Mobile App Development

Exercises: Todo List App

The following is a start to a complete *Todo List App*. The source code can be found at https://github.com/ebbi/TodoListApp/. Please use this as reference only. The following instructions will build the same git branches leading to the final prototype App. Here is the sequential list of git branches and corresponding links to the Android best practice pages implemented in the TodoApp.

1. TodoListApp - initial setup
2. TodoModel - git branch for the `TodoModel` and `Todo` classes
3. Views - git branch for views for the App
4. Controllers - git branch for the controllers for the App

The following Android developer pages has further explanations for the classes used in this implementation.

- Fragments - See https://developer.android.com/training/basics/fragments/index.html
- FragmentManager - See https://developer.android.com/training/basics/fragments/fragment-ui.html
- RecyclerView - See https:/https://developer.android.com/guide/topics/ui/layout/recyclerview.html and https://developer.android.com/training/material/lists-cards.html
- Fragment Arguments - See https://developer.android.com/training/basics/fragments/communicating.html

Visit https://developer.android.com/training/index.html for an overview and further best practice examples.

The following are possible refactor of the current implemetation for a more complete App

- viewpager - See https://developer.android.com/training/implementing-navigation/lateral.html
- toolbar - See https://developer.android.com/training/appbar/setting-up.html

The use cases considered essentially manipulate lists with a seperate view for the detail of each list item. As such the same code can be applied to any list; in this case it is a todo list.

The following class diagram depicts the design to be implemented.

list view

A `TodoListActivity` has an empty `FrameLayout` view which is populated dynamically by a `FragmentManager`. The implementation defines a seperate controller for the detail view of list items.

The support class RecyclerView is utilised for scrolling and list boundary checkes as well as styling. The implementation is suitable for adding a toolbar and menu classes for navigation.

Sketches as well as clarifying the views also identify the set of data attributes to be modelled.

list view dettail view

Todo List App

Create a project with a blank TodoListActivity.

Check dependencies for the project

```
File > Project Structure
app > Dependencies

If the recyclerview is not on the list then try:
+ > Library dependency
  scroll down and click on:
  "com.android.support:recyclerview-v7:27.0.0"
  click ok.

  Similarly, appcompat-v7:27.0.0 should be present otherwise add it
  delete other versions

(N.B. could use API v7:26 so long as both appcompat and recyclerview are the same version)
Wait for any Gradle Sync to complete

Run the App to check for any initial setup errors.
```

Initialise Git

```
Switch to the Project view and edit the .gitignore file
replace the content with the content of the  file here 

Add the following to the start of the .gitignore file

# es added for Studio 3
.idea/misc.xml
.idea/vcs.xml

Save the file

Switch back from Project to Android view.

In the Terminal window, initialise git.

git init
git status

git add .
git status

git commit -am "initial setup"
git status
```

The model

The `TodoModel` has methods for a list of `Todo` objects as well as other helper methods for finding an individual `todo` based on its ID.

Create a git branch for implementing the model

```
git checkout master
git status

git branch todomodel
git checkout todomodel
git status
```

The `todo` class is a Plane Old Java Object(POJO) with getter and setter methods; here is a typical set of attributes to start with. Create the Todo class with the following code.
if prompted to add the file to Git; select Don't ask again and click `Yes`

```
public class Todo {
    private UUID mId;
    private String mTitle;
    private String mDetail;
    private Date mDate;
    private boolean mIsComplete;

    public Todo() {
        mId = UUID.randomUUID();
        mDate = new Date();
    }

}
```

```
alt Enter to resolve imports
right-mouse click on the attributes and create the getter and setter methods
```

In Android a `Singleton` is in scope as long as the application is in memory. The `TodoModel` implements a `Singleton` and its used to pass data between controllers. Create a class named, `TodoModel` and insert the code:

```
import android.content.Context;

import java.util.ArrayList;
import java.util.UUID;

public class TodoModel {

    private static TodoModel sTodoModel;

    private ArrayList<Todo> mTodoList;

    public static TodoModel get(Context context) {
        if (sTodoModel == null) {
            sTodoModel = new TodoModel(context);
        }
        return sTodoModel;
    }

    private TodoModel(Context context){
        mTodoList = new ArrayList<>();

        // refactor to pattern for data plugins
        // simulate some data for testing

        for (int i=0; i < 3; i++){
            Todo todo = new Todo();
            todo.setTitle("Todo title " + i);
            todo.setDetail("Detail for task " + todo.getId().toString());
            todo.setComplete(false);

            mTodoList.add(todo);
        }

    }

    public Todo getTodo(UUID todoId) {
        for (Todo todo : mTodoList) {
            if (todo.getId().equals(todoId)){
                return todo;
            }
        }

        return null;
    }

    public ArrayList<Todo> getTodos() {

        return mTodoList;

    }

    public void addTodo(Todo todo){

        mTodoList.add(todo);

    }

}
```

