Designing Series 40 apps with Nokia UI API and Canvas for Great User Experiences

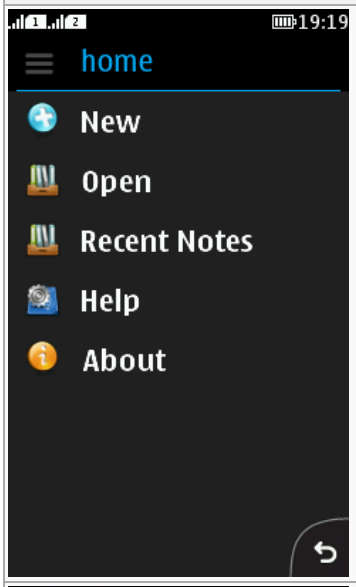
This article gives *Some Ideas! to design UI of Series 40 JavaME Apps for THE NEXT GENERATION FEATURE PHONE. I am considering only Series 40 full touch phones.*

Before going into this article, please take a tour on Nokia UI/Series 40 Full Touch UI. This will help you to understand easily.

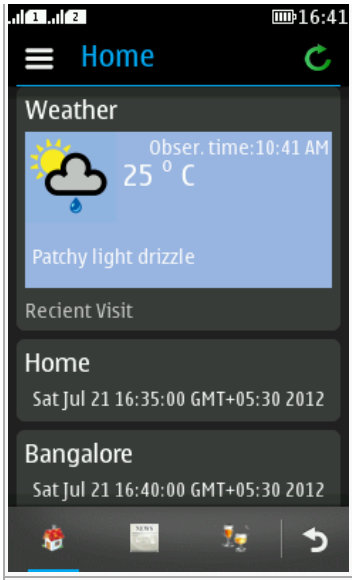
Note: This is an entry in the Asha Touch Competition 2012Q3

Introduction

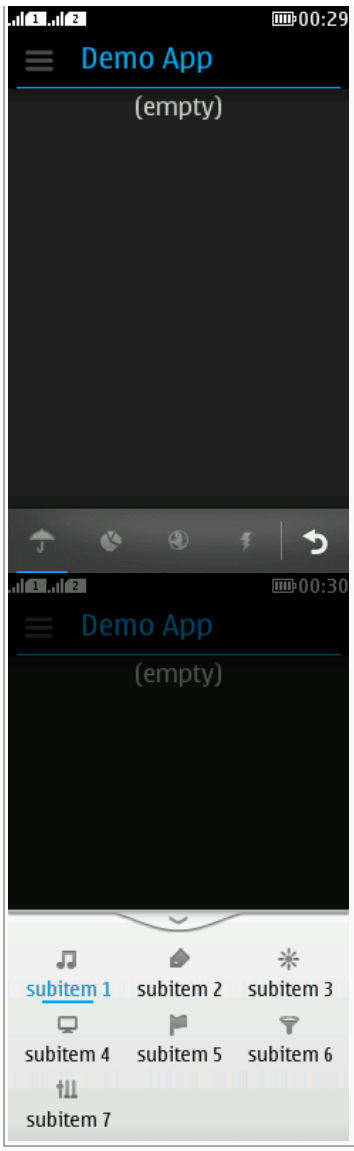
Do you think user will be bored of using default UI (LCDUI default elements). You can make your application UI still better with LCDUI and Canvas.

Image	Description
	Application developed with pure LCDUI components. Skills: Beginner
	Metro UI - Includes Canvas drawing. Skills: Expert

(C) Copyright Nokia 2014. All rights reserved.

	Tab View using LCDUI and Nokia UI API - Icon Command - "Refresh Button" and Category Bar - "Tab Selector". Skills: Beginner to Intermediate
	Dual Work-space application with Tab View!! - makes use of Orientation API. Skills: Beginner to Intermediate

(C) Copyright Nokia 2014. All rights reserved.

	Menu and Sub Menu system - makes use of Category Bar API. Skills: Beginner to Intermediate

You can make out the differences in UI. You can clearly make out the difference in UI of application developed with pure LCDUI components and other UI. Except Metro UI, all other UI development requires basic to intermediate skills. Metro UI requires expert/advance skills, as it includes pure canvas drawing and manipulation of pixels.

