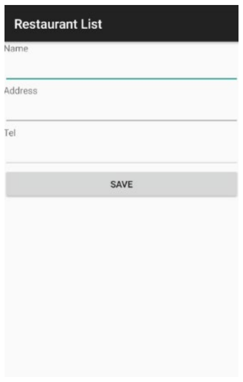
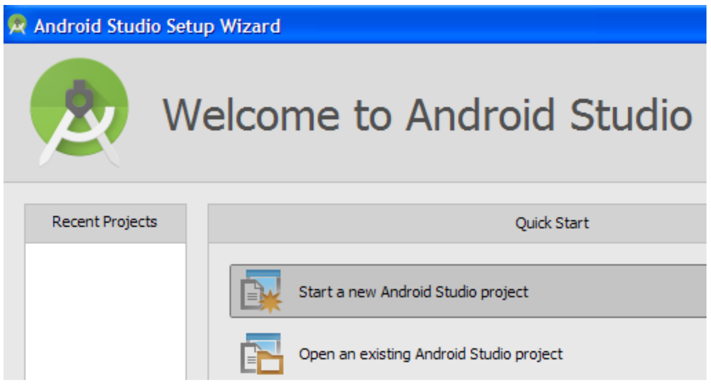
Practical 1: UI Design

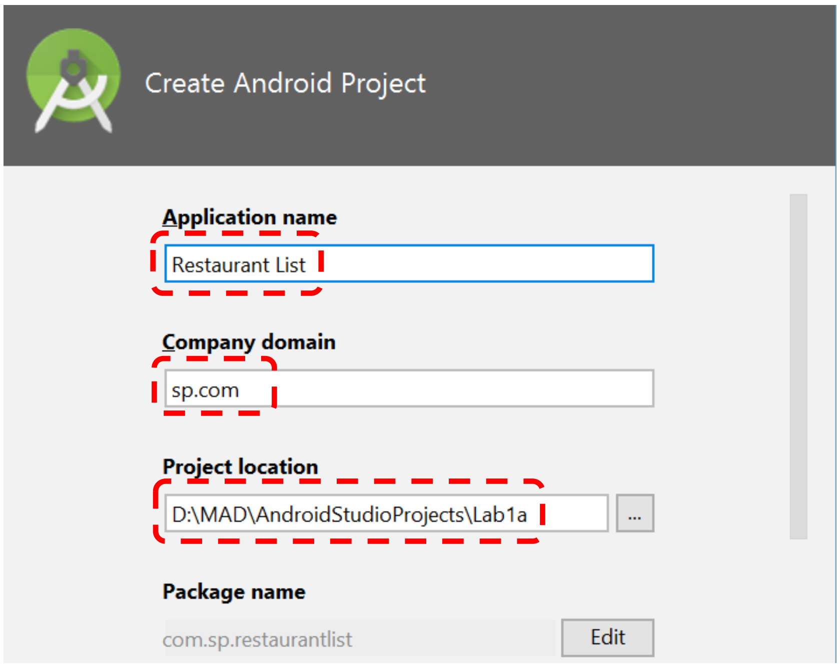
At the end of the session, you will learn how to use Android Layout Editor, both Graphical mode and Text mode, to design an UI as well as setting the attribute of widgets to format data entry



Part I – Creating an Android Application Project

1. Start a new Android Studio project with
   - Application name: *Restaurant List* **(Note: With space in between. This name will appear on phone screen)**
   - Company Domain: *sp.com*
   - Project location: *D:\MAD\AndroidStudioProjects\Lab1a or C:\MAD\AndroidStudioProjects\Lab1a*


Android Studio Setup Wizard
Welcome to Android Studio
Recent Projects
Quick Start
Start a new Android Studio project
Open an existing Android Studio project


Create Android Project

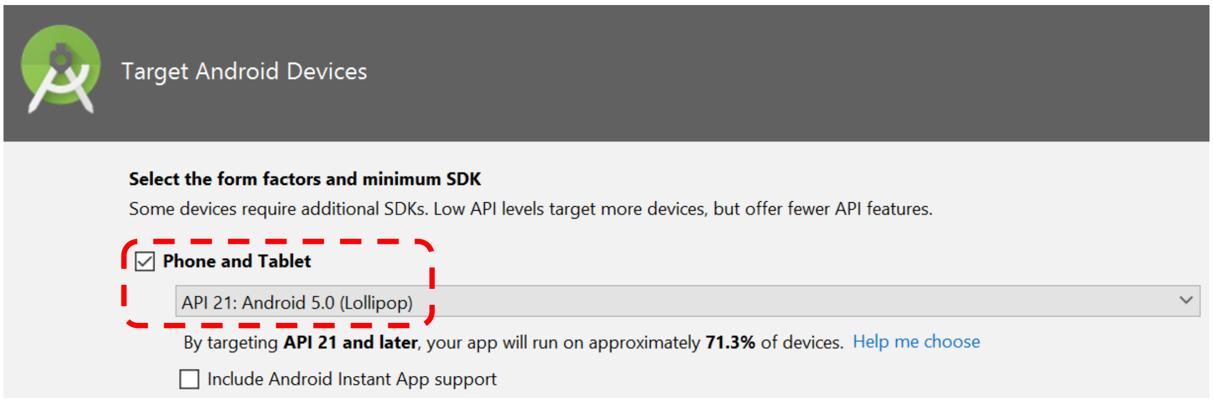
Application name: Restaurant List

Company domain: sp.com

Project location: D:\MAD\AndroidStudioProjects\Lab1a

Package name: com.sp.restaurantlist

2. Select Minimum SDK:
   - *API 21: Android 5.0 (Lollipop)*



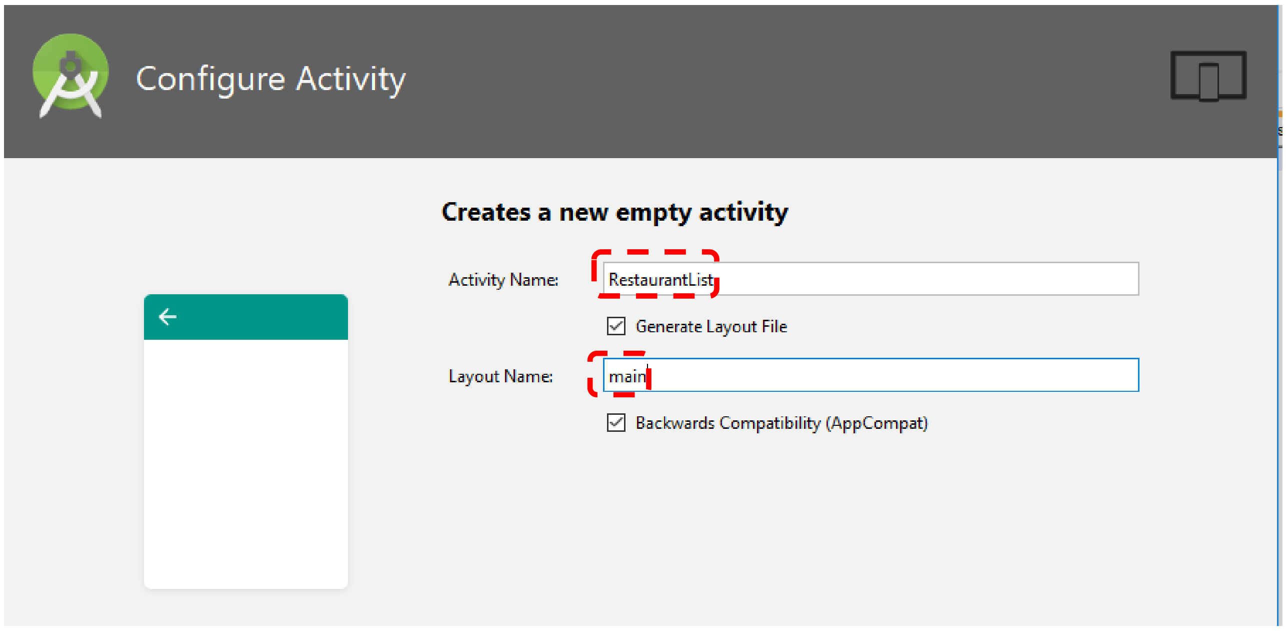
Select the form factors and minimum SDK
Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☑ **Phone and Tablet**
API 21: Android 5.0 (Lollipop)
By targeting **API 21 and later**, your app will run on approximately **71.3%** of devices.
☐ Include Android Instant App support

