Trường Đại học Bách Khoa Hà Nội
Hanoi University of Science and Technology

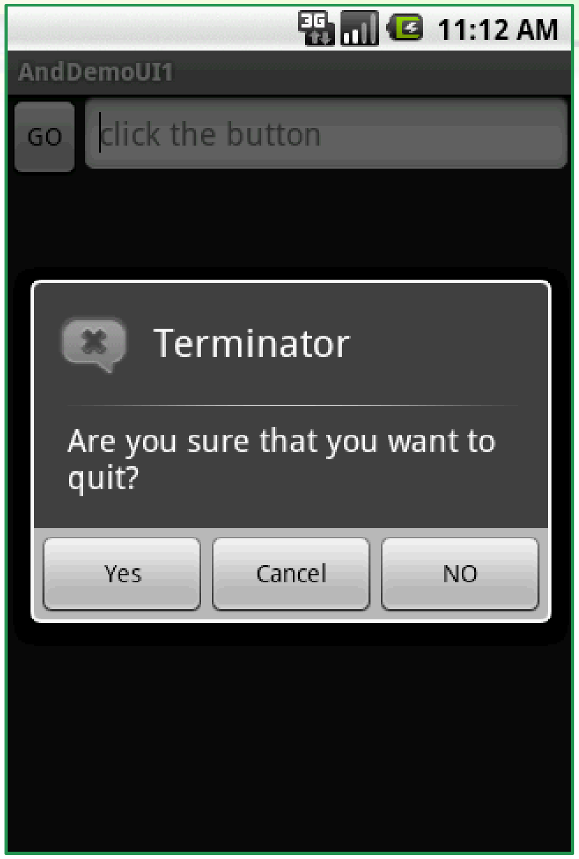
Chapter 9

Intent and Notifications

The DialogBox

Android provides two primitive forms of dialog boxes:

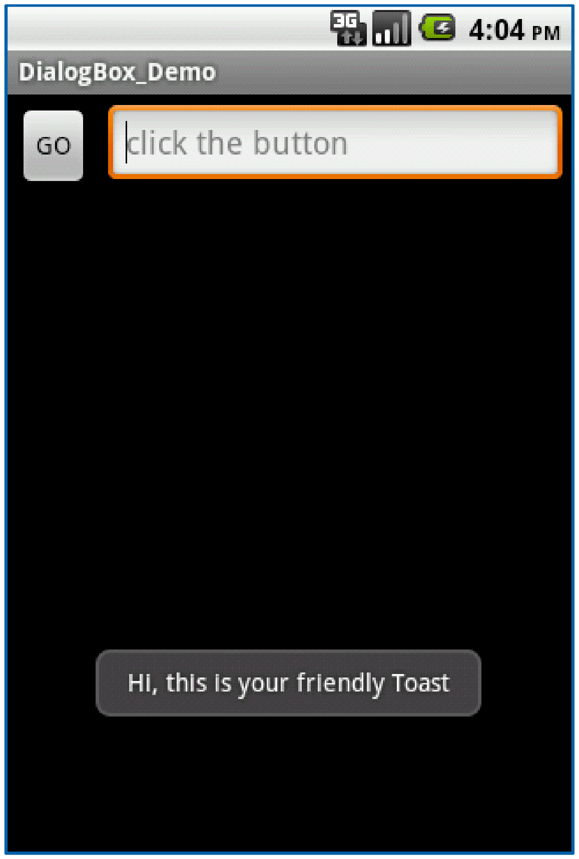
1. **AlertDialog** boxes, and
2. **Toast** controls


11:12 AM

AndDemoUI1

Terminator

Are you sure that you want to quit?


