Reading a text file line by line in Java ME

This code snippet demonstrates how to read a text file line by line (note that unlike Java SE with its `BufferedReader` class, Java ME doesn't provide the functionality).

Source

```
import java.io.IOException;
import java.io.InputStreamReader;
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.midlet.MIDlet;
```

```
public class ReaderMIDlet extends MIDlet implements CommandListener {
    private Form mainForm;
    private Command exitCommand;

    /**
     * Constructor. Constructs the object and initializes displayables.
     */
    public ReaderMIDlet() {
        mainForm = new Form("Example MIDlet");

        exitCommand = new Command("Exit", Command.EXIT, 1);
        mainForm.addCommand(exitCommand);
        mainForm.setCommandListener(this);

        try {
            readFile("text.txt");
        } catch (IOException ex) {
            // TODO: Exception handling
        }
    }

    /**
     * Outputs the specified file onto the form.
     * @throws java.io.IOException if an exception occurs when reading the
     * file
     */
    private void readFile(String filename) throws IOException {
        InputStreamReader reader = new InputStreamReader(
            getClass().getResourceAsStream(filename));
        String line = null;
        // Read a single line from the file. null represents the EOF.
        while ((line = readLine(reader)) != null) {
            // Append the read line to the main form with a linefeed ('\n')
            mainForm.append(line + "\n");
        }
        reader.close();
    }

    /**
     * Reads a single line using the specified reader.
     * @throws java.io.IOException if an exception occurs when reading the
     * line
     */
    private String readLine(InputStreamReader reader) throws IOException {
        // Test whether the end of file has been reached. If so, return null.
        int readChar = reader.read();
        if (readChar == -1) {
            return null;
        }
        StringBuffer string = new StringBuffer("");
        // Read until end of file or new line
        while (readChar != -1 && readChar != '\n') {
            // Append the read character to the string. Some operating systems
            // such as Microsoft Windows prepend newline character ('\n') with
            // carriage return ('\r'). This is part of the newline character
            // and therefore an exception that should not be appended to the
            // string.
            if (readChar != '\r') {
                string.append((char)readChar);
            }
            // Read the next character
            readChar = reader.read();
        }
        return string.toString();
    }

    /**
     * From MIDlet.
     * Called when the MIDlet is started.
     */
    public void startApp() {
        // The initial display is the first form
        Display.getDisplay(this).setCurrent(mainForm);
    }

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

    /**
     * From MIDlet.
     * Called to signal the MIDlet to terminate.
     * @param unconditional whether the MIDlet has to be unconditionally
     * terminated
```


(C) Copyright Nokia 2013. All rights reserved.

```
     */
    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 == exitCommand) {
            // Exit the MIDlet
            notifyDestroyed();
        }
    }
}
```

Supplementary material

- You can test the file handling features in action in a simple, executable application into which this code snippet has been patched. The application is available for download at: Media:JavaMEStub CS001006.zip
- You can examine all the changes that are required to implement file handling in an application. The changes are provided in unified diff and color-coded diff formats: Media:CS001006 Reading a text file line by line.diff.zip
- For general information on applying the patch, see Using Diffs.
- For unpatched stub applications, see Example app stubs with logging framework.

Postconditions

The specified text file is output onto the form.

COPY-PASTE the desired platforms into the wiki text and save. All relevant version categories should be applied.
Windows Phone: [[Category:Windows Phone]]
[[Category:Windows Phone 7.5]]
[[Category:Windows Phone 8]]

Nokia Asha: [[Category:Nokia Asha]]
[[Category:Nokia Asha Platform 1.0]]

Series 40: [[Category:Series 40]]
[[Category:Series 40 1st Edition]] [[Category:Series 40 2nd Edition]]
[[Category:Series 40 3rd Edition (initial release)]] [[Category:Series 40 3rd Edition FP1]] [[Category:Series 40 3rd Edition FP2]]
[[Category:Series 40 5th Edition (initial release)]] [[Category:Series 40 5th Edition FP1]]
[[Category:Series 40 6th Edition (initial release)]] [[Category:Series 40 6th Edition FP1]] [[Category:Series 40 Developer Platform 1.0]] [[Category:Series 40 Developer Platform 1.1]] [[Category:Series 40 Developer Platform 2.0]]

Symbian: [[Category:Symbian]]
[[Category:S60 1st Edition]] [[Category:S60 2nd Edition (initial release)]] [[Category:S60 2nd Edition FP1]] [[Category:S60 2nd Edition FP2]]
[[Category:S60 2nd Edition FP3]]
[[Category:S60 3rd Edition (initial release)]] [[Category:S60 3rd Edition FP1]] [[Category:S60 3rd Edition FP2]]
[[Category:S60 5th Edition]]
[[Category:Symbian^3]] [[Category:Symbian Anna]] [[Category:Nokia Belle]]

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.