CSCI 4100 Laboratory 6

Class List Application

Introduction

In this laboratory we will build the infamous class list application in Android. In the process we will learn about list views and toast. List views in Android are similar to table views in iOS. Toasts put a notification on the screen for a short period of time.

List Views

A list view is a subclass of AdapterView. The other common subclass of AdapterView is the grid view. All subclasses of AdapterView use an adapter to provide the data for the view. In the case of the list view we have a vertical list of cells, where each cell is a separate view. The adapter provides the views that go in the cells. Each adapter has a data source that provides the data for the view. You can write your own adapter class or you can use one of the standard Android adapter classes. The two most common standard adapter classes are ArrayAdapter and SimpleCursorAdapter. ArrayAdapter is used when the data is in an array and it converts each array entry into a string (using its toString() method) and places the result in a TextView widget. The SimpleCursorAdapter is used when the data is stored in a database or provided by a content provider. In our application we have an array of class names so the ArrayAdapter will be used.

The constructor for the ArrayAdapter class has three parameters:

- The application context, which is the current activity.
- The layout to be used for the cell. This layout must contain a TextView widget.
- The array containing the data.

This class is parameterized based on the type of data stored in the array. Since we have an array of Strings we use ArrayAdapter<String>. Once the adapter has been created we add it to the ListView by calling its setAdapter method.

We are now ready to start building our class list application. The resulting application should look like figure 1. Start by creating a new Android application, like you have done in the other labs. You can call the application ClassList, otherwise there is nothing special about this application.

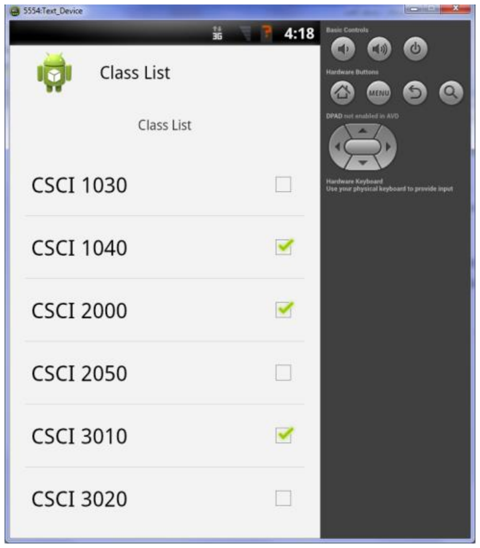


Figure 1 Class list application

Our data is stored in an array of strings so add the following to the start of the main activity:

```
private final String[] courses = {"CSCI 1030", "CSCI 1040", "CSCI 2000",
                    "CSCI 2050", "CSCI 3010", "CSCI 3020", "CSCI 3040", "CSCI 3055",
                    "CSCI 3090", "CSCI 4100", "CSCI 4160"};
```

In the onCreate() method we load the initial user interface in the same way that we always do, but in addition we need to create the adapter and add it to the list view:

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

      ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                   android.R.layout.simple_list_item_multiple_choice, courses);

      ListView listView = (ListView) findViewById(R.id.listView1);
      listView.setAdapter(adapter);

      listView.setOnItemClickListener(myMessageClickedHandler);

}
```

The first parameter to the array adapter constructor is the current activity. Android provides several standard layouts for list view cells. In our case we are going to use a layout that gives us a check box on the right of each cell and allows for multiple selections. We could have constructed our own layout in interface builder, but since the standard layout does everything we need we might as well use it. The next two lines find the list view in the layout and then set its adapter to the new adapter we have just created. The final line of this method sets a listener method that will be called each time one the list view cells is selected. For now this method will just toggle the check mark. The code for this method is:

```
private OnItemClickListener myMessageClickedHandler = new OnItemClickListener() {
        public void onItemClick(AdapterView parent, View v, int position, long id) {
              CheckedTextView cell = (CheckedTextView) v;
              cell.toggle();
        }
};
```

OnItemClickListener is an interface that must be implemented by classes that respond to the selection of cells. This interface has one required method, which is onItemClick. The four parameters to this method are:

- The parent of the cell where the click occurred. In our case this is the list view widget
- The view where the click occurred. This is the list view cell.
- The position of the view in the adapter.
- The row id of the item that was clicked.

The layout that we have selected uses a CheckedTextView widget for the cells in the list view. We cast the second parameter to this type and then call the toggle() method on it to toggle its state.

Before we can run the application we need to design our screen layout. Our layout consists of two widgets, a text view and a list view. The text view has the string “Class List” centered at the top of the screen. The list view is placed below it. For the list view we must make sure that the Choice Mode property is set to multipleChoice. This is shown in figure 2.

We now have the first part of the application running. Try out your code and check that is works okay.

Toast

A toast is a small notification that appears on the screen for a short period of time. They are used to provide feedback to the user or notify them of an external event. By default a toast is placed near the bottom of the screen. The user can continue to interact with the application while the toast is on the screen. We will add a simple toast to our application as shown in figure 3. This toast will appear when we click the Android image in the action bar.

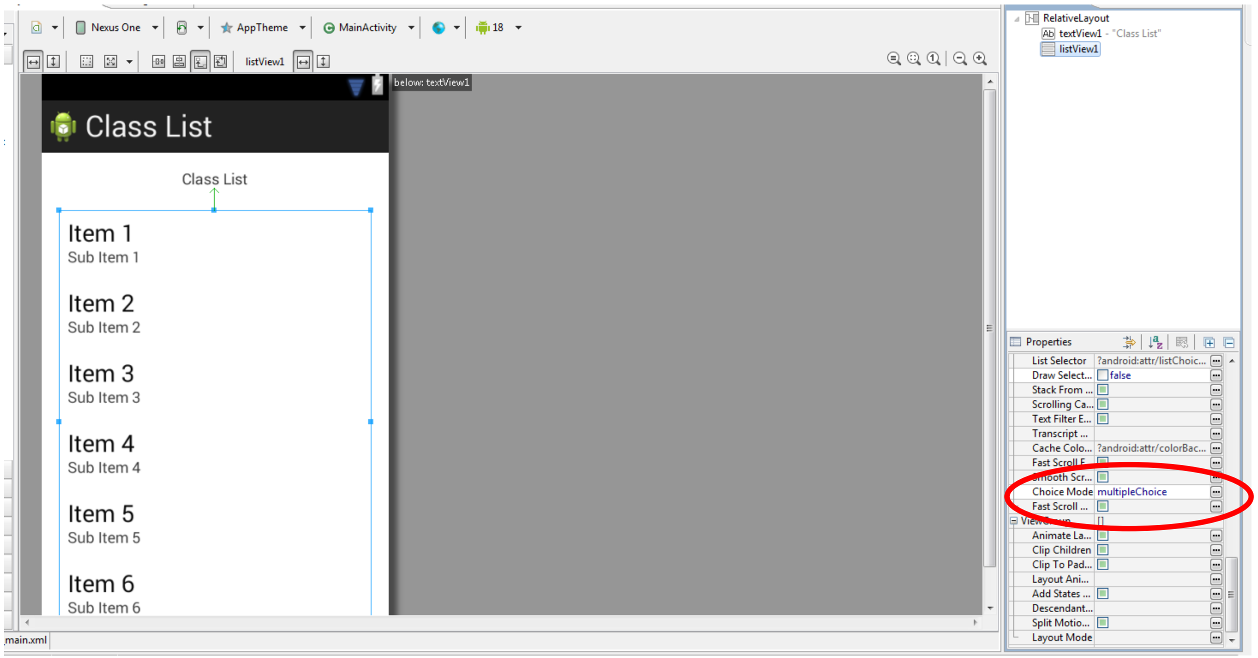
The onCreateOptionsMenu() method is provided for us when we created the new activity. This provides us with basically an empty action menu. We now need to provide an onOptionsItemSelected method that will create a toast when the options bar item is selected. Normally this method would check to see which menu item has been selected, but in this case we don’t have any items, so we will just create the toast. The code for this method is:

```java
public boolean onOptionsItemSelected(MenuItem item) {
	ListView listView = (ListView) findViewById(R.id.listView1);
	Toast toast = Toast.makeText(this,
		String.valueOf(listView.getCheckedItemCount()),
		Toast.LENGTH_LONG);
	toast.show();
	return true;
}
```

This method first finds the list view. The getCheckedItemCount() method in list view returns the number of items that have been selected. This will be the contents of our toast. The makeText method in the Toast class is used to create a new toast with a text string. The three parameters to this method are:

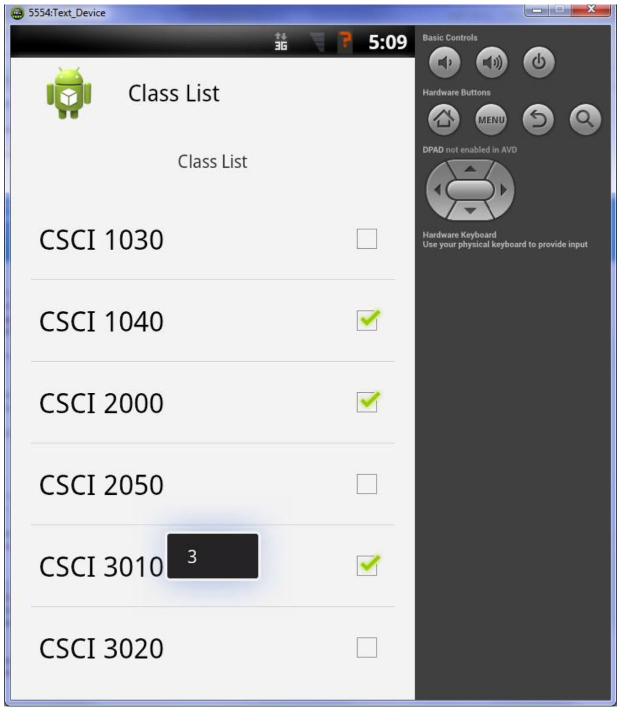
- The context, which in this case is the current activity.
- The string to put in the toast.
- The length of time to display the toast

Once the toast has been created it can be placed on the screen by calling its show() method. There are a few other things that a toast can do, which you will explore in a few minutes. Try running the new version of the application with the toast to make sure that it is working correctly.



List Selector	?android:attr/listChoic...
Draw Select...	false
Stack From ...	
Scrolling Ca...	
Text Filter E...	
Transcript ...	
Cache Colo...	?android:attr/colorBac...
Fast Scroll E...	
Smooth Scr...	
Choice Mode	multipleChoice
Fast Scroll ...	
ViewGroup	[]
Animate La...	
Clip Children	
Clip To Pad...	
Layout Ani...	
Add States ...	
Descendant...	
Split Motio...	
Layout Mode	

Figure 2 Layout for class list application



Class List

Class List

- CSCI 1030 ☐
- CSCI 1040 ✓
- CSCI 2000 ✓
- CSCI 2050 ☐
- CSCI 3010 ✓
- CSCI 3020 ☐

3

Figure 3 Toast

Laboratory Report

It would be nice if when we selected a course a toast came up with a short description of the course over top of the item that had been selected. This would involve a lot of typing to enter the course description, so instead we will just use the course name. So add a toast in the OnItemClickListener that displays the name of the course over top of the cell for the course when it's selected. The cell.isChecked() method will tell you if an item is checked. The getLeft() and getTop() return the position of a view (remember the cell is a subclass of view). The toast API Guide will provide you with the rest of the information you need to complete this task (http://developer.android.com/guide/topics/ui/notifiers/toasts.html). When you are finished show the result to the TA.

Notes:

1. The toast is positioned in screen coordinates, but the position of the cell is within its parent, and the parent is within the layout. Remember that the action bar takes some space at the top of the screen.
2. The OnItemClickListener has its own this variable, which is not the same as the activity's one. You need the activity one, which you can get by parent.getContext().