="Thoát" />

```

Code XML dialog mới:



```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"

```

```

        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="Bạn có thực sự muốn thoát hay không?"
            android:textSize="30dp"
            android:textStyle="bold" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="30dp"
            android:gravity="right"
            android:orientation="horizontal">

            <ImageButton
                android:id="@+id/btnYes"
                android:layout_width="70dp"
                android:layout_height="70dp"
                app:srcCompat="@drawable/yes"
                android:scaleType="fitCenter"
                android:adjustViewBounds="true"
                android:contentDescription="Đồng ý"/>

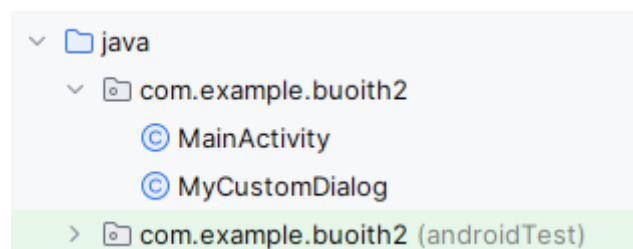
            <ImageButton
                android:id="@+id/btnNo"
                android:layout_width="70dp"
                android:layout_height="70dp"
                android:src="@drawable/decline"
                android:scaleType="fitCenter"
                android:adjustViewBounds="true"
                android:minHeight="48dp"
                android:contentDescription="Ở lại"/>

        </LinearLayout>

    </LinearLayout>

```

Tạo mới một lớp MyCustomDialog để xử lý nghiệp vụ trên Dialog mới:



```

package com.example.buoih2;

import android.app.Activity;
import android.app.Dialog;
import android.view.View;
import android.widget.ImageButton;

public class MyCustomDialog extends Dialog {
    // Khai báo các control trên dialog

```

```

ImageButton btnYes, btnNo;
Activity context;

public MyCustomDialog(Activity context) {
    super(context);
    this.context = context;
    setContentView(R.layout.layout_custom_dialog);
    addViews();
    addEvents();
    // Nhấn ra ngoài dialog thì không bị biến mất
    setCanceledOnTouchOutside(false);
}

private void addViews() {
    btnYes = (ImageButton) findViewById(R.id.btnYes);
    btnNo = (ImageButton) findViewById(R.id.btnNo);
}

private void addEvents() {
    // khi nhấn nút yes thì thoát ứng dụng
    btnYes.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            context.finish();
        }
    });
    // khi nhấn nút no thì đóng dialog, ở lại ứng dụng
    btnNo.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            dismiss();
        }
    });
}
}

```

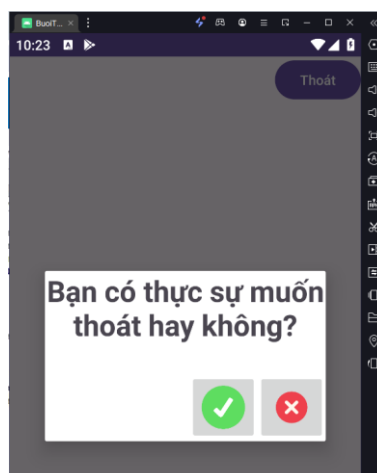
Trong MainActivity, ta lập trình để gọi đến MyCustomDialog khi nhấn button Thoát:

```

public void thoat(View view) {
    MyCustomDialog new_dialog=new MyCustomDialog(MainActivity.this);
    new_dialog.show();
}

```

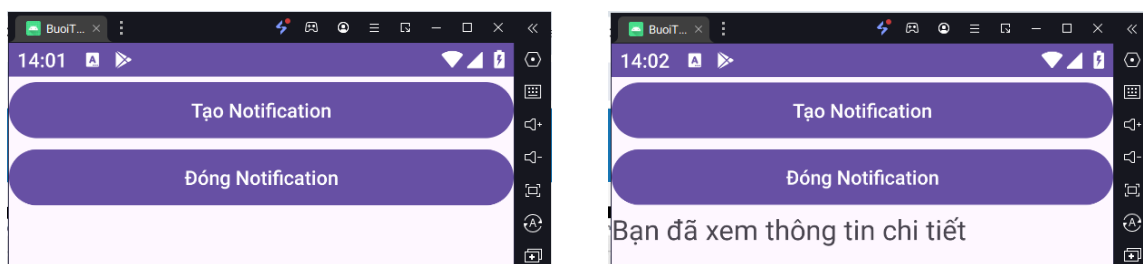
Kết quả:



4. Loại Notification

Một Notification là một thông điệp mà Android hiển thị bên ngoài ứng dụng (thường được dùng khi phần mềm ở trạng thái chạy ngầm). Nó được xuất hiện ở thanh bên trên điện thoại. Notification cho phép cung cấp cho người dùng lời nhắc, tin nhắn hay bất kỳ một thông tin gì từ ứng dụng. Người dùng có thể nhấn vào Notification này để mở trực tiếp ứng dụng hoặc thực hiện một hành động ngay trên notification như gửi tin nhắn.

Ví dụ, ta tạo hai button là Tạo Notification và Đóng Notification như hình minh họa dưới. Khi người dùng nhấn vào notification thì hiển thị màn hình chính ứng dụng và xuất hiện thêm một TextView cho phép nhận biết.



Code file XML:

