Create more flexible table in Java ME

Create more flexible table in Java ME

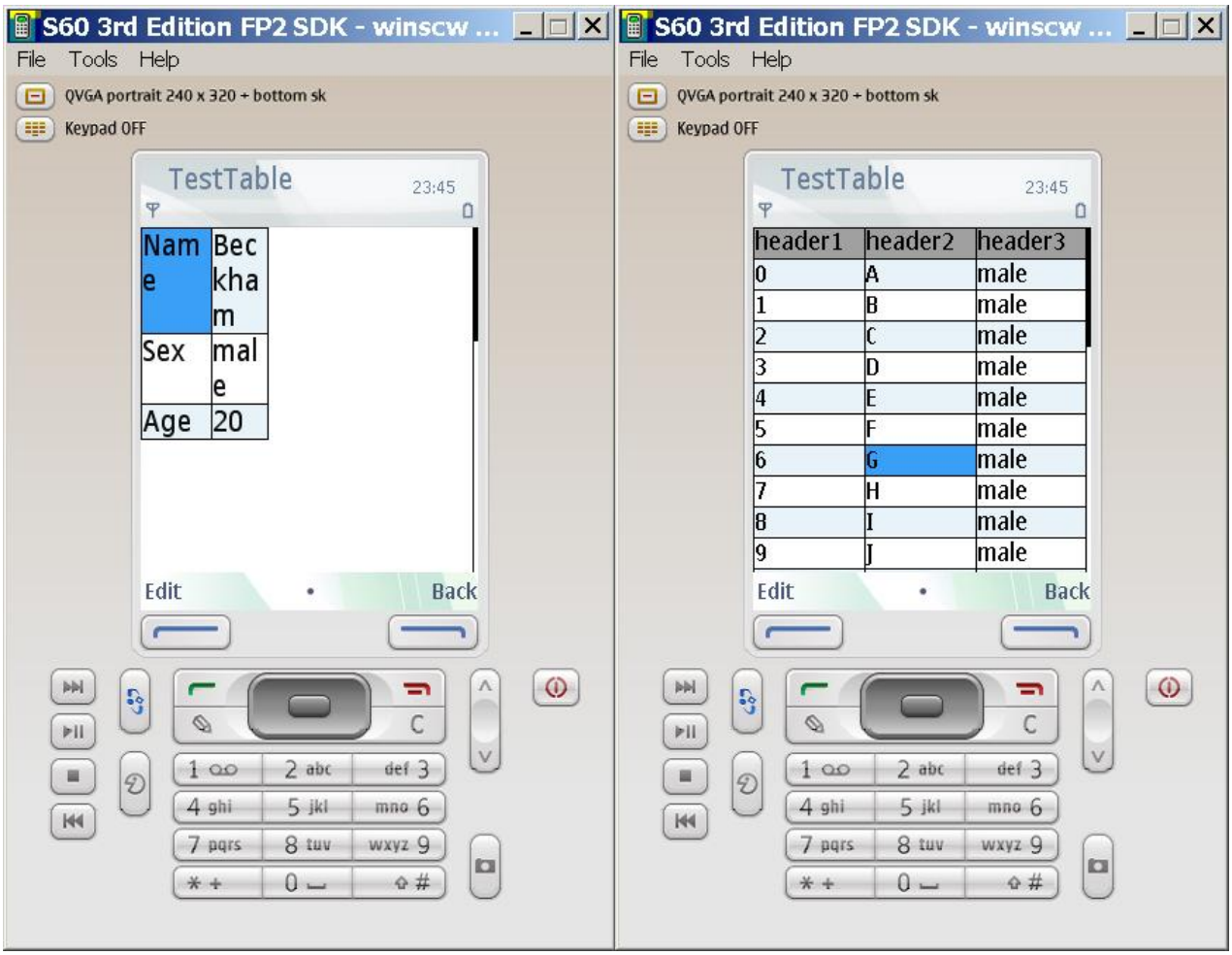
As you perhaps know that there is no table in Java ME. If we want to display data in table, we need to create it ourself. Another choice is NetBeans Mobility Pack. NetBeans Mobility Pack includes a new *CustomItem* that we can add to our project and use to show in a form, a table layout information, like a DataGrid or a spreadsheet. But that table is too weak. It does not support multi-line text in table cell. So I created a more flexible table in Java ME.

This table is based on Canvas where each cell can display multi-line text; each cell can be set as editable; in each cell, content can be text or predefined Option Item.

This table has three claases: *Table*, *CellEditor*, and *TableCell*. Table class draws the UI. *CellEditor* class is a table cell editor. *TableCell* class contain cell's parameter, such as width, height and cell text. Each cell is an instance of this class.

