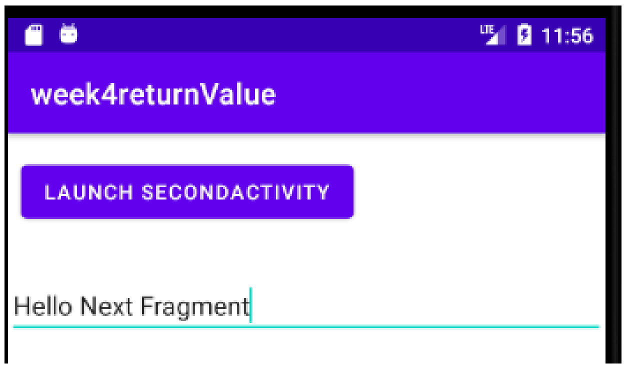
Android Application Development, COMP 10073

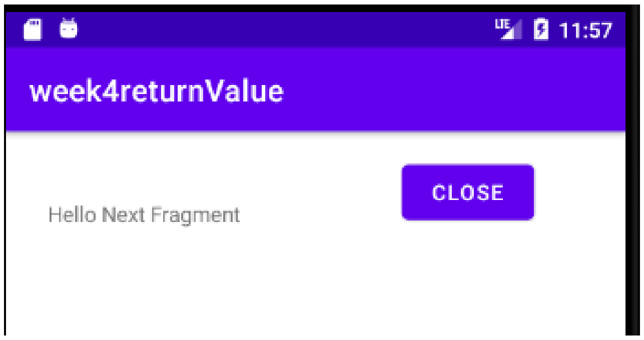
Mohawk College, Winter 2021

Getting Results from Activities

Previously when we discussed starting activities we said that it was a bad idea to share results via references, and that what should happen is that results should be shared in other ways. As discussed in the last lecture sometimes sharing references between fragments is okay. Any shared reference needs to be mediated by the host activity.

We presented a method for sending results to an activity, but what about collecting results from an activity? Start with a simple version of a previous example. Construct a main activity with a button and and EditText widget. Create a second activity with a button and a TextView widget.





Recall how to receive data

Following our previous examples, add this code to MainAcitivity2. If there is an intent, it extracts the data and adds it to the textView.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main2);
    TextView textView = (TextView)findViewById(R.id.textViewText);
    if (getIntent()!=null && getIntent().hasExtra(Intent.EXTRA_TEXT)) {
        textView.setText(getIntent().getStringExtra(Intent.EXTRA_TEXT));
    }
}
```

Include a close button

https://developer.android.com/reference/android/app/Activity#finish()

Rather than restarting the previous activity, or relying on Android's back button we can include our own close button. The handler makes a call to finish().

```
public void onClickClose(View view) {
    finish();
}
```

Start the Fragment using startActivityForResult()

https://developer.android.com/reference/android/app/Activity#startActivityForResult(android.content.Intent,%20int)

Instead of calling startActivity() we can use another method to indicate that we expect a result. The method startActivityForResult() takes two arguments, the intent to start, and a requestCode.

```
public void onClickSwitchActivity(View view) {
    EditText editText = (EditText)findViewById(R.id.editTextData);
    String text = editText.getText().toString();
    Intent intent = new Intent(this, MainActivity2.class);
    intent.putExtra(Intent.EXTRA_TEXT,text);
    startActivityForResult(intent, 1);
}
```

The requestCode should be >=0. This value will be returned in onActivityResult() when the activity exits along with any value from the caller. If there are several ways to start the next activity, by reading the requestCode on return we can know which method was use to start the activity.

When finish() is called the requestCode will be returned along with other details.

Extend our close button to return data

https://developer.android.com/reference/android/app/Activity#setResult(int)

Use setResult() to return a result code and data through an intent. Common result codes are RESULT_OK and RESULT_CANCELLED. If you use the back button you will get the default code RESULT_CANCELLED.

```
public void onClickClose(View view) {
    Intent returnIntent = new Intent();
    returnIntent.putExtra(MainActivity.REQUEST_RESULT, 42);
    setResult(RESULT_OK, returnIntent);
    finish();
}
```

onActivityResult()

https://developer.android.com/reference/android/app/Activity#onActivityResult(int,%20int,%20android.content.Intent)

The sending activity needs some additional code to receive the data. The method onActivityResult() is Called when an activity you launched exits, giving you the requestCode you started it with, the resultCode it returned, and any additional data. The resultCode will be RESULT_CANCELED if the activity explicitly returned that, didn't return any result, or crashed during its operation.

An activity can never receive a result in the resumed state. You can count on onResume() being called after this method, though not necessarily immediately after. If the activity was resumed, it will be paused and the result will be delivered, followed by onResume(). If the activity wasn't in the resumed state, then the result will be delivered, with onResume() called sometime later when the activity becomes active again.