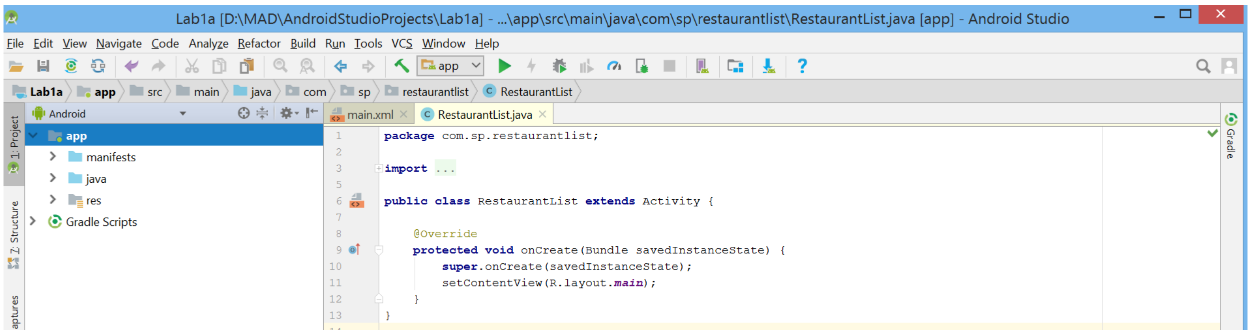
3. Select **Empty Activity** and enter the following information:
   - Activity Name: *RestaurantList*
   - Layout Name: *main*

   (Note: This layout file will bind to RestaurantList activity as user View on phone display)

4. *Lab1a* project will be created as shown



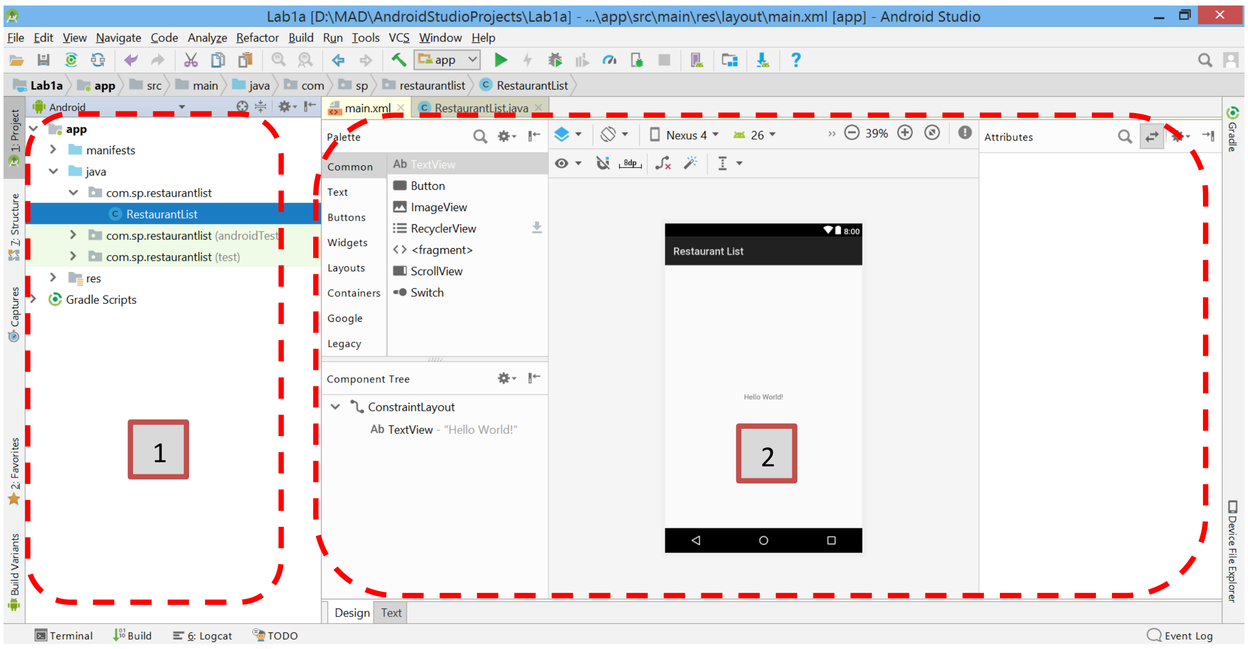
Creates a new empty activity
Activity Name: RestaurantList
☑ Generate Layout File
Layout Name: main
☑ Backwards Compatibility (AppCompat)


Lab1a [D:\MAD\AndroidStudioProjects\Lab1a] - ...\app\src\main\java\com\sp\restaurantlist\RestaurantList.java [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Lab1a app src main java com sp restaurantlist RestaurantList
Android
app
manifests
java
res
Gradle Scripts
main.xml
RestaurantList.java
1: Project
7: Structure
Gradle