4:04 PM

DialogBox_Demo

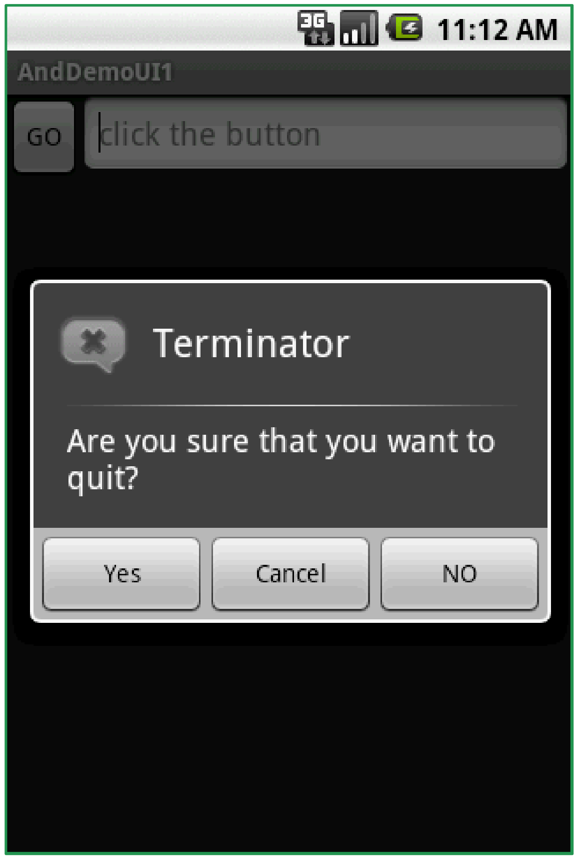
Hi, this is your friendly Toast

The AlertDialog

The *AlertDialog* is an *almost modal* screen that

(1) presents a brief message to the user typically shown as a small floating window that partially obscures the underlying view, and

(2) collects a simple answer (usually by clicking an option button) .



Are you sure that you want to quit?

Note:
A *modal* view remains on the screen waiting for user's input. The rest of the application is on hold. It has to be dismissed by an explicit user's action.

The AlertDialog

Warning !!!

An *AlertDialog* is **NOT** a typical *inputBox* (as in .NET)

Why?

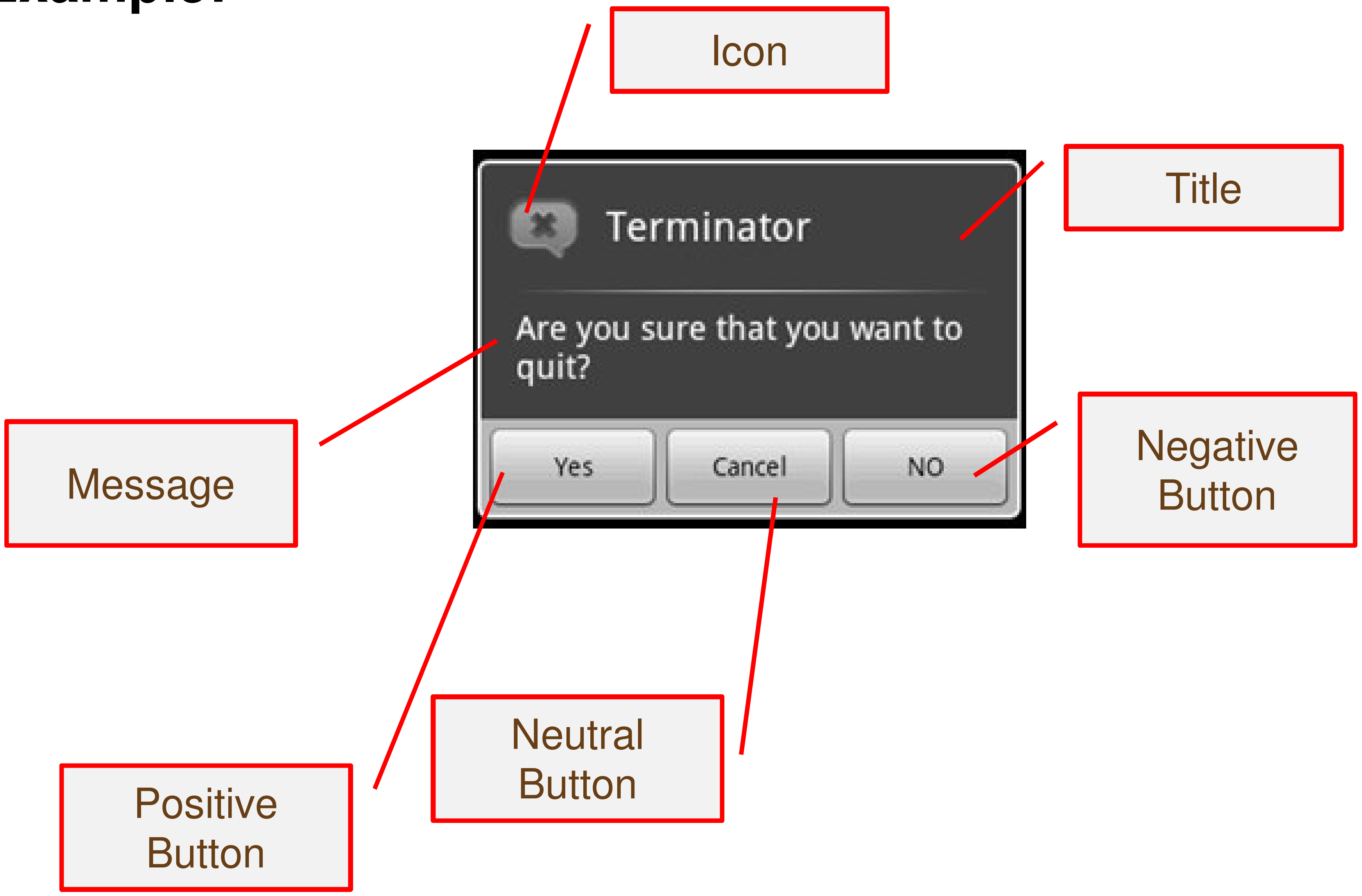
An *AlertDialog* box is modal as it needs user intervention to be terminated