```java
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    Log.d(tag, "onActivityResult(), requestCode = " + requestCode);
    if (resultCode==RESULT_OK) {
        Log.d(tag, "RESULT_OK. Result Data =" +  data.getIntExtra(REQUEST_RESULT, 0));
    } else if (resultCode==RESULT_CANCELED) {
        Log.d(tag, "RESULT_CANCELED. ");
    } else {
        Log.d(tag, "requestCode = " + requestCode +
                " resultCode = " + resultCode + "Intent = " + data);
    }
}
```

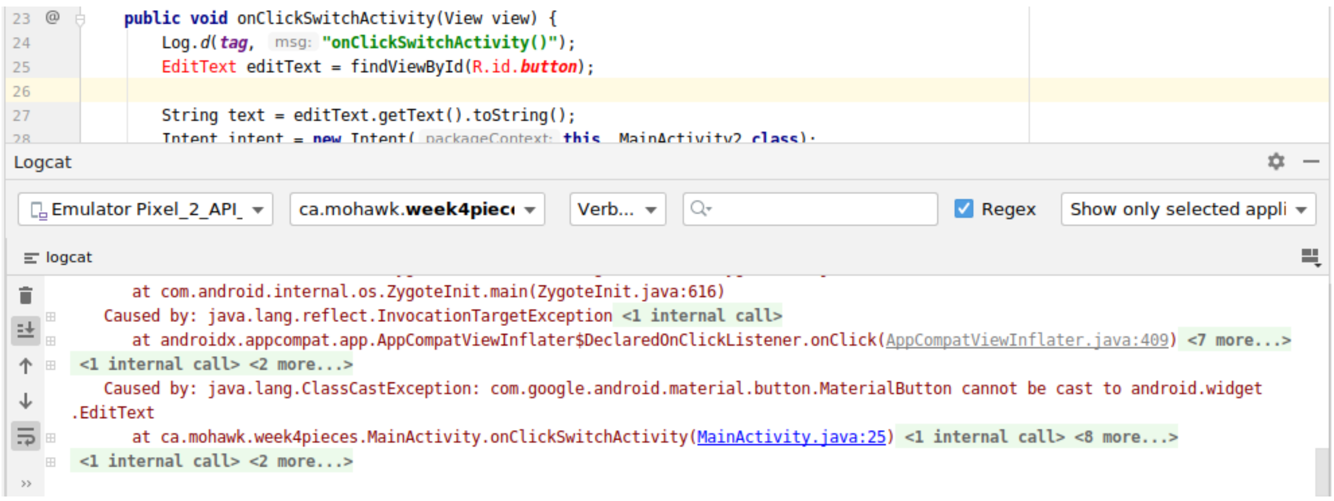
Add a second button to pass in a different result code. Modify the handler to recognize which button was pressed by checking the button’s symbolic ID.

```java
public void onClickSwitchActivity(View view) {
    EditText editText = (EditText)findViewById(R.id.editTextData);
    String text = editText.getText().toString();
    Intent intent = new Intent(this, MainActivity2.class);
    intent.putExtra(Intent.EXTRA_TEXT,text);
    if (view.getId() == R.id.button) {
        startActivityForResult(intent, 1);
    } else if (view.getId() == R.id.button2){
        startActivityForResult(intent, 2);
    } else {
        Log.d(tag, "Unrecognized Button Pressed.");
    }
}
```

Avoiding casting the view to Button type makes this code more robust.

Using the ID is important from a number of perspectives.

By now you ought to have realized that casting creates problems. Casting is the last resort of a programmer who has lost control of a type system and is desperate to make things fit together. Android studio will highlight some type mismatch issues, for example, it knows that the view returned by R.id.button should not be converted to an EditText type. The compiler will allow this, but it will crash at run time, i.e.



