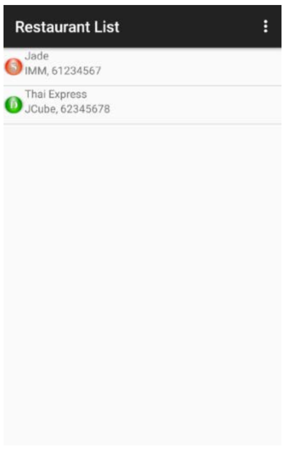
Practical 4: Activities & Preference

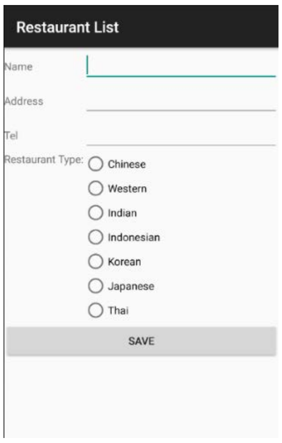
At the end of the session, you will learn to make use of Explicit Intent to activate 'Details' Form activity, edit & update the record saved in restaurants_table and setting preference sort order of the restaurant list displayed

Part I – From Single Activity to Two Activities

1. In **previous lab** exercises we have successfully split the **UI Views** into two ('List' & 'Details') using tabs. The controller for these UI views is still **using single activity class** (*RestaurantList.java* - **AppCompatActivity**)!
2. For better way of managing an Android app will be one UI View with one activity controller. Hence, in this part of the exercise, we will reorganize the **UI View with individual activity** controller and layout file:
   - 'List' – *RestaurantList.java* **ListActivity** and *main.xml* layout
   - 'Details' form – *DetailForm.jav a* **Activity** and *detail_form.xml* layout



ListActivity & main.xml
– 'List' (*RestaurantList.java*)

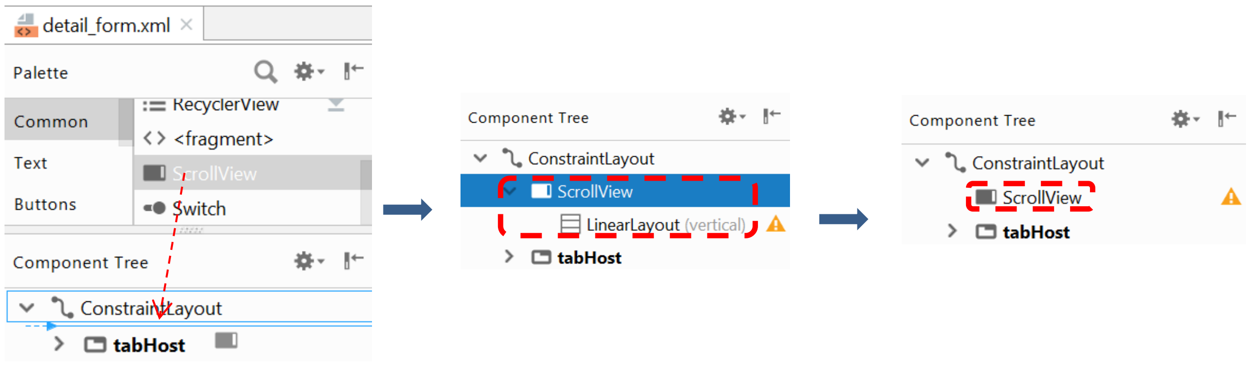


Activity & detail_form.xml
– 'Details' form (*DetailForm.java*)

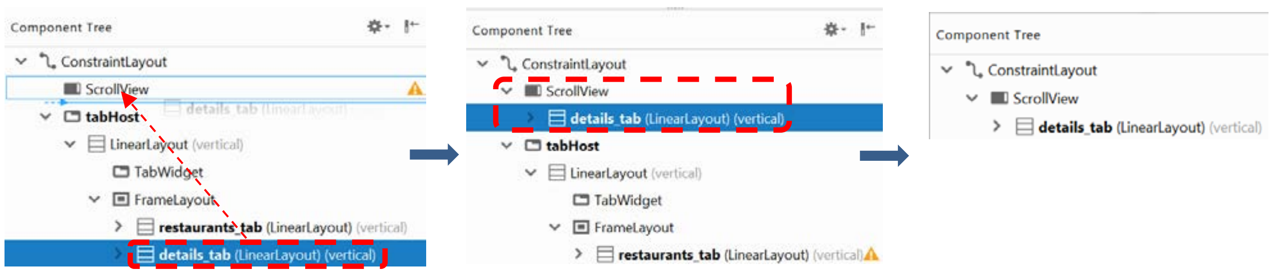
3. Create a new project with the following information:
   - **Application Name** : *Restaurant List*
   - **Company Domain** : *sp.com*
   - **Project Location** : *C:\MAD\AndroidStudioProjects\Lab4a or D:\MAD\AndroidStudioProjects\Lab4a*
   - **Minimum SDK** : *Android 5.0 (Lollipop)*
   - **Empty Activity name** : *RestaurantList*
   - **Layout name** : *main*
4. Close *RestaurantList.java* and *main.xml* files in the **Editor** pane
5. Open your **Windows File Explorer** and navigate to your **Android Studio** workspace where all you projects are created.
6. Copy *AndroidManifest.xml* file, **java** and **res** folders from **Lab3b\app\src\main** project folder and paste into **Lab4a\app\src\main** folder to overwrite the existing file and folder
7. At Windows Explorer, navigate to **Lab4a\app\src\main\res\layout** folder. Copy the *main.xml* file and paste into the same folder to create a duplicated copy
8. Right click on *main – Copy.xml* file and rename it to ***detail_form.xml***

Splitting UI views & Change MENU Item

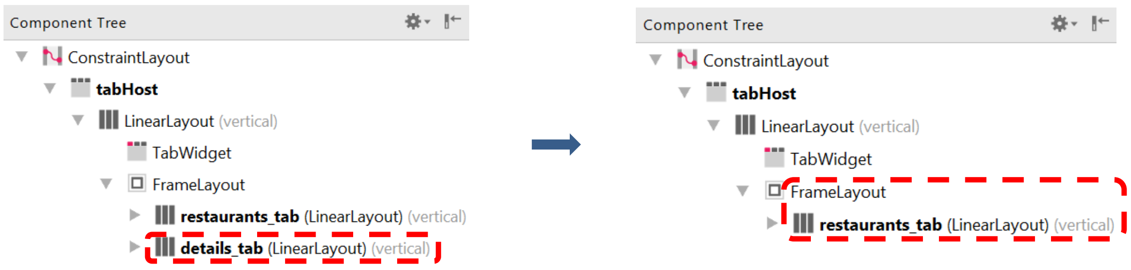
9. At Android Studio, open the *detail_form.xml* from the **res/layout** folder. At the **Design Editor > Palette** pane, drag a **ScrollView** widget and drop into the **ConstraintLayout** widget as a child node. Delete the LinearLayout (Vertical) in the **ScrollView** just added



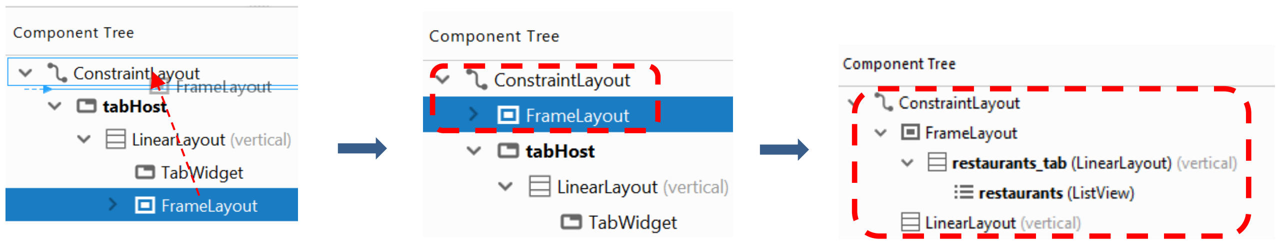
10. Drag the **details_tab (LinearLayout)** widget from **FrameLayout** and drop into **ScrollView** widget. Delete the **tabHost** widget from the **ConstraintLayout**child node