However

it *does not stop the main thread* (code following the call to show the *DialogAlert* box is executed without waiting for the user's input)

The AlertDialog

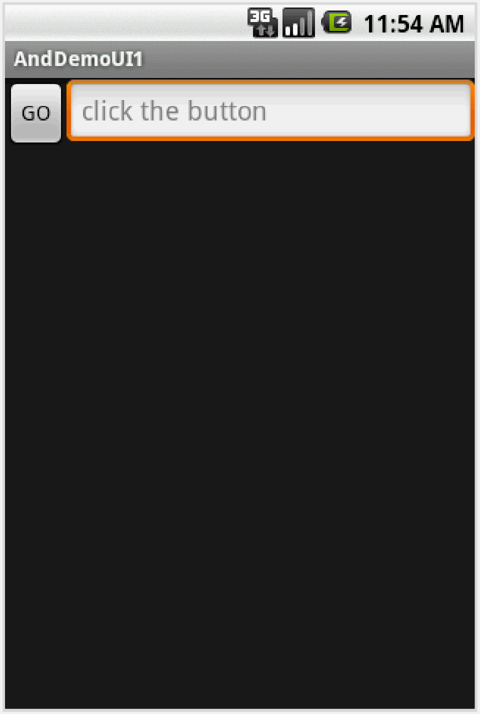
Example:



The AlertDialog

Example: A simple Dialog Box

```
<LinearLayout
    android:id="@+id/LinearLayout01"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal">
<Button
   android:text="GO"
   android:id="@+id/btnGo"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content">
 </Button>
 <EditText
   android:hint="click the button"
   android:id="@+id/txtMsg"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content">
 </EditText>

</LinearLayout>
```



The AlertDialog

Example: A simple dialog box

```
package it3660.selectionwidgets;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class AndDemoUI1 extends Activity {

   Button btnGo;
   EditText txtMsg;
   String msg;
```

The AlertDialog

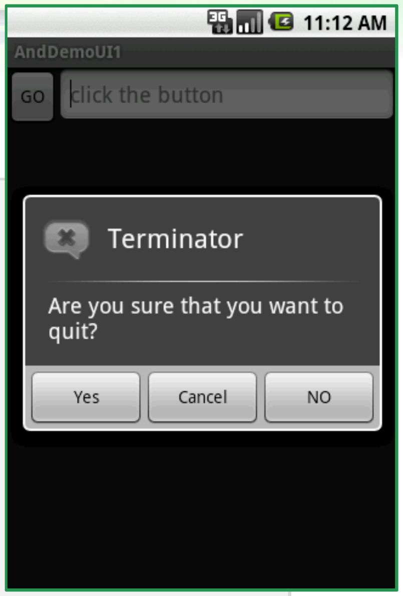
Example: A simple dialog box

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtMsg =  (EditText)findViewById(R.id.txtMsg);
        btnGo = (Button) findViewById(R.id.btnGo);
        btnGo.setOnClickListener(new OnClickListener() {
          @Override
          public void onClick(View arg0) {

            AlertDialog dialBox = createDialogBox();
            dialBox.show();

            // WARNING: (in general...)
            // after showing a dialog you should have NO more code. Let the buttons of
            // the dialog box handle the rest of the logic. For instance, in this
            // example a modal dialog box is displayed (once shown you can not do
            // anything to the parent until the child is closed) however the code in
            // the parent continues to execute after the show() method is
            // called.
            txtMsg.setText("I am here!");
          }
    });

