Mobile App Development

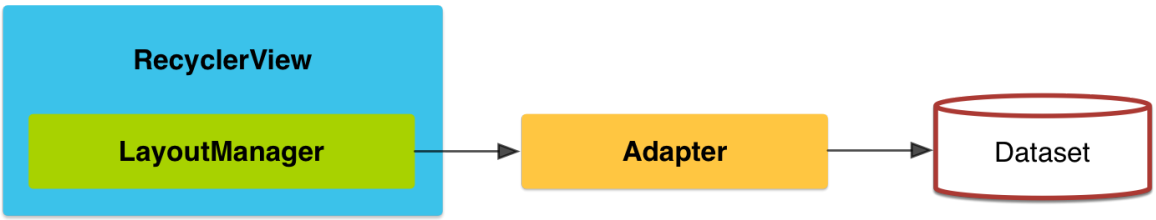
> Exercises: Todo RecyclerView App

Best practice guidelines - Creating Lists

This exercise implements the best practice guidelines for creating Lists. The `RecyclerView` object is a more advanced and flexible version of `ListView`. It has a:

- Layout manager for positioning list items
- Default animation for common operations such as addition or removal of items.

The idea is depicted below.



The `RecyclerView` has a layout manager which determines when to reuse view items (fragments) as they appear and disappear from the display. To reuse (or recycle) the view, the layout manager in turn uses an `adapter` object to replace the content of the view with a different element from the dataset.

> Create a new project with a blank activity.
>
> Consider the following code for the `adapter` and `RecyclerView`. Create a new class and name it, `TodoListFragment` and include the following code.

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

public class TodoListFragment extends Fragment {

    private RecyclerView mTodoRecyclerView;
    private TodoAdapter mTodoAdapter;
    private Todo mTodo;

    @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;

    }

    private void updateUI(){

        /*
        TODO: refactor to data layer
         */
        ArrayList todos = new ArrayList<>();

        for (int i=0; i < 100; i++){
            Todo todo = new Todo();
            todo.setTitle("Todo number " + i);
            todo.setIsComplete(i % 2 == 0);
            todos.add(todo);
        }

        mTodoAdapter = new TodoAdapter(todos);
        mTodoRecyclerView.setAdapter(mTodoAdapter);

    }

    public class TodoHolder extends RecyclerView.ViewHolder {

        private TextView mTextViewTitle;
        private TextView mTextViewDate;

        public TodoHolder(LayoutInflater inflater, ViewGroup parent) {
            super(inflater.inflate(R.layout.todo_list_item, parent, false));

            mTextViewTitle = (TextView) itemView.findViewById(R.id.textViewTodoTitle);
            mTextViewDate = (TextView) itemView.findViewById(R.id.textViewTodoDate);

        }

        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();
        }
    }

}
```

Following best practice, the `TodoListFragment` view can be added dynamically in a fragment transaction; see the code below and include the code in the `MainActivity` class.

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

public class MainActivity extends AppCompatActivity {

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

        FragmentManager fm = getSupportFragmentManager();

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

    }

}
```

Each `todo` is encapsulated in a POJO (plain Old Java Object) with getter and setter methods. `UUID` is from a Unix utility library and provides a unique ID for each Todo (analagous to a Primary Key, making each Todo instance unique or first normal form). Hete is the code for the `Todo` class.

```
import java.util.Date;
import java.util.UUID;

/**
 * Created by Ebbi on 11/10/2017.
 */

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();
    }

    public void setIsComplete(boolean todoIsComplete) {
        mIsComplete = todoIsComplete;
    }

    public boolean isIsComplete() {

        return mIsComplete;
    }

    public UUID getId() {
        return mId;
    }

    public String getTitle() {
        return mTitle;
    }

    public String getDetail() {
        return mDetail;
    }

    public Date getDate() {
        return mDate;
    }

    public void setId(UUID todoId) {
        mId = todoId;
    }

    public void setTitle(String title) {
        mTitle = title;
    }

    public void setDetail(String detail) {
        mDetail = detail;
    }

    public void setDate(Date todoDate) {
        mDate = todoDate;
    }
}
```

The corresponding views include an empty `FrameLayout` with a `fragment_container` to dynamically populate with `todo list items`. Here is the code for the `res/layout/activity_main.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_height="match_parent"
    android:layout_width="match_parent"
    />
```

And the `todo_recycler_view` is a container for dynamically loaded `todo_list_items`.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.RecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/todo_recycler_view"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

And the view for the `res/layout/todo_list_item.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
        android:id="@+id/textViewTodoDate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/todo_date"
        />

    <TextView
        android:id="@+id/textViewTodoTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/todo_title"
        android:textSize="24sp"/>
</LinearLayout>
```

Run and see a view with a smooth scrolling acheived by reuse of the Recycler class.

And the parent `Activity` would implement the `interface`

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

public class MainActivity extends AppCompatActivity {

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

        FragmentManager fm = getSupportFragmentManager();

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

```
    }
}
```

The above is esseentially the answer. The following is the complete code as an example of a dynamically loaded fragment todo app.

Create a new project with a blank Activity and name it, `FragB`

A `interface` in the `Fragment` class which is implemented in the parent `Activity` class is standard pattern for passing data from the fragment to its parent activity.