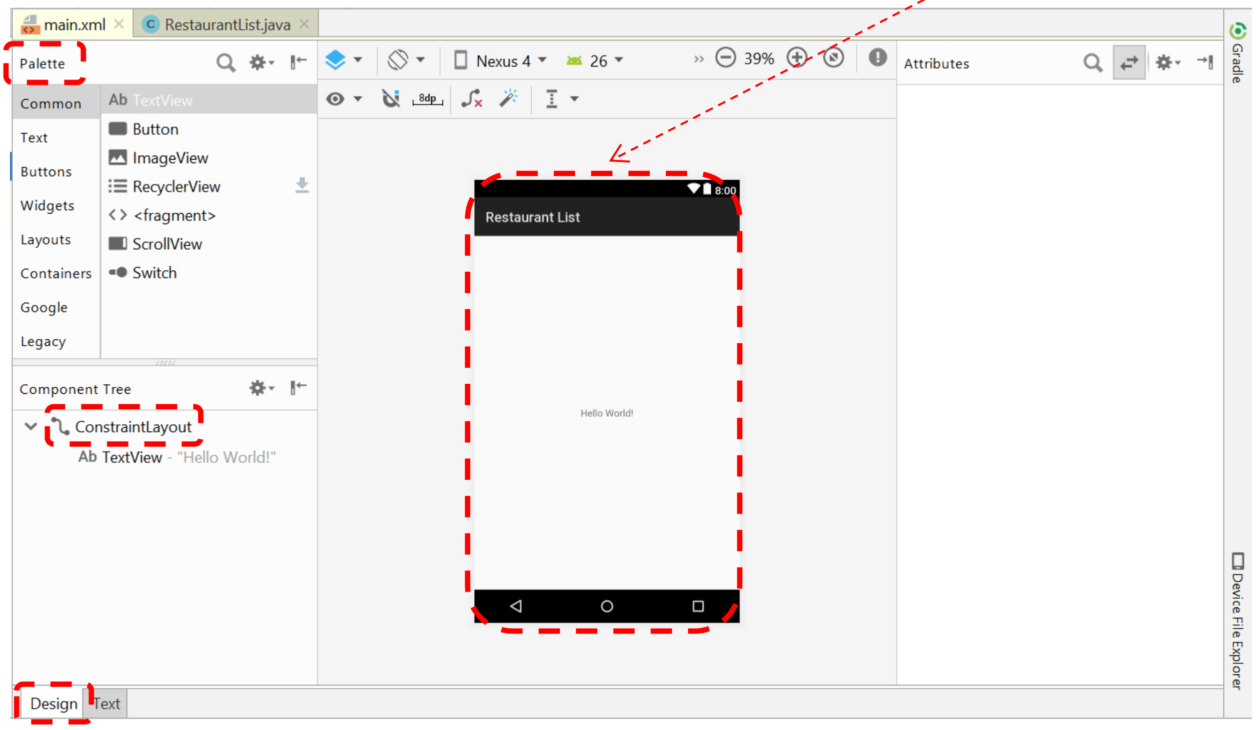
```java
package com.sp.restaurantlist;

import ...

public class RestaurantList extends Activity {

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

5. Take a look at Android Studio IDE view.
   - View 1 – Project View showing the contents of the project structure

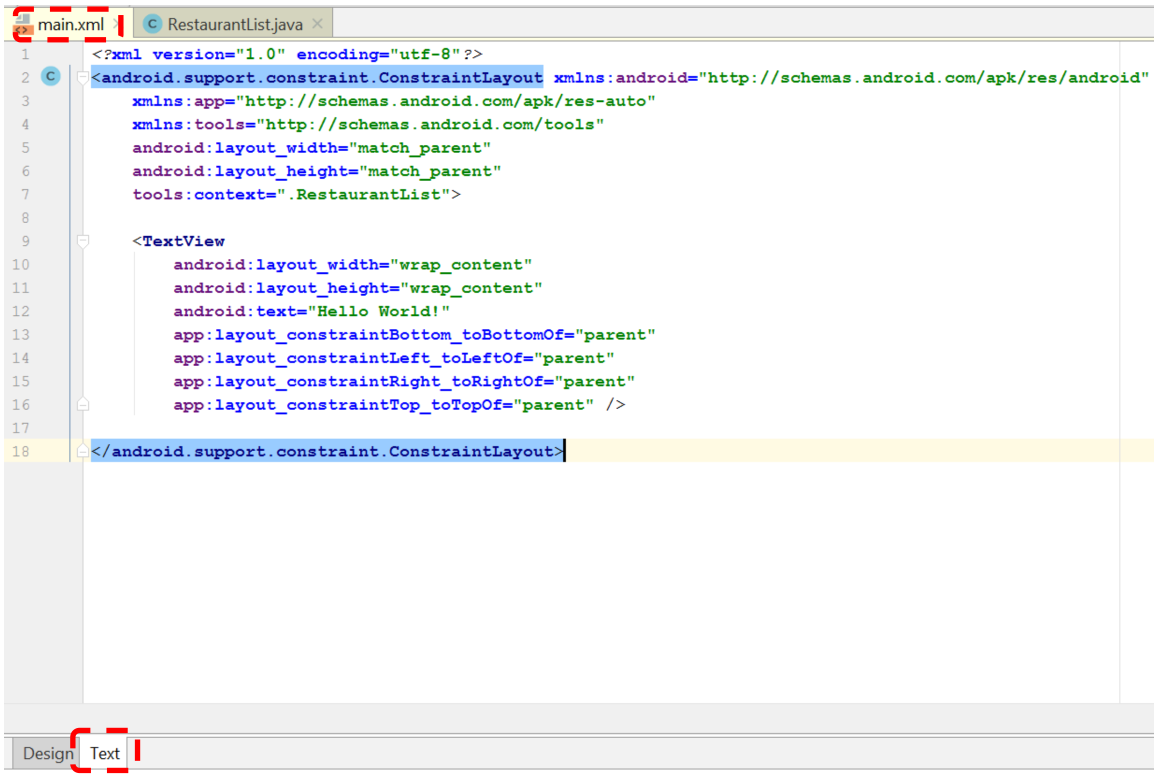
- View 2 – Editor View to allow user to edit User Interface (UI) layout and program



6. At the Layout Editor, the *main.xml* file is the layout file created for *RestaurantList.java* Activity
7. By default, all layout files shall be stored in the **res/layout** folder
8. The IDE has provided to two options of **Layout Editor** to allow user to create User Interface (UI) View:
   - **Design** Editor – allow user to drag and drop widgets from **Palette** into the Design Editor view or into Component Tree

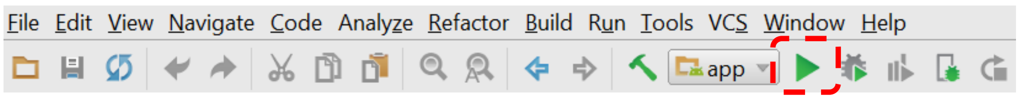


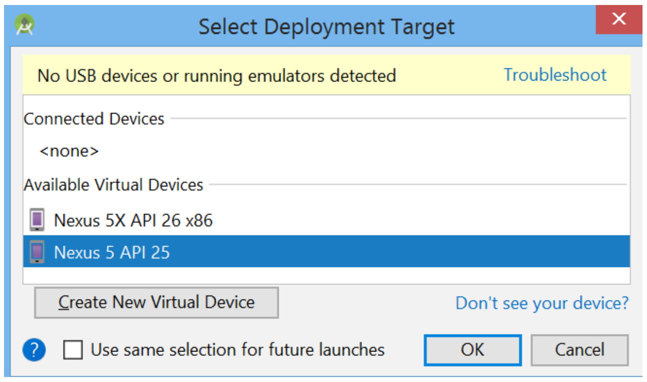
- **Text** Editor – create UI View using XML text file
  (**Note** to indent the file, at Top MENU Bar, click on **Code > Reformat Code** at the top menu bar)



```
<?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=".RestaurantList">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

9. To run the app create in the *Lab1* project, click on ▶ button at the top menu bar


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
app

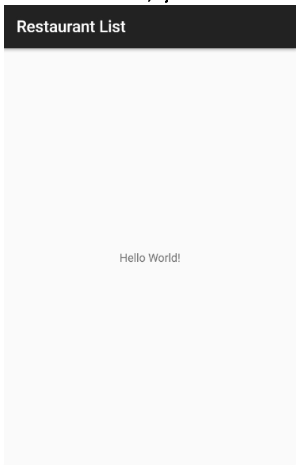


No USB devices or running emulators detected

Connected Devices
<none>
Available Virtual Devices
Nexus 5X API 26 x86
Nexus 5 API 25

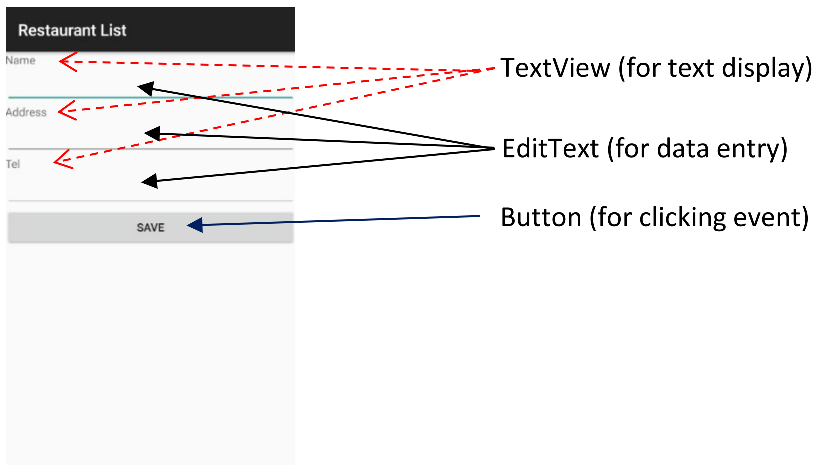
☐ Use same selection for future launches