    }//onCreate
```

The AlertDialog

Example: A simple dialog box

```
private AlertDialog createDialogBox(){

    AlertDialog myQuittingDialogBox =

        new AlertDialog.Builder(this)

        //set message, title, and icon
        .setTitle("Terminator")
        .setMessage("Are you sure that you want to quit?")
        .setIcon(R.drawable.ic_menu_end_conversation)

        //set three option buttons
        .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
             public void onClick(DialogInterface dialog, int whichButton) {
                    //whatever should be done when answering "YES" goes here
                    msg = "YES " + Integer.toString(whichButton);
                    txtMsg.setText(msg);
             }
        })//setPositiveButton
```



Are you sure that you want to quit?

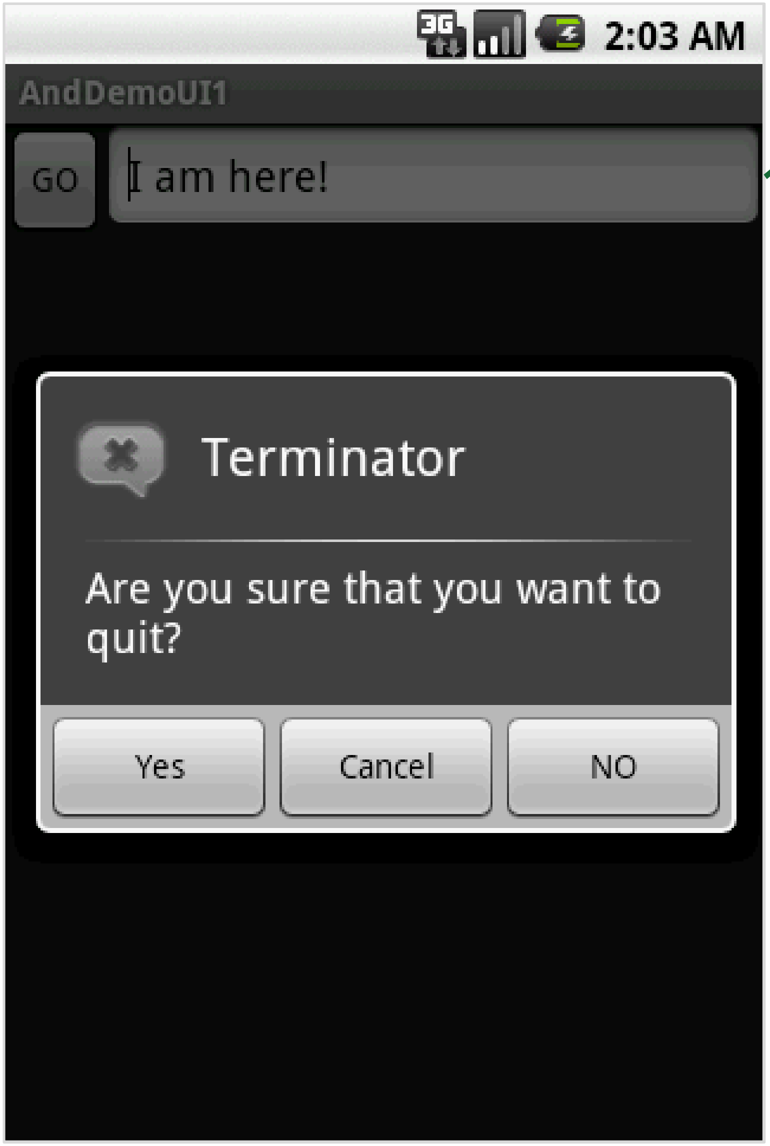
The AlertDialog

Example: A simple dialog box

```
        .setNeutralButton("Cancel",new DialogInterface.OnClickListener() {
             public void onClick(DialogInterface dialog, int whichButton) {
                    //whatever should be done when answering "CANCEL" goes here
                    msg = "CANCEL " + Integer.toString(whichButton);
                    txtMsg.setText(msg);
             }//OnClick
        })//setNeutralButton

        .setNegativeButton("NO", new DialogInterface.OnClickListener() {
             public void onClick(DialogInterface dialog, int whichButton) {
                    //whatever should be done when answering "NO" goes here
                    msg = "NO " + Integer.toString(whichButton);
                    txtMsg.setText(msg);
              }
        })//setNegativeButton

