How to utilize different keyboards in Java ME

Overview

The most of the mobile phones have ITU-T keyboard. ITU-T keypad is the traditional mobile device keypad with 12 basic keys, that is, number key 0-9, *-key and #-key. ITU-T is a standard of International Telecommunication Union. However, some of the latest S60 3rd Edition FP2 and 5th Edition devices have different keypads and keyboards, for example mini-QWERTY or half-QWERTY (a.k.a compact QWERTY) keyboard. This article explains, how to use effectively these keyboards with different keyboard layouts.

Keyboard types

Currently (June 2009) there are three keypad or keyboard layouts in use in S60 devices:

- ITU-T keypad (for example Nokia 6220 classic)
- half-QWERTY keyboard (for example Nokia E55)
- mini-QWERTY keyboard (for example Nokia N97)

Below are pictures showing these three layouts in Nokia devices.

Figure 1: ITU-T keypad in Nokia 6220 classic, half-QWERTY keyboard in Nokia E55 and mini-QWERTY keyboard in Nokia N97

Issues with the half-QWERTY keyboard

It should be noticed, that it is not possible to get key codes for all the characters in half-QWERTY keyboard. For example, if the key with "G", "H" and "5" is pressed, you will get keycode 103 (scan code 71), which means character "g". It is not possible to get keycodes for "h" or "H" by using Canvas.keyPressed() method.

About the scancodes

A system property "com.nokia.key.scancode" returns the the scancode of the latest pressed key. The scancode can be thought to be the code of a certain key(, not a key for a certain character). For example in QWERTY keyboard pressing "Q" gives 113 as keycode and 81 as scancode. But then in French-speaking countries AZERTY keyboard layout is commonly used. In these keyboards pressing "Q" would give the same keycode, but the scancode would be different. Practical example of using scancodes could be games, in which keys in certain pre-defined locations are needed. This means, that compact QWERTY is more close to ITU-T keypad than to full character keyboard. If Java developer wants to implement e.g. text input with low-level key events then application needs to do it's own text input handling in very similar way as it is done in devices having ITU-T keypad.

Extra system properties

There are some additional system properties, which are needed for effective usage of the different keypads and keyboards. Note, that these system properties are currently available only in S60 devices (JRT 1.4 and newer), not in Series 40 devices.

System property	Purpose	Values
com.nokia.keyboard.type	Currently use keyboard type	One of the following values: None, PhoneKeypad, HalfKeyboard, FullKeyboard, LimitedKeyboard4x10, LimitedKeyboard3x11, Custom, Unknown
com.nokia.key.modifier	Key event modifiers (SHIFT/FN/CTRL, etc.) used in the latest key press	A number (one or more digits), see separate Table 2.
com.nokia.key.scancode	Indicates the scancode of the latest key press	A number (one or more digits)

(C) Copyright Nokia 2014. All rights reserved.

Table 1: MiniController MIDlet classes

Modifier key	Native identifier examples (below are the S60 identifiers for reference)	Value	Decimal value
Left shift	EModifierShift	0x00000400	1280
Right shift	EModifierShift	0x00000400	1536
Left ctrl	EModifierCtrl	0x00000080	160
Right ctrl	EModifierCtrl	0x00000080	192
Fn	EModifierFunc	0x00002000	12288
Chr	EModifierFunc	0x00002000	10240

Table 2: Bitflags for the modifier keys

The system property values are updated, when the actual feature related to the property is changed, for example, when a modifier key is pressed or if the keyboard layout is changed. Thus it makes sense to track the changes of the properties in Canvas.keyPressed() method. One way to get a notification of the keyboard layout change is to read the "com.nokia.keyboard.type" property in Canvas.paint() method, because commonly also screen size changes at the same time.

Different keypad and keyboard layouts in high-level UIs

The support for different keypad and keyboard layouts works in the same way as in the native editors. Note however, that some of the text constraints may be meaningless in full keyboard editors. For example, normally full keyboard editors do not always have an automatic casing functionality, so INITIAL_CAPS_WORD and INITIAL_CAPS_SENTENCE are ignored in them. Also since predictive text input (T9) is always turned off (or completely unavailable) in full keyboard input, the NON_PREDICTIVE mode does not have any effect on the editor.

Different keypad and keyboard layouts in low-level UIs

In Canvas, GameCanvas, FullCanvas (in the Nokia UI API) and CustomItem low-level key events are delivered to the applications. The MIDP specification defines that the keycode values must be equal to Unicode values for characters that represent the keys. For keys for which there is no obvious correspondence to a Unicode character (for example soft keys in Nokia devices), negative values must be used.

Modifier keys are keys that modify the result of a key press after the modifier key press, e.g. shift, control and chr-key are modifier keys. The modifier keys, like a shift key, in a keyboard will affect both the keycode and the key name delivered from that key. For example, pressing a key labeled ‘A’ will result a keycode ‘97’ to be delivered, but pressing shift together with a key labeled ‘A’ will result a keycode ‘65’. The key names will be ‘a’ for keycode 97 and ‘A’ for keycode 65 respectively. Commonly modifier keys work as sticky modifier keys. This means, that a modifier key can be pressed and released alone, not combined with any other key press, and the next key press will result a keycode with a modified value.

