Android Dialogs

Dialogs are simple visual objects that pop up and display a message or prompt for some user input.

Create a new Android project with a suitable name and package. To begin with it will have a Button to pick up events and a TextView for output

Toast

A simple way to get a message to the user is with a Toast. This is a message that appears on the screen and soon fades away. It does not interrupt the running program but there is no way to capture any user response from a Toast.

The following import is required:

```
import android.widget.Toast;
```

A Toast can be created and displayed in one, rather long, line.

```
Toast.makeText(this, "A toast, cheers",
Toast.LENGTH_LONG).show();
```

AlertDialog

This is a simple dialog that can be customised with extra functions. Out of the box the AlertDialog has no Buttons so it will grab control of the application with no way of closing itself.

These imports are going to be needed:

```
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
```

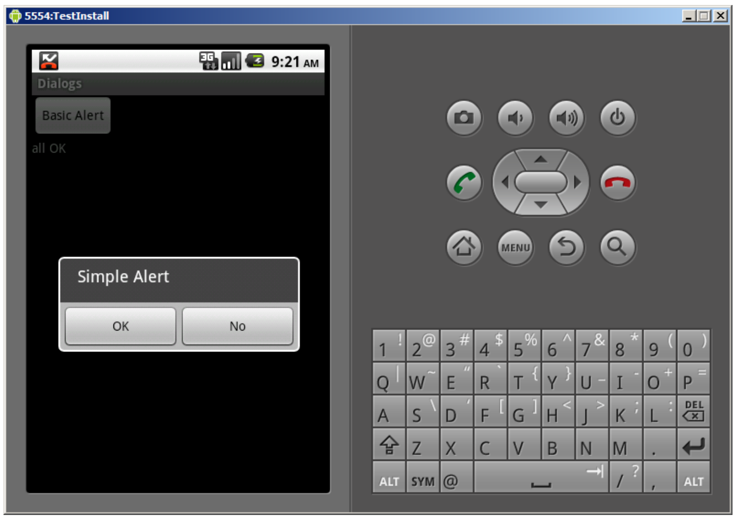
An AlertDialog can be declared and instantiated like a Button.

```
AlertDialog.Builder builder;

builder = new AlertDialog.Builder(this);
builder.setMessage("Simple Alert")
```

Here the onClick () method of the Button is used to show the AlertDialog:

```
public void onClick(View v) {
      vOut.setText("all OK");
      builder.show();
}
```

This code can work but the results are disappointing:



Additional parameters of the AlertDialog can be used to add Buttons and have these do something the cancel method is a good place to start. Note that 2 methods have been passed to the builder object with the dot operator. There is no semi colon between the 1st and 2nd lines below. To the code they are on the same line.

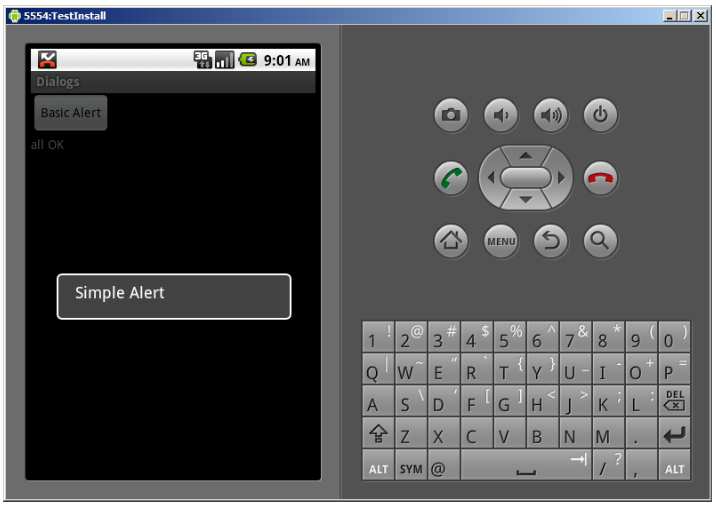
```
builder.setMessage("Simple Alert")
.setPositiveButton("OK", new DialogInterface.OnClickListener()
        {
          public void onClick(DialogInterface dialog, int id)
          {
               vOut.setText("OK pressed");
               dialog.cancel();
          }
        });
```

A second Button can be added with its own action:

```
builder.setMessage("Simple Alert")
.setPositiveButton("OK", new DialogInterface.OnClickListener()
        {
          public void onClick(DialogInterface dialog, int id)
          {
               vOut.setText("OK pressed");
               dialog.cancel();
          }
        })
.setNegativeButton("No", new DialogInterface.OnClickListener()
        {
          public void onClick(DialogInterface dialog, int id)
          {
               vOut.setText("No pressed");
               dialog.cancel();
          }
        });
```

The results are now more useful.

There are only 3 possible buttons



- PositiveButton
- NegativeButton
- NeutralButton

Handling more than 1 click event

In the application above there is only a single Button so there will only be 1 case of a click. With more than 1 Button or other control that requires clicking there needs to be a way of knowing which has been pressed.

There is more than 1 solution to this problem. This solution uses an if statement within `public void onClick(View v` to pick up which object has been pressed. Note that the name used is that from the XML design not that given to the control in the java code.

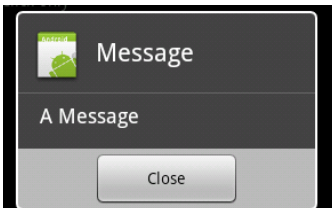
```
if (v == findViewById(R.id.bAlert))

{
     vOut.setText("Alert so far");

}
```

So far nothing exciting has been achieved. The program will work as before. We can move up a notch by adding a dangerous 2nd Button to the XML UI and in code with its own if clause within `onClick`.