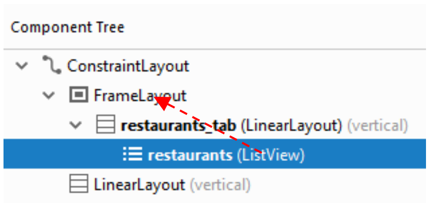
11. Open the *main.xml* layout file. Delete **details_tab (LinearLayout)** from **FrameLayout** child node

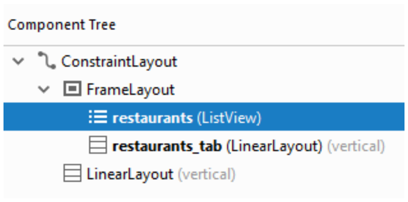


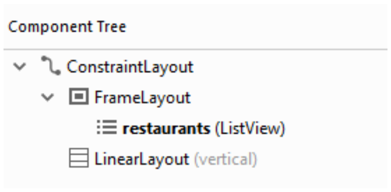
12. Drag and drop the **FrameLayout** widget into **ConstraintLayout** of *main.xml* file. Delete the whole **TabHost** tag



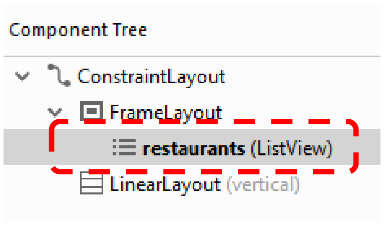
13. Drag and drop the **restaurants (ListView)** widget into the **FrameLayout** of *main.xml* file. Delete the **restaurants** _tab (LinearLayout) tag

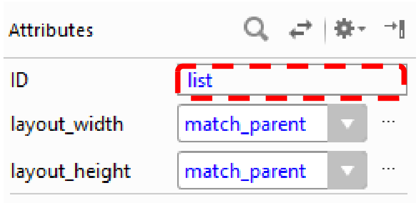

Component Tree
ConstraintLayout
FrameLayout
restaurants_tab (LinearLayout) (vertical)
restaurants (ListView)
LinearLayout (vertical)

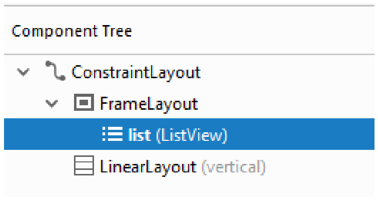

Component Tree
ConstraintLayout
FrameLayout
restaurants (ListView)
restaurants_tab (LinearLayout) (vertical)
LinearLayout (vertical)


Component Tree
ConstraintLayout
FrameLayout
restaurants (ListView)
LinearLayout (vertical)

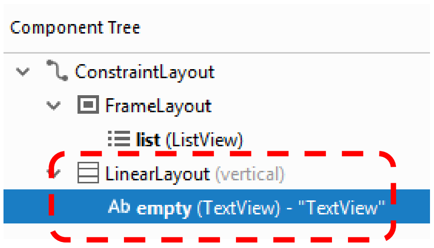
14. Change restaurants (ListView) widget ID to **list**.

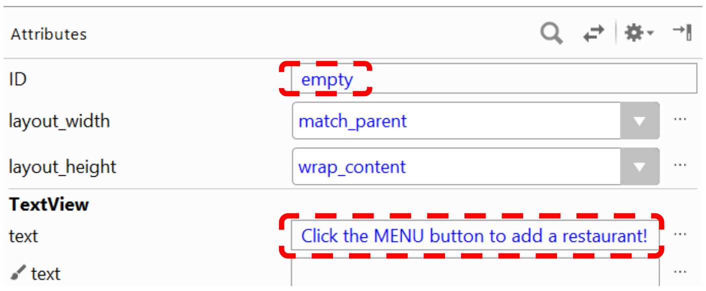

Component Tree
ConstraintLayout
FrameLayout
restaurants (ListView)
LinearLayout (vertical)


Attributes
ID: list
layout_width: match_parent
layout_height: match_parent

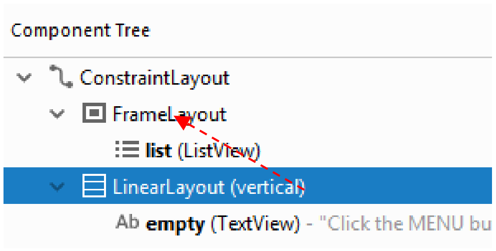

Component Tree
ConstraintLayout
FrameLayout
list (ListView)
LinearLayout (vertical)

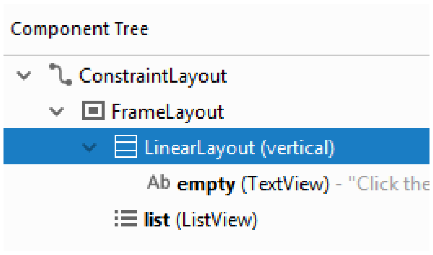
15. Drag a **TextView** widget from **Palette** pane and into the **LinearLayout** widget. Update the **TextView ID** to 'empty' and text as 'Click the MENU button to add a restaurant!'.


Component Tree
ConstraintLayout
FrameLayout
list (ListView)
LinearLayout (vertical)
Ab **empty** (TextView) - "TextView"

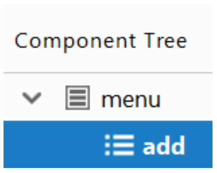

Attributes
ID: empty
layout_width: match_parent
layout_height: wrap_content
TextView
text: Click the MENU button to add a restaurant!
text

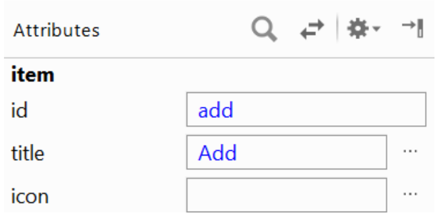
16. Drag the **LinearLayout (vertical)** child node and drop into the **FrameLayout**


Component Tree
ConstraintLayout
FrameLayout
list (ListView)
LinearLayout (vertical)
Ab **empty** (TextView) - "Click the MENU bu


Component Tree
ConstraintLayout
FrameLayout
LinearLayout (vertical)
Ab **empty** (TextView) - "Click the
list (ListView)

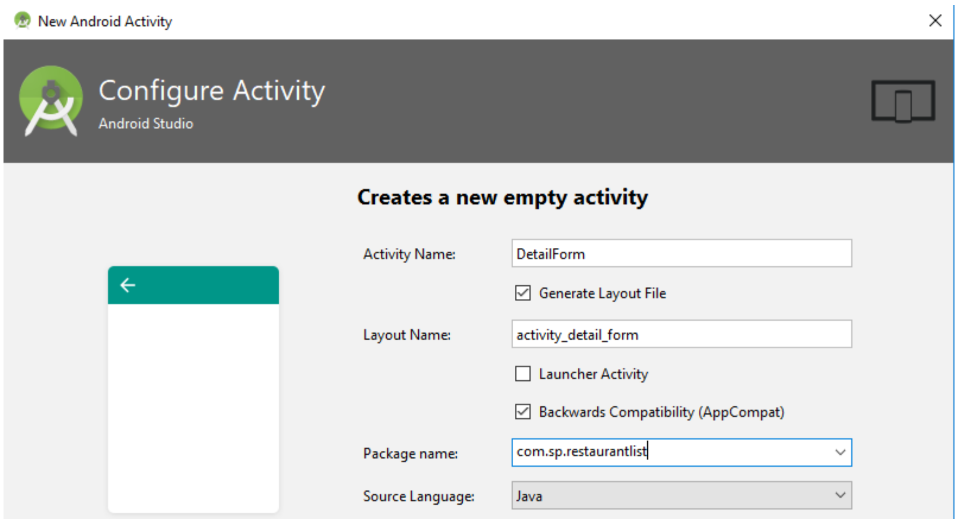
17. Open the *option.xml* file in the **res > menu** folder, change the option item **id** and **title** to "add" and "Add" respectively.