```
<?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="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/btnTaoNotification"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="taoNotification"
        android:text="Tạo Notification" />

    <Button
        android:id="@+id/btnDongNotification"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="dongNotification"
        android:text="Đóng Notification" />

    <TextView
        android:id="@+id/txvDaXem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:textSize="20dp" />

</LinearLayout>
```

Code file java:

```

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.learn_notification);

        Intent intent = getIntent();
        String noiDung = intent.getStringExtra("CHITIET");
        TextView txvDaXem = (TextView) findViewById(R.id.txvDaXem);
        txvDaXem.setText(noiDung);
    }

    public void taoNotification(View view) {
        String chanelID = "CHANNEL_ID_NOTIFICATION";
        NotificationCompat.Builder builder = new
NotificationCompat.Builder(getApplicationContext(), chanelID);
        builder.setSmallIcon(R.drawable.ic_notification);
        builder.setContentTitle("Bạn có thông báo");
        builder.setContentText("Nhấn vào để xem thông tin chi tiết");
        builder.setAutoCancel(true);
        builder.setPriority(NotificationCompat.PRIORITY_DEFAULT);

        Intent intent = new Intent(getApplicationContext(),
MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        intent.putExtra("CHITIET", "Bạn đã xem thông tin chi tiết");

        PendingIntent pendingIntent =
PendingIntent.getActivity(getApplicationContext(), 0, intent,
PendingIntent.FLAG_MUTABLE);
        builder.setContentIntent(pendingIntent);
        NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel notificationChannel =
notificationManager.getNotificationChannel(chanelID);
            if (notificationChannel == null) {
                int important = NotificationManager.IMPORTANCE_HIGH;
                notificationChannel = new NotificationChannel(chanelID,
"Some description", important);
                notificationChannel.setLightColor(Color.GREEN);
                notificationChannel.enableVibration(true);

                notificationManager.createNotificationChannel(notificationChannel);
            }
        }

        notificationManager.notify(0, builder.build());
    }

    public void dongNotification(View view) {
        NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
        notificationManager.cancel(0);
    }
}

```

Chú ý, trong file AndroidManifests.xml cần phải có lệnh ở dòng số 5:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools">
4
5     <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
6     <application
```

Nội dung 3: Câu hỏi lý thuyết và bài tập thực hành

Câu 1: So sánh trường hợp sử dụng của các loại sự kiện.

Câu 2: Trong những trường hợp nào sử dụng của số dạng Toast, AlertDialog, CustomDialog là phù hợp nhất.

Câu 3: Cửa sổ thông báo dạng Notification được sử dụng trong những trường hợp nào.

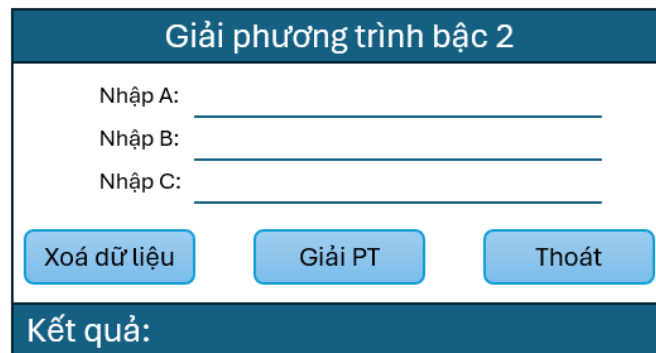
Bài 1: Viết chương trình tạo giao diện và thực hiện các yêu cầu dưới đây:

| Tính tổng hai số | |
|--|----------------------|
| Nhập A: | <input type="text"/> |
| Nhập B: | <input type="text"/> |
| Kết quả: | |
| <input type="button" value="Tổng hai số"/> | |
| <input type="button" value="Hiệu hai số"/> | |
| <input type="button" value="Tích hai số"/> | |
| <input type="button" value="Thương hai số"/> | |
| <input type="button" value="Ước số chung lớn nhất"/> | |
| <input type="button" value="Thoát chương trình"/> | |