Test application

This table is tested on S60 3rd Edition Feature Pack 2 emulator and Sun Java Wireless Toolkit (WTK) 2.5.2. A small application show how to use this table. The source code and test class can be found at: File:MTable.zip



Source Code

```
package table;

import java.util.Vector;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;

public class Table extends Canvas implements CommandListener {

    private MIDlet midlet;
    private Displayable previousScreen;
    private String tableName;
    private int oneRowHeight;
    private int rowCount = 2;
    private int colCount = 2;
    protected int focusRow = 0;
    protected int focusCol = 0;
```


(C) Copyright Nokia 2013. All rights reserved.

```
    private int rowOffset = 0;
    private int scrollBarWidth = 4;
    private int headerHeight;
    private String header[] = {"a", "b"};
    private int colWidth[] = {118, 118};
    public TableCell cells[][] = {{new TableCell(TableCell.CellType.STRING, "Name", false), new TableCell(TableCell.CellType.STRING,
    private Font font = Font.getDefaultFont();
    private Command editCommand;
    private Command backCommand;
    private CellEditor cellEditor;

    public Table(MIDlet midlet, Displayable previousScreen, String tableName) {
        this.midlet = midlet;
        this.previousScreen = previousScreen;
        this.tableName = tableName;
        initialize();
        repaint();
    }

    private void initialize() {
        setTitle(tableName);
        editCommand = new Command("Edit", Command.ITEM, 1);
        backCommand = new Command("Back", Command.BACK, 1);
        addCommand(editCommand);
        addCommand(backCommand);
        setCommandListener(this);
        calculateHeight();
        repaint();
    }

    /*
     *This method set table data.
     *colWidth contain each column's width.
     *header contain header strings,can be null.
     *cells: all table cells
     *
     */
    public void setData(int colWidth[], String header[], TableCell[][] cells) {
        if (colWidth.length != cells[0].length || (header != null && colWidth.length != header.length)) {
            System.out.println("Invalid Argument.");
            return;
        }
        this.colWidth = colWidth;
        this.cells = cells;
        this.header = header;
        rowCount = cells.length;
        colCount = cells[0].length;
        calculateHeight();
        repaint();
    }
    /*
     * Set table's font
     */
    public void setFont(Font font) {
        this.font = font;
        calculateHeight();
        repaint();
    }

    /*
     *This method calculate all cells' height.
     *Long cell text will be splited into several segments(several lines).
     */
    protected void calculateHeight() {
        oneRowHeight = font.getHeight() + 1;
        if(header==null){
            headerHeight=0;
        }else{
            headerHeight = oneRowHeight;
        }

        int xTemp = 0;
        int yTemp = headerHeight;
        int rowHeight=oneRowHeight;
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < rowCount; i++) {
            rowHeight = oneRowHeight;
            xTemp = 0;

            for (int j = 0; j < colCount; j++) {
                cells[i][j].x = xTemp;
                xTemp += colWidth[j];
                cells[i][j].y = yTemp;
                cells[i][j].width = colWidth[j];

                if (cells[i][j].cellText == null || font.stringWidth(cells[i][j].cellText) < colWidth[j]) {
                    cells[i][j].height = oneRowHeight;
                    cells[i][j].multiLine = false;
                } else {
                    cells[i][j].multiLine = true;
                    cells[i][j].cellStrings = new Vector();
                    sb.setLength(0);
                    for (int k = 0; k < cells[i][j].cellText.length(); k++) {
                        sb.append(cells[i][j].cellText.charAt(k));
                        if (font.stringWidth(sb.toString()) > colWidth[j]) {
                            sb.deleteCharAt(sb.length() - 1);
                            cells[i][j].cellStrings.addElement(sb.toString());
                            sb.setLength(0);
                            sb.append(cells[i][j].cellText.charAt(k));
                        }
                    }
                    if (sb.length() > 0) {
                        cells[i][j].cellStrings.addElement(sb.toString());
                    }
                    cells[i][j].height = oneRowHeight * cells[i][j].cellStrings.size();
                }

                if (rowHeight < cells[i][j].height) {
                    rowHeight= cells[i][j].height;
                }
            }
            for (int j = 0; j < colCount; j++) {
                cells[i][j].height = rowHeight;
            }
            yTemp += cells[i][0].height;
        }
    }

    protected void paint(Graphics g) {
        g.setFont(font);
```


