List in Dialog

```
public void onItemCheckedStateChanged(ActionMode mode, int position,
        long id, boolean checked) {
    int count = mList.getCheckedItemCount();
    mode.setTitle(String.format("%d Selected", count));
}
```

To use our ListView to activate a multiple selection ActionMode, we set its choiceMode attribute to CHOICE_MODE_MULTIPLE_MODAL. This is different from the traditional CHOICE_MODE_MULTIPLE, which will provide selection widgets on each list item to make the selection. The modal flag applies this selection mode only while an ActionMode is active.

A series of callbacks are required to implement an ActionMode that are not built directly into an activity like the ContextMenu. We need to implement the ActionMode.Callback interface to respond to the events of creating the menu and selecting options. ListView has a special interface called MultiChoiceModeListener, a subinterface of ActionMode.Callback, which we implement in the example.

In onCreateActionMode(), we respond similarly to onCreateContextMenu(), just inflating our menu options for the overlay to display. Your menu does not need to contain icons; ActionMode can display the item names instead. The onItemCheckedStateChanged() method is where we will get feedback for each item selection. Here, we use that change to update the title of ActionMode to display the number of items that are currently selected.

The onActionItemClicked() method will be called when the user has finished making selections and taps an option item. Because there are multiple items to work on, we go back to the list to get all the items selected with getCheckedItemPositions() so we can apply the selected operation.

2-6. Displaying a User Dialog Box

Problem

You need to display a simple pop-up dialog box to the user to either notify of an event or present a list of selections.

Solution

(API Level 1)

AlertDialog is the most efficient method of displaying important modal information to your user quickly. The content it displays is easy to customize, and the framework provides a convenient AlertDialog.Builder class to construct a pop-up quickly.

How It Works

When you use AlertDialog.Builder, you can construct a similar alert dialog box but with some additional options. AlertDialog is a versatile class for creating simple pop-ups to get feedback from the user. With AlertDialog.Builder, a single or multichoice list, buttons, and a message string can all be easily added into one compact widget.

To illustrate this, let's create the same pop-up selection as before by using AlertDialog. This time, we will add a Cancel button to the bottom of the options list (see Listing 2-21).

Listing 2-21. Action Menu Using `AlertDialog`

```
public class DialogActivity extends Activity implements
        DialogInterface.OnClickListener,
        View.OnClickListener {

    private static final String[] ZONES = {
            "Pacific Time", "Mountain Time",
            "Central Time", "Eastern Time",
            "Atlantic Time"};

    Button mButton;
    AlertDialog mActions;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        mButton = new Button(this);
        mButton.setText("Click for Time Zones");
        mButton.setOnClickListener(this);

        AlertDialog.Builder builder =
                new AlertDialog.Builder(this);
        builder.setTitle("Select Time Zone");
        builder.setItems(ZONES, this);
        //Cancel action does nothing but dismiss, we could
        // add another listener here to do something extra
        // when the user hits the Cancel button
        builder.setNegativeButton("Cancel", null);
        mActions = builder.create();

        setContentView(mButton);
    }

    //List selection action handled here
    @Override
    public void onClick(DialogInterface dialog, int which) {
        String selected = ZONES[which];
        mButton.setText(selected);
    }

    //Button action handled here (pop up the dialog)
    @Override
    public void onClick(View v) {
        mActions.show();
    }
}
```

In this example, we create a new `AlertDialog.Builder` instance and use its convenience methods to add the following items:

- A title, using `setTitle()`
- The selectable list of options, using `setItems()` with an array of strings (also works with array resources)
- A Cancel button, using `setNegativeButton()`

The listener that we attach to the list items returns which list item was selected as a zero-based index into the array we supplied, so we use that information to update the text of the button with the user's selection. We pass in `null` for the Cancel button's listener, because in this instance we just want Cancel to dismiss the dialog. If there is some important work to be done upon pressing Cancel, another listener could be passed in to the `setNegativeButton()` method.

The builder provides several other options for you to set the content of the dialog to something other than a selectable list:

- `setMessage()` applies a simple text message as the body content.
- `setSingleChoiceItems()` and `setMultiChoiceItems()` create a list similar to this example but with selection modes applied so that the items will appear as being selected.
- `setView()` applies any arbitrary custom view as the dialog's content.

The resulting application looks like Figure 2-4 when the button is pressed.

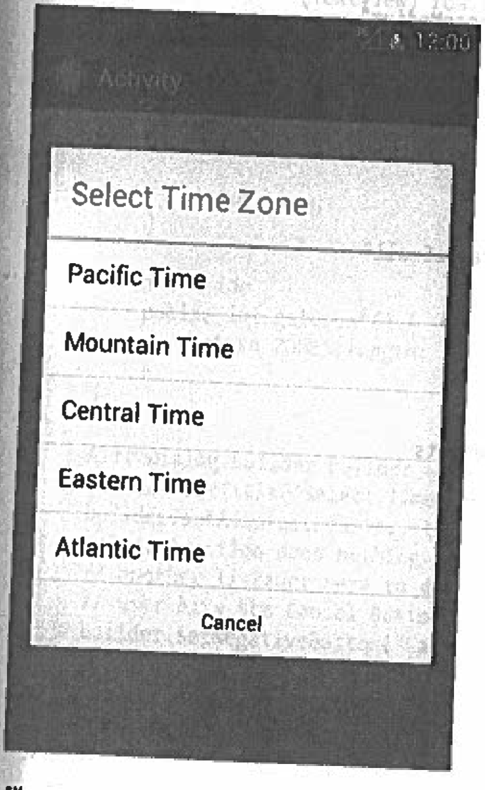

Figure 2-4. Alert dialog box with items list