Controlling vibra settings in Java ME

This article needs to be updated: If you found this article useful, please fix the problems below then delete the {{ArticleNeedsUpdate}} template from the article to remove this warning.

Reasons:
Code doesn't seem to work on Nokia 701 or Nokia Asha 305

Overview

This article explains how to use device's vibration function.

To start vibrating device a call to `vibrate()`-method of Display-instance is used.

Duration of the vibration is given as a parameter. Note that this method returns immediately, so it doesn't wait till vibration is fnished.

The `Thread.sleep()` method is used for waiting until the end of the vibration.

Source file: ControllingVibraMIDlet.java

```
import javax.microedition.midlet.MIDlet;

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;

/**
 *
 */
public class ControllingVibraMIDlet extends MIDlet implements CommandListener {
    /**
     * Command for calling vibration
     */
    private Command controllingVibraCommand;
    /**
     * exit midlet command
     */
    private Command exitCommand;
    private Form mainForm;

    /**
     * The ControllingVibraMIDlet constructor.
     */
    public ControllingVibraMIDlet() {
    }

    /**
     * Performs an action assigned to the Mobile Device - MIDlet Started point.
     */
    public void startMIDlet() {
        switchDisplayable(getForm());
    }

    /**
     * Switches a current displayable in a display. The Display instance is
     * taken from getDisplay() method. This method is used by all actions
     * in the design for switching displayable.
     * @param nextDisplayable the Displayable to be set
     */
    public void switchDisplayable(Displayable nextDisplayable) {
        Display display = getDisplay();
        display.setCurrent(nextDisplayable);
    }

    /**
     * From CommandListener.
     * Called by a system to indicated 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 (displayable == mainForm) {
            if (command == exitCommand) {
                exitMIDlet();
            } else if(command == controllingVibraCommand) {
                try {
                    //displayObject.vibrate(int) Requests operation of the
                    //device's vibration.
                    //This method switches on the vibration for the requested
                    //duration, or switches it off if the requested duration
                    //is zero. This method returns immediately; that is, it
                    //must not block the caller while the vibration is running.
                    //that's why next we call Thread.sleep() function
                    mainForm.append("1000 milliseconds duration");
                    Display.getDisplay(this).vibrate(1000);
                    Thread.sleep(1100);
                    mainForm.append("700 milliseconds duration");
                    Display.getDisplay(this).vibrate(700);
                    Thread.sleep(800);
                    mainForm.append("starting with 500 milliseconds duration" +
                        " but we will stop vibration after first 300 " +
                        "milliseconds using 0 duration");
                    Display.getDisplay(this).vibrate(500);
                    Thread.sleep(300);
```


(C) Copyright Nokia 2013. All rights reserved.

```
                    Display.getDisplay(this).vibrate(0);
                }  catch (InterruptedException ex) {
                    ex.printStackTrace();
                }

            }

        }
    }
    /**
     * Returns an initiliazed instance of mainForm component.
     * @return the initialized component instance
     */
    public Form getForm() {
        if (mainForm == null) {
            mainForm = new Form("Vibra test!");

            exitCommand = new Command("Exit", Command.EXIT, 0);
            mainForm.addCommand(exitCommand);
            controllingVibraCommand = new Command("Vibra", Command.SCREEN, 0);
            mainForm.addCommand(controllingVibraCommand);
            mainForm.setCommandListener(this);
        }
        return mainForm;
    }
    /**
     * Returns a display instance.
     * @return the display instance.
     */
    public Display getDisplay () {
        return Display.getDisplay(this);
    }

    /**
     * Exits MIDlet.
     */
    public void exitMIDlet() {
        switchDisplayable(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /**
     * From MIDlet.
     * Called when MIDlet is started.
     * Checks whether the MIDlet have been already started and
     * initialize/starts or resumes the MIDlet.
     */
    public void startApp() {
        switchDisplayable(getForm());
    }
    /**
     * From MIDlet.
     * Called when MIDlet is paused.
     */
    public void pauseApp() {
        //Empty implementation
    }

    /**
     * From MIDlet.
     * Called to signal the MIDlet to terminate.
     * @param unconditional if true, then the MIDlet has to be unconditionally
     * terminated and all resources has to be released.
     */
    public void destroyApp(boolean unconditional) {
        //Empty implementation
    }
}
```

Postconditions

This MIDlet implements a simple example for using the device's vibration function.

When the "Vibra"-command has been selected from the menu, the device will vibrate for 1000 milliseconds, and then 700 milliseconds with a small delay between.

Next starts a vibration for 500 milliseconds, but after first 300 milliseconds it is turned off by the duration of vibra being set to 0 milliseconds.

Supplementary material

You can download this MIDlet and source code from here: Controlling Vibra Settings.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.