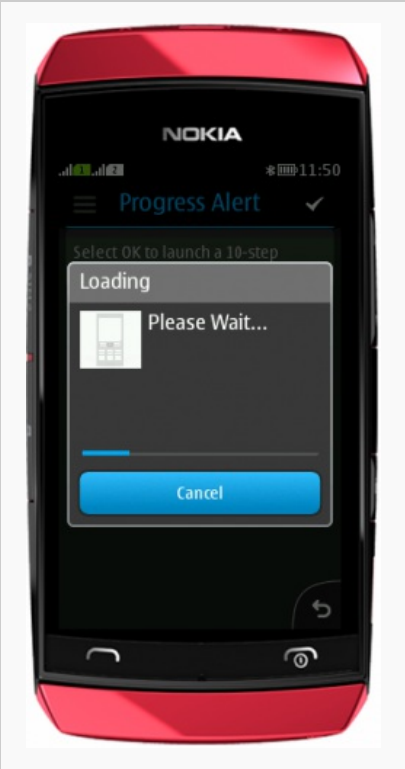
How to add an activity or progress indicator to an Alert

This code example demonstrates how to add an activity indicator to an Alert, that displays the progress for the completion of a process and dismisses itself, when the process is completed.

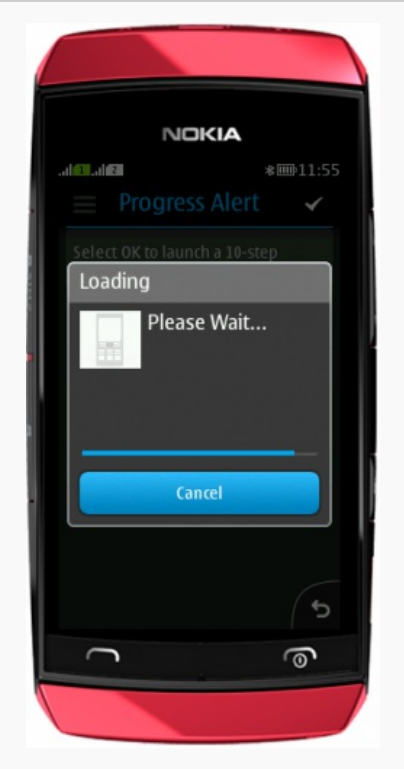
Overview

This code example uses two high-level LCDUI components, an Alert and a Gauge. The Alert is manually launched by the end user, while the Gauge is used as activity indicator inside the Alert. The activity indicator describes the progress for completing a 10-step task. The range of the indicator is between 0 and 100, in increments of 10.

On full touch Series 40, the Alert Launcher is placed in Action Button 1



Step 2 in a 10-step progress activity



Step 9 in a 10-step progress activity

Main Components

There are three Commands in this code example. An OK command that launches the Alert and an Exit Command for calling the `notifyDestroyed()` method, are appended to the Form. A Cancel Command is added to the Alert. The running thread moves the progress indicator every one second in increments of 10. There is also an image that is being displayed inside the alert.

Note: In order for the image to be properly displayed with this code, you need to add an image file to the resource directory of your project and rename it to **generic_phone.png**.

```
 Form mainForm;

//OK and Exit are added to the Form
Command exitCommand = new Command("Exit", Command.EXIT, 0);
Command okCommand = new Command("OK", Command.OK, 0);

//Cancel is added to the Alert
Command cancelCommand = new Command("Cancel", Command.CANCEL, 0);

//A 1 second per step thread
Thread thread;
Alert alert;

//A random image for the alert
Image img;

//A bar indicator attached to the alert
Gauge indicator;

protected void startApp() throws MIDletStateChangeException {
 //The Form's components
 mainForm = new Form("Progress Alert");
```


(C) Copyright Nokia 2013. All rights reserved.

```
 mainForm.append("Select OK to launch a 10-step process");
 mainForm.addCommand(okCommand);
 mainForm.addCommand(exitCommand);
 mainForm.setCommandListener(this);
 display = Display.getDisplay(this);
 display.setCurrent(mainForm);
```

Conditions for adding a Gauge as activity indicator within an Alert

Only under certain conditions, can a high-level Gauge be used within an Alert as activity indicator. The Gauge needs to satisfy the following conditions:

- it must be non-interactive
- it must not be owned by another container (Alert or Form)
- it must not have any `Commands`
- it must not have an `ItemCommandListener`
- it must not have a label (that is, its label must be null
- its preferred width and height must both be unlocked, and
- its layout value must be `LAYOUT_DEFAULT`.

This practically means that we need to instantiate the Gauge object with `null` for the first argument that denotes the label and `false` for the second argument that denotes the boolean that determines whether the Gauge is interactive or not:

```
indicator = new Gauge(null, false, 100, 0);
alert.setTimeout(Alert.FOREVER);
alert.setCommandListener(this);
alert.setIndicator(indicator);
```

As seen from the code above, the alert uses a `CommandListener` for canceling the activity. That requires that we set the Timeout to `FOREVER` so that the end user can only dismiss the alert by clicking on the **Cancel** button rather than by tapping anywhere on the screen.

The thread that simulates the progress of the activity

The indicator in this example can be used in connection with measuring the progress of a task with definite range. This means that we know in advance how many steps the task consists of. As examples of tasks with definite ranges can be considered downloading a file whose size is known. The example can be adjusted to the measuring quantity. Here, we assume that a task's range is from 0 up to 100 and consists of 10 steps. The thread pauses for 1 second then increments the value of the indicator by 10 until it reaches 100. At that point, the application exits the while loop and dismisses the Alert as shown below:

```
public void run() {
      try {
  int current = indicator.getValue();
  //A ten step increment from 0 to 100 in steps of 10
  while(current < 100){
           current = indicator.getValue();
   indicator.setValue(current + 10);
   //One second pause between the steps
   Thread.sleep(1000);
                 }
         }
         catch (InterruptedException e) {
  //Dismisses the alert by interrupting the progress thread
  e.printStackTrace();
 }
 //Displays the original screen and resets the progress bar value
 display.setCurrent(mainForm);
 indicator.setValue(0);
}
```

The alert can also be dismissed by clicking on the **Cancel** button. The code inside this action, simply interrupts the thread:

```
public void commandAction(Command c, Displayable arg1) {
 if(c == cancelCommand) {
  //Interrupts the thread that moves the progress bar
  thread.interrupt();
 }
}
```

See also

- The Gauge Class
- The Alert Class

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.