



<b>ID</b>	CS000973	<b>Creation date</b>	May 20, 2008
<b>Platform</b>	S60 3rd Edition, FP1	<b>Tested on devices</b>	Nokia N95 8GB
<b>Category</b>	Java ME	<b>Subcategory</b>	UI

**Keywords (APIs, classes, methods, functions):** javax.microedition.lcdui.Display, javax.microedition.lcdui.Displayable, javax.microedition.lcdui.Alert, javax.microedition.lcdui.AlertType, javax.microedition.lcdui.Display.getDisplay(), javax.microedition.lcdui.Display.setCurrent()

## Overview

This code snippet demonstrates how to navigate between three different Screen instances. Screen is the superclass of all high-level user interface classes, such as Form and List.

In practice, the navigation is done by setting the current Displayable to the specific Screen:

```
private void switchCurrentScreen(Displayable displayable) {
    Display.getDisplay(this).setCurrent(displayable);
}
```

## Source file

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.StringItem;
import javax.microedition.midlet.MIDlet;

public class NavigationMIDlet extends MIDlet implements CommandListener {
    private Form firstScreen;
    private Form secondScreen;
    private Form thirdScreen;

    // Navigation commands
    private static final Command NEXT_COMMAND =
        new Command("Next", Command.SCREEN, 1);
    private static final Command PREV_COMMAND =
```

## CS000973\_-\_Navigating\_between\_Screens

```
        new Command("Previous", Command.BACK, 1);
private static final Command EXIT_COMMAND =
        new Command("Exit", Command.EXIT, 1);

/**
 * Constructor. Constructs the object and initializes displayables.
 */
public NavigationMIDlet() {
    // Initialize the screens
    initFirstScreen();
    initSecondScreen();
    initThirdScreen();
}

private void initFirstScreen() {
    firstScreen = new Form("First example screen");
    StringItem item = new StringItem("Description",
        "This is the first example screen.");
    firstScreen.append(item);
    firstScreen.addCommand(NEXT_COMMAND);
    firstScreen.addCommand(EXIT_COMMAND);
    firstScreen.setCommandListener(this);
}

private void initSecondScreen() {
    secondScreen = new Form("Second example screen");
    StringItem item = new StringItem("Description",
        "This is the second example screen.");
    secondScreen.append(item);
    secondScreen.addCommand(NEXT_COMMAND);
    secondScreen.addCommand(PREV_COMMAND);
    secondScreen.setCommandListener(this);
}

private void initThirdScreen() {
    thirdScreen = new Form("Third example screen");
    StringItem item = new StringItem("Description",
        "This is the third example screen.");
    thirdScreen.append(item);
    thirdScreen.addCommand(PREV_COMMAND);
    thirdScreen.setCommandListener(this);
}

private void switchCurrentScreen(Displayable displayable) {
    Display.getDisplay(this).setCurrent(displayable);
}

/**
 * From MIDlet.
 * Called when the MIDlet is started.
 */
public void startApp() {
    // The initial display is the first form
    switchCurrentScreen(firstScreen);
}

/**
 * From MIDlet.
 * Called to signal the MIDlet to enter the Paused state.
 */
public void pauseApp() {
    // No implementation required
}
```

## CS000973\_-\_Navigating\_between\_Screens

```
}

/**
 * From MIDlet.
 * Called to signal the MIDlet to terminate.
 * @param unconditional whether the MIDlet has to be unconditionally
 * terminated
 */
public void destroyApp(boolean unconditional) {
    // No implementation required
}

/**
 * From CommandListener.
 * Called by the system to indicate that a command has been invoked on a
 * particular displayable.
 * @param command the command that was invoked
 * @param displayable the displayable where the command was invoked
 */
public void commandAction(Command command, Displayable displayable) {
    if (command == EXIT_COMMAND) {
        // Exit the MIDlet
        notifyDestroyed();
    } else if (command == NEXT_COMMAND) {
        if (displayable == firstScreen) {
            switchCurrentScreen(secondScreen);
        } else if (displayable == secondScreen) {
            switchCurrentScreen(thirdScreen);
        }
    } else if (command == PREV_COMMAND) {
        if (displayable == secondScreen) {
            switchCurrentScreen(firstScreen);
        } else if (displayable == thirdScreen) {
            switchCurrentScreen(secondScreen);
        }
    }
}
}
```

It is also possible to display an Alert before switching the screen. In addition, the switching can be done only after the user has dismissed the alert. `AlertType` class defines the type of the alert and whether user interaction is required before the screen is switched (`AlertType.CONFIRMATION`). To implement showing an informative alert (`AlertType.INFO`; doesn't require user interaction) before switching the screen, add the following code to the example above:

```
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;

private void switchCurrentScreen(Alert alert, Displayable displayable) {
    Display.getDisplay(this).setCurrent(alert, displayable);
}

Alert alert = new Alert("Screen changing",
    "Switching current screen in a moment...", null, AlertType.INFO);
switchCurrentScreen(alert, firstScreen);
```

## **Postconditions**

The MIDlet demonstrates navigation between three different screens.