Note, that the modifier keys might look different in different devices. In some devices FN-key is just an arrow-kind of symbol and in some devices it it marked as "Fn". Below is an image showing the modifier key locations and their appearance in N97 and E55. Note also, that several modifier keys can be pressed simultaneously. Such cases have their own system property values, as can be seen by using the attached MIDlet. Keycode for all the modifier keys is -50, even if several modifier keys are pressed at the same time.

Figure 2. The modifier keys in Nokia N97 and Nokia E55

Example MIDlet

Here is a MIDlet for testing the new system properties and other keypad and keyboard related features. The MIDlet prints out the property values and also the keycodes of the pressed keys (if available).

(C) Copyright Nokia 2014. All rights reserved.

Figure 3: KeyboardMIDlet running in Nokia N97 SDK

Source code: KeyboardMIDlet.java

```
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;

public class KeyboardMIDlet extends MIDlet {
    private KeyboardCanvas canvas;

    public void startApp() {
        canvas = new KeyboardCanvas(this);
        Display.getDisplay(this).setCurrent(canvas);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

Source code: KeyboardCanvas.java

```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Font;
import javax.microedition.lcdui.Graphics;

public class KeyboardCanvas extends Canvas implements CommandListener {
    private static final int LSHIFT = 0x00000500;
    private static final int RSHIFT = 0x00000600;
    private static final int LCTRL  = 0x000000A0;
    private static final int RCTRL  = 0x000000C0;
    private static final int FN     = 0x00003000;
    private static final int CHR    = 0x00002800;
```


(C) Copyright Nokia 2014. All rights reserved.

```
    private KeyboardMIDlet midlet;
    private Command exitCommand;
    private Font font;
    private int keyCode;
    private int keyScanCode = 0;
    private int keyModifiers = 0;
    private String keyboard = "";
    private String modifier = "";


    public KeyboardCanvas(KeyboardMIDlet midlet) {
        this.midlet = midlet;
        exitCommand = new Command("Exit", Command.EXIT, 1);
        this.addCommand(exitCommand);
        this.setCommandListener(this);
        font = Font.getFont(Font.FACE_SYSTEM, Font.STYLE_PLAIN, Font.SIZE_SMALL);
    }

    public void paint(Graphics g) {
        g.setColor(255,255,255);
        g.fillRect(0, 0, getWidth(), getHeight());
        g.setColor(0, 0, 0);
        g.setFont(font);
        int height = font.getHeight();
        g.drawString("Keyboard: " + this.keyboard, 0, 0, Graphics.TOP | Graphics.LEFT);
        g.drawString("Keycode: " + this.keyCode, 0, 0 + height*1, Graphics.TOP |
Graphics.LEFT);
        g.drawString("Scancode: " + this.keyScanCode, 0, 0 + height*2, Graphics.TOP |
Graphics.LEFT);
        g.drawString("Modifier: " + this.keyModifiers, 0, 0 + height*3, Graphics.TOP |
Graphics.LEFT);
        g.drawString("Modifiers: " + this.modifier, 0, 0 + height*4, Graphics.TOP |
Graphics.LEFT);
        g.drawString("Character: " + (char)this.keyCode, 0, 0 + height*5, Graphics.TOP |
Graphics.LEFT);
    }

    protected void keyPressed(int keyCode) {
        this.keyCode = 0;
        this.keyScanCode = 0;
        this.keyModifiers = 0;
        this.keyCode = keyCode; //keyCode is argument of keyPressed method
        try {
            this.keyScanCode =
Integer.parseInt(System.getProperty("com.nokia.key.scancode"));
            this.keyModifiers =
Integer.parseInt(System.getProperty("com.nokia.key.modifier"));
        }
            catch (NumberFormatException nfe) {
            nfe.printStackTrace();
        }

        //if keyModifiers value contains some of modifier bits,
        //name of corresponding key is added to modifier string
        this.modifier = "";
        this.modifier += ((this.keyModifiers & LCTRL) == LCTRL) ? "LCTRL " : "";
        this.modifier += ((this.keyModifiers & RCTRL) == RCTRL) ? "RCTRL " : "";
        this.modifier += ((this.keyModifiers & LSHIFT) == LSHIFT) ? "LSHIFT " : "";
        this.modifier += ((this.keyModifiers & RSHIFT) == RSHIFT) ? "RSHIFT " : "";
        this.modifier += ((this.keyModifiers & FN) == FN) ? "FN " : "";
        this.modifier += ((this.keyModifiers & CHR) == CHR) ? "CHR " : "";
```


(C) Copyright Nokia 2014. All rights reserved.

```
        this.keyboard = System.getProperty("com.nokia.keyboard.type");
        repaint();
    }

    public void commandAction(Command c, Displayable d) {
        if (c == exitCommand) {
            midlet.notifyDestroyed();
        }
    }
}
```

Example application

- KeyboardMIDlet.zip containing KeyboardMIDlet.jad, KeyboardMIDlet.jar and the sources

See also

- Java Runtime 1.4 for S60 Release Notes

(C) Copyright Nokia 2014. All rights reserved.