Mobile App Development

> Exercises: Todo Detail App

The previous lab exercises introduced the Android Studio IDE and the "Hello World" Android App. Two classes central to Android architectural design are the `Activity` and the `Intent` classes. The aim of this exercise is to understand and implement Activity and Intent objects.

Activity and Intent

Activity objects are created by the Android OS Activity Manager and Intent is an object that can be used to communicate with the Android OS. Intent is used here to pass data between two activities but it provides different constructors for other data communication to services, broadcast receivers and content providers (more about this later, we will concentrate on Activity for now).

The point to note is the App manifest defines the launch activity and the complexity here is dealing with intent objects and the methods for communicating with the Activity Manager in the Android OS. The following sequence diagram gives an overview of the implementation.

TodoDetail App exercise

Snippetts of code are used here for explanation; the full working code can be accessed here:
https://github.com/ebbi/android

To help achieve the aim of understanding Activity and Intent, our objective is to create a *todoDetailActivity* and pass the todo array index from the main *todo activity* to it. The idea illustrates the pattern for passing small amounts of data, such as indexes between activities. For the todo app, it is a use case to display details of a particular activity.

Always start with a new *git* branch to try new ideas and merge or delete at the end.

Make sure you are on master branch and create a new branch `todo_detail`

```
git status
git checkout master
git branch todo_detail
git checkout todo_detail
git status
```

Use Android Studio to create a new Activity and name it `TodoDetailActivity`:

```
App > New > Activity > Empty Activity
```

Update the `TodoDetailActivity` view and include a textView object. We will display the data passed from the main activity in this textView. Name the textView, `textViewTodoDetail`. The complete code is here for reference.

In Android an `extra` is a *key, value* pair. Intent objects have `putExtra` methods that the calling activity can use to `putExtra` arbitary key, value data pairs into the intent object. This is similar to passing

parameters to a constructer, except through the Activity Manager in the Android OS.

Being event driven an activity may be started from many events. A design question arises, where should the intent data be declared? In the calling activity or the activity that uses the data. We will follow the second pattern as it provides for better encapsulation; the logic being the todoDetail object is better placed to define the data it needs than the calling activities. Here is the static method for TodoDetailActivity, declaring the data it needs, namely, the `todoIndex`. Any calling activity would call this static method and pass the necessary key, value data pair in an intent object which is acheived by the put and get extra helper methods. Here is the code:

```
private static final String TODO_INDEX = "com.example.todoIndex";

public static Intent newIntent(Context packageContext, int todoIndex){
  Intent intent = new Intent(packageContext, TodoDetailActivity.class);
  intent.putExtra(TODO_INDEX,todoIndex);
  return intent;
}

and within the onCreate(Bundle) lifecycle method retrive the key value:

int todoIndex = getIntent().getIntExtra(TODO_INDEX, 0);
```

The calling Activity would now have to create an intent object and include the *todoIndex* value as a key value pair. Here is the code:

```
Intent intent = TodoDetailActivity.newIntent(TodoActivity.this, mTodoIndex);
startActivity(intent);
```

See if you can include the code above in a listener for a detail button in the Activity page. Here is the code for the `TodoActivity` class and here the code for TodoDetailActivity.

For further testing, try to rotate with the detail view and you notice the same crash as before with the Activity lifecycle loosing the todo index with the change in state from portrait to landscape.

> Try the previous solution with the "rotation problem" for the todoActivity with todoDetailActivity. Also, remember the `-land` config convention for an alternate landscape view for the todoDetailActivity.

The complete code is here for reference.

Return data from child Activity

Not suprisingly, intent objects are used to return data from the called activity. The following code would return a boolean value from the todoDetail activity back to the todoActivity. We will use this value to check the user repsponse and dislay a tick character for todos checked as completed.

Here is the method that adds a key, value pair to the intent object. The OS will make this object available to the calling method can the `onActivityResult(int, int, intent)` methid can be used to retreive the int result status ints as well the intent object. Finally, convinience getExtra methods can retreive the key, value pairs of date. Here the code:

```
/*
    requestCode is the integer request code originally supplied to startActivityForResult
    resultCode is the integer result code returned by the child activity through its setResult()
    intent date attached with intent "extras"
*/
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
```

```
    if (requestCode == IS_SUCCESS ){
        if (intent != null) {
            // data in intent from child activity
            boolean isTodoComplete = intent.getBooleanExtra(IS_TODO_COMPLETE, false);
            updateTodoComplete(isTodoComplete);
        } else {
            Toast.makeText(this, R.string.back_button_pressed, Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(this, R.string.request_code_mismatch,
                Toast.LENGTH_SHORT).show();
    }

}

private void updateTodoComplete(boolean is_todo_complete) {

    final TextView textViewTodo;
    textViewTodo = (TextView) findViewById(R.id.textViewTodo);

    if (is_todo_complete) {
        textViewTodo.setBackgroundColor(
                ContextCompat.getColor(this, R.color.backgroundSuccess));
        textViewTodo.setTextColor(
                ContextCompat.getColor(this, R.color.colorSuccess));

        setTextViewComplete("\u2713");
    }

}
```

Reflection and QA

What is an Android Activity?

What is an Android intent?

What are Intent filters intent?

What is the role of an `ActivityManager` in an Android App?

Waht is the difference between `startActivity(Intent)` and `startActivityForResult(intent, int)`

Being event driven an activity may be started from many events. A design question arises, where should the intent data be declared? In the calling activity or the activity that uses the data?