Component Tree
menu
add


Attributes
item
id: add
title: Add
icon:

Creating Separate Activities

18. With the ‘List’ UI view (*main.xml*) and ‘Details’ form UI view (*detail_form.xml*) created, we will create the necessary activity controller to link up individual UI view
19. Right click on the java/com.sp.restaurantlist folder, select New > Activity > Empty Activity to create DetailForm.java controller to manage the ‘Details’ form UI view



Creates a new empty activity

Activity Name: DetailForm
☑ Generate Layout File
Layout Name: activity_detail_form
☐ Launcher Activity
☑ Backwards Compatibility (AppCompat)
Package name: com.sp.restaurantlist
Source Language: Java

20. The function of *DetailForm.java* Activity is:
    1. to load the *detail_form.xml* layout as user’s UI view using the command `setContentView(R.layout.detail_form)` method
    2. to link up to widgets on UI view using command `findViewById` method
    3. to capture the data entries of each widgets using command `getText().toString()`method
    4. to create a new record in *restaurants_table* using the insert method provided by *RestaurantHelper* (*SQLiteOpenHelper* subclass)

21. Update the *DetailForm.java* **AppCompatActivity** file with the following content and save (Note: Most of the code can be transferred from *RestaurantList.java* file)

```
package com.sp.restaurantlist;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioGroup;

public class DetailForm extends AppCompatActivity {
    private EditText restaurantName;
    private EditText restaurantAddress;
    private EditText restaurantTel;
    private RadioGroup restaurantTypes;
    private Button buttonSave;

    private RestaurantHelper helper = null;

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

        restaurantName = (EditText) findViewById(R.id.restaurant_name);
        restaurantAddress = (EditText) findViewById(R.id.restaurant_address);
        restaurantTel = (EditText) findViewById(R.id.restaurant_tel);
        restaurantTypes = (RadioGroup) findViewById(R.id.restaurant_types);

        buttonSave = (Button) findViewById(R.id.button_save);
        buttonSave.setOnClickListener(onSave);
```

Cut from RestaurantList & paste here

Cut from RestaurantList & paste here

```java

        helper = new RestaurantHelper(this);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        helper.close();
    }

    private View.OnClickListener onSave = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // To read date from EditText
            String nameStr = restaurantName.getText().toString();
            String addrStr = restaurantAddress.getText().toString();
            String telStr = restaurantTel.getText().toString();
            String restType = "";
            //To read selection of restaurantTypes RadioGroup
            switch (restaurantTypes.getCheckedRadioButtonId()) {
                case R.id.chinese:
                    restType = "Chinese";
                    break;
                case R.id.western:
                    restType = "Western";
                    break;
                case R.id.indian:
                    restType = "Indian";
                    break;
                case R.id.indonesian:
                    restType = "Indonesian";
                    break;
                case R.id.korean:
                    restType = "Korean";
                    break;
                case R.id.japanese:
                    restType = "Japanese";
                    break;
                case R.id.thai:
                    restType = "Thai";
                    break;
            }

            //Create Restaurant Data Model
            helper.insert(nameStr, addrStr, telStr, restType);

            //To close current Activity class and exit
            finish();
        }
    };
}
```

Cut from
RestaurantList
& paste here

22. With the ‘Details’ form UI view controller done, we will look into the ‘List’ UI view controller (*RestaurantList.java* - **AppCompatActivity** subclass)

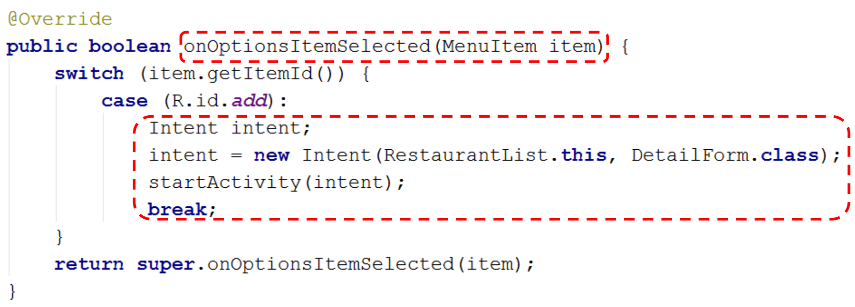
23. The function of *RestaurantList.java* is:
    1. to load the *main.xml* layout as user’s UI view
    2. to display record(s) saved in *restaurants_table* in *ListView* using *CursorAdapter*
    3. to provide an ‘Add’ MENU item to make an **Explicit Intent** call to launch *DetailForm.java* activity

24. Update the *RestaurantList.java* with the following content and save

```java
package com.sp.restaurantlist;

import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.app.Activity;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;

public class RestaurantList extends AppCompatActivity
{
    private Cursor model = null;
    private RestaurantAdapter adapter = null;
    private ListView list;
    private RestaurantHelper helper = null;
    private TextView empty = null;

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

        empty = (TextView) findViewById(R.id.empty);
        helper = new RestaurantHelper(this);
        list = (ListView) findViewById(R.id.list);
        model = helper.getAll();
        adapter = new RestaurantAdapter(this, model, 0);
        list.setAdapter(adapter);
    }

    @Override
    protected void onResume() {
        super.onResume();
        if (model != null) {
            model.close();
        }
        model = helper.getAll();

        if (model.getCount() > 0) {
            empty.setVisibility(View.INVISIBLE);
        }
        adapter.swapCursor(model);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        helper.close();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.option, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case (R.id.add):
                Intent intent;
                intent = new Intent(RestaurantList.this, DetailForm.class);
                startActivity(intent);
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    class RestaurantAdapter extends CursorAdapter {
        RestaurantAdapter(Cursor c) {
            super(RestaurantList.this, c);
        }

        @Override
        public void bindView(View row, Context ctxt, Cursor c) {
            RestaurantHolder holder = (RestaurantHolder) row.getTag();

            holder.populateFrom(c, helper);
        }

        @Override
        public View newView(Context ctxt, Cursor c, ViewGroup parent) {
            LayoutInflater inflater = getLayoutInflater();
            View row = inflater.inflate(R.layout.row, parent, false);
            RestaurantHolder holder = new RestaurantHolder(row);

            row.setTag(holder);
            return (row);
        }
    }

    static class RestaurantHolder {
        private TextView restName = null;
        private TextView addr = null;
        private ImageView icon = null;

        RestaurantHolder(View row) {
            restName = (TextView) row.findViewById(R.id.restName);
            addr = (TextView) row.findViewById(R.id.restAddr);
            icon = (ImageView) row.findViewById(R.id.icon);
        }

        void populateFrom(Cursor c, RestaurantHelper helper) {
            restName.setText(helper.getRestaurantName(c));
            String temp = helper.getRestaurantAddress(c) + ", " + helper.getRestaurantTel(c);
            addr.setText(temp);

            if (helper.getRestaurantType(c).equals("Chinese")) {
                icon.setImageResource(R.drawable.ball_red);
            } else if (helper.getRestaurantType(c).equals("Western")) {

                icon.setImageResource(R.drawable.ball_yellow);
            } else {
                icon.setImageResource(R.drawable.ball_green);
            }
        }
    }
}
```

25. When MENU button is pressed at the ‘List’ UI view, a menu with ‘**Add**’ item will pop-up.

26. If the '**Add**' menu item is selected, the item selected event generated will be captured by the `onOptionsItemSelected(MenuItem item)` method. The item.getItemId() method will read in the item which the user has clicked on the phone screen. In this case, it is "Add". After comparing the **id** returned, if it is R.id.add, an **Explicit Intent** call will be used to start (`startActivity`) a new user interface (UI) - the *DetailForm.java* Activity



```java
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case (R.id.add):
            Intent intent;
            intent = new Intent(RestaurantList.this, DetailForm.class);
            startActivity(intent);
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

27. ***IMPORTANT – new Activity (DetailForm.java) has been added**
In order for the system to recognize a new added **Activity**, the *DetailForm.java* **Activity** need to be recorded in the *AndroidManifest.xml* file, otherwise, an error will occur whenever the 'Add' menu item is selected during runtime. Check that line 19 is added to the *AndroidManifest.xml* file. If not add the line and save.

