Assignment: ExpandableListAdapter.

For ExpandableListAdapter, we mainly need 2 java files and 3 xml files. To create a suitable adapter we have to extend BaseExpandableListAdapter.

There are two methods that are used to create the View corresponding to our layout, one for the items (childs) and one for the groups (parents).
These methods are:

- `public View getChildView(int groupPosition, final int childPosition, boolean isLastChild, View convertView, ViewGroup parent)` used when we need to create a child view.
- `public View getGroupView(int groupPosition, boolean isExpanded, View convertView, ViewGroup parent)` used when we need to create a parent view.

public view getChildView (int groupPosition, int childPosition, Boolean isLastChild, View convertView, ViewGroup parent)

Parameters

groupPosition	the position of the group that contains the child
childPosition	the position of the child (for which the View is returned) within the group
isLastChild	Whether the child is the last child within the group
convertView	The old view to reuse, if possible. You should check that this view is non-null and of an appropriate type before using. If it is not possible to convert this view to display the correct data, this method can create a new view. It is not guaranteed that the convertView will have been previously created by getChildView(int, int, boolean, View, ViewGroup).
Parent	the parent that this view will eventually be attached to

Returns

- the View corresponding to the child at the specified position

public View getGroupView (int groupPosition, boolean isExpanded, View convertView, ViewGroup parent)

Parameters

groupPosition	the position of the group for which the View is returned
isExpanded	whether the group is expanded or collapsed
convertView	the old view to reuse, if possible. You should check that this view is non-null and of an appropriate type before using. If it is not possible to convert this view to display the correct data, this method can create a new view. It is not guaranteed that the convertView will have been previously created by getGroupView(int, boolean, View, ViewGroup).
Parent	the parent that this view will eventually be attached to

Returns

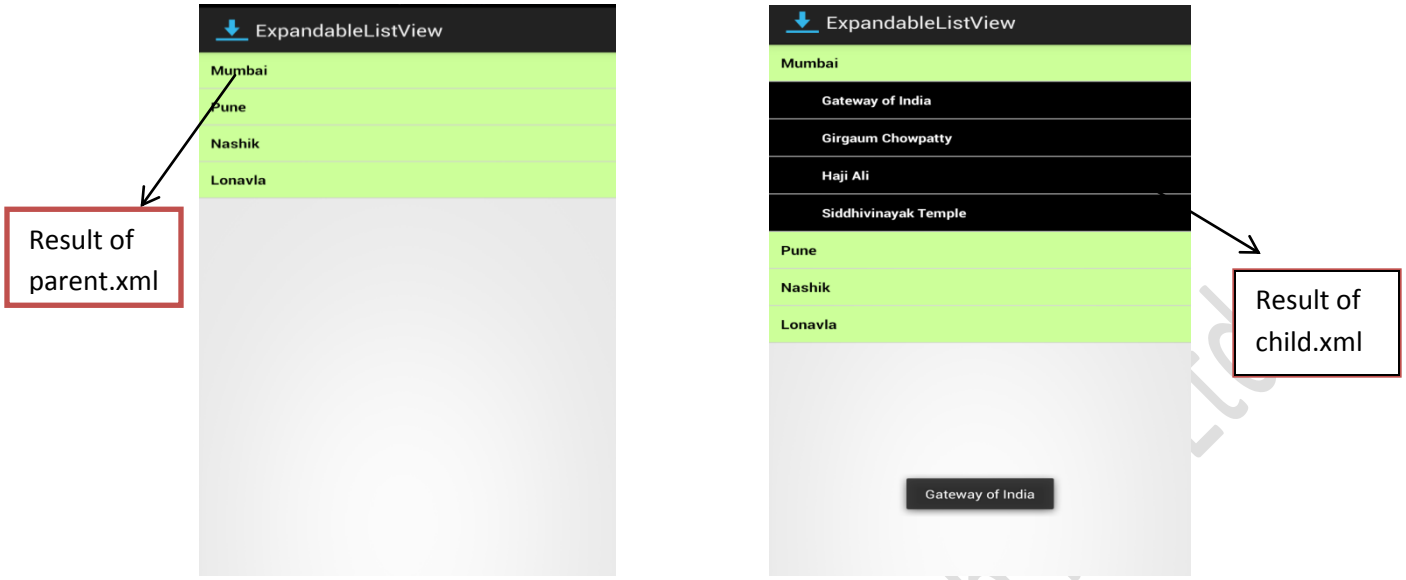
- the View corresponding to the group at the specified position

Java files:

- MainActivity.java
- MyExpandableAdapter.java

Xml files:

- activity_main.xml
- parent.xml
- child.xml



As it is shown above, we need two different layouts: one for the group and another one for items. So it is bit more complex than a listView even if the concept behind this component it is the same. We need to develop an adapter and two layouts.

