Extending LWUIT layouts on Series 40

Note: This is an entry in the Asha Touch Competition 2012Q3

This code example shows how to extend the LWUIT framework to create new UI layouts.

Introduction

Lightweight UI Toolkit (LWUIT) is a UI framework and "widget" library for Java ME. It comes with a number of in-built Layouts, which allow you to organize components such as buttons, labels, checkbox, radiobuttons etc on the screen inside a container. The most common layouts include:

- BoxLayout - allows to organize the components in a linear way either along the X axis or the Y axis
- BorderLayout - allows to organize the components in 5 different locations North,South,Center,East,West
- GridLayout - allows to organize the components in a grid of cells.
- FlowLayout,TableLayout,etc

The default layouts are more than sufficient for most apps, but if you want to define your own cool and unique layout style, LWUIT allows you to extend the basic Layout class and implement your own organization of the components inside of the container.

This code example shows you how to create a new Layout called `CircularLayout`, which organizes its components in a circle around a central point. This sort of organization is almost impossible to achieve using the build-in layouts.

Implement the abstract classes

When extending the Layout class you need to implement the two abstract methods:

```
public abstract Dimension getPreferredSize(Container parent);
public abstract void layoutContainer(Container parent);
```

Both methods receive the parent `Container` parameter which they are associated with. The `getPreferredSize()` method should return the preffered size in pixels of Width x Height the required layout. The `ayoutContainer()` method does the actual layout of the components inside the parent `Container` (which basically means setting the components sizes and location (x,y) relative to the container location).

Source code for the layout

(C) Copyright Nokia 2013. All rights reserved.

```
import com.sun.lwuit.Component;
import com.sun.lwuit.Container;
import com.sun.lwuit.geom.Dimension;
import com.sun.lwuit.layouts.Layout;
import com.sun.lwuit.plaf.Style;

/**
 * @author ifrach shai
 */
public class CircularLayout extends Layout {

 private int startAngel;
 private int clockWise;

 public CircularLayout()
 {
  this(270, true);
 }

 public CircularLayout(int startAngel, boolean clockWise)
 {
  this.startAngel = startAngel;
  this.clockWise = clockWise ? -1 : 1;
 }

 /**
 * returns the preffered size the layout needs in order to properly arrange the components
 */
 public Dimension getPreferredSize(Container parent)
 {
  int components = parent.getComponentCount();
  double largestDiameter = 0;
  for (int iter = 0; iter < components; iter++)
  {
   Component current = parent.getComponentAt(iter);
   Dimension d = current.getPreferredSize();
   largestDiameter = Math.max(largestDiameter, Math.sqrt(d.getWidth() * d.getWidth() + d.getHeight()
     * d.getHeight()));
  }
  double angelPart = 360f / components;
  double otherAngel = (180 - angelPart) / 2;
  int radius = (int) (largestDiameter * Math.sin(Math.toRadians(otherAngel)) / Math
    .sin(Math.toRadians(angelPart)));
  return new Dimension((int) (radius * 2 + largestDiameter), (int) (radius * 2 + largestDiameter));

 }

 /**
 * this method arranges the components within the container by setting their relative x,y values as well as their width,height
 * even if the size allocated to this container is smaller than the one returned in the getPrefferedSize method this method will sti
 * the components in a nice circle but there will be some overlapping.
 */
 public void layoutContainer(Container parent)
 {
  int components = parent.getComponentCount();
  double angelPart = 360f / components;
  Style parentStyle = parent.getStyle();
  int centerPosX = (parent.getLayoutWidth() - parentStyle.getMargin(Component.LEFT) - parentStyle
    .getMargin(Component.RIGHT))
    / 2 + parentStyle.getMargin(Component.LEFT);
  int centerPosY = (parent.getLayoutHeight() - parentStyle.getMargin(Component.TOP) - parentStyle
    .getMargin(Component.BOTTOM))
    / 2 + parentStyle.getMargin(Component.TOP);
  double largestDiameter = 0;
  for (int iter = 0; iter < components; iter++)
  {
   Component current = parent.getComponentAt(iter);
   Dimension d = current.getPreferredSize();
   largestDiameter = Math.max(largestDiameter, Math.sqrt(d.getWidth() * d.getWidth() + d.getHeight()
     * d.getHeight()));
  }
  int radius = (int) (Math.min(parent.getLayoutWidth() - largestDiameter, parent.getLayoutHeight()
    - largestDiameter) / 2);
  for (int iter = 0; iter < components; iter++)
  {
   Component current = parent.getComponentAt(iter);
   Dimension d = current.getPreferredSize();
   current.setSize(d);
   double radianAgnel = Math.toRadians(startAngel + clockWise * iter * angelPart);
   current.setX(centerPosX + (int) (radius * Math.cos(radianAgnel)) - current.getWidth() / 2);
   current.setY(centerPosY + (int) (radius * Math.sin(radianAgnel)) - current.getHeight() / 2);
  }

 }

}
```

Screenshots

The images below shows how different number of components are arrange in the center container using the `CircularLayout`.

(C) Copyright Nokia 2013. All rights reserved.

3 components

4 components

5 components

6 components

7 components

8 components

Summary

In this article we focused on creating a new layout for LWUIT, you can think of other layouts you wish to have and implement them yourself to create new and exciting UI for your apps.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.