        .create();
        .return myQuittingDialogBox;

    }// createDialogBox

}// class
```

The AlertDialog

Example: A simple AlertDialog box



2:03 AM

AndDemoUI1

GO

I am here!

Terminator

Are you sure that you want to quit?

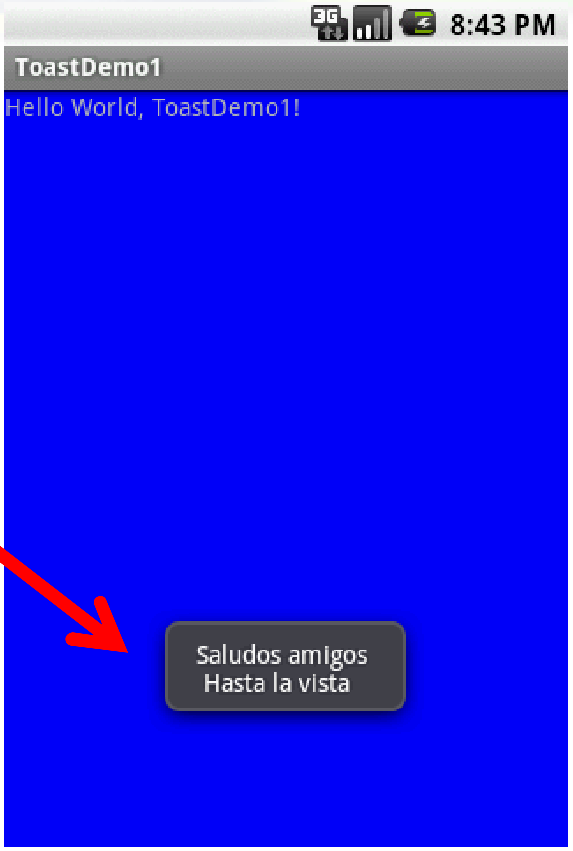
This text is set right after showing the dialog box

The Toast View

A **Toast** is a transient view containing a quick little message for the user.

They appear as a floating view over the application.

They never receive focus.


8:43 PM
ToastDemo1
Hello World, ToastDemo1!
Saludos amigos
Hasta la vista

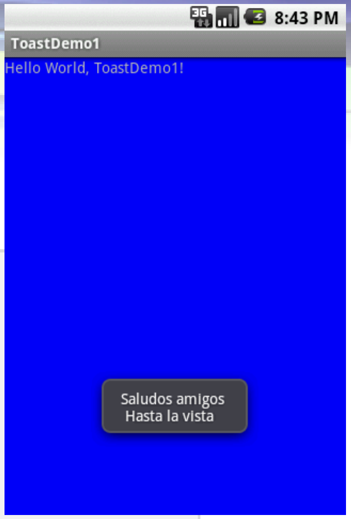
The Toast View

Example: A simple Toast

```
Toast.makeText ( context,  message,  duration ).show();
```

Context:	A reference to the view's environment (what is around me…)
Message:	The thing you want to say
Duration:	SHORT or LONG exposure

The Toast View


8:43 PM
ToastDemo1
Hello World, ToastDemo1!
Saludos amigos
Hasta la vista

Example: A simple Toast

```
package it3660.dialogboxes;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Toast;

public class ToastDemo1 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Toast.makeText(
                getApplicationContext(),
                "Saludos amigos \n Hasta la vista",
                Toast.LENGTH_LONG).show();
    }
}
```

The Toast View

As an aside

Context:

On Android a Context is mostly used to load and access resources.

All widgets receive a Context parameter in their constructor.

In a regular Android application, you usually have two kinds of Context, *Activity* and *Application*. The first one is typically passed to classes and methods that need a Context.

Views have a reference to the entire activity and therefore to anything your
activity is holding onto; usually the entire View hierarchy and all its resources.

The Toast View

Customizing a Toast View

By default Toast views are displayed at the center-bottom of the screen.

However the user may change the placement of a Toast view by using either of the following methods:

```
void setGravity (int gravity, int xOffset, int yOffset)
```
Set the location at which the notification should appear on the screen.

```
void setMargin (float horizontalMargin, float verticalMargin)
```
Set the margins of the view.

The Toast View

Customizing a Toast View

The following method uses offset values based on the pixel resolution of the actual device. For instance, the G1 phone screen contains 360x480 pixels.

