Lesson 4 Part 2

# The sequence of Intents

A screenshot of a cell phone

Description automatically generated

# 

# TODO 8.6, in DataEntryActivity, onCreate()

This is an implicit intent to launch the image gallery.

**REQUEST\_CODE\_IMAGE** is an arbitrary unique integer constant that identifies this particular implicit intent and is used in **onActivityResult().**

This is necessary as you may have more than one intent from which you are expecting a result from.

//*TODO 8.6 when the selectImage button is clicked, set up an Implicit Intent to the gallery*buttonSelectImage.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent();  
 intent.setType("image/\*");  
 intent.setAction(Intent.*ACTION\_GET\_CONTENT*);  
 startActivityForResult(intent, REQUEST\_CODE\_IMAGE);  
 }  
});

# TODO 8.7 In DataEntryActivity, onActivityResult callback

After the user has selected an image in the image gallery, the result is handled in this callback. This code gets the return intent, extracts the bitmap from the Uri, stores it in **bitmapSelected** and displays it in the **imageViewSelected** widget.

//*TODO 8.7 Complete OnActivityResult so that the selected image is displayed in the imageView*@Override  
public void onActivityResult(int requestCode, int resultCode, Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
  
 if( requestCode == REQUEST\_CODE\_IMAGE  
 && resultCode == Activity.*RESULT\_OK*){  
 //do stuff here - recipe code follows, don't fret ..  
 try{  
 Uri uri = data.getData();  
 InputStream inputStream = this.getContentResolver()  
 .openInputStream(uri);  
 bitmapSelected = Utils.*convertStreamToBitmap*(inputStream);  
 imageViewSelected.setImageBitmap(bitmapSelected);  
 }catch(FileNotFoundException ex){  
 ex.printStackTrace();  
 }  
  
 }  
}

# TODO 8.8 In DataEntryActivity, onCreate callback

When the OK button is created, the data is extracted and added to the database using the **insertOneRow()** method of the **CharaDbHelper** class.

Because **DataEntryActivity** was invoked using **startActivityForResult()** within **MainActivity**, it must return a result. The last two lines of code here take care of that. In this particular case, as long as an image is selected, we pass back the result code **Activity.RESULT\_OK**. Note that we don’t pass back any other data.

//*TODO 8.8 when the OK button is clicked, add the data to the db*Button buttonOK = findViewById(R.id.*buttonOK*);  
buttonOK.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 String name = editTextName.getText().toString();  
 String description = editTextDescription.getText().toString();  
  
 if( bitmapSelected == null){  
 Toast.*makeText*(DataEntryActivity.this,  
 "no image selected",  
 Toast.*LENGTH\_LONG*).show();  
 }else{  
  
 CharaDbHelper.CharaData charaData  
 = new CharaDbHelper.CharaData(name,description,bitmapSelected);  
 charaDbHelper.insertOneRow( charaData );  
  
 Toast.*makeText*(DataEntryActivity.this,  
 "inserting to database",  
 Toast.*LENGTH\_LONG*).show();  
  
 //In MainActivity we started DataEntryActivity using startActivityForResult  
 //Hence,DataEntryActivity must return a result and hence this set of code  
 Intent returnIntent = new Intent();  
 setResult(Activity.*RESULT\_OK*, returnIntent);  
 }  
  
 }  
});

# TODO 8.9 Back in MainActivity, onActivityResult

Now that **DataEntryActivity** has returned a result, **onActivityResult()** is written to process that.

Apart from the result code, **DataEntryActivity** did not really return any data back, hence we don’t have any data to use.

Hence, this code just displays a toast showing the number of rows in the database.

//*TODO 8.9 Complete on activity result to decide what happens after the data is uploaded*//*TODO 8.10 [on your own] Add a cancel button to DataEntryActivity and complete the result-cancelled portion*@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
  
 if(requestCode == *REQUEST\_CODE\_FAB*){  
  
 if(resultCode == Activity.*RESULT\_OK*) {  
  
 //My choice is to display a Toast that says how many rows our database has  
 long numberOfRows = charaDbHelper.queryNumRows();  
 Toast.*makeText*(MainActivity.this,  
 "Number of rows: " + numberOfRows,  
 Toast.*LENGTH\_LONG*).show();  
 }  
  
 if(resultCode == Activity.*RESULT\_CANCELED*){  
 //complete it yourself if you'd like  
 }  
  
 }  
  
  
}//onActivityResult