Activities & Preference

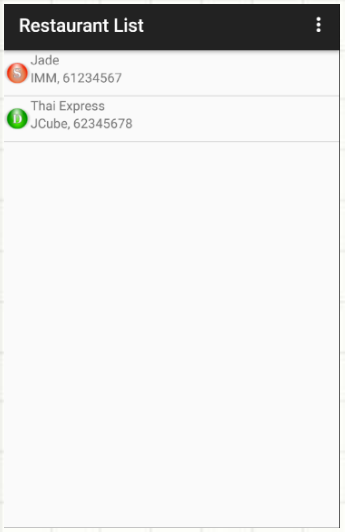
Today's Overview

1. Activities
2. Preferences

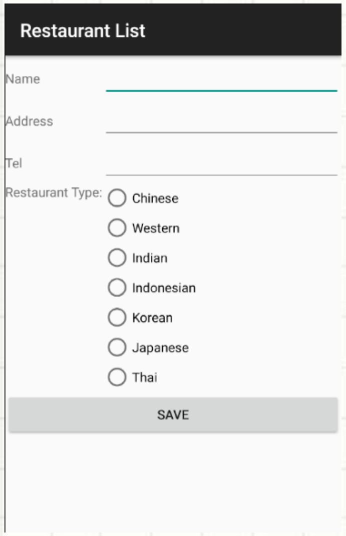
Activities

Activity

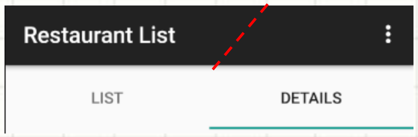
- In first part of Practical 4 exercise, the *RestaurantActivity* will be split into two Activities
  - one *AppCompatActivity (RestaurantList)* to handle 'List' UI view
  - one *AppCompatActivity (DetailForm)* to handle 'Details' form UI view



RestaurantList.java



DetailForm.java



Activity

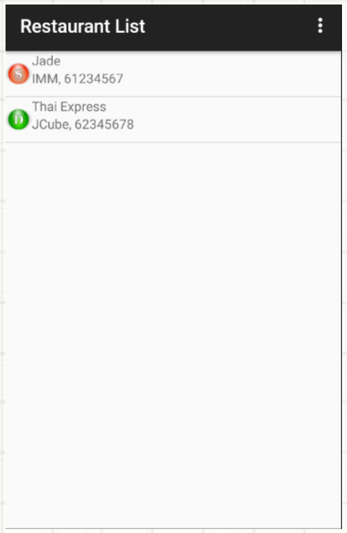
- Let’s check what do we need to modify from the previous exercise to split the UI views to be controlled by separate *Activity*?
  - ❑ Model - Any change in Data Model?
  - ❑ View - Do you need to modify any of the user interface view?
  - ❑ Controller - Do you need to tell the Controller to do any thing new?

Activity

- ❑ Model - NO
- ❑ View - YES

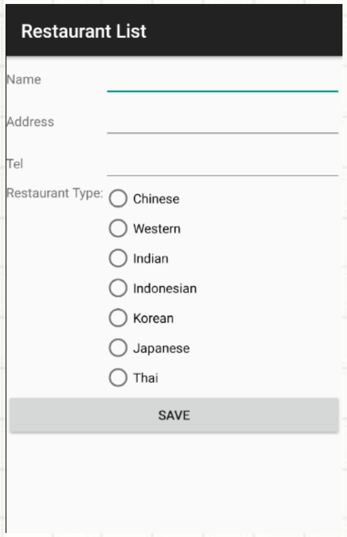
the UI View will be split into to separate XML layout files

- ✓ *main.xml* layout for ‘List’ UI view
- ✓ *detail_form.xml* for ‘Details’ form UI view


Restaurant List

Jade
IMM, 61234567

Thai Express
JCube, 62345678


Restaurant List

Name

Address

Tel

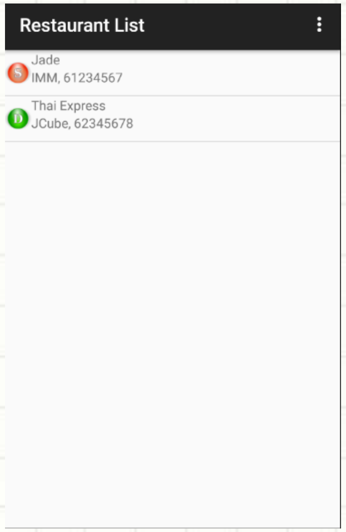
Restaurant Type:
- Chinese
- Western
- Indian
- Indonesian
- Korean
- Japanese
- Thai

SAVE

Activity

❑View - YES

an "Add" MENU option is created at the bottom of the UI View for adding new restaurant data to the restaurants table Model. The *option.xml* layout file in **res/menu** folder will control the View



Activity

- ❑ Controller - YES

  it will be slit into two Controller activities

  - ✓ *RestaurantList AppCompatActivity* Controller – to handle ‘List’ UI view update from *Cursor* Model, calling *DetailForm.java* Activity through Explicit Intent and handle MENU option presses
  - ✓ *DetailForm AppCompatActivity* Controller – to handle restaurants table Model for new record adding