```xml
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3.      package="com.sp.restaurantlist">
4.
5.      <application
6.          android:allowBackup="true"
7.          android:icon="@mipmap/ic_launcher"
8.          android:label="@string/app_name"
9.          android:roundIcon="@mipmap/ic_launcher_round"
10.         android:supportsRtl="true"
11.         android:theme="@style/AppTheme">
12.         <activity android:name=".RestaurantList">
13.             <intent-filter>
14.                 <action android:name="android.intent.action.MAIN" />
15.
16.                 <category android:name="android.intent.category.LAUNCHER" />
17.             </intent-filter>
18.         </activity>
19.         <activity android:name=".DetailForm"></activity>
20.     </application>
21.
22. </manifest>
```

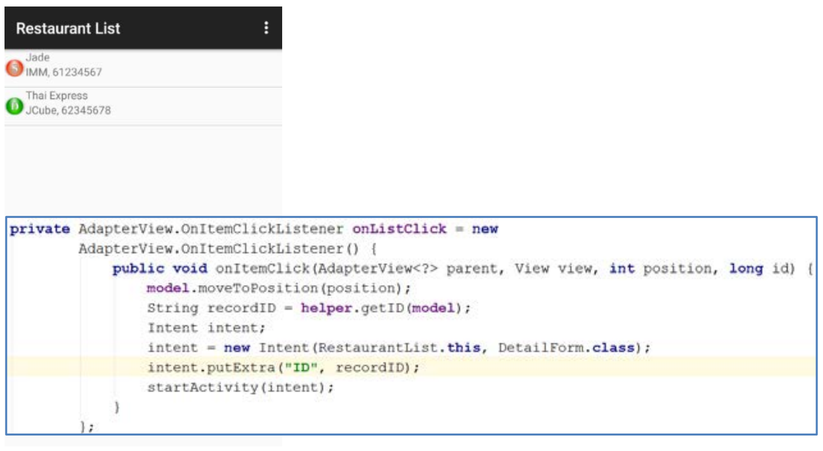
28. Run the *Lab4a* project. Test the persistence data storage by adding new restaurant to the list and save.
29. Show the result to your lecturer when you have completed this section successfully.
30. Notice that the UI view will be switched to 'List' view automatically after the 'Save' button is pressed. Which command in the program causes such effect? Record down the Line number and the command from your program

Java Program Name: ________________ Line No.: ______ Command: __________________

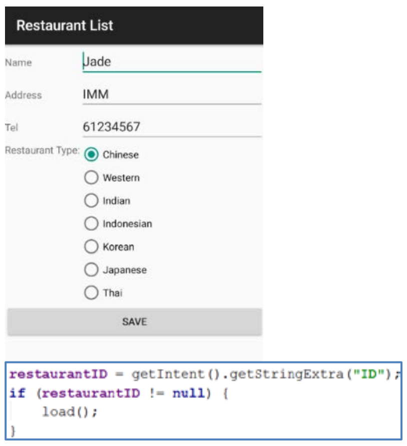
Lecturer Signature : ____

Part II – Passing values between Activities using Intent putExtra

31. We have successfully split the UI views and controllers with separate layout files and activities. User can save restaurant record to *restaurants_table* and update the 'List'.
32. What about reading a record from the *restaurants_table* when a 'List' item is selected and display the result in the 'Details' form?
33. Looking into the process flow, the UI view will switch from 'List' to 'Details' form when a list item is selected i.e. there is a change in activity too. For the 'Details' form to know which list item has been selected, the **'List' controller passes the** `"ID"` as string data **to 'Details' form controller** using putExtra method. The *DetailForm* controller will use `getIntent().getStringExtra("ID")` method to read the value passed over



```
private AdapterView.OnItemClickListener onListClick = new
        AdapterView.OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                model.moveToPosition(position);
                String recordID = helper.getID(model);
                Intent intent;
                intent = new Intent(RestaurantList.this, DetailForm.class);
                intent.putExtra("ID", recordID);
                startActivity(intent);
            }
        };
```



```
restaurantID = getIntent().getStringExtra("ID");
if (restaurantID != null) {
    load();
}
```

Activity – *RestaurantList.java*

Activity – *DetailForm.java*

34. Edit the **RestaurantHelper** model to allow user to get record by "ID" and update the existing restaurant data

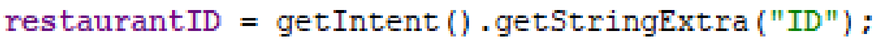
```
package com.sp.restaurantlist;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class RestaurantHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "restaurantlist.db";
    private static final int SCHEMA_VERSION = 1;

    public RestaurantHelper(Context context) {
        super(context, DATABASE_NAME, null, SCHEMA_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // Will be called once when the database is not created
        db.execSQL("CREATE TABLE restaurants_table ( _id INTEGER PRIMARY KEY AUTOINCREMENT, restaurantName TEXT, restaurantAddress TEXT, restaurantTel TEXT, restaurantType TEXT);");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Will not be called until SCHEMA_VERSION increases
        // Here we can upgrade the database e.g. add more tables
    }

    /* Read all records from restaurants_table */
    public Cursor getAll() {
        return (getReadableDatabase().rawQuery(
                "SELECT _id, restaurantName, restaurantAddress, restaurantTel, restaurantType FROM restaurants_table ORDER BY restaurantName", null));
    }

    /* Read a particular record from restaurants_table with id provided */
    public Cursor getById(String id) {
        String[] args = {id};
        return (getReadableDatabase().rawQuery(
                "SELECT _id, restaurantName, restaurantAddress, restaurantTel, restaurantType FROM restaurants_table WHERE _ID=?", args));
    }


    /* Write a record into restaurants_table */
    public void insert(String restaurantName, String restaurantAddress, String restaurantTel, String restaurantType) {
        ContentValues cv = new ContentValues();

        cv.put("restaurantName", restaurantName);
        cv.put("restaurantAddress", restaurantAddress);
        cv.put("restaurantTel", restaurantTel);
        cv.put("restaurantType", restaurantType);
        getWritableDatabase().insert("restaurants_table", "restaurantName", cv);
    }

    /* Update a particular record in restaurants_table with id provided */
    public void update(String id, String restaurantName, String restaurantAddress,
                       String restaurantTel, String restaurantType) {
        ContentValues cv = new ContentValues();
        String[] args = {id};
        cv.put("restaurantName", restaurantName);
        cv.put("restaurantAddress", restaurantAddress);
        cv.put("restaurantTel", restaurantTel);
        cv.put("restaurantType", restaurantType);
        getWritableDatabase().update("restaurants_table", cv, "_ID=?", args);
    }

    /* Read a record id value from restaurants_table */
    public String getID(Cursor c) {
        return (c.getString(0));
    }

    public String getRestaurantName(Cursor c) {
        return (c.getString(1));
    }

    public String getRestaurantAddress(Cursor c) {
        return (c.getString(2));
    }

    public String getRestaurantTel(Cursor c) {
        return (c.getString(3));
    }

    public String getRestaurantType(Cursor c) {
        return (c.getString(4));
    }

}
```

35. Open the *RestaurantList.java* file to add in the `onListItemClick(ListView list, View view, int position, long id)` method to capture the list item click event and start the **Explicit Intent** call to switch to *DetailForm.java* Activity with 'ID' passing over

```
package com.sp.restaurantlist;

import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.Menu;
```

```java
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.CursorAdapter;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;


public class RestaurantList extends AppCompatActivity {
    private Cursor model = null;
    private RestaurantAdapter adapter = null;
    private ListView list;
    private RestaurantHelper helper = null;
    private TextView empty = null;

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

        empty = (TextView) findViewById(R.id.empty);
        helper = new RestaurantHelper(this);
        list = (ListView) findViewById(R.id.list);
        model = helper.getAll();
        adapter = new RestaurantAdapter(this, model, 0);
        list.setOnItemClickListener(onListClick);
        list.setAdapter(adapter);
    }

    @Override
    protected void onResume() {
        super.onResume();
        if (model != null) {
            model.close();
        }
        model = helper.getAll();
        if (model.getCount() > 0) {
            empty.setVisibility(View.INVISIBLE);
        }
        adapter.swapCursor(model);
    }

    @Override
    protected void onDestroy() {
        helper.close();
        super.onDestroy();
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.option, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case (R.id.add):
                Intent intent;
                intent = new Intent(RestaurantList.this, DetailForm.class);
                startActivity(intent);
                break;
        }
        return super.onOptionsItemSelected(item);

    }

    private AdapterView.OnItemClickListener onListClick = new AdapterView.OnItemClickListener() {
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            model.moveToPosition(position);
            String recordID = helper.getID(model);
            Intent intent;
            intent = new Intent(RestaurantList.this, DetailForm.class);
            intent.putExtra("ID", recordID);
            startActivity(intent);
        }
    };


    static class RestaurantHolder {
        private TextView restName = null;
        private TextView addr = null;
        private ImageView icon = null;