10. If successful, you will see an emulator is launched with the 'Hello world!' message shown on screen.

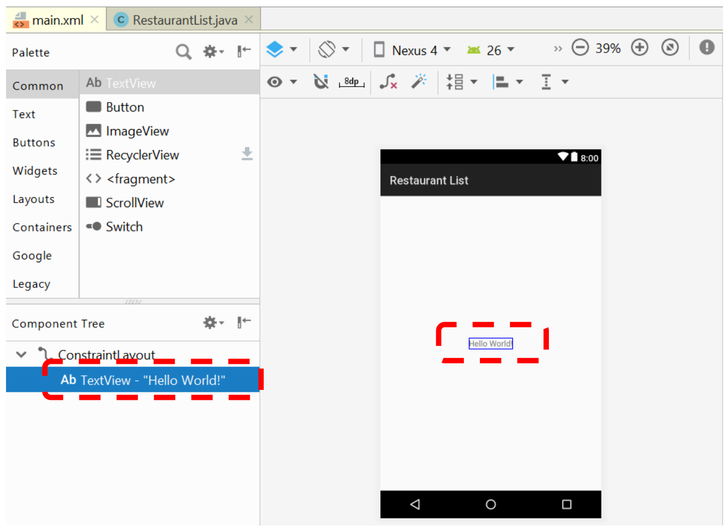


Part II - Creating Simple Form UI View

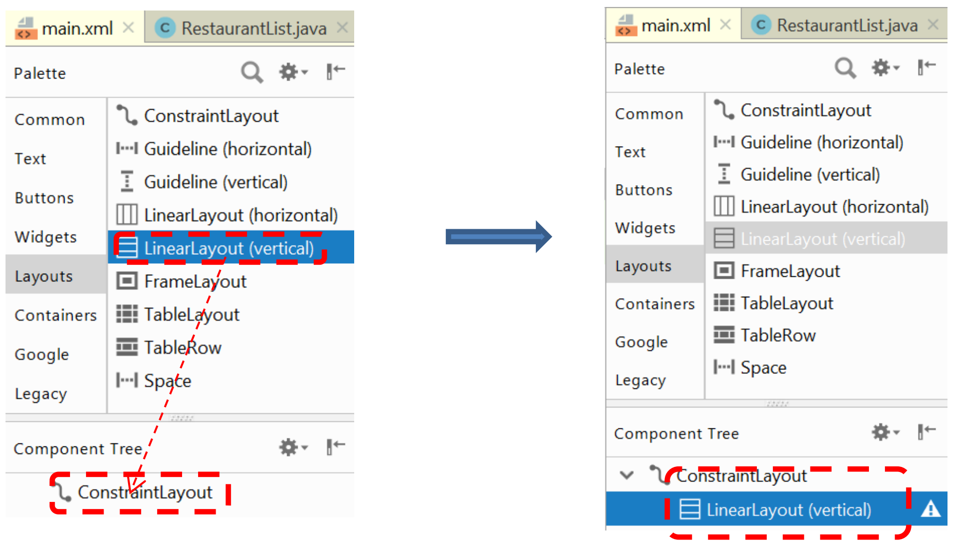
11. The subsequence steps will lead you to create a new layout for the *Restaurant List* application. It comprises three *TextView*, three *EditText* and a *Button* widgets


Restaurant List
Name
Address
Tel
SAVE
TextView (for text display)
EditText (for data entry)
Button (for clicking event)

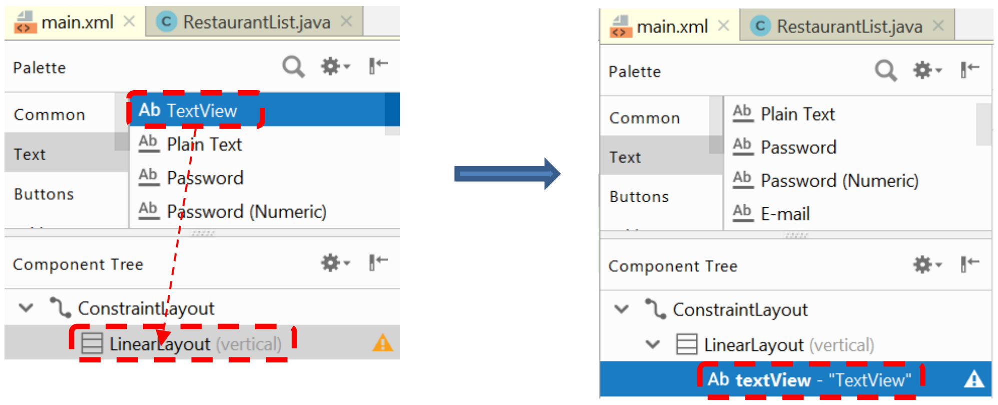
12. At the Design Editor pane of *main.xml* file, click on the *TextView* widget in the **Component Tree** and press delete key from keyboard to remove the widget from the View



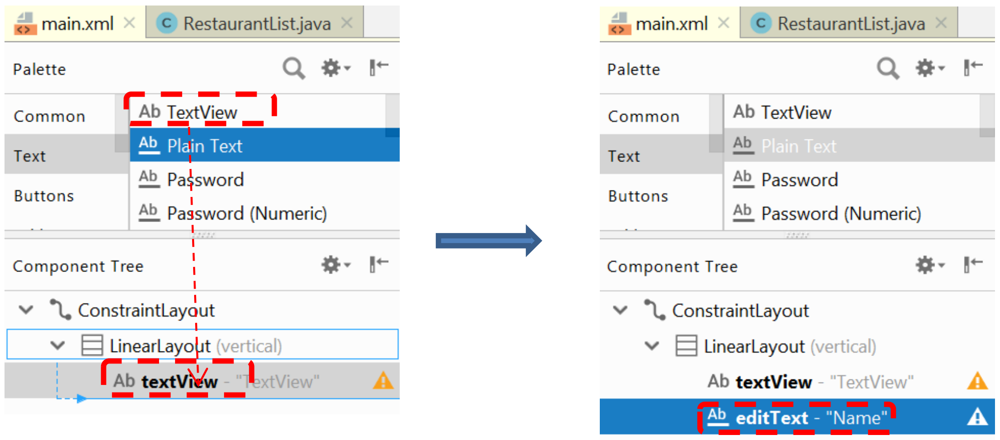
13. Select the *LinearLayout (Vertical)* from **Palette pane**. Drag and drop it onto the *ConstraintLayout* in the **Component Tree**



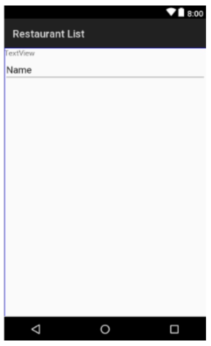
14. At **Palette** pane, drag **TextView** component from **Text** option and drop into **LinearLayout**



15. Drag a **Plain Text** component from **Text** option and drop below **textView** component

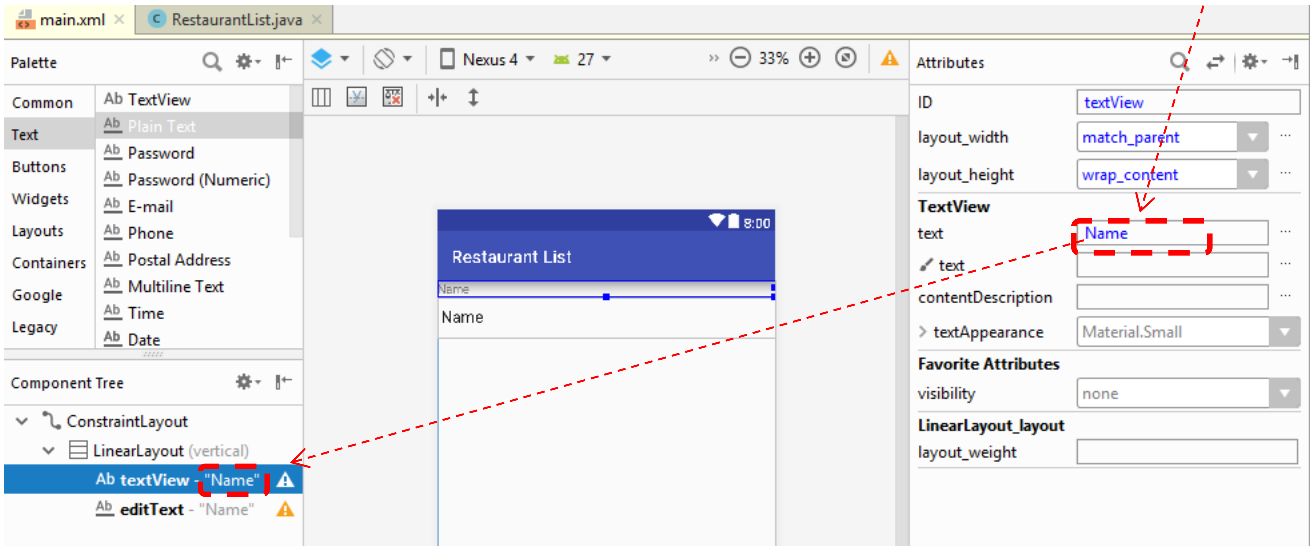


16. What you have done so far is to create an user interface display using components provided
    - *TextView* Widget – to display information on user screen
    - *EditText* Widget – to allow user to enter information