```
    public void onClickSwitchActivity(View view) {
        Log.d(tag,  msg: "onClickSwitchActivity()");
        EditText editText = findViewById(R.id.button);

        String text = editText.getText().toString();
        Intent intent = new Intent( packageContext: this, MainActivity2.class);
```

Logcat

Emulator Pixel_2_API_ | ca.mohawk.week4piec | Verb... | Regex ✓ | Show only selected appli

logcat

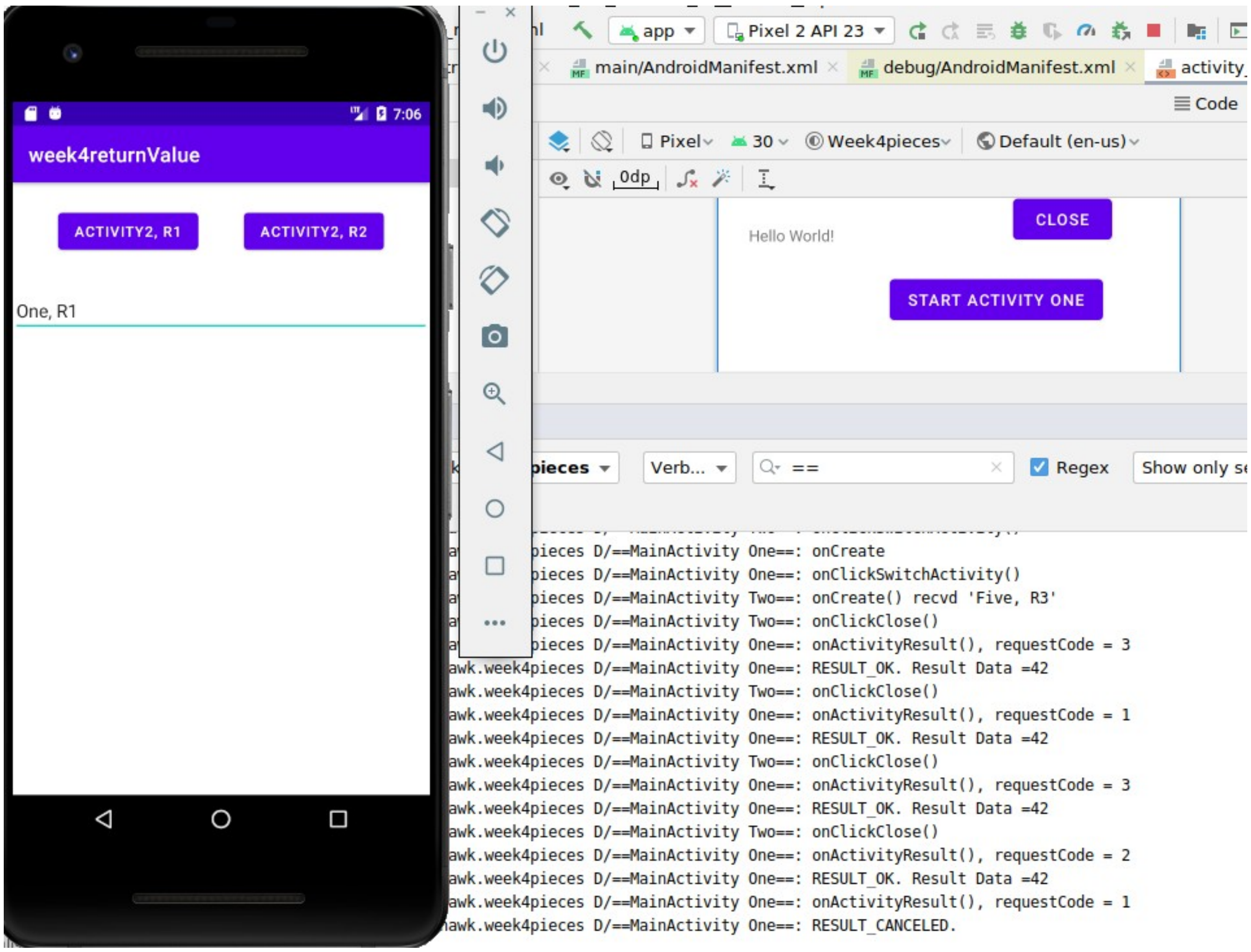
```
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:616)
     Caused by: java.lang.reflect.InvocationTargetException <1 internal call>
        at androidx.appcompat.app.AppCompatViewInflater$DeclaredOnClickListener.onClick(AppCompatViewInflater.java:409) <7 more...>
 <1 internal call> <2 more...>
     Caused by: java.lang.ClassCastException: com.google.android.material.button.MaterialButton cannot be cast to android.widget
.EditText
        at ca.mohawk.week4pieces.MainActivity.onClickSwitchActivity(MainActivity.java:25) <1 internal call> <8 more...>
 <1 internal call> <2 more...>
```

Android's type system is particularly complex and getting a solid handle on that type system takes time.

Try for example adding the onClick handler to the Constraint Layout for Activity One. You are allowed to do this, and we can use the activity background to handle the onclick listener. Now the editor cannot help you if you choose to use the same handler.

You could also make use of java's instanceof operator to check the type. This makes casting safer, although using getId() sidesteps the problem completely.

Add a button to the second activity. Use this button to launch the first activity (instead of closing it), similar to what we did in the second assignment. Add appropriate logging messages and see how deep you can get the stack to go.