The Concept of Tab View design

Developing Tab view application is simple. It requires basic skills of developing JavaME application as we will be using pure LCDUI components with Category Bar.
We will be using category bar. To know more about using category bar - Category Bar

Creating and handling Category Bar and Elements Listener

To produce Tab View effect, we will switch the display between various forms, canvas.

```
/* Creating Category bar */
CategoryBar portraitBar;
```

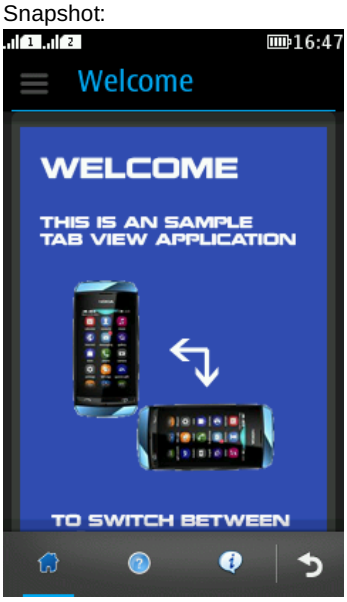
```
/* Initializing Category Bar */
try{
   Image[] unsel = { Image.createImage("/home.png"),
     Image.createImage("/help.png"),
     Image.createImage("/about.png") };
   Image[] sel = { Image.createImage("/home.png"),
     Image.createImage("/help.png"),
     Image.createImage("/about.png") };
   String[] lab = { "Home", "Help", "About" };
   portraitBar = new CategoryBar(unsel, sel, lab);
```


(c) Copyright Nokia 2014. All rights reserved.

```
    portraitBar.setElementListener(this);
    portraitBar.setTransitionSupport(true);
}catch(Exception ex){}
```

```
/* Category Bar Element Listener to switch between displays */
public void notifyElementSelected(CategoryBar c, int item) {
  if(c == portraitBar){
   switch(item){
   case 0:
    Display.getDisplay(this).setCurrent(welcomescreen);
    break;
   case 1:
    Display.getDisplay(this).setCurrent(helpscreen);
    break;
   case 2:
    Display.getDisplay(this).setCurrent(aboutscreen);
    break;
   }
  }
}
```

Snapshot:



Note: You should always keep in mind that: In Tab View application design, there is no back button to goto previous display. However, this is an exceptional case for inner hierarchical structure. i.e node level of display tree should not be *root node/level 0/base*. Application should exit/return to home screen, on back key pressed only if it is *root/level 0/base* node of the display tree.

The Concept of Dual Work-space view

I will extend the *Tab View* application and I will add some functionality of Orientation to the application. Since I will be using Tab view concept here also; I will call this as *Concept of dual work-space with tab view*.

This requires some Intermediate Skills to understand.

To be clear, The whole application is split in to two parts. Portrait and Landscape and both are independent of each other. However, A Form or a Canvas cannot have both Portrait and Landscape mode displays support. Switching between Portrait to Landscape or Landscape to portrait - switches the work-space.

To get started with Orientation API - Orientation

2 Category Bars

2 bars - 1 for landscape and another for portrait. It will be easy to switch between bar rather than updating the elements. :)

```
/* creating */
CategoryBar portraitBar; //for portrait mode display
```


(C) Copyright Nokia 2014. All rights reserved.

```
CategoryBar landscapeBar; //for landscape mode display
```

```
/* initializing */
try{
   Image[] unsel = { Image.createImage("/home.png"),
     Image.createImage("/help.png"),
     Image.createImage("/about.png") };
   Image[] sel = { Image.createImage("/home.png"),
     Image.createImage("/help.png"),
     Image.createImage("/about.png") };
   String[] lab = { "Home", "Help", "About" };
   portraitBar = new CategoryBar(unsel, sel, lab);
   portraitBar.setElementListener(this);
   portraitBar.setTransitionSupport(true);
  }catch(Exception ex){}
  try{
   Image[] unsel = { Image.createImage("/tab_unsel.png"),
     Image.createImage("/tab_unsel.png"),
     Image.createImage("/tab_unsel.png"),
     Image.createImage("/tab_unsel.png"),
     Image.createImage("/tab_unsel.png")};
   Image[] sel = { Image.createImage("/tab_sel.png"),
     Image.createImage("/tab_sel.png"),
     Image.createImage("/tab_sel.png"),
     Image.createImage("/tab_sel.png"),
     Image.createImage("/tab_sel.png")};
   String[] lab = { "Tab 1", "tab 2", "Tab 3", "Tab 4", "Tab 5" };
   landscapeBar = new CategoryBar(unsel, sel, lab);
   landscapeBar.setElementListener(this);
   landscapeBar.setTransitionSupport(true);
}catch(Exception ex){}
```

```
/* detection of element selection */
public void notifyElementSelected(CategoryBar c, int item) {
  if(c == portraitBar){
   switch(item){
   case 0:
    Display.getDisplay(this).setCurrent(welcomescreen);
    break;
   case 1:
    Display.getDisplay(this).setCurrent(helpscreen);
    break;
   case 2:
    Display.getDisplay(this).setCurrent(aboutscreen);
    break;
   }
  }
  else if(c == landscapeBar){
   switch(item){
   case 0:
    Display.getDisplay(this).setCurrent(f1);
    break;
   case 1:
    Display.getDisplay(this).setCurrent(f2);
    break;
   case 2:
    Display.getDisplay(this).setCurrent(f3);
    break;
   case 3:
```