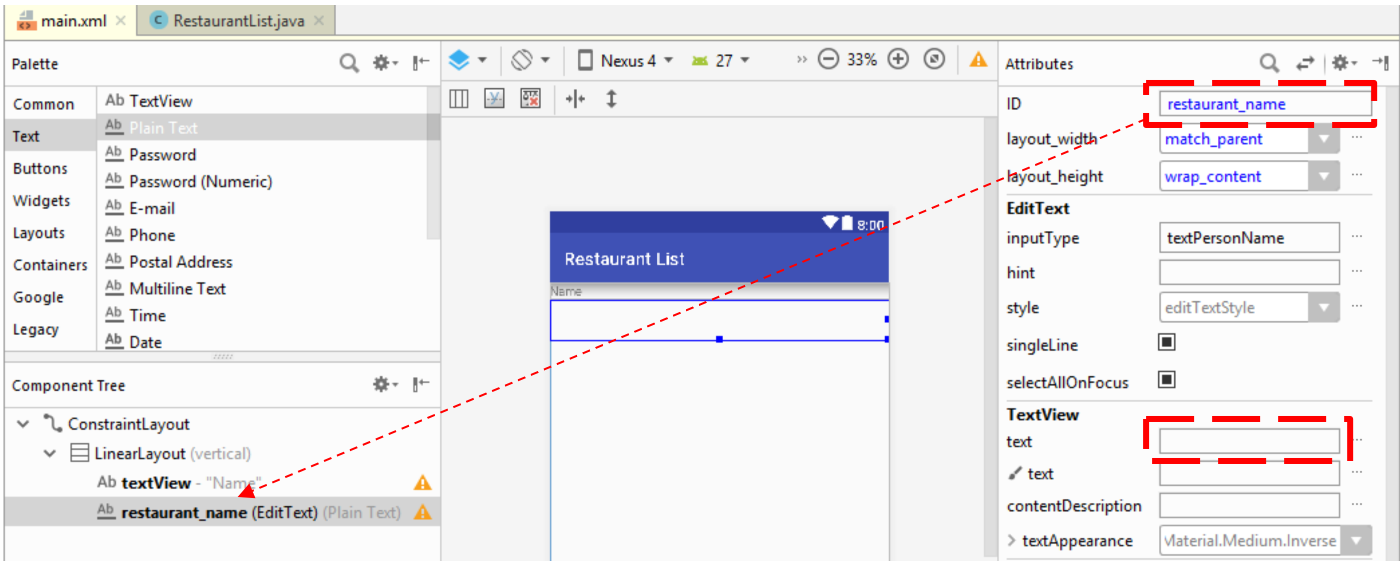
17. As can see from the *Design Editor View*, the label displayed on screen is not correct. The intention of *TextView* widget in this form is to 'show' user that the following data entry (*EditText* widget) is for 'Restaurant's Name'. To edit the values, you will need to update the 'Properties' of these widgets. To make changes on these value, you will need to click on the individual widgets and update the content of the widget's properties

18. To change the *TextView* widget to show 'Name' as label, click on the **textView widget** at the **Component Tree panel**. At the **Properties** panel (on the right hand side of the *Design Editor View*), update the *text* value to 'Name'



ID: textView
layout_width: match_parent
layout_height: wrap_content
TextView
text: Name
text:
contentDescription:
textAppearance: Material.Small
Favorite Attributes
visibility: none
LinearLayout_layout
layout_weight:

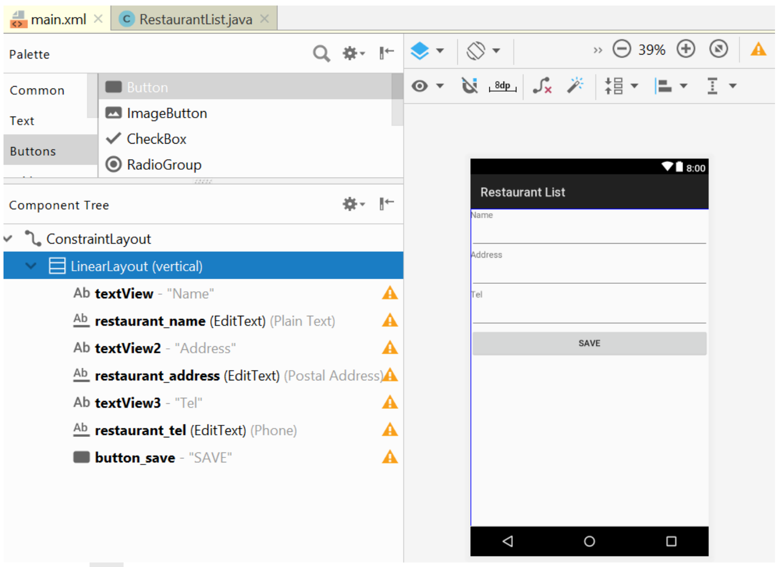
19. To change the *EditText* widget to display blank entry by default and the widget ID to '*restaurant_name*', click on the *EditText* widget at the Component Tree panel. At the Properties panel, update the *ID* to '*restaurant_name*' and clear the content of text property



ID: restaurant_name
layout_width: match_parent
layout_height: wrap_content
EditText
inputType: textPersonName
hint:
style: editTextStyle
singleLine
selectAllOnFocus
TextView
text:
text:
contentDescription:
textAppearance: Material.Medium.Inverse

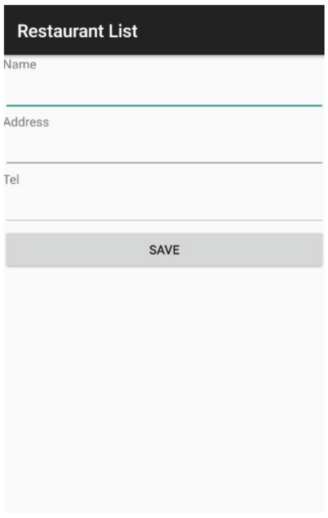
20. Repeat **Step 14 to 19** to add 5 more widgets to the **Component Tree** of *main.xml* layout
    - Two *TextView* widgets – to display as label 'Address' and 'Tel' (update *text* property)
    - Two *EditText* widgets (Postal Address and Phone) – for user to enter address and telephone number of the restaurant. Update IDs to '*reataurant_address*' and '*restaurant_tel*' respectively

- One *Button* widget – to allow user to click for some action taking. Update ID to '*button_save*' and *text* to 'SAVE'



Component Tree
- ConstraintLayout
  - LinearLayout (vertical)
    - textView - "Name"
    - restaurant_name (EditText) (Plain Text)
    - textView2 - "Address"
    - restaurant_address (EditText) (Postal Address)
    - textView3 - "Tel"
    - restaurant_tel (EditText) (Phone)
    - button_save - "SAVE"

21. Click on ▶ button at the top menu bar to run the app
22. If successfully, an emulator will be launched and the form designed will be loaded. It will take quite some time for the emulator to run. Please be patient.
23. This simple UI View allows user to enter information by clicking the respective input box and type in the text.