        RestaurantHolder(View row) {
            restName = (TextView) row.findViewById(R.id.restName);
            addr = (TextView) row.findViewById(R.id.restAddr);
            icon = (ImageView) row.findViewById(R.id.icon);
        }

        void populateFrom(Cursor c, RestaurantHelper helper) {
            restName.setText(helper.getRestaurantName(c));
            String temp = helper.getRestaurantAddress(c) + ", " + helper.getRestaurantTel(c);
            addr.setText(temp);

            if (helper.getRestaurantType(c).equals("Chinese")) {
                icon.setImageResource(R.drawable.ball_red);
            } else if (helper.getRestaurantType(c).equals("Western")) {
                icon.setImageResource(R.drawable.ball_yellow);
            } else {
                icon.setImageResource(R.drawable.ball_green);
            }
        }
    }

    class RestaurantAdapter extends CursorAdapter {
        RestaurantAdapter(Context context, Cursor cursor, int flags) {
            super(context, cursor, flags);
        }

        @Override
        public void bindView(View view, Context context, Cursor cursor) {
            RestaurantHolder holder = (RestaurantHolder) view.getTag();
            holder.populateFrom(cursor, helper);
        }

        @Override
        public View newView(Context context, Cursor cursor, ViewGroup parent) {
            LayoutInflater inflater = getLayoutInflater();
            View row = inflater.inflate(R.layout.row, parent, false);
            RestaurantHolder holder = new RestaurantHolder(row);
            row.setTag(holder);
            return (row);
        }
    }
}
```

36. When the controller is switched to *DetailForm.java* Activity, the value of the record 'ID' that is passed over will be read using the **getIntent** method and save to local variable ***restaurantID***

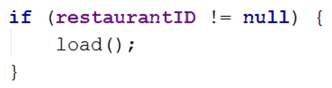


```
restaurantID = getIntent().getStringExtra("ID");
```

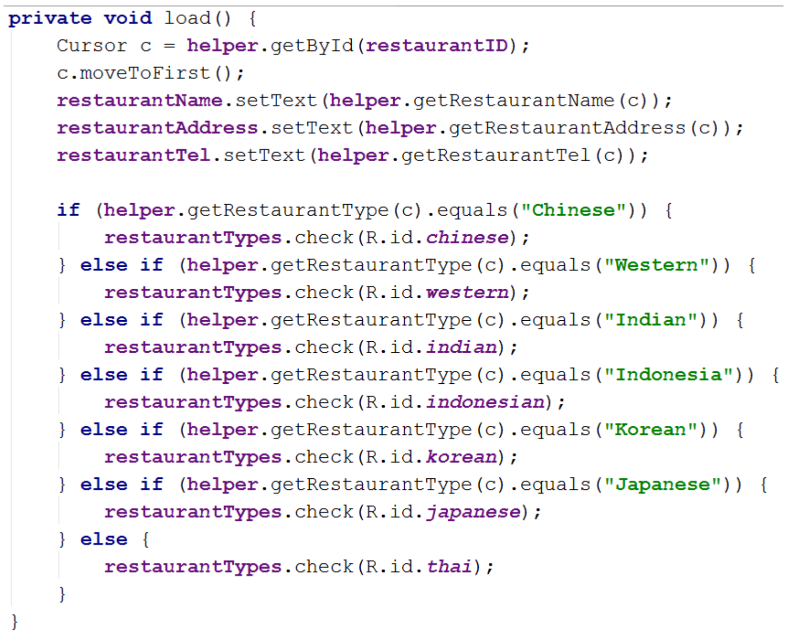
37. How does the *DetailForm.java* Activity differentiate if it is supposed
    - to **read a record** from the *restaurants_table* **and display** in the 'Details' form or
    - to **provide an empty 'Details' form** for user to **add a new restaurant record**?

Answer: ____

38. If the intension is to add a new record, the value of "*restaurantID*" will be null. Otherwise, the record will be retrieved through the load() method call



```
if (restaurantID != null) {
    load();
}
```



```
private void load() {
    Cursor c = helper.getById(restaurantID);
    c.moveToFirst();
    restaurantName.setText(helper.getRestaurantName(c));
    restaurantAddress.setText(helper.getRestaurantAddress(c));
    restaurantTel.setText(helper.getRestaurantTel(c));

    if (helper.getRestaurantType(c).equals("Chinese")) {
        restaurantTypes.check(R.id.chinese);
    } else if (helper.getRestaurantType(c).equals("Western")) {
        restaurantTypes.check(R.id.western);
    } else if (helper.getRestaurantType(c).equals("Indian")) {
        restaurantTypes.check(R.id.indian);
    } else if (helper.getRestaurantType(c).equals("Indonesia")) {
        restaurantTypes.check(R.id.indonesian);
    } else if (helper.getRestaurantType(c).equals("Korean")) {
        restaurantTypes.check(R.id.korean);
    } else if (helper.getRestaurantType(c).equals("Japanese")) {
        restaurantTypes.check(R.id.japanese);
    } else {
        restaurantTypes.check(R.id.thai);
    }
}
```

39. When the restaurant record has been read and displayed in 'Details' form, what will happen if user clicks on the "Save" button?
    Answer: ____

40. To prevent adding the same restaurant record to the *restaurants_table* in *restaurants* SQLite database we can make use of the same checking method as **Step 37** i.e. if the "restaurantID" is **null**, **add new record** to *restaurants_table*. **Otherwise**, **update** the **existing record** in the *restaurants_table*

41. Update the *detail_form.java* file with the following content and save

```
package com.sp.restaurantlist;

import android.database.Cursor;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioGroup;

public class DetailForm extends AppCompatActivity {
    private EditText restaurantName;
    private EditText restaurantAddress;
    private EditText restaurantTel;
    private RadioGroup restaurantTypes;
    private Button buttonSave;

    private RestaurantHelper helper = null;
    private String restaurantID = "";


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.detail_form);
        restaurantName = (EditText) findViewById(R.id.restaurant_name);
        restaurantAddress = (EditText) findViewById(R.id.restaurant_address);
        restaurantTel = (EditText) findViewById(R.id.restaurant_tel);
        restaurantTypes = (RadioGroup) findViewById(R.id.restaurant_types);
```

```java

        buttonSave = (Button) findViewById(R.id.button_save);
        buttonSave.setOnClickListener(onSave);

        helper = new RestaurantHelper(this);

        restaurantID = getIntent().getStringExtra("ID");
        if (restaurantID != null) {
            load();
        }

    }
    private void load() {
        Cursor c = helper.getById(restaurantID);
        c.moveToFirst();
        restaurantName.setText(helper.getRestaurantName(c));
        restaurantAddress.setText(helper.getRestaurantAddress(c));
        restaurantTel.setText(helper.getRestaurantTel(c));

        if (helper.getRestaurantType(c).equals("Chinese")) {
            restaurantTypes.check(R.id.chinese);
        } else if (helper.getRestaurantType(c).equals("Western")) {
            restaurantTypes.check(R.id.western);
        } else if (helper.getRestaurantType(c).equals("Indian")) {
            restaurantTypes.check(R.id.indian);
        } else if (helper.getRestaurantType(c).equals("Indonesian")) {
            restaurantTypes.check(R.id.indonesian);
        } else if (helper.getRestaurantType(c).equals("Korean")) {
            restaurantTypes.check(R.id.korean);
        } else if (helper.getRestaurantType(c).equals("Japanese")) {
            restaurantTypes.check(R.id.japanese);
        } else {
            restaurantTypes.check(R.id.thai);
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        helper.close();
    }

    View.OnClickListener onSave = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // To read date from EditText
            String nameStr = restaurantName.getText().toString();
            String addrStr = restaurantAddress.getText().toString();
            String telStr = restaurantTel.getText().toString();
            String restType = "";
            //To read selection of restaurantTypes RadioGroup
            switch (restaurantTypes.getCheckedRadioButtonId()) {
                case R.id.chinese:
                    restType = "Chinese";
                    break;
                case R.id.western:
                    restType = "Western";
                    break;
                case R.id.indian:
                    restType = "Indian";
                    break;
                case R.id.indonesian:
                    restType = "Indonesian";
                    break;
                case R.id.korean:
                    restType = "Korean";
                    break;
                case R.id.japanese:
                    restType = "Japanese";
                    break;
                case R.id.thai:
                    restType = "Thai";
```

