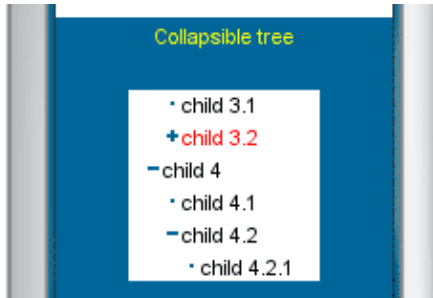




This code will show how to draw trees with J2me with the following features:

- collapsible/expandable nodes
- vertical scrolling



You can see a sample midlet showing this code in action on [this page](#).

MenuNode class

To start, we need a class to represent single tree nodes. This class will have also features related to its graphic appearance: even if it's not a best practice, we'll do that to reduce the number of classes within our J2me code.

```
import java.util.Vector;

public class MenuNode
{
    /* graphic properties */
    public int x = 0;
    public int y = 0;

    /* node properties */
    public MenuNode parentNode = null;
    public String label = null;
    public boolean expanded = false;

    public int index = 0;

    Vector children = null;

    public MenuNode(String label)
    {
        this.label = label;

        this.children = new Vector();
    }
    public void appendChild(MenuNode node)
    {
        parentNode = this;

        index = this.children.size();

        this.children.addElement(node);
    }
    public int getChildrenNum()

```

Drawing_Collapsible_Trees_in_Java_ME

```
{
return this.children.size();
}
public MenuNode getNextSibling()
{
if(parentNode != null)
{
return parentNode.getChild(index + 1);
}
return null;
}
public MenuNode getPrevSibling()
{
if(parentNode != null)
{
return parentNode.getChild(index - 1);
}
return null;
}
public MenuNode getLastChild()
{
return getChild(getChildrenNum() - 1);
}
public MenuNode getChild(int i)
{
if(i >= 0 && i < this.children.size())
{
return (MenuNode)this.children.elementAt(i);
}
return null;
}
public void removeChildren(int index)
{
this.children.removeElementAt(index);

for(int i = index; i < this.children.size(); i++)
{
this.getChild(i).index--;
}
}
public boolean hasChildren()
{
return this.getChildrenNum() > 0;
}
public void expand()
{
if(this.getChildrenNum() > 0)
{
this.expanded = true;
}
}
public void collapse()
{
this.expanded = false;
}
}
```

TreeMenu

Now, we'll write down our TreeMenu component, that will accept a root node, width and height within its constructor, and then will manage user key actions through the keyAction() method. The only other public method will be paint(), which will paint our tree on the specified Graphic object.

```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Font;
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.Image;

public class ScrollableIconTreeMenu
{
    Font font = Font.getDefaultFont();
    int foreColor = 0x000000;
    int foreFocusedColor = 0xff0000;

    int viewportHeight = 0;
    int viewportWidth = 0;
    int viewportY = 0;

    int nodeHeight = 0;

    int childPadding = 10;
    int verticalPadding = 2;

    MenuNode root;
    MenuNode current;

    Image expandIcon = null;
    Image collapseIcon = null;
    Image leafIcon = null;

    public static final int ICON_WIDTH = 6;
    public static final int ICON_HEIGHT = 6;

    public ScrollableIconTreeMenu(MenuNode root, int width, int height)
    {
        try
        {
            expandIcon = Image.getImageClass().getResourceAsStream("/plus.png");
            collapseIcon = Image.getImageClass().getResourceAsStream("/minus.png");
            leafIcon = Image.getImageClass().getResourceAsStream("/leaf.png");

            nodeHeight = Math.max(ICON_HEIGHT, font.getHeight());
        }
        catch (Exception e)
        {

        }

        this.viewportHeight = height;
        this.viewportWidth = width;

        this.root = root;

        this.current = root;
    }

    public void keyAction(int key)
```

Drawing_Collapsible_Trees_in_Java_ME

```
{
switch(key)
{
case Canvas.LEFT:
    ();          keyLeft
break;
case Canvas.UP:
    ();          keyUp
break;
case Canvas.RIGHT:
    ();          keyRight
break;
case Canvas.DOWN:
    ();          keyDown
break;
}
}

void keyLeft()
{
    MenuNode; nextNode

if(current.expanded)
{
    collapse();    current.
}
else
{
    = current.parentNode
}
    (nextNode; nextNode
}
void keyRight()
{
    MenuNode; nextNode

if(current.expanded)
{
    = current.getChildNode
}
else
{
    expand();      current.
}
    (nextNode; nextNode
}
void keyDown()
{
    MenuNode; nextNode

if(current.expanded)
{
    = current.getChildNode
}
if(nextNode == null)
{
    MenuNode; nextNode; searchingSibling

while(searchingSibling != null && nextNode == null)
{
    = searchingSibling.getNextSibling();
}
```

Drawing_Collapsible_Trees_in_Java_ME

```

        = searchingSibling.getPrevSibling();
    }
}

    (nextNode, parentNode
}
void keyUp()
{
    MenuNode.nextNode.getPrevSibling();

if(nextNode == null)
{
    = current.parentNode
}
else
{
while(nextNode.expanded)
{
    = nextNode.getLastChild(nextNode
}
}

    (nextNode, parentNode
}

void setCurrentNode(MenuNode node)
{
if(node != null)
{
this.current = node;

if(this.current.y < this.viewportY)
{
this.viewportY = this.current.y;
}
else if(this.current.y + font.getHeight() > this.viewportY + this.viewportHeight)
{
this.viewportY = this.current.y + font.getHeight() - this.viewportHeight;
}
}
}

public void paint(Graphics g)
{
int cx, cy, cw, ch;

    = g.getClipX(cx
    = g.getClipY(cy
    = g.getClipWidth(cw);
    = g.getClipHeight(ch);

setClip(0, 0, viewportWidth, viewportHeight);

translate(0, -viewportY);

    (g, root.parentNode
translate(0, viewportY);

setClip(cx, cy, cw, ch);
}
int paintNode(Graphics g, MenuNode node, int left, int top)
{

```

Drawing_Collapsible_Trees_in_Java_ME

```
x = left;  node.
y = top;   node.

Image icon = node.hasChildren() ?
(node.expanded ? collapseIcon : expandIcon) :
    ;           leafIcon

drawImage(icon.g.left, top + nodeHeight / 2, Graphics.LEFT | Graphics.VCENTER);

if(node == current)
{
    setColor(foreFocusedColor);
}
else
{
    setColor(foreColor);  g.
}
drawString(node.g.label, left + 2 + ICON_WIDTH, top + (nodeHeight - font.getHeight()) / 2, Graphics.LEFT | Graphics.VCENTER);

int nodeHeight = 0;

    += font.getHeight() + verticalPadding;

if(node.expanded)
{
    += childPadding;  left

int childrenNum = node.getChildrenNum();

for(int i = 0; i < childrenNum; i++)
{
    += paintNode(g, node.getChildren(i), left, top + nodeHeight);
}
    -= childPadding;  left
}
return nodeHeight;
}
}
```