Extra Credit

24. If you notice, the input fields (*EditText* widgets) have no constrained on the input data type or length! For instance, at the moment, users are not stopped from entering alphabet letter as telephone number. The control can easily be done through adding extra attribute to the respective *EditText* widgets in *main.xml* layout. Please modify the properties of the following *EditText* widgets
    - **'restaurant_name'** – maximum 30 characters
    - **'restaurant_address'** – maximum 60 characters
    - **'restaurant_tel'** – maximum 8 digits number

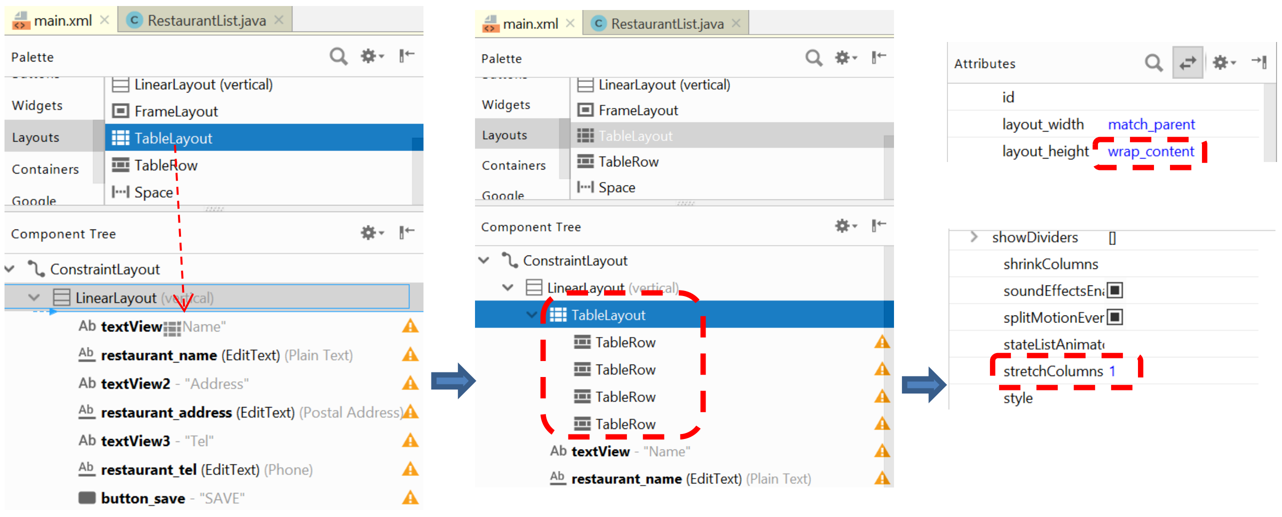
    Hint: You can edit the **Maximum Length** attribute display in **Properties** pane.

    (**Note**: You do not have to close the emulator. For every new run of your Android application, the system will overwrite the old application in the emulator)
25. Show your lecturer after you have achieved the extra credit.

Lecturer Signature : ______________________________

Part III – Creating a Fancier UI View

1. In this section of the exercise, *TableLayout* will be used to enhance the look and feel of the previous form design. The *TableLayout* allows user to assign widgets in rows and columns of a table. Here, the table will display rows of widgets in the form of a column. It looks better as comparing to *LinearLayout*
2. At **Component Tree**, drag a *TableLayout* widget and drop into *LinearLayout* as shown below. Click on *TableLayout* widget and update the **layout_height** to '*wrap_content*' and **stretchColumns** with '1' at **Properties** panel

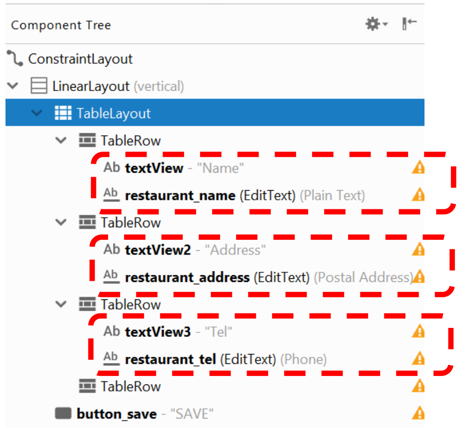


```
Component Tree
ConstraintLayout
  LinearLayout (vertical)
    Ab textView - "Name"
    restaurant_name (EditText) (Plain Text)
    Ab textView2 - "Address"
    restaurant_address (EditText) (Postal Address)
    Ab textView3 - "Tel"
    restaurant_tel (EditText) (Phone)
    button_save - "SAVE"
```

```
Component Tree
ConstraintLayout
  LinearLayout (vertical)
    TableLayout
      TableRow
      TableRow
      TableRow
      TableRow
    Ab textView - "Name"
    restaurant_name (EditText) (Plain Text)
```

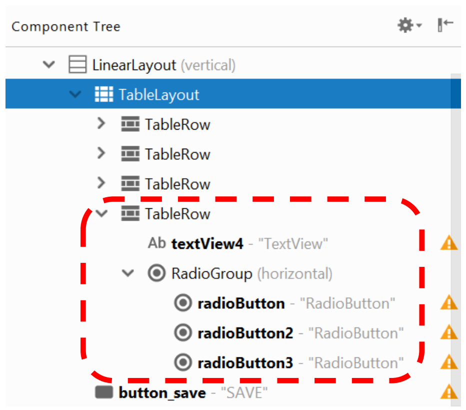
```
Attributes
id
layout_width    match_parent
layout_height   wrap_content

showDividers
shrinkColumns
soundEffectsEn
splitMotionEver
stateListAnimat
stretchColumns  1
style
```

3. Drag and drop all the *TextView* and *EditText* widgets into the *TableRow* as shown



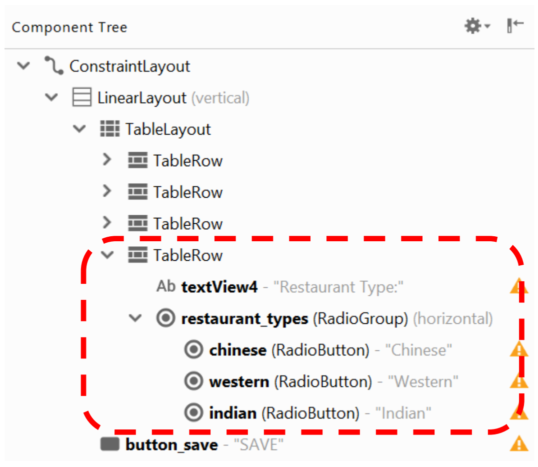
```
Component Tree
ConstraintLayout
  LinearLayout (vertical)
    TableLayout
      TableRow
        Ab textView - "Name"
        restaurant_name (EditText) (Plain Text)
      TableRow
        Ab textView2 - "Address"
        restaurant_address (EditText) (Postal Address)
      TableRow
        Ab textView3 - "Tel"
        restaurant_tel (EditText) (Phone)
      TableRow
    button_save - "SAVE"
```

4. Drag and drop **one** *TextView* widget and **one** *RadioGroup* widget (from *Buttons* option) into the forth empty *TableRow*. Drag **three** *RadioButton* widgets into the *RadioGroup* widget

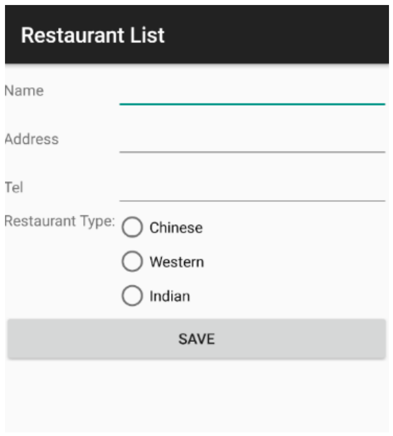


```
Component Tree
  LinearLayout (vertical)
    TableLayout
      TableRow
      TableRow
      TableRow
      TableRow
        Ab textView4 - "TextView"
        RadioGroup (horizontal)
          radioButton - "RadioButton"
          radioButton2 - "RadioButton"
          radioButton3 - "RadioButton"
    button_save - "SAVE"
```

5. Do the necessary update on the display text and IDs of *TextView*, *RadioGroup* and *RadioButton* widgets as below
   i. TextView – change ***text*** to "Restaurant Type:"
   ii. RadioGroup – change ***id*** to "**restaurant_types**"
   iii. RadioButton1 – change ***ID*** to "**chinese**" and display text to "**Chinese**"
   iv. RadioButton2 – change ***ID*** to "**western**" and display text to "**Western**"
   v. RadioButton3 – change ***ID*** to "**indian**" and display text to "**Indian**

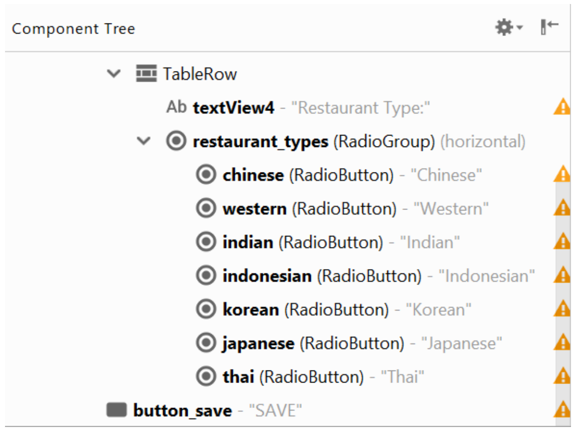

```
Component Tree
ConstraintLayout
  LinearLayout (vertical)
    TableLayout
      TableRow
      TableRow
      TableRow
      TableRow
        Ab textView4 - "Restaurant Type:"
        restaurant_types (RadioGroup) (horizontal)
          chinese (RadioButton) - "Chinese"
          western (RadioButton) - "Western"
          indian (RadioButton) - "Indian"
    button_save - "SAVE"
```