(c) Copyright Nokia 2014. All rights reserved.

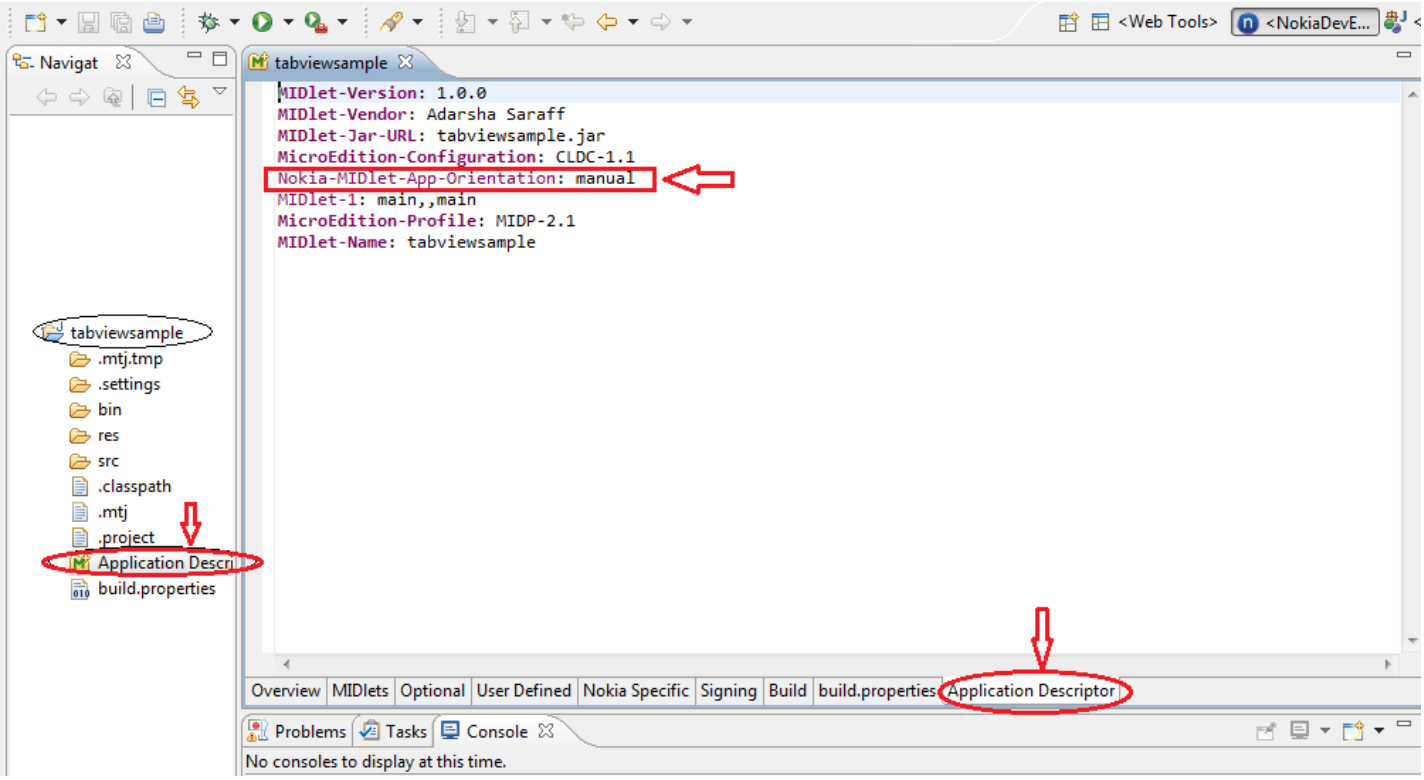
```
    Display.getDisplay(this).setCurrent(f4);
    break;
   case 4:
    Display.getDisplay(this).setCurrent(f5);
    break;
   }
  }

}
```

Orientation Detection

Setting up "Application Descriptor". In Nokia Eclipse Integrated IDE: double click on *Application descriptor* in that select *Application descriptor* tab and add the following.

```
Nokia-MIDlet-App-Orientation: manual
```



Steps to be performed. Your MIDlet should implement *OrientationListener* in order to detect screen orientation.