```
void setGravity (int gravity, int xOffset, int yOffset)
```

Gravity: Overall placement. Typical values include:
Gravity.CENTER, Gravity.TOP,
Gravity.BOTTOM, …

xOffset: Assume Gravity.CENTER placement on a G1 phone.
The *xOffset* range is -160,…,0,…160 (left, center, right)

yOffset: The range on a G1 is: -240,…,0,…240 (top, center, bottom)

The Toast View

Customizing the Toast View

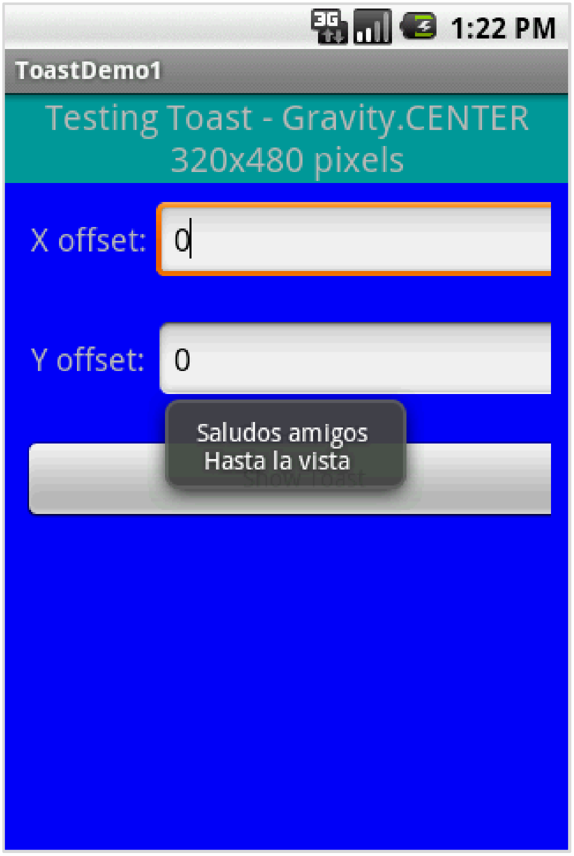
A second method to place a Toast is ***setMargin***. The screen is considered to have a center point where horizontal and vertical center lines meet. There is 50% of the screen to each side of that center point (top, botton, left, right). Margins are expressed as a value between: -50,…, 0, …, 50.

