Creating threads in Java ME

This code snippet demonstrates how to create and execute a separate thread.

Overview

There are two ways to implement new threads in Java ME:

- by extending the base `Thread` class, or
- by implementing the `Runnable` interface.

In both cases, a new class based on the `Thread` class or the `Runnable` interface must be defined and it must override the `run()` method. This method defines the code that the thread will run, and thus defines the thread's functionality.

To use the previously defined `Thread` class, an object should be instanced and its `start()` method should be invoked.

If the `Runnable` interface is used instead, an object should be instanciated, after which the object should be passed to the `Thread` class constructor. Also in this case, the `start()` method of the newly created `Thread` class should be invoked.

The following code snippet uses the first way of implementing a thread - it extends the base `Thread` class and overrides the `run()` method.

Note that the state of the UI control in this snippet is changed from separate thread. UI controls in Java are thread-safe, so it is safe to update states and execute methods of UI controls from separate threads.

This MIDlet consists of 2 source files:

1. CreatingThreadsMidlet.java - contains the MIDlet class.
2. GaugeThread.java - contains the GaugeThread class which updates the specified gauge in a separate thread.

Source file: CreatingThreadsMidlet.java

```
    // Gauge to be updated by our gauge thread.
    private Gauge gauge;
    // Gauge thread
    private GaugeThread gaugeThread;

    /**
     * Sets up the main form.
     */
    private void setupMainForm() {
        mainForm = new Form("Creating Threads");

        // Create gauge for our thread
        gauge = new Gauge("Thread activity", false, Gauge.INDEFINITE,
                Gauge.INCREMENTAL_IDLE);
        mainForm.append(gauge);

        logField = new TextField("Log", null, 512, TextField.PLAIN);
        mainForm.append(logField);

        mainForm.addCommand(EXECUTE_COMMAND);
        mainForm.addCommand(EXIT_COMMAND);
        mainForm.setCommandListener(this);
    }
    /**
     * Executes the snippet.
     */
    private void executeSnippet() {
```


(C) Copyright Nokia 2014. All rights reserved.

```
        startGaugeThread();
    }
    /**
     * Creates gauge thread and starts it
     */
    private void startGaugeThread() {
        stopGaugeThread();
        // Create thread for gauge and run it
        gaugeThread = new GaugeThread(gauge);
        gaugeThread.start();
    }
    /**
     * Stops gauge thread
     */
    private void stopGaugeThread() {
        // Stop gauge thread
        if(gaugeThread != null && gaugeThread.isAlive() == true) {
            gaugeThread.quit();
            try {
                // Wait for gauge thread to die
                gaugeThread.join();
            } catch(InterruptedException exc) {
                // This thread was interrupted, do nothing.
            }
        }
    }
```

Source file: GaugeThread.java

```
import javax.microedition.lcdui.Gauge;

/**
 * Gauge thread functionality. Updates specified gauge.
 */
public class GaugeThread extends Thread {
    // Gauge to be updated.
    private Gauge gauge = null;
    // Flag indicating that thread must quit.
    private boolean isQuit = false;

    /**
     * Constructor.
     * @param gauge - gauge to be updated by this thread
     */
    public GaugeThread(Gauge gauge) {
        this.gauge = gauge;
    }

    /**
     * Terminates this thread.
     */
    public void quit() {
        isQuit = true;
    }
```


(C) Copyright Nokia 2014. All rights reserved.

```
    /**
     * Implements functionality of thread.
     */
    public void run() {
        // If no gauge specified exit from thread
        if(gauge == null) {
            return;
        }

        // Thread loop
        while(isQuit == false) {
            // Update gauge value
            gauge.setValue(Gauge.INCREMENTAL_UPDATING);
            try {
                // Make thread sleep for 100 milliseconds
                sleep(100);
            } catch(InterruptedException interruptExc) {
                // Some other thread interrupts this thread so exit from here
                return;
            }
        }
    }
}
```

Postconditions

The gauge and the logging text box are shown on the display. Press 'Execute snippet' to start a thread which updates the gauge's state so that its animation is displayed.

(C) Copyright Nokia 2014. All rights reserved.