Commit the git branch before moving onto implementing the todo fragment class

```
git status
git add .
git commit -am "todomodel complete"
git checkout master
git merge todomodel
```

Starting an Activity from a fragment

A typical use case would be a selection of a todo list item which leads to the item detail being displayed by a second controller.

Not surprisingly, starting an activity from a fragment is very similar to starting an activity from another activity.

Views

```
git checkout master
git branch views
git checkout views
```

Save the following string constants in the `res/values/strings.xml`

```
<resources>
    <string name="app_name">TodoListApp</string>
    <string name="todo_title">todo title</string>
    <string name="todo_title_hint">todo title hint</string>
    <string name="todo_title_label">One line description of the todo</string>
    <string name="todo_detail_label">detail of what to do</string>
    <string name="todo_complete_label">is it complete?</string>
    <string name="todo_date">Todo date</string>
</resources>
```

We need to define a generic `FrameLayout` as a container view for fragments that are added by Activity controllers.

Save the following in `res/layout/activity_fragment.xml`

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
             android:id="@+id/fragment_container"
             android:layout_width="match_parent"
             android:layout_height="match_parent" />
```

The support class RecyclerView is utilised for scrolling and list boundary checkes as well as styling. This requires a `RecyclerView` for the todo list fragments.

Save the following in the `res/layout/fragment_todo_list.xml`

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- A RecyclerView with some commonly used attributes -->
<android.support.v7.widget.RecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/todo_recycler_view"
    android:scrollbars="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
/>
```

The `RecyclerView` list items display the list item rows with each row containing the `todo_title` and `todo_date`.

Save the following in the `res/layout/list_item_todo.xml`

```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="wrap_content"
              android:orientation="vertical"
              android:padding="8dp">

    <TextView xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/todo_title"
              android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:text="@string/todo_title"/>

    <TextView xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/todo_date"
              android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:text="@string/todo_date"/>

</LinearLayout>
```

Each todo is a `LinearLayout` with `TextView`, `EditText`, `Button` and `CheckBox` view widgets

Save the following string constants in the `res/layout/fragment_todo.xml`

```
<?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"
              android:layout_margin="16dp"
              android:weightSum="1">

    <TextView
        style="?android:listSeparatorTextViewStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/todo_title_label"/>

    <EditText
        android:id="@+id/todo_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/todo_title_hint"/>

    <TextView
        style="?android:listSeparatorTextViewStyle"
        android:layout_width="match_parent"
        android:layout_height="@android:dimen/notification_large_icon_height"
        android:text="@string/todo_detail_label"/>

    <Button
        android:id="@+id/todo_date"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />

    <CheckBox
        android:id="@+id/todo_complete"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/todo_complete_label"/>

</LinearLayout>
```

```
git status
git add .
git commit -am "views complete"

git checkout master
git merge views
```

Controllers

An Activity controller defines a placeholder in its layout for the fragment's view and also manages the lifecycle of the fragment instance.

Activity and fragment life cycles are understandably similar with typical overrides of `stopped`, `paused`, and `resumed` states; *the crucial difference is the fragment methods are called by the Activity and not the OS.*

There are two approaches to hosting a `fragment` in an `activity`, namely, to add the fragments to the activity's layout or alternatively to the activity's code. The second approach is more complex but more flexible as the fragments can be controlled at run time. This second approach is implemented here.

```
git checkout master
git branch controllers
git checkout controllers
```

Begin by `TodoListActivity` class

Save the following in `TodoListActivity` in the java folder

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v7.app.AppCompatActivity;

public class TodoListActivity extends AppCompatActivity {

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

        FragmentManager fm = getSupportFragmentManager();
        Fragment fragment = fm.findFragmentById(R.id.fragment_container);

        if (fragment == null){
            TodoListFragment todoListFragment = new TodoListFragment();
            fm.beginTransaction()
                    .add(R.id.fragment_container, todoListFragment)
                    .commit();
        }