UI View - Separate Layouts & Option Menu

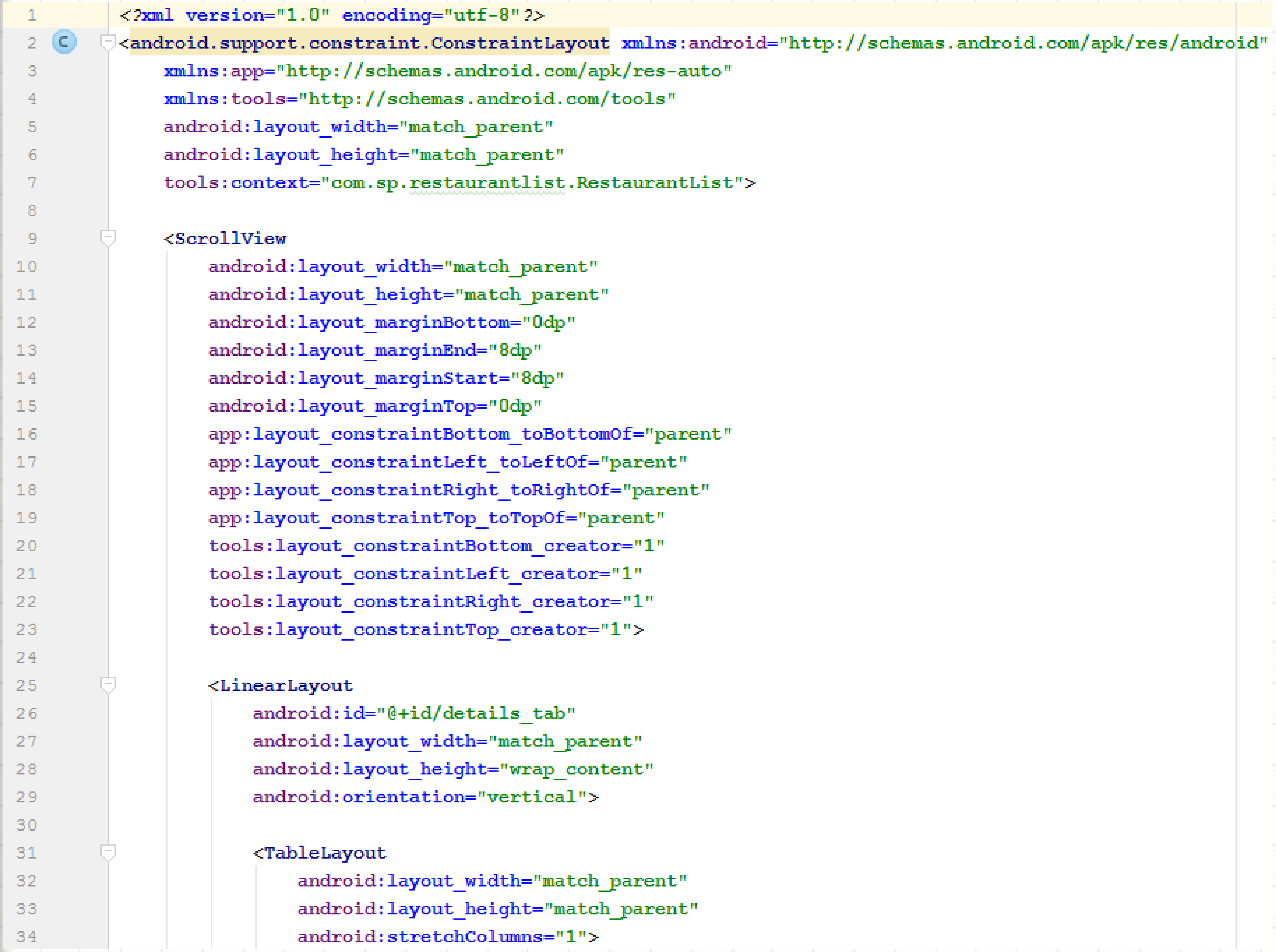
View

‘Details’ Form UI View

- The layout for showing ‘Details’ form is taken out from the *main.xml* layout and saved as a separated layout file *detail_form.xml* in **res/layout** folder

View

detail_form.xml Listing

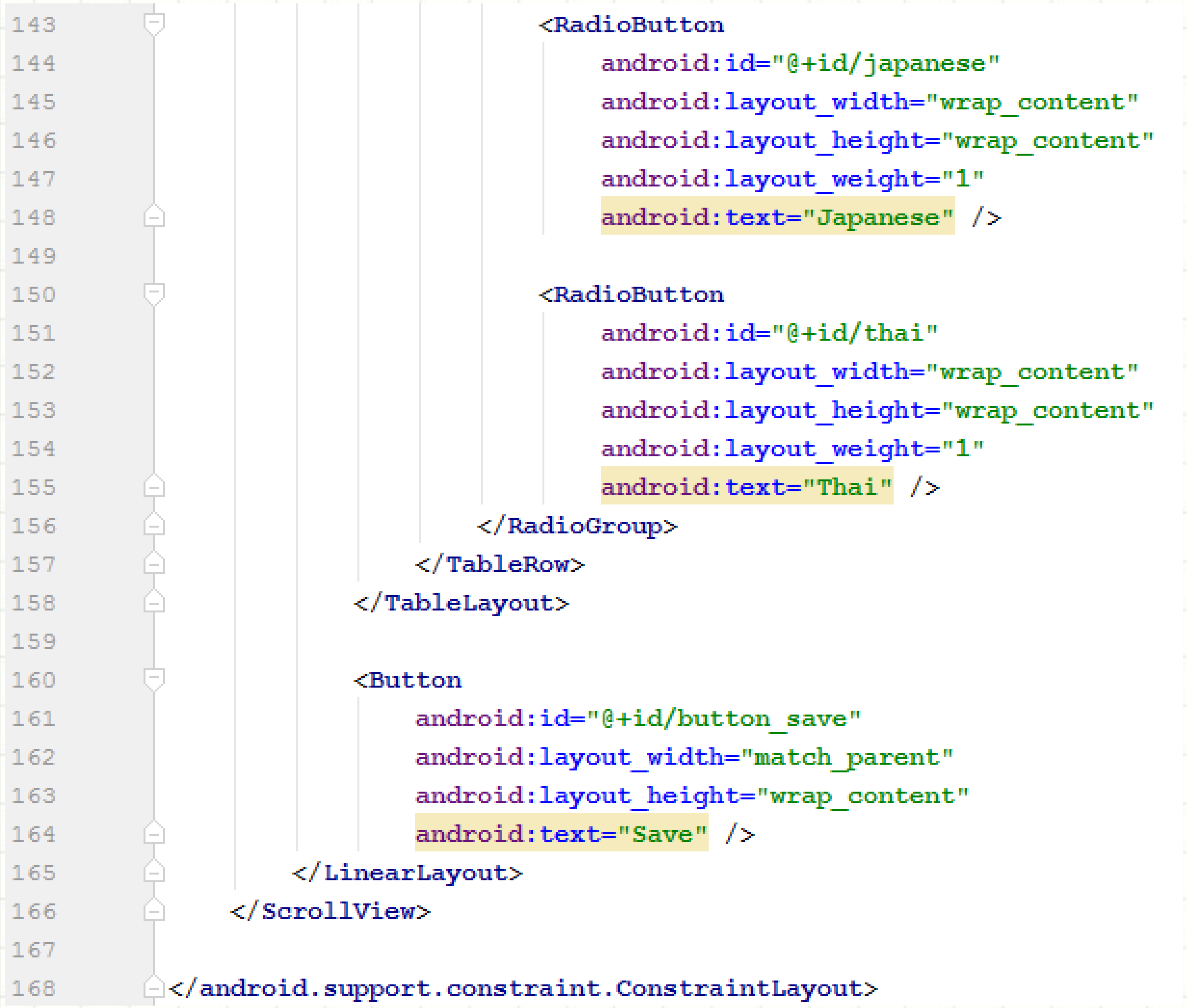


```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.sp.restaurantlist.RestaurantList">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginBottom="0dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:layout_constraintBottom_creator="1"
        tools:layout_constraintLeft_creator="1"
        tools:layout_constraintRight_creator="1"
        tools:layout_constraintTop_creator="1">

        <LinearLayout
            android:id="@+id/details_tab"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <TableLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:stretchColumns="1">
```