```
                        break;
                    }

                    if (restaurantID == null) {
                        helper.insert(nameStr, addrStr, telStr, restType);
                    } else {
                        helper.update(restaurantID, nameStr, addrStr, telStr, restType);
                    }
                    //To close current Activity class and exit
                    finish();
                }
            };
        }
```

42. Run the *Lab4a* project. Test reading a record from *restaurants_table*. Change some data when record is loaded in the 'Details' form and "save". No new record should be added to the 'List'
43. Show the result to your lecturer when completed this section successfully

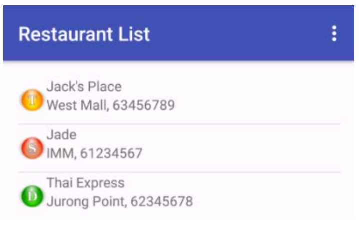
Lecturer Signature : ____

Part II - Setting Preference

44. In this part of the exercise, we will further enhance the Restaurant List app by introduction sorting function for the record(s) in 'List' UI view
45. Create a new project with the following information:
    - **Application Name** : *Restaurant List*
    - **Company Domain** : *sp.com*
    - **Project Location** : *C:\MAD\AndroidStudioProjects\Lab4b or D:\MAD\AndroidStudioProjects\Lab4b*
    - **Minimum SDK** : *Android 5.0 (Lollipop)*
    - **Empty Activity name** : *RestaurantList*
    - **Layout name** : *main*

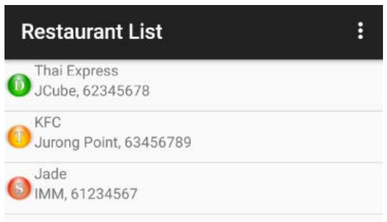
46. Close *RestaurantList.java* and *main.xml* files in the **Editor** pane
47. Open your **Windows File Explorer** and navigate to your **Android Studio** workspace where all you projects are created.
48. Copy *AndroidManifest.xml* file, **java** and **res** folders from **Lab4a\app\src\main** project folder and paste into **Lab4b\app\src\main** folder to overwrite the existing file and folder
49. At this stage of time, record(s) saved is(are) being read from the *restaurants_table* through the getAll method provided in *RestaurantHelper.java* (**SQLiteOpenHelper** subclass). The SQL command "*SELECT _id, restaurantName, restaurantAddress, restaurantTel, restaurantType FROM restaurants_table ORDER BY restaurantName*" is to read all record(s) with the specified fields (_id, restaurantName, restaurantAddress, restaurantTel and restaurantType) from the *restaurants_table* and sort by "restaurantName" field in ascending order

```
/* Read all records from restaurants_table */
public Cursor getAll() {
    return (getReadableDatabase().rawQuery(
            "SELECT _id, restaurantName, restaurantAddress, restaurantTel, restaurantType " +
                    "FROM restaurants_table ORDER BY restaurantName", null));
}
```


Restaurant List
Jack's Place
West Mall, 63456789
Jade
IMM, 61234567
Thai Express
Jurong Point, 62345678

50. We will add a sorting feature that will allows user to retrieve and display the record(s) from the *restaurants_table* in the 'List' UI view, according to the "sorting order" preference setting. The getALL method in *RestaurantHelper.java* file need to be modified to receive the "**orderBy**" string from the "sorting order" preference setting . For instance, to sort the 'name' field in descending order the "orderBy" string received will be "restaurantName DESC" and the completed SQL command constructed will be "*SELECT _id, restaurantName, restaurantAddress, restaurantTel, restaurantType FROM restaurants_table ORDER BY* ***restaurantName DESC***"

```
public Cursor getAll(String orderBy) {
    return (getReadableDatabase().rawQuery(
            "SELECT _id, name, address, telephone, restaurantType "
                    + "FROM restaurants_table ORDER BY " + orderBy, null));
}
```


Restaurant List
Thai Express
JCube, 62345678
KFC
Jurong Point, 63456789
Jade
IMM, 61234567

"orderBy" string:

- `restaurantName ASC` - sort 'restaurantName' field in ascending order
- `restaurantName DESC` - sort 'restaurantName' filed in descending order (**see above**)
- `restaurantType ASC` - sort 'restaurantType' field in ascending order
- `restaurantAddress ASC` - sort 'restaurantAddress' field in ascending order
- `restaurantAddress DESC` - sort 'restaurantAddress' field in descending order

51. Open the *RestaurantHelper.java* file. Update the existing getAll method with the following content and save

```java
public Cursor getAll(String orderBy) {
        return (getReadableDatabase().rawQuery(
                "SELECT _id, restaurantName, restaurantAddress,
                restaurantTel, restaurantType FROM
                restaurants_table ORDER BY " + orderBy, null));
    }
```

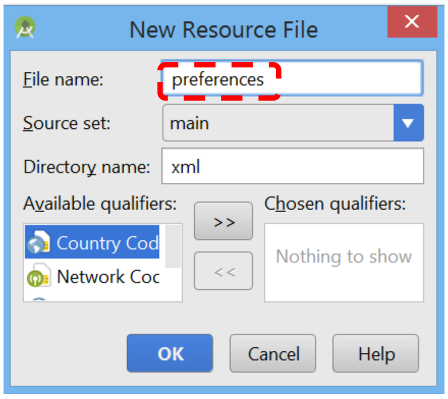
52. Right click on the **res > values** folder and select **New > XML > Values XML File.** Under **Values File Name** enters *arrays*. This file will provide sorting options for user to select as well as the '**orderBy**' string for passing to the getAll method.
53. Open the *arrays.xml* file to update with the following content and save

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="sort_names">
        <item>By Restaurant Name, Ascending</item>
        <item>By Restaurant Name, Descending</item>
        <item>By Restaurant Type</item>
        <item>By Restaurant Address, Ascending</item>
        <item>By Restaurant Address, Descending</item>
    </string-array>
    <string-array name="sort_clauses">
        <item>restaurantName ASC</item>
        <item>restaurantName DESC</item>
        <item>restaurantType ASC</item>
        <item>restaurantAddress ASC</item>
        <item>restaurantAddress DESC</item>
    </string-array>
</resources>
```

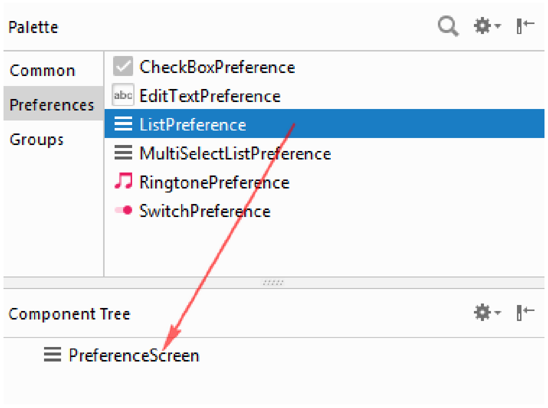
'orderBy' string

54. When user has selected the preference sorting format, we will need to save the choice into the phone. For phone Data Storage, we have explored SQLite database to store restaurants records. We will explore another way of local phone storage called ***SharedPreferences***. Shared Preferences store small amount of data and it is private to your application. The *SharedPreferences* class provides a general framework that allows you to save and retrieve persistent key-value pairs of primitive data types. You can use Shared Preferences to save any primitive data: booleans, floats, ints, longs, and strings. This data will persist across user sessions (even if your application is killed)
55. We will create a preference setting view in the **res/xml** folder.
56. Right click on the **res** folder and select **New > Directory**. Enter the directory name as "**xml**"

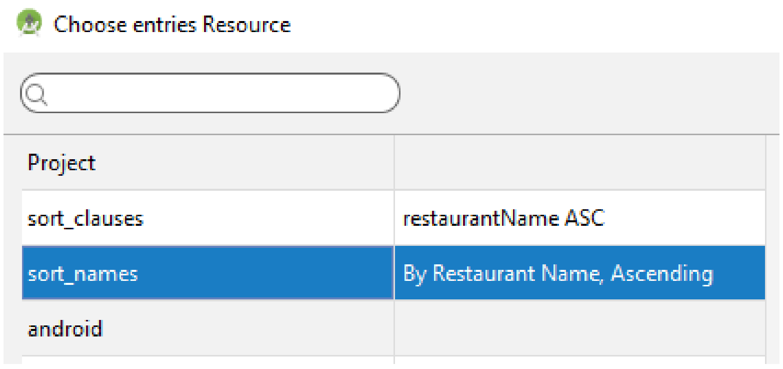
57. Right click on the **res > xml** folder and select **New > XML resources file.** Type in File name "*preferences*"