    }
}
```

And, similarly `TodoActivity` class

Save the following in `TodoActivity` in the java folder

```
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v7.app.AppCompatActivity;

import java.util.UUID;

public class TodoActivity extends AppCompatActivity {

    public static final String EXTRA_TODO_ID = "todo_id";

    public static Intent newIntent(Context packageContext, UUID todoId) {
        Intent intent = new Intent(packageContext, TodoActivity.class);
        intent.putExtra(EXTRA_TODO_ID, todoId);
        return intent;
    }

    /*
    To decouple the fragment and make it reusable, the TodoFragment has a newInstance method
    that receives a todoId and returns the fragment
     */
    protected Fragment createFragment(){
        UUID todoId = (UUID) getIntent().getSerializableExtra(EXTRA_TODO_ID);
        return TodoFragment.newInstance(todoId);
    }

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

        FragmentManager fm = getSupportFragmentManager();
        Fragment fragment = fm.findFragmentById(R.id.fragment_container);

        if (fragment == null){

            Fragment todoFragment = createFragment();

            fm.beginTransaction()
                    .add(R.id.fragment_container, todoFragment)
                    .commit();
        }

    }

}
```

Almost identical `onCreate` method and room to refactor to an Abstract class

Note, the explicit `intent` uses `putExtra` and passes in a key, value map for the `todoId`. Any event such as `onClick` can use the `newIntent` while passing in the `todoId`; the code would be similat to the following `onClick` event.

```
public void onClick(View view) {
    Intent todoIntent = TodoActivity.newIntent(getActivity(), mTodo.getId());
    startActivity(todoIntent);
}
```

The `TodoListFragment` controller is an example of calling the explicit intent to display the todo fragment for the particular `todoId`.

Save the following in `TodoListfragment` in the java folder

```
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.List;

public class TodoListFragment extends Fragment {
```

```
private RecyclerView mTodoRecyclerView;
TodoAdapter mTodoAdapter;

@Override
public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

}

@Override
public View onCreateView(LayoutInflater inflater,
                         ViewGroup container,
                         Bundle savedInstanceState) {

    View view = inflater.inflate(R.layout.fragment_todo_list, container, false);

    mTodoRecyclerView = (RecyclerView) view.findViewById(R.id.todo_recycler_view);
    mTodoRecyclerView.setLayoutManager( new LinearLayoutManager(getActivity()) );

    updateUI();

    return view;

}

@Override
public void onResume() {
    super.onResume();
    updateUI();
}


private void updateUI(){

    ArrayList todos = new ArrayList<>();
    TodoModel todoModel = TodoModel.get(getContext());
    todos = todoModel.getTodos();

    if (mTodoAdapter == null) {
        mTodoAdapter = new TodoAdapter(todos);
        mTodoRecyclerView.setAdapter(mTodoAdapter);
    } else {
        mTodoAdapter.notifyDataSetChanged();
    }

}

public class TodoHolder extends RecyclerView.ViewHolder
        implements View.OnClickListener {

    private Todo mTodo;
    private TextView mTextViewTitle;
    private TextView mTextViewDate;

    public TodoHolder(LayoutInflater inflater, ViewGroup parent) {

        super(inflater.inflate(R.layout.list_item_todo, parent, false));

        itemView.setOnClickListener(this);

        mTextViewTitle = (TextView) itemView.findViewById(R.id.todo_title);
        mTextViewDate = (TextView) itemView.findViewById(R.id.todo_date);

    }

    @Override
    public void onClick(View view) {
        // have a Toast for now
        Toast.makeText(
                getActivity(),
                    mTodo.getTitle() + " clicked",
                    Toast.LENGTH_SHORT)
                .show();

                Intent intent = TodoActivity.newIntent(getActivity(), mTodo.getId());
                startActivity(intent);

            }

            public void bind(Todo todo){
                mTodo = todo;
                mTextViewTitle.setText(mTodo.getTitle());
                mTextViewDate.setText(mTodo.getDate().toString());
            }

        }

        public class TodoAdapter extends RecyclerView.Adapter<TodoListFragment.TodoHolder> {

            private List<Todo> mTodos;

            public TodoAdapter(List<Todo> todos) {
                mTodos = todos;
            }