View

detail_form.xml Listing

⋮



```
                        <RadioButton
                            android:id="@+id/japanese"
                            android:layout_width="wrap_content"
                            android:layout_height="wrap_content"
                            android:layout_weight="1"
                            android:text="Japanese" />

                        <RadioButton
                            android:id="@+id/thai"
                            android:layout_width="wrap_content"
                            android:layout_height="wrap_content"
                            android:layout_weight="1"
                            android:text="Thai" />
                    </RadioGroup>
                </TableRow>
            </TableLayout>

            <Button
                android:id="@+id/button_save"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Save" />
        </LinearLayout>
    </ScrollView>

</android.support.constraint.ConstraintLayout>
```

View

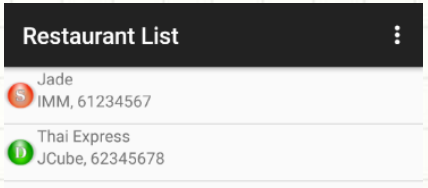
‘List’ UI View

- The *main.xml* layout contains only *ListView* layout to display records saved and a *TextView* to display an instruction message

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

⋮

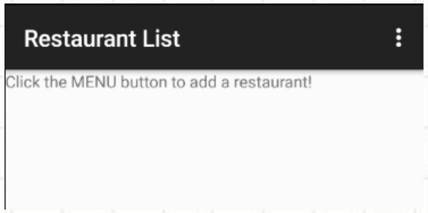
```
  <ListView
      android:id="@+id/list"
      android:layout_width="match_parent"
      android:layout_height="match_parent" />

  <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical">

      <TextView
          android:id="@+id/empty"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:text="Click the MENU button to add a restaurant!" />
  </LinearLayout>

</FrameLayout>
```



Restaurant List

Jade
IMM, 61234567

Thai Express
JCube, 62345678



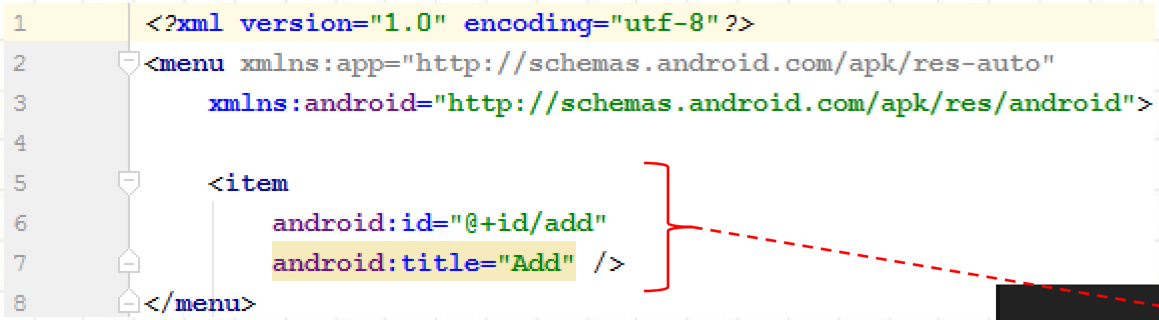
Restaurant List

Click the MENU button to add a restaurant!

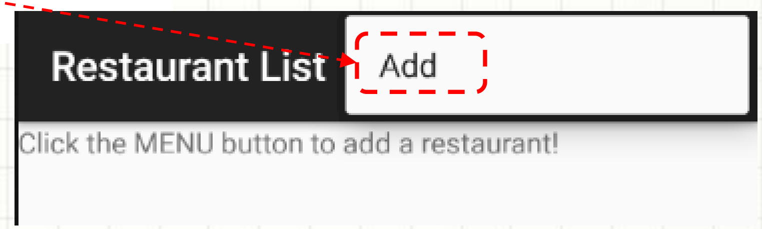
View

Add Menu Option

- The *option.xml* layout file under **res/menu** folder will be modified to show “Add” MENU option