58. Open the *preferences.xml* file and drag the "ListPreference" widget onto "PreferenceScreen".

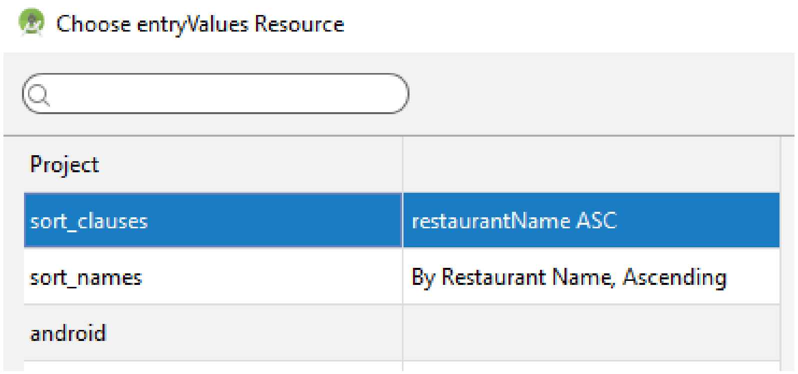


59. Click "sort_names" and click "OK".



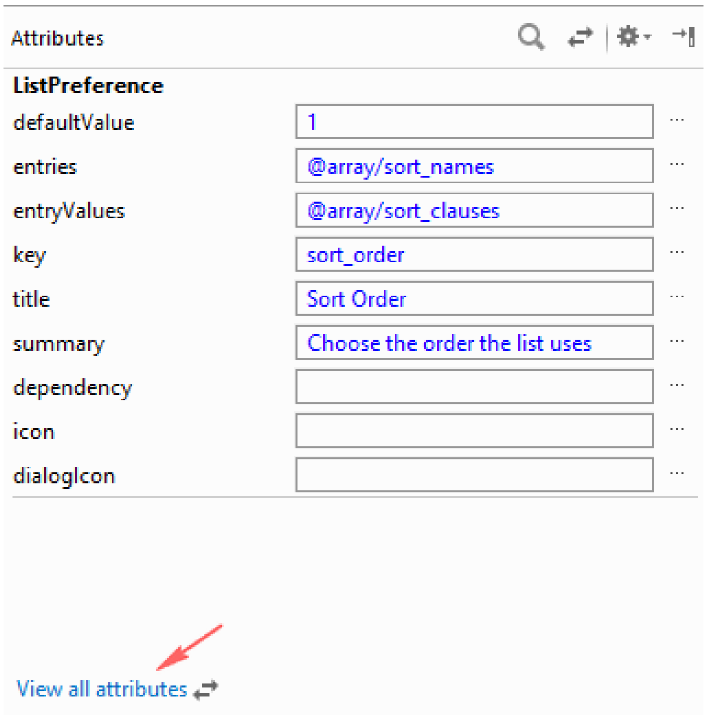
Project	
sort_clauses	restaurantName ASC
sort_names	By Restaurant Name, Ascending
android	

60. Click "sort_clauses" and click "OK"



Project	
sort_clauses	restaurantName ASC
sort_names	By Restaurant Name, Ascending
android	

61. Edit the "ListPreference" attributes as below.

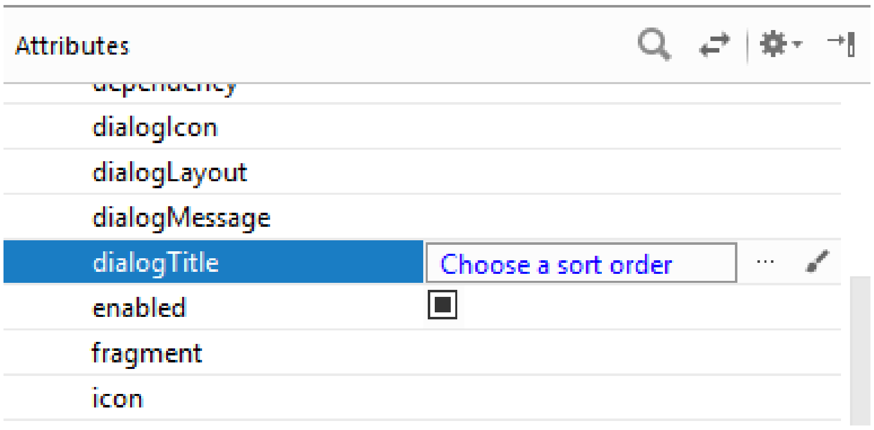


Attributes

ListPreference

defaultValue	1
entries	@array/sort_names
entryValues	@array/sort_clauses
key	sort_order
title	Sort Order
summary	Choose the order the list uses
dependency	
icon	
dialogIcon	

62. Click "View all attributes" to display all attributes. Edit "dialogTitle" attribute as below.

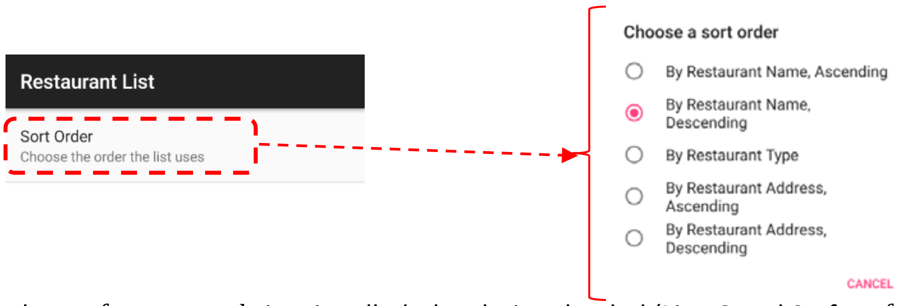


Attributes

dialogIcon	
dialogLayout	
dialogMessage	
dialogTitle	Choose a sort order
enabled	■
fragment	
icon	

63. Click to "Text" mode and check that the content is as below

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <ListPreference
        android:defaultValue="1"
        android:dialogTitle="Choose a sort order"
        android:entries="@array/sort_names"
        android:entryValues="@array/sort_clauses"
        android:key="sort_order"
        android:summary="Choose the order the list uses"
        android:title="Sort Order" />
</PreferenceScreen>
```


Restaurant List

Sort Order
Choose the order the list uses

Choose a sort order

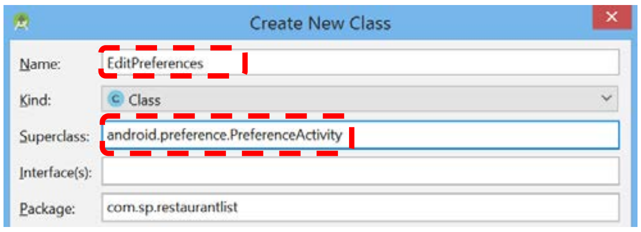
- By Restaurant Name, Ascending
- By Restaurant Name, Descending
- By Restaurant Type
- By Restaurant Address, Ascending
- By Restaurant Address, Descending

64. When the *preferences.xml* view is called, the choices loaded (Line 8 and 9 of *preferences.xml* file) will be gotten from the *array.xml* created in STEP 52. The "orderBy" value will be stored in "sort_order" key (Line 10 of *preferences.xml* file) as *SharedPreferences* data

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <PreferenceScreen xmlns:tools="http://schemas.android.com/tools"
3.      xmlns:android="http://schemas.android.com/apk/res/android">
4.
5.      <ListPreference
6.          android:defaultValue="1"
7.          android:dialogTitle="Choose a sort order"
8.          android:entries="@array/sort_names"
9.          android:entryValues="@array/sort_clauses"
10.         android:key="sort_order"
11.         android:summary="Choose the order the list uses"
12.         android:title="Sort Order" />
13. </PreferenceScreen>
```