Consider the following and create a class and save it as `TodoListFragment`

```
import android.app.Activity;
import android.os.Build;
import android.os.Bundle;
import android.support.v4.app.ListFragment;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class TodoListFragment extends ListFragment {
    OnTodoSelectedListener mCallback;

    // The container Activity must implement this interface so the frag can deliver messages
    public interface OnTodoSelectedListener {
        /** Called by TodoListFragment when a list item is selected */
        public void onTodoSelected(int position);
    }

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

        // We need to use a different list item layout for devices older than Honeycomb
        int layout = Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB ?
                android.R.layout.simple_list_item_activated_1 : android.R.layout.simple_list_item_1;

        // Create an array adapter for the list view, using the TodoModel headlines array
        setListAdapter(new ArrayAdapter<String>(getActivity(), layout, TodoModel.Todos));
    }

    @Override
    public void onStart() {
        super.onStart();

        // When in two-pane layout, set the listview to highlight the selected list item
        // (We do this during onStart because at the point the listview is available.)
        if (getFragmentManager().findFragmentById(R.id.todo_fragment) != null) {
            getListView().setChoiceMode(ListView.CHOICE_MODE_SINGLE);
        }
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);

        // This makes sure that the container activity has implemented
        // the callback interface. If not, it throws an exception.
```

```
        try {
            mCallback = (OnTodoSelectedListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnTodoSelectedListener");
        }
    }

    @Override
    public void onListItemClick(ListView l, View v, int position, long id) {
        // Notify the parent activity of selected item
        mCallback.onTodoSelected(position);

        // Set the item as checked to be highlighted when in two-pane layout
        getListView().setItemChecked(position, true);
    }

}
```

The `MainActivity` implements the `OnTodoSelectedListener`. Consider the `MainActivity` class and replace it with the following code:

```
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentTransaction;

public class MainActivity extends FragmentActivity
        implements TodoListFragment.OnTodoSelectedListener {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Check that the activity is using the layout version with
        // the fragment_container FrameLayout
        if (findViewById(R.id.fragment_container) != null) {

            // However, if we're being restored from a previous state,
            // then we don't need to do anything and should return or else
            // we could end up with overlapping fragments.
            if (savedInstanceState != null) {
                return;
            }

            // Create a new Fragment to be placed in the activity layout
            TodoListFragment firstFragment = new TodoListFragment();

            // In case this activity was started with special instructions from an
            // Intent, pass the Intent's extras to the fragment as arguments
            firstFragment.setArguments(getIntent().getExtras());

            // Add the fragment to the 'fragment_container' FrameLayout
            getSupportFragmentManager().beginTransaction()
                    .add(R.id.fragment_container, firstFragment).commit();
        }
    }


    public void onTodoSelected(int position) {
        // Implement interface onTodoSelected
        // The user selected the todo item from the TodoListFragment

        // Capture the todo fragment from the activity layout
        TodoFragment todoFragment = (TodoFragment)
```

```
                getSupportFragmentManager().findFragmentById(R.id.todo_fragment);

        if (todoFragment != null) {
            // If todo frag is available, we're in two-pane layout...

            // Call a method in the TodoFragment to update its content
            todoFragment.updateTodoView(position);

        } else {
            // If the frag is not available, we're in the one-pane layout and must swap frags...

            // Create fragment and give it an argument for the selected article
            TodoFragment newFragment = new TodoFragment();
            Bundle args = new Bundle();
            args.putInt(TodoFragment.ARG_POSITION, position);
            newFragment.setArguments(args);
            FragmentTransaction transaction =
                    getSupportFragmentManager().beginTransaction();

            // Replace whatever is in the fragment_container view with this fragment,
            // and add the transaction to the back stack so the user can navigate back
            transaction.replace(R.id.fragment_container, newFragment);
            transaction.addToBackStack(null);

            // Commit the transaction
            transaction.commit();
        }
    }

}
```

And create the `TodoFragment` class with the following code:

```
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

public class TodoFragment extends Fragment {

    final static String ARG_POSITION = "position";
    int mCurrentPosition = -1;

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

        // If activity recreated (such as from screen rotate), restore
        // the previous todo selection set by onSaveInstanceState().
        // This is primarily necessary when in the two-pane layout.
        if (savedInstanceState != null) {
            mCurrentPosition = savedInstanceState.getInt(ARG_POSITION);
        }

        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.todo_view, container, false);
    }

    @Override
    public void onStart() {
        super.onStart();

        // During startup, check if there are arguments passed to the fragment.
```

```
        // onStart is a good place to do this because the layout has already been
        // applied to the fragment at this point so we can safely call the method
        // below that sets the todo text.
        Bundle args = getArguments();
        if (args != null) {
            // Set todo based on argument passed in
            updateTodoView(args.getInt(ARG_POSITION));
        } else if (mCurrentPosition != -1) {
            // Set todo based on saved instance state defined during onCreateView
            updateTodoView(mCurrentPosition);
        }
    }

    public void updateTodoView(int position) {
        TextView todo = (TextView) getActivity().findViewById(R.id.todo);
        todo.setText(TodoModel.Todos[position]);
        mCurrentPosition = position;
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);

        // Save the current todo selection in case we need to recreate the fragment
        outState.putInt(ARG_POSITION, mCurrentPosition);
    }

}
```

Save the view Framelayout for the list of todos in `res/layout/activity_main.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" />
```

And save the view for each todo item in `res/layout/todo_view.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
          android:id="@+id/todo"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:padding="16dp"
          android:textSize="18sp" />
```

The model is a simplified class with static string arrays; create a class `TodoModel` with the following content.

```
class TodoModel {

    static String[] Todos = {
            "Todo One",
            "Todo Two"
    };

    static String[] Todo = {
```

```
            "Todo One\n\nWake up!",
            "Todo Two\n\nGo to sleep!"
    };

}
```

`Run` the App

Reflection and QA

With reference to the code in this example, describe the code necessary to add a fragment to an Activity at run time.

Why is it a bad idea for fragments to pass parameters to each other?

How does defining an `interface` help a fragment to communicate with its parent activity?