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

    <item
        android:id="@+id/add"
        android:title="Add" />
</menu>
```



Click the MENU button to add a restaurant!

Controller – Explicit Intent & Option Menu

Controller

- The Controller tasks are split and handled by individual activities (*RestaurantList* and *DetailForm*)
- *RestaurantList* takes care of *ListView* item. When 'Add' MENU item is selected, it will then activate the *DetailForm Activity* to load the 'Details' form UI View

Controller

- *RestaurantList* also takes care of *ListView* and *Cursor* Model update for ‘List’ UI View through *CursorAdapter*
- *DetailForm Activity* handles ‘Details’ form UI View and interaction of restaurant table Model through *RestaurantHelper*

Data Updating

Activity

- In second part of Practical 4 exercise, an extra feature is added to allow user to update the existing restaurant data and save

Activity

- Let’s check what do we need to modify from the previous exercise to allow restaurant data to be updated?
  - ❑Model - Any change in Data Model?
  - ❑View - Do you need to modify any of the user interface view?
  - ❑Controller - Do you need to tell the Controller to do any thing new?

Activity

- Model - YES

to add in two new methods to *RestaurantHelper*

- ✓ getById() - to retrieve existing record from restaurants_table Model with the specified record 'ID' provided
- ✓ update() - to update edited record to restaurants_table Model with the specified record 'ID' provided

Activity

- ❑ View - NO
- ❑ Controller - YES
  - ✓ onListClick - to handle 'List' item selection at 'List' UI View, get the record ID and pass to *DetailForm* Controller
  - ✓ DetailForm - to decide update record or insert new record to restaurants_table data model

Model – RestaurantHelper

Model

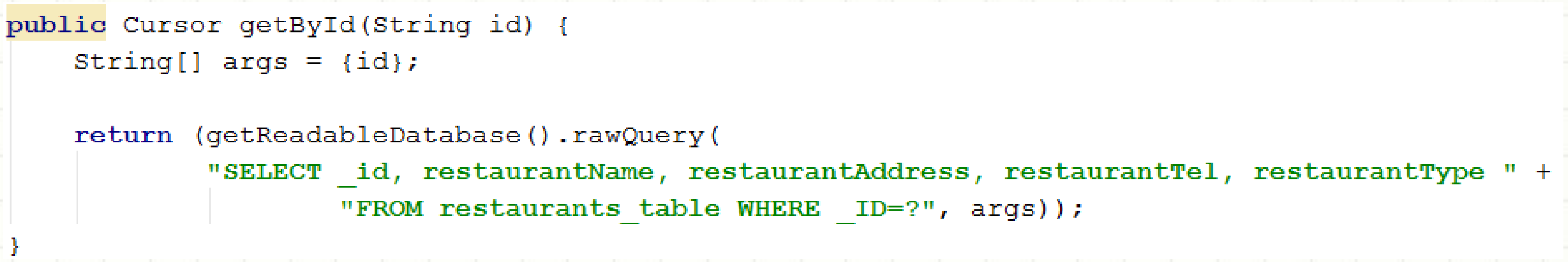
RestaurantHelper Model

- There is no change in restaurant table Model. The data elements will remain unchanged
- Due to the extra features in retrieving and updating existing record , the *RestaurantHelper* will be added with three new methods for the purpose

Model

RestaurantHelper Model

- Method to retrieve record from the restaurants_table model by record “_id” field

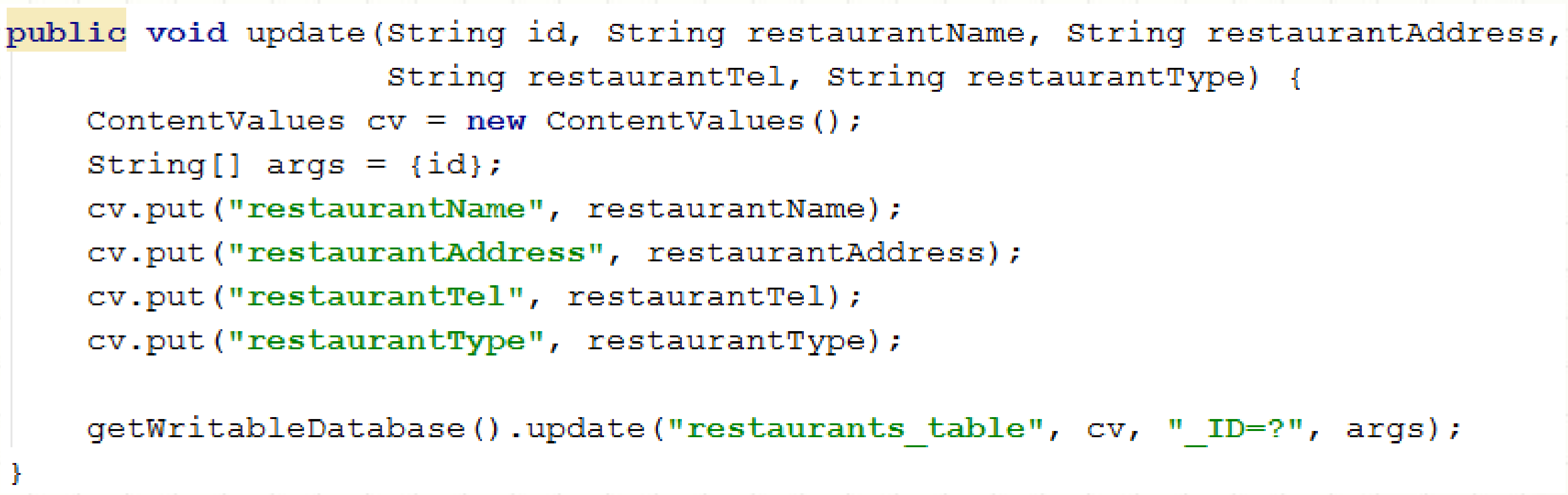