            @Override
            public TodoListFragment.TodoHolder onCreateViewHolder(ViewGroup parent, int viewType) {
                LayoutInflater layoutInflater = LayoutInflater.from(getActivity());

                return new TodoHolder(layoutInflater, parent);
            }

            @Override
            public void onBindViewHolder(TodoHolder holder, int position) {
                Todo todo = mTodos.get(position);
                holder.bind(todo);
            }

            @Override
            public int getItemCount() {
                return mTodos.size();
            }

        }
    }
```

The `todoId` is now in the `TodoActivity` intent. The `todoId` needs to be retreived in the `todoFragment` class to retrieve the fragment and display it.

The easiest and incorrect way is for the `TodoFragment` to use the `getActivity` method to access the `TodoActivity`'s intent directly. However, this is bad practice as it makes the `TodoFragment` coupled with activity that has an intent with a todoId. This may seem a reasonable assumption but it restricts the todofragment from being a generic fragment.

The solution is to use a `Bundle` for the `todoId` in the fragment rather than the activity; hence decoupling the fragment with its own `argument bundle`.

Save the following in `Todofragment` in the java folder

```
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
```

```java
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.EditText;

import java.util.UUID;

public class TodoFragment extends Fragment {

    private static final String ARG_TODO_ID = "todo_id";

    private Todo mTodo;
    private EditText mEditTextTitle;
    private Button mButtonDate;
    private CheckBox mCheckBoxIsComplete;

    /*
    Rather than the calling the constructor directly, Activity(s) should call newInstance
    and pass required parameters that the fragment needs to create its arguments.
     */
    public static TodoFragment newInstance(UUID todoId) {
        Bundle args = new Bundle();
        args.putSerializable(ARG_TODO_ID, todoId);

        TodoFragment fragment = new TodoFragment();
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);

        /*
         Fragment accessing the intent from the hosting Activity as in the following code snippet
         allows for simple code that works.

        UUID todoId = (UUID) getActivity()
                .getIntent().getSerializableExtra(TodoActivity.EXTRA_TODO_ID);

         The disadvantage: TodoFragment is no longer reusable as it is coupled to Activities whoes
         intent has to contain the todoId.

         Solution: store the todoId in the fragment's arguments bundle.
            See the TodoFragment newInstance(UUID todoId) method.

         Then to create a new fragment, the TodoActivity should call TodoFragment.newInstance(UUID)
         and pass in the UUID it retrieves from its extra argument.

        */

        UUID todoId = (UUID) getArguments().getSerializable(ARG_TODO_ID);

        mTodo = TodoModel.get(getActivity()).getTodo(todoId);

    }

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater,
                             @Nullable ViewGroup container,
                             @Nullable Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_todo, container, false);

        mEditTextTitle = (EditText) view.findViewById(R.id.todo_title);
        mEditTextTitle.setText(mTodo.getTitle());
        mEditTextTitle.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {
                // This line is intentionally left blank
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                mTodo.setTitle(s.toString());
            }

            @Override
            public void afterTextChanged(Editable s) {
                // This line is intentionally left blank
            }
        });

        mButtonDate = (Button) view.findViewById(R.id.todo_date);
        mButtonDate.setText(mTodo.getDate().toString());
        mButtonDate.setEnabled(false);

        mCheckBoxIsComplete = (CheckBox) view.findViewById(R.id.todo_complete);
        mCheckBoxIsComplete.setOnCheckedChangeListener(new OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                Log.d("DEBUG **** TodoFragment","called onCheckedChanged");
                mTodo.setComplete(isChecked);
            }
        });

        return view;

    }
}
```

Android studio does a good job of updating the manifest file. Double check that there is an entry for the `TodoActivity` class in the `manifests > AndroidManifest.xml`, if not, enter the following activity declaration:

```
<activity android:name=".TodoActivity">
</activity>
```

Run the App

Reflection and QA

There are two approaches a `fragment` can access data in it's `Activity`'s intent, directly with similar code to this:

```
UUID todoId = (UUID) getActivity()
                .getIntent().getSerializableExtra(TodoActivity.EXTRA_TODO_ID);
```

Alternatively, the ID is stored in the `Fragment argument bundle`. And rather than calling the constructor, a `newInstance` method is used to receive the ID parameter from any calling Activity.

Explain with a code walkthrough how the second approach is implemented

Which is the better approach and why?