(C) Copyright Nokia 2013. All rights reserved.

```
        //draw table background
        g.setColor(0xffffff);
        g.fillRect(0, 0, this.getWidth(), this.getHeight());

        //draw table border line
        g.setColor(0x000000);
        g.drawLine(0, 0, cells[0][colCount-1].x+cells[0][colCount-1].width, 0);
        g.drawLine(0, 0, 0, cells[rowCount-1][0].y+cells[rowCount-1][0].height);
        g.drawLine(cells[0][colCount-1].x+cells[0][colCount-1].width, 0, cells[0][colCount-1].x+cells[0][colCount-1].width, cells[ro
        g.drawLine(0, cells[rowCount-1][0].y+cells[rowCount-1][0].height, cells[0][colCount-1].x+cells[0][colCount-1].width, cells[r

        //draw cells
        for (int i = 0; i < rowCount; i++) {
            //draw cell background
            if (i % 2 == 0) {
                g.setColor(0xe7f3f8);
                g.fillRect(1, cells[i][0].y - rowOffset + 1, cells[i][colCount - 1].x + cells[i][colCount - 1].width, cells[i][0].he
            }
            g.setColor(0x000000);
            g.drawLine(0, cells[i][0].y - rowOffset + cells[i][0].height, cells[i][colCount - 1].x + cells[i][colCount - 1].width, c

            //draw cell text
            for (int j = 0; j < colCount; j++) {
                //draw single-line text
                if (!cells[i][j].multiLine) {
                    if (cells[i][j].cellText != null) {
                        g.drawString(cells[i][j].cellText, cells[i][j].x + 1, cells[i][j].y - rowOffset + 1, 0);
                    }
                } else {
                    //draw multi-line text
                    for (int a = 0; a < cells[i][j].cellStrings.size(); a++) {
                        g.drawString(cells[i][j].cellStrings.elementAt(a).toString(), cells[i][j].x + 1, cells[i][j].y + oneRowHeigh
                    }
                }
            }
        }
        //draw table header
        if (header != null) {
            g.setColor(0xA0A0A0);
            g.fillRect(1, 1, cells[0][colCount - 1].x + cells[0][colCount - 1].width, headerHeight);
            g.setColor(0x000000);
            g.drawLine(0, 0, cells[0][colCount - 1].x + cells[0][colCount - 1].width, 0);
            g.drawLine(0, headerHeight, cells[0][colCount - 1].x + cells[0][colCount - 1].width, headerHeight);
            for (int i = 0; i < header.length; i++) {
                g.drawString(header[i], cells[0][i].x + 1, 0, 0);
            }
        }

        //draw vertical line
        int temp = 0;
        for (int i = 0; i < colWidth.length; i++) {
            temp += colWidth[i];
            g.drawLine(temp, 0, temp, cells[rowCount - 1][0].y + cells[rowCount - 1][0].height);
        }

        //draw scrollbar
        g.drawLine(this.getWidth() - scrollBarWidth, 0, this.getWidth() - scrollBarWidth, this.getHeight() - 1);
        g.fillRect(this.getWidth() - scrollBarWidth, 0, scrollBarWidth, ((int) (this.getHeight() * (focusRow + 1) / rowCount)));

        //draw focus cell
        g.setColor(0x3a9ff7);
        g.fillRect(cells[focusRow][focusCol].x + 1, cells[focusRow][focusCol].y - rowOffset + 1, cells[focusRow][focusCol].width - 1
        g.setColor(0x000000);
        if (!cells[focusRow][focusCol].multiLine) {
            if (cells[focusRow][focusCol].cellText != null) {
                g.drawString(cells[focusRow][focusCol].cellText, cells[focusRow][focusCol].x + 1, cells[focusRow][focusCol].y - rowO
            }
        } else {
            for (int i = 0; i < cells[focusRow][focusCol].cellStrings.size(); i++) {
                g.drawString(cells[focusRow][focusCol].cellStrings.elementAt(i).toString(), cells[focusRow][focusCol].x + 1, cells[f
            }
        }
    }

    public void commandAction(Command com, Displayable display) {
        if (com == backCommand) {
            Display.getDisplay(midlet).setCurrent(previousScreen);
        } else if (com == editCommand) {
            if (cells[focusRow][focusCol].editable) {
                editCell(cells[focusRow][focusCol]);
            }
        }
    }

    private void editCell(TableCell cell) {
        if (cellEditor == null) {
            cellEditor = new CellEditor(midlet, this, "");
        }
        cellEditor.setCell(cell);
        Display.getDisplay(midlet).setCurrent(cellEditor);
    }

    public void keyPressed(int keyCode) {
        switch (keyCode) {
            case -3: //left

                focusCol--;
                if (focusCol <= 0) {
                    focusCol = 0;
                }
                break;

            case -4: //right

                focusCol++;
                if (focusCol >= colCount - 1) {
                    focusCol = colCount - 1;
                }

                break;

            case -1:  //up

                focusRow--;
                if (focusRow <= 0) {
                    focusRow = 0;
                }
                if (cells[focusRow][0].y < rowOffset+headerHeight) {
                    rowOffset = cells[focusRow][0].y-headerHeight;
                }
```