```
public Cursor getById(String id) {
    String[] args = {id};

    return (getReadableDatabase().rawQuery(
            "SELECT _id, restaurantName, restaurantAddress, restaurantTel, restaurantType " +
                    "FROM restaurants_table WHERE _ID=?", args));
}
```

Model

RestaurantHelper Model

- Method to update existing record to the restaurants_table model by record “_id” field

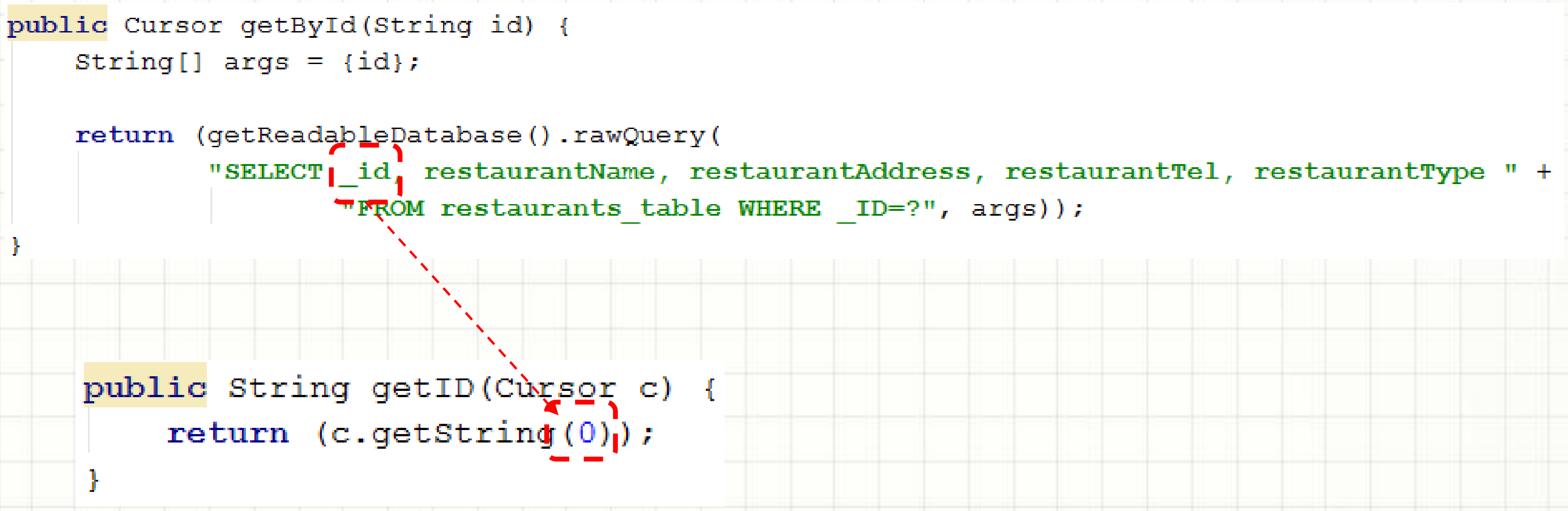


```java
public void update(String id, String restaurantName, String restaurantAddress,
                   String restaurantTel, String restaurantType) {
    ContentValues cv = new ContentValues();
    String[] args = {id};
    cv.put("restaurantName", restaurantName);
    cv.put("restaurantAddress", restaurantAddress);
    cv.put("restaurantTel", restaurantTel);
    cv.put("restaurantType", restaurantType);

    getWritableDatabase().update("restaurants_table", cv, "_ID=?", args);
}
```

Model

RestaurantHelper Model

- Method to get the value of the “_id” field of the record read from restaurants_table model



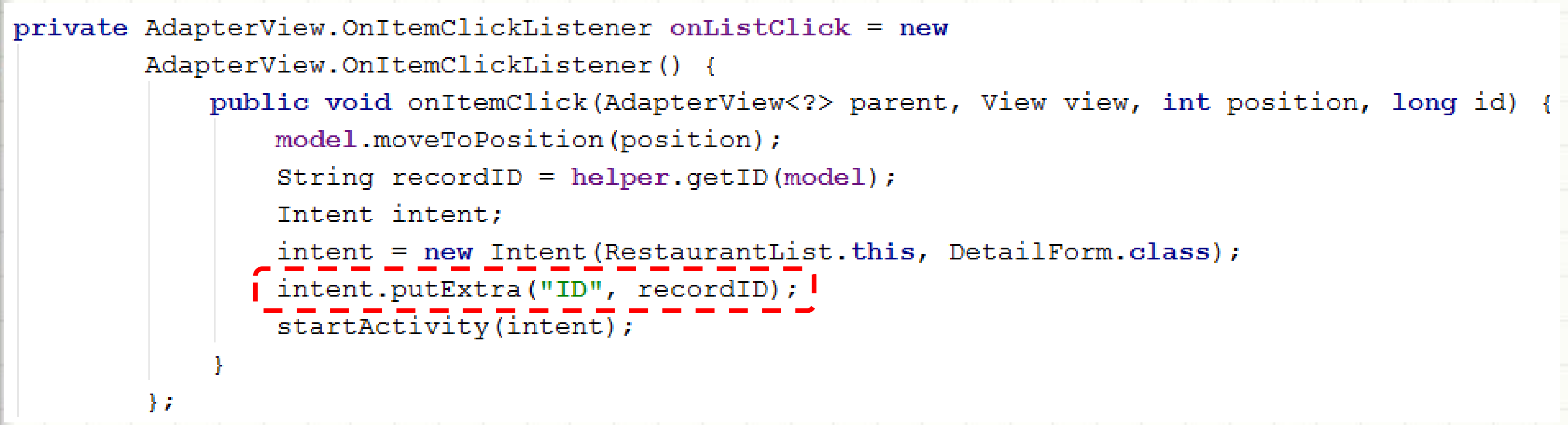
```
public Cursor getById(String id) {
    String[] args = {id};

    return (getReadableDatabase().rawQuery(
            "SELECT _id, restaurantName, restaurantAddress, restaurantTel, restaurantType " +
                    "FROM restaurants_table WHERE _ID=?", args));
}
```

```
public String getID(Cursor c) {
    return (c.getString(0));
}
```

Controller – ‘List’ Item Select & Passing ‘ID’

Controller

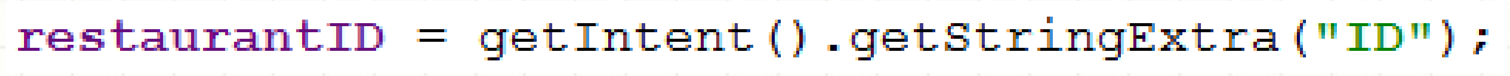
- Detect ‘List’ item click through OnItemClickListener and make an Explicit Intent call to *DetailForm*
- Pass the record ID over to ‘Details’ form using Intent putExtra method