1. activity_main.xml:
   This is a simple xml file consisting of `ExpandableListView` within the linear layoutas shown below:

```
<ExpandableListView
android:id="@+id/list"
android:layout_height="match_parent"
android:layout_width="match_parent"
android:groupIndicator="@null"/>
```

2. parent.xml :
   It consists of a Linear layout containing following attributes:

```
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="40dp"
android:background="@android:color/black"
android:clickable="true"
android:orientation="vertical"
android:paddingLeft="40dp"
tools:context=".MainActivity">
</LinearLayout>
```

Linear layout consists of a normal text view consisting of attribute textIsSelectable and set it to True.

```
<TextView
android:id="@+id/textView1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="5dp"
android:textIsSelectable="true"
android:textColor="#FFFFFF"
android:textSize="14sp"
android:textStyle="bold"/>
```

3. child.xml:
   It consists of aCheckedTextView without any Layout as it is to be displayed under a ListView.

```
<CheckedTextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="90dip"
            android:checkMark="?android:attr/listChoiceIndicatorMultiple"
            android:clickable="true"
            android:checked="true"
            android:gravity="center"
            android:hint="Checked"/>
```

We have made things really simple. Now we have our layouts we need to create an adapter to handle items and groups and bind them together.

4. MyExpandableAdapter.java:
   This class should extend BaseExpandableListAdapter.
   We'll first focus on getChildView(), here is the code snippet:

```
public View getChildView(intgroupPosition, finalintchildPosition, booleanisLastChild,
View convertView, ViewGroup parent) {

            child = (ArrayList<String>) childtems.get(groupPosition);

            TextViewtextView = null;

            if (convertView == null) {
                  convertView = inflater.inflate(R.layout.group, null);
            }

            textView = (TextView) convertView.findViewById(R.id.textView1);
            textView.setText(child.get(childPosition));

            convertView.setOnClickListener(newOnClickListener() {
```

```
                @Override
                publicvoidonClick(View view) {
                      Toast.makeText(activity, child.get(childPosition),
                                  Toast.LENGTH_SHORT).show();
                }
          });
returnconvertView;
               }
```

As we can notice we get position of the child from from`child.get(childPosition)` and display it as toast.

When we have to create a group View corresponding to our Category class we simply do the same thing, like shown below:

```
public View getGroupView(intgroupPosition, booleanisExpanded, View convertView,
ViewGroup parent) {

          if (convertView == null) {
                convertView = inflater.inflate(R.layout.row, null);
          }

          ((CheckedTextView) convertView).setText(parentItems.get(groupPosition));
          ((CheckedTextView) convertView).setChecked(isExpanded);

          return convertView;
     }
```

- In the method overridden, we should add the following code to obtain the size or count of the parents and child element.

```
@Override
     public int getChildrenCount(int groupPosition) {
          return((ArrayList<String>) childtems.get(groupPosition)).size();
     }



@Override
     public int getGroupCount() {
          return parentItems.size();
     }
```

5. MainActivity.java

This activity extends ExpandableListActivityand is responsible to create our ExpandableListView:(Refer following code snippet)

```
     public void onCreate(Bundle savedInstanceState) {

          super.onCreate(savedInstanceState);
```

```
ExpandableListViewexpandableList=(ExpandableListView) findViewById(R.id.list);

              expandableList.setDividerHeight(2);
              expandableList.setGroupIndicator(null);
              expandableList.setClickable(true);

              setGroupParents();
              setChildData();

              MyExpandableAdapter adapter = newMyExpandableAdapter(parentItems,
childItems);

adapter.setInflater((LayoutInflater)getSystemService(Context.LAYOUT_INFLATER_SERVICE)
, this);
              expandableList.setAdapter(adapter);
              expandableList.setOnChildClickListener(this);
       }
```

In the code above we simply get the ExpandableListView widget reference in our main layout and create the Adapter.

- Code snippet to generate GroupCategory:

```
publicvoidsetGroupParents() {
              parentItems.add("Mumbai");
              parentItems.add("Pune");
              parentItems.add("Nashik");
              parentItems.add("Lonavla"); }
```

Code snippet to generate ItemCategory:

```
publicvoidsetChildData() {

// Mumbai
              ArrayList<String> child = newArrayList<String>();
              child.add("Gateway of India");
              child.add("GirgaumChowpatty");
              child.add("Haji Ali");
              child.add("Siddhivinayak Temple");
              childItems.add(child);


// Nashik
              child = newArrayList<String>();
              child.add("Pandavleni Caves");
              child.add("Muktidham");
              child.add("Sula Vineyards");
              childItems.add(child);

}
```

Similarly, add various places in the city as done for Mumbai inside `setChildData()`.
Note : `setGroupParents()` and`setChildData()`should be coded under onCreate() of MainActivity.