The 2nd Button can perform the same actions on click as the 1st to ensure that everything still works.

Here a 2nd Button click is linked to a simple 1 line Alert object and its methods

```
if(v == findViewById(R.id.bSimple))
{
     new AlertDialog.Builder(this).
     setTitle("Message").
     setMessage("A Message").
     setIcon(R.drawable.icon).
     setNeutralButton("Close", null).show();
}
```



Picking Dates

Android can pick up the current date from the `Calendar` object. This requires an import to be accessed.

```
import java.util.Calendar;
```

These variables are declared at the top of the class so they can be accessed by several methods

```
private int mYear;
private int mMonth;
private int mDay;
```

This is in `onClick` and is used to pick up the current date from `Calendar`. Only the year is output below.

```
if (v == findViewById(R.id.bTt))
{

    final Calendar c = Calendar.getInstance();
    mYear = c.get(Calendar.YEAR);
    mMonth = c.get(Calendar.MONTH);
    mDay = c.get(Calendar.DAY_OF_MONTH);
    vOut.setText(String.valueOf(mYear));
}
```

To set up the date picker we need a static declaring to hold the id of the Dialog

```
static final int DATE_DIALOG_ID = 0;
```

The usual additional imports are required.

```
import android.app.DatePickerDialog;
import android.widget.DatePicker;
```

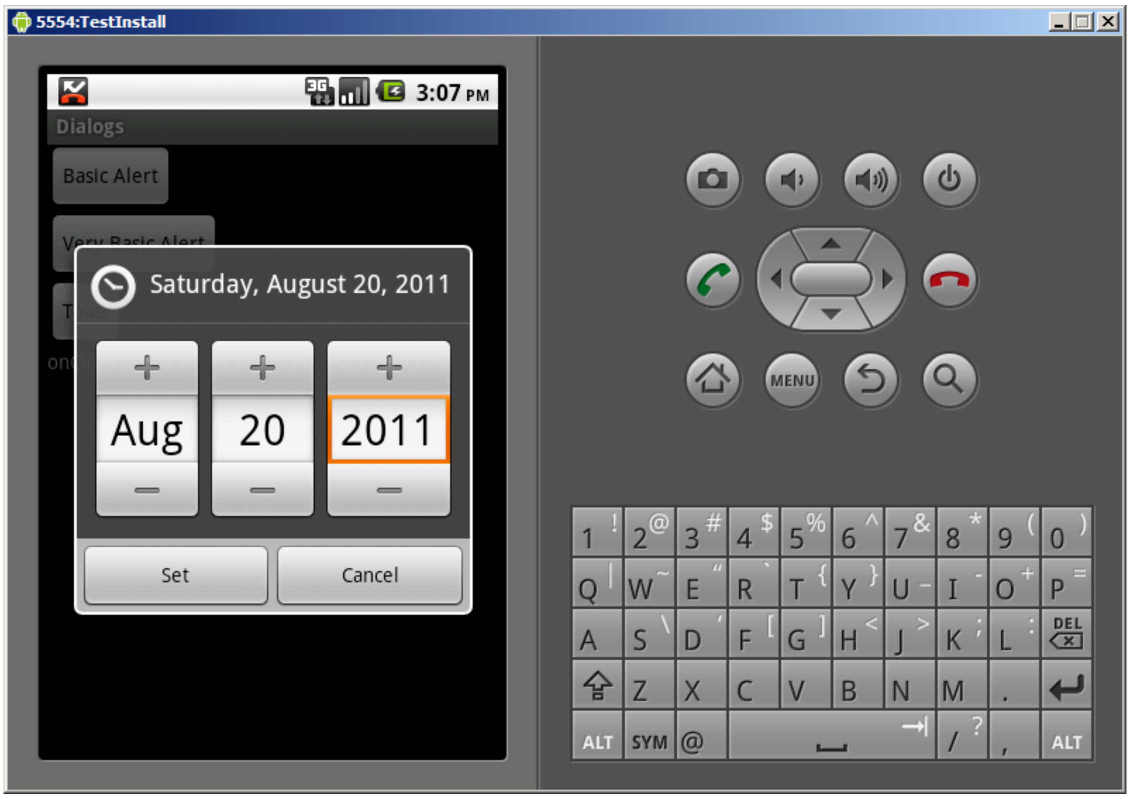
The following methods are then used to display the date picker and capture the date input. These all need to be within the class but not inside any other methods (such as `onClick(View v)` or `onCreate(Bundle savedInstanceState)`)

```
protected Dialog onCreateDialog(int id) {
    switch (id) {
        case DATE_DIALOG_ID:
            return new DatePickerDialog(this,
                mDateSetListener, mYear, mMonth, mDay);
            }
        return null;
}
```

```
private DatePickerDialog.OnDateSetListener mDateSetListener =
     new DatePickerDialog.OnDateSetListener() {
               public void onDateSet(DatePicker view, int
          year,  int monthOfYear, int dayOfMonth)
          {
               mYear = year;
               mMonth = monthOfYear;
               mDay = dayOfMonth;
               updateDisplay(); //called below
          }
     };
private void updateDisplay() {
     vOut.setText(
                    new StringBuilder()
                    .append(mDay).append("/")
                    // Month is 0 based so add 1
                    .append(mMonth + 1).append("/")
                    .append(mYear));
}
```

Having sorted all this out the code to show the DatePicker is

```
showDialog(DATE_DIALOG_ID);
```


5554:TestInstall
3:07 PM
Dialogs
Basic Alert
Very Basic Alert
Saturday, August 20, 2011
Aug
20
2011
Set
Cancel
MENU