```
pieces D/==MainActivity One==: onCreate
pieces D/==MainActivity One==: onClickSwitchActivity()
pieces D/==MainActivity Two==: onCreate() recvd 'Five, R3'
pieces D/==MainActivity Two==: onClickClose()
pieces D/==MainActivity One==: onActivityResult(), requestCode = 3
awk.week4pieces D/==MainActivity One==: RESULT_OK. Result Data =42
awk.week4pieces D/==MainActivity Two==: onClickClose()
awk.week4pieces D/==MainActivity One==: onActivityResult(), requestCode = 1
awk.week4pieces D/==MainActivity One==: RESULT_OK. Result Data =42
awk.week4pieces D/==MainActivity Two==: onClickClose()
awk.week4pieces D/==MainActivity One==: onActivityResult(), requestCode = 3
awk.week4pieces D/==MainActivity One==: RESULT_OK. Result Data =42
awk.week4pieces D/==MainActivity Two==: onClickClose()
awk.week4pieces D/==MainActivity One==: onActivityResult(), requestCode = 2
awk.week4pieces D/==MainActivity One==: RESULT_OK. Result Data =42
awk.week4pieces D/==MainActivity One==: onActivityResult(), requestCode = 1
hawk.week4pieces D/==MainActivity One==: RESULT_CANCELED.
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    public static final String REQUEST_RESULT="REQUEST_RESULT";
    private static final String tag = "==MainActivity One==";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(tag,"onCreate");
        setContentView(R.layout.activity_main);
    }

    public void onClickSwitchActivity(View view) {
        Log.d(tag, "onClickSwitchActivity()");

        EditText editText = findViewById(R.id.editTextData);
        String text = editText.getText().toString();
        Intent intent = new Intent(this, MainActivity2.class);
        intent.putExtra(Intent.EXTRA_TEXT,text);
        if (view.getId() == R.id.button) {
            startActivityForResult(intent, 1);
        } else if (view.getId() == R.id.button2) {
            startActivityForResult(intent, 2);
        } else if (view.getId() == R.id.constraintlayout) {
            startActivityForResult(intent, 3);
        } else {
            Log.d(tag, "Unrecognized Button Pressed.");
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        Log.d(tag, "onActivityResult(), requestCode = " + requestCode);
        if (resultCode==RESULT_OK) {
            Log.d(tag, "RESULT_OK. Result Data =" +  data.getIntExtra(REQUEST_RESULT, 0));
        } else if (resultCode==RESULT_CANCELED) {
            Log.d(tag, "RESULT_CANCELED. ");
        } else {
            Log.d(tag, "requestCode = " + requestCode +
                    " resultCode = " + resultCode + "Intent = " + data);
        }
    }
}
```

MainAcitivity2.java

```
public class MainActivity2 extends AppCompatActivity {

    private static final String tag = "==MainActivity Two==";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        TextView textView = (TextView)findViewById(R.id.textViewText);
        String extra = "";
        if (getIntent()!=null && getIntent().hasExtra(Intent.EXTRA_TEXT)) {
            extra = getIntent().getStringExtra(Intent.EXTRA_TEXT);
            textView.setText(extra);
        }
        Log.d(tag, "onCreate() recvd '" + extra + "'");
    }

    public void onClickClose(View view) {
        Log.d(tag, "onClickClose()");
        Intent returnIntent = new Intent();
        returnIntent.putExtra(MainActivity.REQUEST_RESULT,42);
        setResult(RESULT_OK, returnIntent);
        finish();
    }

    public void onClickSwitchActivity(View view) {
        Log.d(tag, "onClickSwitchActivity()");
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }
}
```