```
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);
            }
        };
```

Controller

- *DetailForm* Activity uses the getIntent.getStringExtra to retrieve the ‘ID’ string passed over by *RestaurantList AppCompatActivity* and save to local variable restaurantID



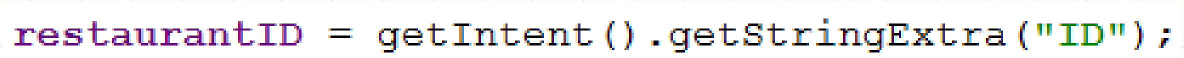
```
restaurantID = getIntent().getStringExtra("ID");
```

Controller

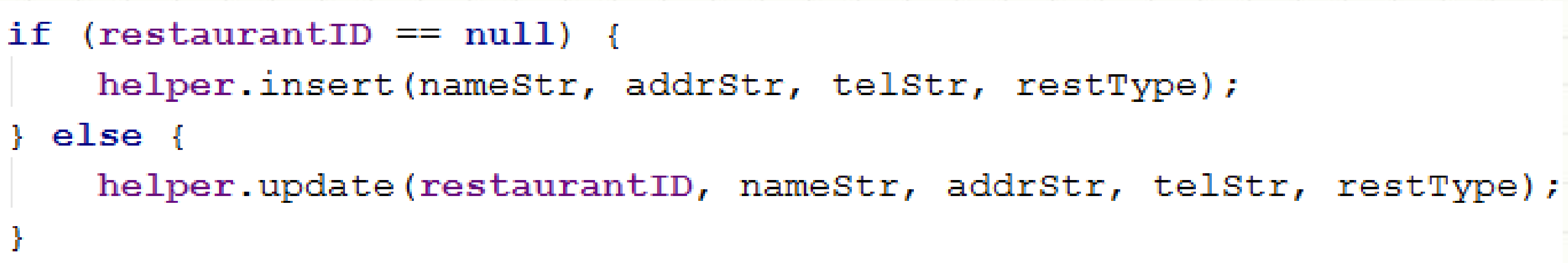
- How does *DetailForm* able to differentiate whether the *Explicit Intent* calls from *RestaurantList* (Call 1 - ‘Add’ MENU item is selected; Call 2 - ‘List’ Item is selected) is to add new restaurant record or to update existing restaurant record when “Save” button is pressed?

Controller

- The solution is to check the restaurantID value
- For adding new record, the Explicit Intent Call will not pass and 'ID' to 'Details' form i.e. restaurantID equals to 'null'



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

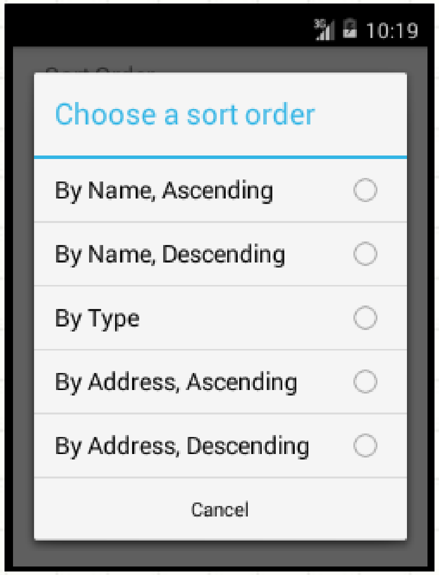