(C) Copyright Nokia 2013. All rights reserved.

```
                }
                break;

            case -2: //down

                focusRow++;
                if (focusRow >= rowCount - 1) {
                    focusRow = rowCount - 1;
                }

                if (cells[focusRow][0].y + cells[focusRow][0].height-rowOffset> this.getHeight()) {
                    rowOffset = cells[focusRow][0].y + cells[focusRow][0].height - this.getHeight();
                }

                break;
            case 13:
                if (cells[focusRow][focusCol].editable) {
                    editCell(cells[focusRow][focusCol]);
                }
                break;
        }
        repaint();
    }
}
```

```
package table;

import java.util.Vector;

public class TableCell {

    public static class CellType {
        public static final int STRING = 0;
        public static final int NUMBER = 1;
        public static final int ITEM = 2;
    }

    int x;
    int y;
    int width;
    int height;

    int cellType;
    boolean editable = false;
    boolean multiLine = false;
    public String cellText;
    /*
     *Long cell text will be splited into several segments(several lines).
     *This Vecotr contains all segments.
     */
    public Vector cellStrings;
    /*
     *If a cell's type is CellType.ITEM,itemOptions must be set.
     */
    public String[] itemOptions;

    public TableCell() {
        this(CellType.STRING,"",true);
    }

    public TableCell(int cellType, String cellText, boolean editable) {
        this.cellType = cellType;
        this.cellText = cellText;
        this.editable = editable;
    }

    public TableCell(int cellType, String cellText, String[] itemOption, boolean editable) {
        this.cellType = cellType;
        this.cellText = cellText;
        this.itemOptions = itemOption;
        this.editable = editable;
    }

    public String toString() {
        return "TableCell: x=" + x + ",y=" + y + ",width=" + width + ",height=" + height+",cellText="+cellText;
    }
}
```

```
package table;

import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;


public class CellEditor extends Form implements CommandListener {

    private MIDlet midlet = null;
    private Table previousScreen = null;
    private Command backCommand = null;
    private Command OKCommand = null;
    private TextField textField;
    private ChoiceGroup choiceGroup;
    private TableCell cell;

    public CellEditor(MIDlet midlet, Table previousScreen, String title) {
        super(title);
        this.midlet = midlet;
        this.previousScreen = previousScreen;
        initialize();
    }

    private void initialize() {
        backCommand = new Command("Back", Command.BACK, 1);
        OKCommand = new Command("OK", Command.OK, 1);
        addCommand(backCommand);
        addCommand(OKCommand);
        setCommandListener(this);
    }

    public void setCell(TableCell cell) {
        this.cell = cell;
        deleteAll();
        if (cell.cellType == TableCell.CellType.STRING) {
            textField = getTextField();
```


(C) Copyright Nokia 2013. All rights reserved.

```
            textField.setConstraints(TextField.ANY);
            textField.setString(cell.cellText);
            append(textField);
        } else if (cell.cellType == TableCell.CellType.NUMBER) {
            textField = getTextField();
            textField.setConstraints(TextField.NUMERIC);
            textField.setString(cell.cellText);
            append(textField);
        } else {
            choiceGroup=getChoiceGroup();
            choiceGroup.deleteAll();
            for (int i = 0; i < cell.itemOptions.length; i++) {
                choiceGroup.append(cell.itemOptions[i], null);
                if (cell.cellText.equals(cell.itemOptions[i])) {
                    choiceGroup.setSelectedIndex(i, true);
                }
            }
            append(choiceGroup);
        }
    }

    public TextField getTextField() {
        if (textField == null) {
            textField = new TextField("", "", 300, TextField.ANY);
        }
        return textField;
    }

    public ChoiceGroup getChoiceGroup() {
        if (choiceGroup == null) {
            choiceGroup = new ChoiceGroup("", ChoiceGroup.EXCLUSIVE);
        }
        return choiceGroup;
    }

    public void commandAction(Command com, Displayable display) {

        if (com == backCommand) {
            Display.getDisplay(midlet).setCurrent(previousScreen);
        } else if (com == OKCommand) {
            if (cell.cellType == TableCell.CellType.ITEM) {
                previousScreen.cells[previousScreen.focusRow][previousScreen.focusCol].cellText = choiceGroup.getString(choiceGroup.
            } else {
                previousScreen.cells[previousScreen.focusRow][previousScreen.focusCol].cellText = textField.getString();
            }
            previousScreen.calculateHeight();
            Display.getDisplay(midlet).setCurrent(previousScreen);
        }
    }
}
```

--loveboylxy 19:24, 30 May 2008 (EEST)

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.