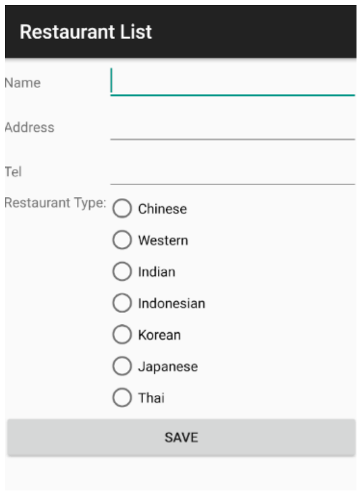
6. Run the *Lab1a* project. The emulator will load the view as designed



Extra Credit

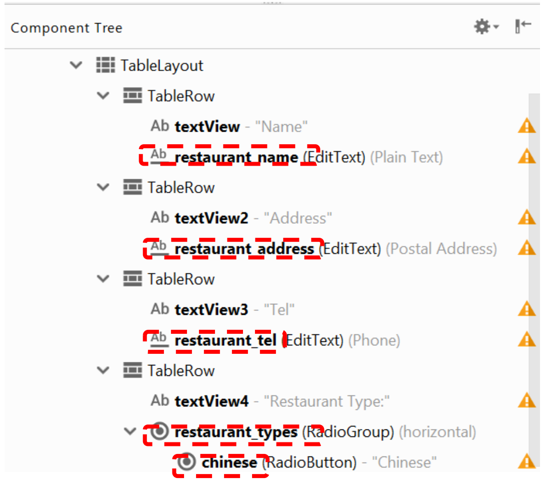
7. Add the 4 more food types (Indonesia, Korean, Japanese and Thai) to the existing *RadioGroup* widget. Update the ID and display text for the new food types. Save and run the app


```
Component Tree
TableRow
  Ab textView4 - "Restaurant Type:"
  restaurant_types (RadioGroup) (horizontal)
    chinese (RadioButton) - "Chinese"
    western (RadioButton) - "Western"
    indian (RadioButton) - "Indian"
    indonesian (RadioButton) - "Indonesian"
    korean (RadioButton) - "Korean"
    japanese (RadioButton) - "Japanese"
    thai (RadioButton) - "Thai"
button_save - "SAVE"
```



Part IV – Getting Data from Form & Detecting Button Click

8. We have created a form UI for user to enter Restaurant data. The subsequence steps will demonstrate how program can response to "Save" button click and read data from *EditText* widget as well as selection of *RadioGroup*
9. **Each** of the **widgets** created in *main.xml* Layout have an **unique ID**



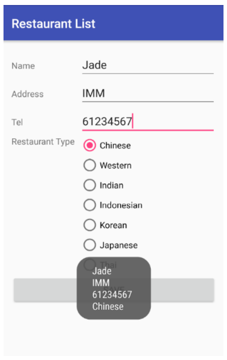
Component Tree
- TableLayout
  - TableRow
    - Ab **textView** - "Name"
    - Ab **restaurant_name** (EditText) (Plain Text)
  - TableRow
    - Ab **textView2** - "Address"
    - Ab **restaurant_address** (EditText) (Postal Address)
  - TableRow
    - Ab **textView3** - "Tel"
    - Ab **restaurant_tel** (EditText) (Phone)
  - TableRow
    - Ab **textView4** - "Restaurant Type:"
    - **restaurant_types** (RadioGroup) (horizontal)
      - **chinese** (RadioButton) - "Chinese"

10. In order for program (Controller) to have accessed to all these widgets, the program must have a variable to bind to them
11. For example, if the Controller needs to access the data entered at the "*restaurant_name*" *EditText* widget
    i. Declare an *EditText* variable with name as '*restaurantName*':
       `private EditText restaurantName;`
    ii. Bind the '*restaurantName*' **EditText** variable created with the **EditText** widget '*restaurant_name*':
       `restaurantName = (EditText)findViewById(R.id.restaurant_name);`
    iii. Use the *getText()* method provided by the *EditText* widget to read the data from the form UI
       `restaurantName.getText().toString();`
12. To detect a *Button* click event we will need to declare a **Button** variable and bind it to the **Button** widget '*button_save*'. To response to a click event we will bind an Onclick Event Listener to the button to listen to any clicking event generated by a *Button* in a form UI
    i. Declare a *Button* variable with name as '*buttonSave*':
       `private Button buttonSave;`
    ii. Bind the '*buttonSave*' Button variable created with the Button widget '*button_save*':
       `buttonSave = (Button)findViewById(R.id.button_save);`
    iii. Register the '*buttonSave*' button to an Event Listener. When '*Save*' Button is clicked on View, the program will jump to the **onSave** View Listener object where we can start to read the data from the form UI
       `buttonSave.setOnClickListener(onSave);`
13. Update the *RestaurantList* Activity class with the following content.

```
package com.sp.restaurantlist;
import android.support.v7.app.AppCompatActivity;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioGroup;
import android.widget.Toast;

public class RestaurantList extends AppCompatActivity {
    private EditText restaurantName;
    private RadioGroup restaurantTypes;
    private Button buttonSave;

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

        restaurantName = (EditText) findViewById(R.id.restaurant_name);
        restaurantTypes = (RadioGroup) findViewById(R.id.restaurant_types);

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

    private View.OnClickListener onSave = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // To read data from restaurantName EditText
            String nameStr = restaurantName.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;
            }
            String combineStr = nameStr + "\n" + restType;
            Toast.makeText(v.getContext(), combineStr, Toast.LENGTH_LONG).show();
        }
    };
}
```

Extra Credit

14. Complete the *RestaurantList* Activity class to include reading data from address and telephone *EditText* widgets and display the result using **Toast**
15. Show your lecturer after you have achieved the extra credit.


Restaurant List

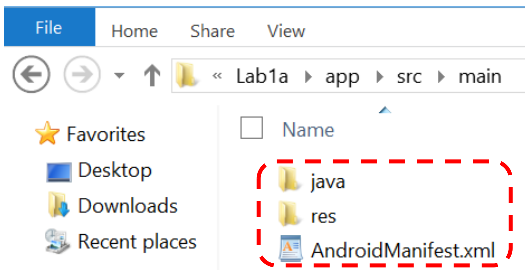
Name Jade
Address IMM
Tel 61234567
Restaurant Type
● Chinese
○ Western
○ Indian
○ Indonesian
○ Korean
○ Japanese

Jade
IMM
61234567
Chinese

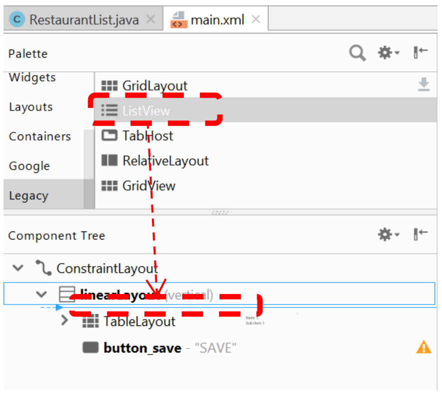
Lecturer Signature : ____________________________________

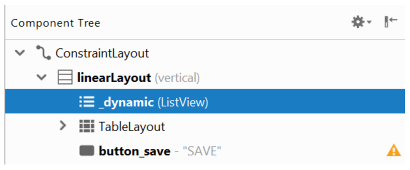
Part V – Creating a Form with List and Model Class to Hold Restaurant Data