- Mỗi lần click vào các button thì thực hiện các chức năng tương ứng của button đó.
- Tổng hai số => dùng sự kiện onClick XML.
- Hiệu hai số => dùng sự kiện anonymous listener.
- Tích của hai số => dùng sự kiện activity listener.

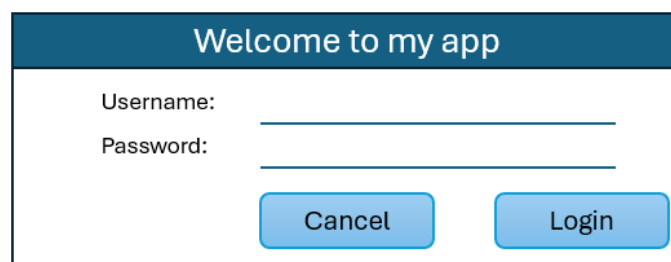
- Thương hai số => dùng sự kiện variable listener.
- Ước số chung lớn nhất => dùng sự kiện explicit class listener.
- Thoát chương trình => dùng sự kiện view subclassing.

Bài 2: Viết chương trình thiết kế giao diện theo mẫu và thực hiện các chức năng dưới đây:

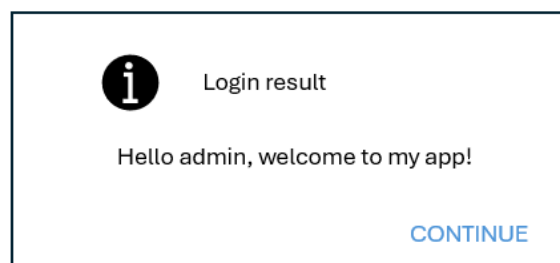


- Xoá dữ liệu: xoá trắng dữ liệu của các EditText tương ứng với nhập A, nhập B và nhập C. Focus vào EditText của nhập A.
- Giải PT: thực hiện giải phương trình bậc 2 và hiển thị kết quả ra màn hình.
- Thoát: thoát khỏi ứng dụng.

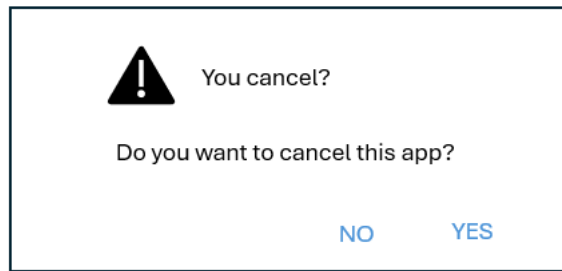
Bài 3: Viết chương trình thiết kế giao diện và thực hiện các chức năng dưới đây:



Khi bấm nút Login, nếu Username đúng bằng “admin” và Password đúng bằng “admin” thì hiển thị AlertDialog như sau:



Khi bấm Cancel, cửa sổ xác nhận thoát phần mềm dạng AlertDialog như sau được bật lên:



Tài liệu tham khảo

1. *Head First: Android Development* (2015), Dawn Griffiths, David Griffiths.
2. *Đề cương chi tiết học phần: Lập trình cho các thiết bị di động* (2019), Trường đại học Cần Thơ.
3. *Đề cương chi tiết học phần: Phát triển ứng dụng di động* (2024), Học viện Nông nghiệp Việt Nam.
4. *Giáo trình Phát triển ứng dụng di động cơ bản bản và nâng cao* (2019), chủ biên Lê Hoàn Sử.
5. Website <https://developer.android.com/>
6. Website: <https://www.geeksforgeeks.org/>

Hà Nội, ngày 08 tháng 03 năm 2024

Biên soạn và tổng hợp

Hoàng Văn Tuấn

----- **Hết** -----