65. To handle setting up the preference setting view, a **PreferenceActivity** controller (*EditPreferences.java*) will be created

66. Right click on the **java/com.sp.restaurantlist** folder and create a new Java Class named *EditPreferences.java* (**PreferenceActivity** subclass)

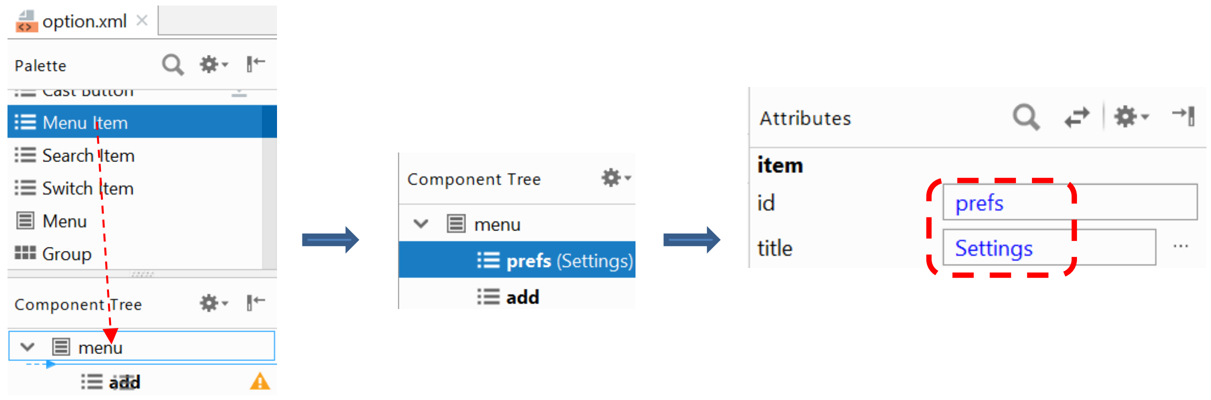

Create New Class

Name: EditPreferences
Kind: Class
Superclass: android.preference.PreferenceActivity
Interface(s):
Package: com.sp.restaurantlist

67. Update with the following content and save

```
package com.sp.restaurantlist;

import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.preference.PreferenceFragment;

public class EditPreferences extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getFragmentManager().beginTransaction().replace(android.R.id.content, new
        MyPreferenceFragment()).commit();
    }

    public static class MyPreferenceFragment extends PreferenceFragment {
        @Override
        public void onCreate(final Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            addPreferencesFromResource(R.xml.preferences);
        }
    }
}
```

68. For the next steps, we will add an extra MENU item “Settings” to allow user to bring out the preference setting view
69. Open the *option.xml* file from **res/menu** folder. Drag a **Menu Item** widget and drop into **menu** of **Component Tree**. Update the new item id as ‘**prefs**’ and title as ‘**Settings**’

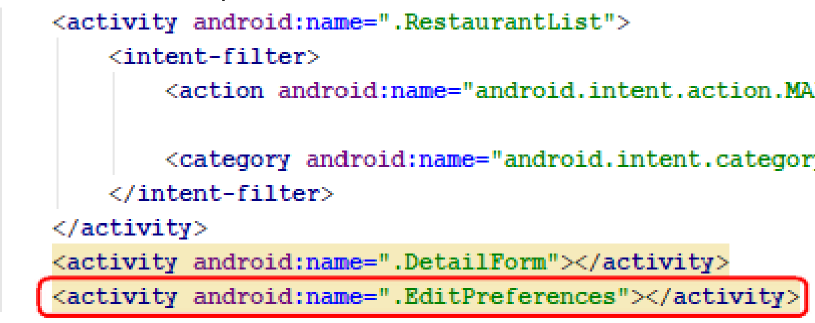


id	prefs
title	Settings

70. With the UI view and PreferenceActivity (*EditPreferences.java*) done, we are ready to link them up and take necessary action to update view for any change in selection of sorting order
71. **In case you have “R.id” errors, select “Build -> Clean Project”**.
72. Edit the *RestaurentList.java* file to incorporate the preference option and save

```
package com.sp.restaurantlist;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.CursorAdapter;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;

public class RestaurantList extends AppCompatActivity {
    private Cursor model = null;
    private RestaurantAdapter adapter = null;
    private ListView list;
    private RestaurantHelper helper = null;
    private TextView empty = null;
    private SharedPreferences prefs = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        empty = (TextView) findViewById(R.id.empty);
        helper = new RestaurantHelper(this);
        list = (ListView) findViewById(R.id.list);

        adapter = new RestaurantAdapter(this, model, 0);
        list.setOnItemClickListener(onListClick);
        list.setAdapter(adapter);
        prefs = PreferenceManager.getDefaultSharedPreferences(this);
        prefs.registerOnSharedPreferenceChangeListener(prefListener);
    }

    private void initList() {
        if (model != null) {
            model.close();
        }
        model = helper.getAll(prefs.getString("sort_order", "restaurantName"));
        adapter.swapCursor(model);
    }

    private SharedPreferences.OnSharedPreferenceChangeListener prefListener = new SharedPreferences.OnSharedPreferenceChangeListener() {
        public void onSharedPreferenceChanged(SharedPreferences sharedPrefs,String key) {
            if (key.equals("sort_order")) {
                initList();
            }
        }
    };

    @Override
    protected void onResume() {
        super.onResume();
        initList();
        if (model.getCount() > 0) {
            empty.setVisibility(View.INVISIBLE);
        }
    }

    @Override
    protected void onDestroy() {
        helper.close();
        super.onDestroy();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.option, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case (R.id.add):
                Intent intent;
                intent = new Intent(RestaurantList.this, DetailForm.class);
                startActivity(intent);
                break;
            case (R.id.prefs):
                startActivity(new Intent(this, EditPreferences.class));
                break;
        }
        return super.onOptionsItemSelected(item);

    }

private AdapterView.OnItemClickListener onListClick = new AdapterView.OnItemClickListener() {
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            model.moveToPosition(position);
            String recordID = helper.getID(model);
                  Intent intent;
                  intent = new Intent(RestaurantList.this, DetailForm.class);
                  intent.putExtra("ID", recordID);
                  startActivity(intent);
              }
          };

          static class RestaurantHolder {
              private TextView restName = null;
              private TextView addr = null;
              private ImageView icon = null;

              RestaurantHolder(View row) {
                  restName = (TextView) row.findViewById(R.id.restName);
                  addr = (TextView) row.findViewById(R.id.restAddr);
                  icon = (ImageView) row.findViewById(R.id.icon);
              }

              void populateFrom(Cursor c, RestaurantHelper helper) {
                  restName.setText(helper.getRestaurantName(c));
                  String temp = helper.getRestaurantAddress(c) + ", " +
helper.getRestaurantTel(c);
                  addr.setText(temp);
                  if (helper.getRestaurantType(c).equals("Chinese")) {
                      icon.setImageResource(R.drawable.ball_red);
                  } else if (helper.getRestaurantType(c).equals("Western")) {
                      icon.setImageResource(R.drawable.ball_yellow);
                  } else {
                      icon.setImageResource(R.drawable.ball_green);
                  }
              }
          }

          class RestaurantAdapter extends CursorAdapter {
              RestaurantAdapter(Context context, Cursor cursor, int flags) {
                  super(context, cursor, flags);
              }

              @Override
              public void bindView(View view, Context context, Cursor cursor) {
                  RestaurantHolder holder = (RestaurantHolder) view.getTag();
                  holder.populateFrom(cursor, helper);
              }

              @Override
              public View newView(Context context, Cursor cursor, ViewGroup parent) {
                  LayoutInflater inflater = getLayoutInflater();
                  View row = inflater.inflate(R.layout.row, parent, false);
                  RestaurantHolder holder = new RestaurantHolder(row);
                  row.setTag(holder);
                  return (row);
              }
          }
      }
```

73. *IMPORTANT
    Open the *AndroidManifest.xml* file and add in the extra EditPreferences *Activity* to the **Application** tag



```
<activity android:name=".RestaurantList">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity android:name=".DetailForm"></activity>
<activity android:name=".EditPreferences"></activity>
```

74. Run the *Lab4b* project. Enter some data to create a restaurant list and do test on preference the settings
75. Show your lecturer when you have completed this section

Lecturer Signature: ______________________________

-END-