```
if (restaurantID == null) {
    helper.insert(nameStr, addrStr, telStr, restType);
} else {
    helper.update(restaurantID, nameStr, addrStr, telStr, restType);
}
```

Preference

Preference

- In third part of Practical 4 exercise, a preference setting for sorting the order of restaurant list is introduced



Choose a sort order

By Name, Ascending ○

By Name, Descending ○

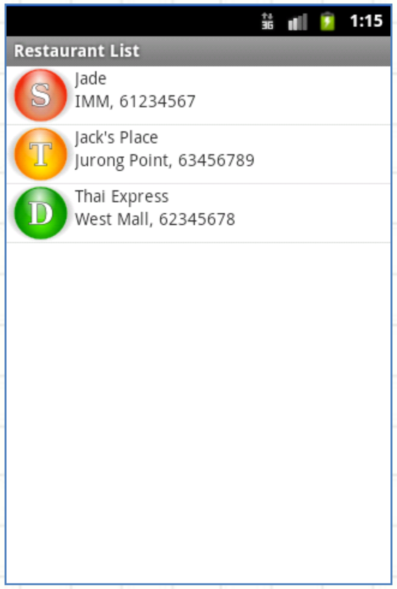
By Type ○

By Address, Ascending ○

By Address, Descending ○

Cancel

Sort Preference Setting



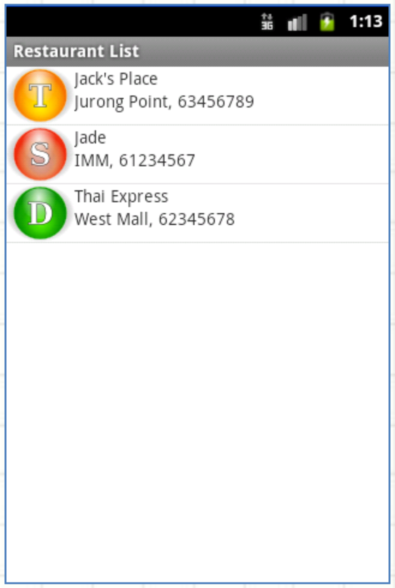
Restaurant List

Jade
IMM, 61234567

Jack's Place
Jurong Point, 63456789

Thai Express
West Mall, 62345678

List sorted by Address in ascending order



Restaurant List

Jack's Place
Jurong Point, 63456789

Jade
IMM, 61234567

Thai Express
West Mall, 62345678

List sorted by Name in ascending order

Preference

- Let's check what do we need to modify from the previous exercise to allow user to set sorting preference on records displayed in 'List' UI View?
  - ❑Model - Any change in Data Model?
  - ❑View - Do you need to modify any of the user interface view?
  - ❑Controller - Do you need to tell the Controller to do any thing new?

Preference

- Model - YES

  the getAll() method will be changed to allow restaurant records read from *restaurants_table* Data Model in preference sorting order

Preference

- View - YES

  an extra **MENU** item “Setting” is added to the *option.xml* layout. When selected, a *preferences.xml* layout will be shown on UI View

Preference

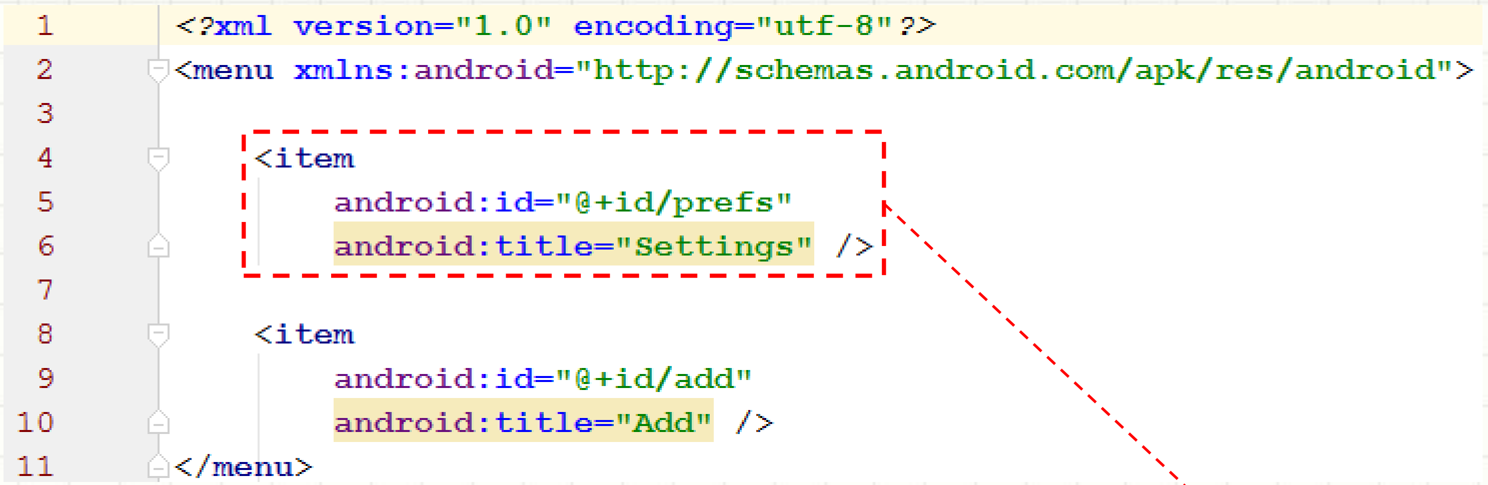
- Controller - YES

  EditPreferences, sub-class of *PreferenceActivity*, is used to handle preference selection and control the sort order through capturing the *SharedPreference* changed

View – Preference

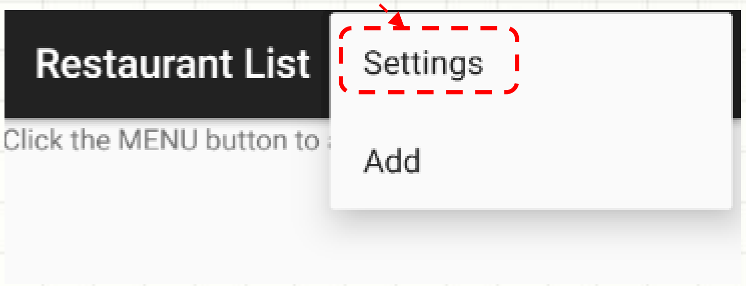
View

- An extra MENU option item “Setting” is added to *option.xml* layout for UI View



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

    <item
        android:id="@+id/prefs"
        android:title="Settings" />

    <item
        android:id="@+id/add"
        android:title="Add" />
</menu>
```

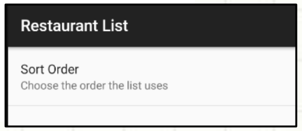
Listing - options.xml



View

- The preference view is constructed by *preferences.xml* layout file and *arrays.xml* file for the pop-up choices

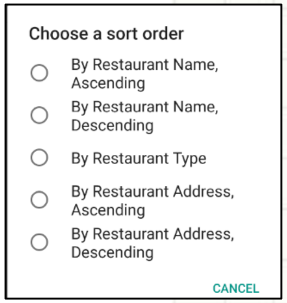
```
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android">
  <ListPreference
    android:key="sort_order"
    android:title="Sort Order"
    android:summary="Choose the order the list uses"
    android:entries="@array/sort_names"
    android:entryValues="@array/sort_clauses"
    android:dialogTitle="Choose a sort order" />
</PreferenceScreen>
```


Restaurant List

Sort Order
Choose the order the list uses

Listing - preferences.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>

```


Choose a sort order

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

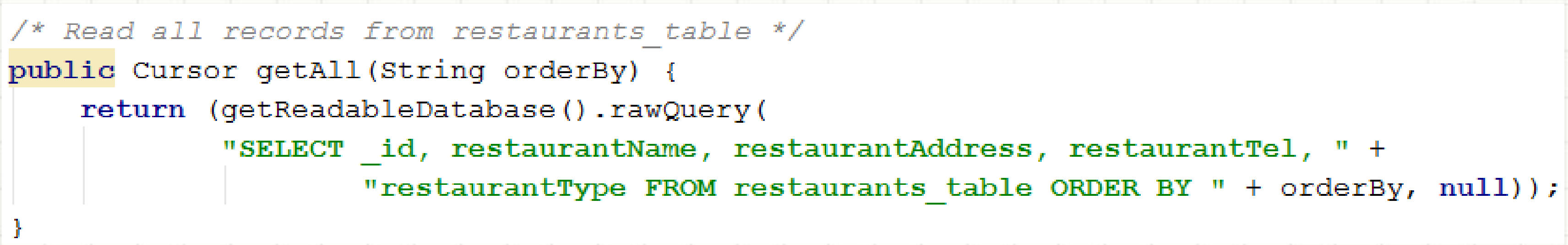
CANCEL

Listing - arrays.xml

Model - RestaurantHelper

Model

- The getAll method in *RestaurantHelper.java* is changed to allow records in restaurants table Model to be retrieved in a specific display order to *Cursor* Model which is used for rows in *ListView*



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

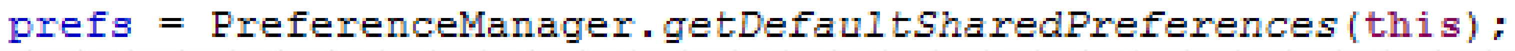
Controller – Setting up

Controller

- Android provides several options for you to save persistent application
- One of the data storage option is Shared Preferences which stores private primitive data in key-value pairs

Controller

- *EditPreferences*, sub-class of **PreferenceActivity**, provides an Activity framework for you to create user preferences data for your application, which will automatically persists (using Shared Preference)
- *RestaurantList* initializes variable ‘prefs’ to be the *SharedPreferences* of *EditPreferences* Activity