```
void setMargin (float horizontalMargin, float verticalMargin)
```

Note:
The pair of margins (-50, -50) represent the upper-left corner of the screen,
(0, 0) is the center, and (50, 50) the lower-right corner.

The Toast View

Example: Changing the placement of a Toast view.



1:22 PM

ToastDemo1

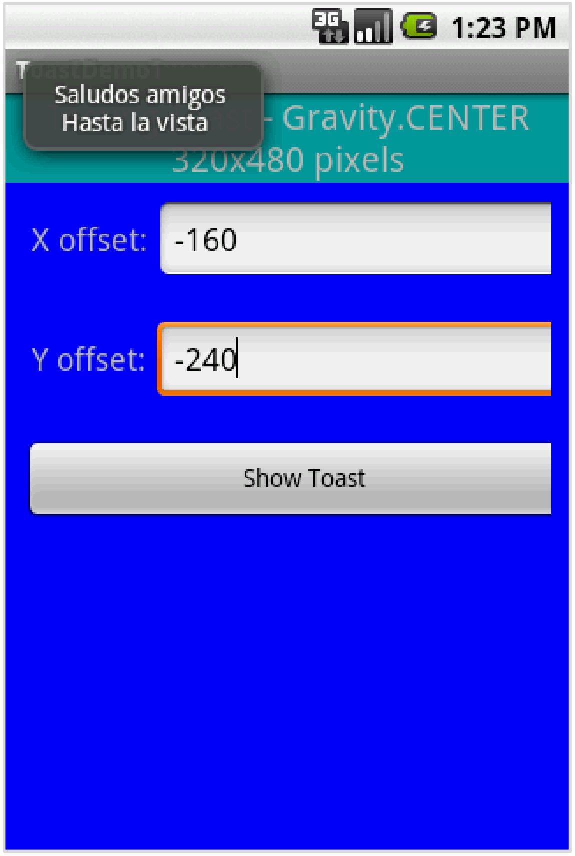
Testing Toast - Gravity.CENTER
320x480 pixels

X offset: 0

Y offset: 0

Saludos amigos
Hasta la vista

Show Toast



1:23 PM

ToastDemo1

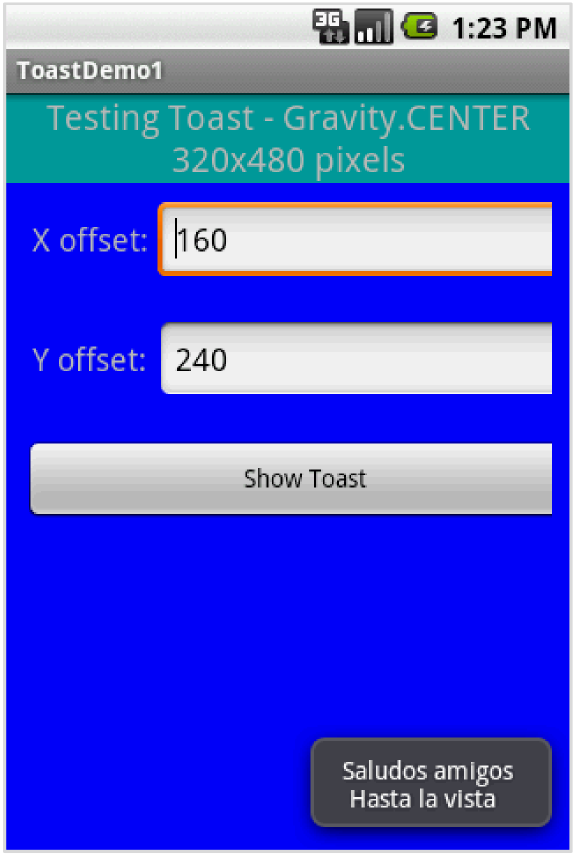
Saludos amigos
Hasta la vista

Testing Toast - Gravity.CENTER
320x480 pixels

X offset: -160

Y offset: -240

Show Toast



1:23 PM

ToastDemo1

Testing Toast - Gravity.CENTER
320x480 pixels

X offset: 160

Y offset: 240

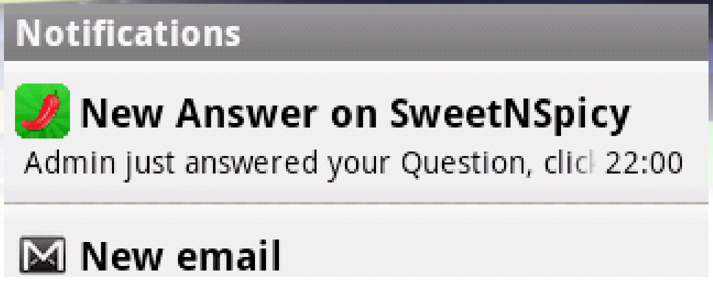
Show Toast

Saludos amigos
Hasta la vista

Using the **`setGravity(…)`** method with Gravity.CENTER, and x and y offsets of (resp.):

0, 0	(center)
-160, -240	(top-left)
160, 240	(right-bottom)

Notifications


Notifications

New Answer on SweetNSpicy

Admin just answered your Question, clicl 22:00

New email

NotificationManager

- Allows an application to put a message in the status bar at the top of the display
- Specify the icon and “alert text” to appear
- Also specify the “title” and the “message” that are shown when the user pulls down the status bar
- Can add sound, vibration, flashing lights, etc.

Notification

In a class that extends Activity...

```
private static final int NOTIFICATION_ID = 1;
// call this method to show the notification/alert
protected void showNotification() {
    String ns = Context.NOTIFICATION_SERVICE;
    NotificationManager mNotificationManager =
         (NotificationManager) getSystemService(ns);
    int icon = R.drawable.icon;
    CharSequence tickerText = "Alert!"; // text at top
    long when = System.currentTimeMillis();

    Notification notification = new Notification(icon,
                                       tickerText, when);
    Context context = getApplicationContext();
    CharSequence contentTitle = "Important message"; // title
    CharSequence contentText = "Hello World!";       // message
    Intent notificationIntent = new Intent(this,
                                         this.getClass());
    PendingIntent contentIntent = PendingIntent.getActivity(
                       this, 0, notificationIntent, 0);

    notification.setLatestEventInfo(context, contentTitle,
                              contentText, contentIntent);

    mNotificationManager.notify(NOTIFICATION_ID, notification);
}
```

End of Lecture