16. In this part of the exercise, List and Model class will be added
17. Create a new project *Lab1b* with the following information:
    - **Application Name** : *Restaurant List*
    - **Company Domain** : *sp.com*
    - **Project Location** : *C:\MAD\AndroidStudioProjects\Lab1b or D:\MAD\AndroidStudioProjects\Lab1b*
    - **Minimum SDK** : *Android 5.0 (Lollipop)*
    - **Empty Activity name** : *RestaurantList*
    - **Layout name** : *main*
18. Close the *main.xml* and *RestaurantList.java* files in the **Editor** pane
19. Open Windows File Explorer and navigate to your Android Studio workspace (*C:\MAD\AndroidStudioProjects* or *D:\MAD\AndroidStudioProjects*) where all your projects are created and saved
20. Double click to open the *Lab1a* project folder and navigate down to "**app\src\main**" folder. Copy **AndroidManifest.xml** file, **java** and **res** folders

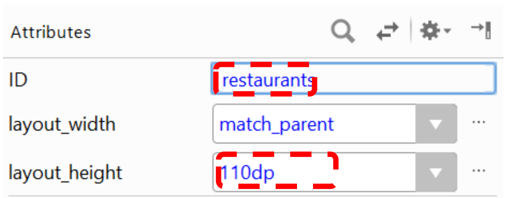


21. Go to newly created project "*Lab1b\app\src\main*" folder and paste into it to overwrite existing file and folders
22. Go back to Android Studio. Open the *RestaurantList.java* and *main.xml* files and they should show the content from *Lab1a*
23. To enhance the *Restaurant List* app, a *ListView* widget will be added to the UI View to display a list of the restaurants information entered
24. Open the *main.xml* at Design pane. Drag a *ListView* widget from **Legacy** option and drop into the **LinearLayout**.



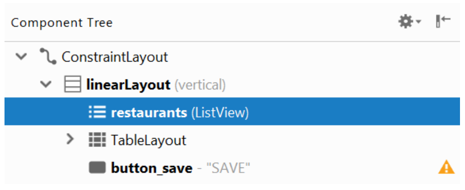


25. Update the *ListView* ***ID*** to "**restaurants**" and set the *layout_height* to **110dp**



Attributes

ID	restaurants
layout_width	match_parent
layout_height	110dp

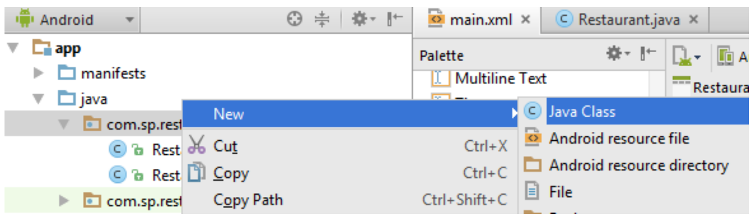


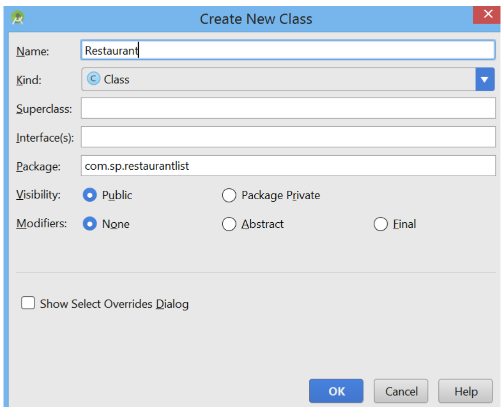
Component Tree

- ConstraintLayout
  - linearLayout (vertical)
    - restaurants (ListView)
    - TableLayout
    - button_save - "SAVE"

26. With the *ListView* widget created, every new set of restaurant information will be treated as a record (which contains restaurantName, restaurantAddress, restaurantTel and restaurantTypes widgets data)
27. Individual restaurant's record will be stored using *Restaurant* **Data Model** and added to **ArrayList**
28. Data entered in the form is collected and kept in memory using **ArrayList**
29. To display all the records collected in *ArrayList* to *ListView* on the form, an **ArrayAdapter** (acts as Controller) will update the *ListView* whenever a new restaurant record is added to the **ArrayAdapter** (update both **ArrayList** and *ListView* automatically)
30. To create the Restaurant Data Model class, right click on **com.sp.restaurantlist** folder and select **New > Java Class**

31. Enter the Java Class name as *Restaurant* and click **OK**





Create New Class

Name:	Restaurant
Kind:	Class
Superclass:	
Interface(s):	
Package:	com.sp.restaurantlist
Visibility:	[✓ Public] Package Private
Modifiers:	[✓ None] Abstract Final

☐ Show Select Overrides Dialog

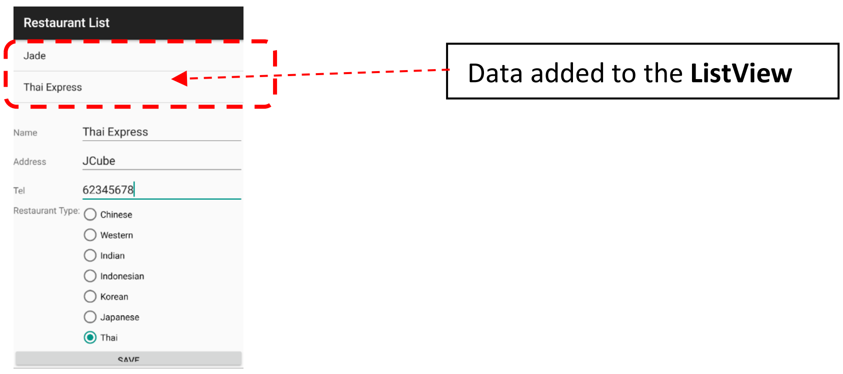
32. Complete the *Restaurant.java* class with the following code and save

```
package com.sp.restaurantlist;

public class Restaurant {
    private String restaurantName = "";
    private String restaurantAddress = "";
    private String restaurantTel = "";
    private String restaurantType = "";

    public String getName() {
        return (restaurantName);
    }

    public void setName(String restaurantName) {
        this.restaurantName = restaurantName;
    }

    public String getAddress() {
        return (restaurantAddress);
    }

    public void setAddress(String restaurantAddress) {
        this.restaurantAddress = restaurantAddress;
    }

    public String getTelephone() {
        return (restaurantTel);
    }

    public void setTelephone(String restaurantTel) {
        this.restaurantTel = restaurantTel;
    }

    public String getRestaurantType() {
        return (restaurantType);
    }

    public void setRestaurantType(String restaurantType) {
        this.restaurantType = restaurantType;
    }

    public String toString() {
        return (getName());
    }
}
```

33. Implement the Controller (**ArrayAdapter**), **ArrayList**, Restaurant Data Model and *ListView* in the *RestaurantList* Activity class and save

```
package com.sp.restaurantlist;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.RadioGroup;

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

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

    private List<Restaurant> model = new ArrayList<Restaurant>();
    private ArrayAdapter<Restaurant> adapter = null;
    private ListView list;

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

        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);

        list = (ListView) findViewById(R.id.restaurants);
        adapter = new ArrayAdapter<Restaurant>(this, android.R.layout.simple_list_item_1, model);
        list.setAdapter(adapter);
    }

    private View.OnClickListener onSave = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // To read date from name 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;
            }
            String combineStr = nameStr + "\n" + restType + "\n" + addrStr + "\n" + telStr;
            //Toast.makeText(v.getContext(), combineStr, Toast.LENGTH_LONG).show();
            //Create Restaurant Data Model
            Restaurant restaurant = new Restaurant();
            //Update the Restaurant Data Model
            restaurant.setName(nameStr);
            restaurant.setAddress(addrStr);
            restaurant.setTelephone(telStr);
            restaurant.setRestaurantType(restType);
            //Pass the record to ArrayAdapter.
            //It will update the ListArray and the ListView
            adapter.add(restaurant);
        }
    };
 }
```

34. Run the *Lab1b* project. The **ListView** will remain empty until we do something to populate it
35. Enter some test data and click the **Save** button



Restaurant List

Jade

Thai Express

Name	Thai Express
Address	JCube
Tel	62345678
Restaurant Type:	○ Chinese
	○ Western
	○ Indian
	○ Indonesian
	○ Korean
	○ Japanese
	● Thai

SAVE

36. Show to your lecturer after you have completed the exercise.

Lecturer Signature : ____

-END-