```
prefs = PreferenceManager.getDefaultSharedPreferences(this);
```

Controller

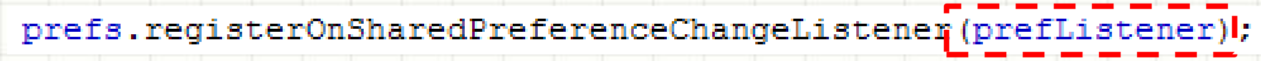
- If user has not specified a sort_order, "*restaurantName*" will be used as default sort value to retrieve records from *restaurants_table* Model to Cursor Model

```
Cursor model = null;
```

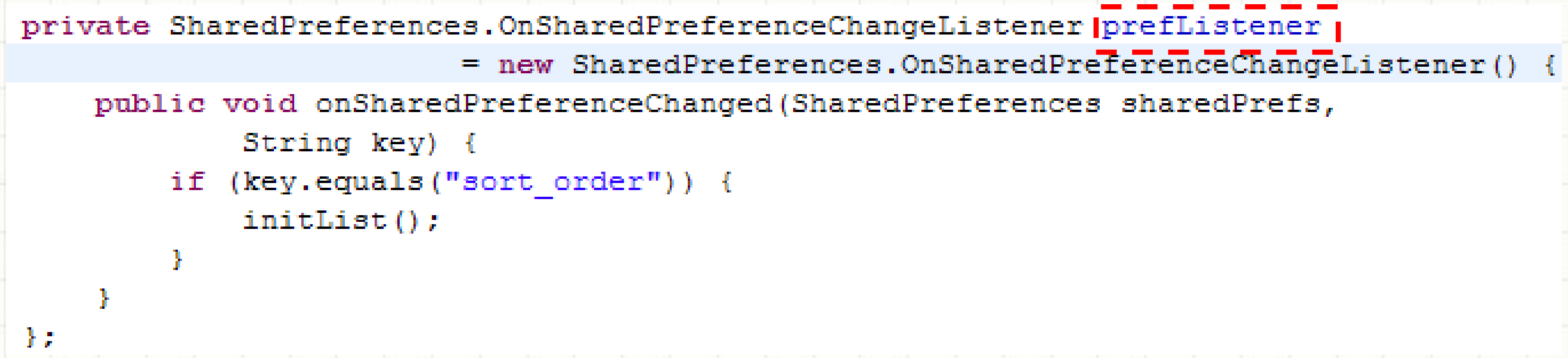
```
model = helper.getAll(prefs.getString("sort order", "restaurantName"));
```

Controller

- In order for the *RestaurantList* to be able to capture any change of sort order preference at *preferences.xml* layout View, *SharedPreferences* has the notion of a preference listener object, to be notified on such changes



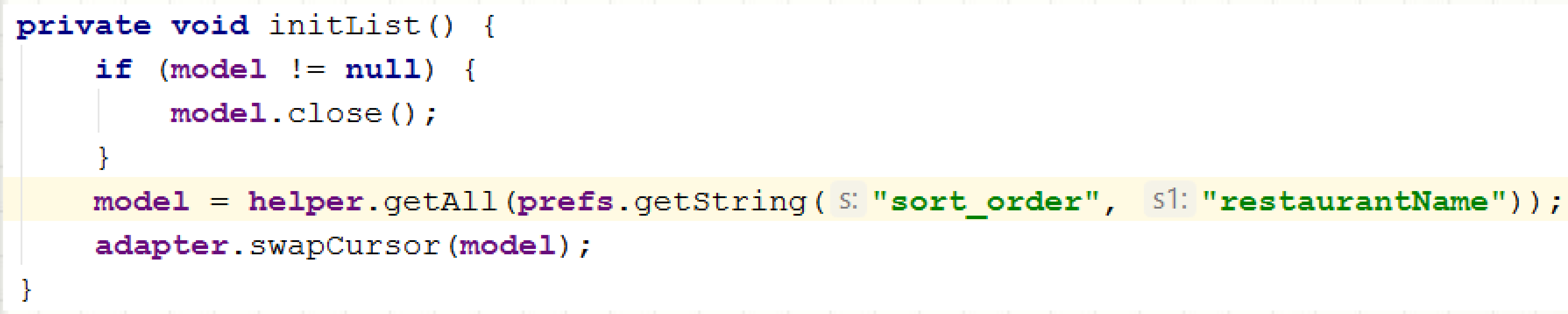
```
prefs.registerOnSharedPreferenceChangeListener(prefListener);
```



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

Controller

- In the initList() method, the model.close() will cause the old *Cursor* to be ignored and followed by getting a fresh *Cursor* representing new sort order list through getAll(….) method and update the ListView through adapter.swapCursor(model)



```java
private void initList() {
    if (model != null) {
        model.close();
    }
    model = helper.getAll(prefs.getString( s: "sort_order",  s1: "restaurantName"));
    adapter.swapCursor(model);
}
```

END