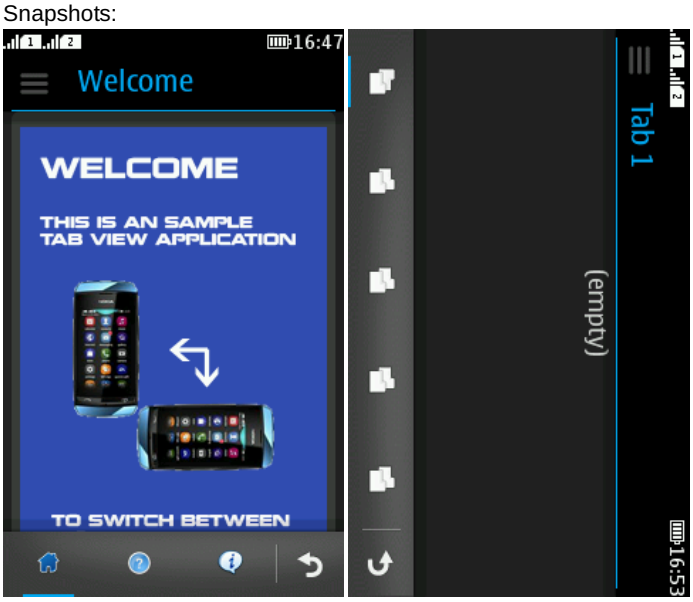
```
/* registering orientation listener */
Orientation.addOrientationListener(this);
```

```
/* detection of current orientation and switching between the work-space */
public void displayOrientationChanged(int mode) {
  switch (mode) {
  case Orientation.ORIENTATION_PORTRAIT:
  case Orientation.ORIENTATION_PORTRAIT_180:
   Orientation.setAppOrientation(Orientation.ORIENTATION_PORTRAIT);
   if(landscapeBar.getVisibility()){
    landscapeBar.setVisibility(false);
   }
   portraitBar.setVisibility(true);
   Display.getDisplay(this).setCurrent(welcomescreen);
   break;
  case Orientation.ORIENTATION_LANDSCAPE:
  case Orientation.ORIENTATION_LANDSCAPE_180:
```


(c) Copyright Nokia 2014. All rights reserved.

```
    Orientation.setAppOrientation(Orientation.ORIENTATION_LANDSCAPE);
    if(portraitBar.getVisibility()){
     portraitBar.setVisibility(false);
    }
    landscapeBar.setVisibility(true);
    Display.getDisplay(this).setCurrent(f1);
   default:
    break;
   }
  }
```

Snapshots:



Note: This also inherits the principle of Tab view.

Metro UI

Metro UI is an design language designed by Microsoft.
I have already written an article on Metro UI. To understand this, you require intermediate to expert skills in JavaME. Please follow the link to find out more.Developing Metro or Panorama UI for Series 40 full touch

Menu and Sub menu with Category Bar

We will be using more than 1 category bar. Each for menu, we will create a separate Category Bar.
To create and initialize Category Bar:

```
/* declaration */
CategoryBar menu, subMenu1, subMenu2;
/* initialization */
try {
   Image[] unsel = { Image.createImage("/m1i1.png"),
     Image.createImage("/m1i2.png"),
     Image.createImage("/m1i3.png"),
     Image.createImage("/m1i4.png") };
   Image[] sel = { Image.createImage("/m1i1.png"),
     Image.createImage("/m1i2.png"),
     Image.createImage("/m1i3.png"),
     Image.createImage("/m1i4.png") };
   String[] lab = { "item 1", "item 2", "item 3", "item 4" };
   menu = new CategoryBar(unsel, sel, lab);
   menu.setTransitionSupport(false);
   menu.setElementListener(this);
  } catch (Exception ex) {
  }
  try {
```


(c) Copyright Nokia 2014. All rights reserved.

```
    Image[] unsel = { Image.createImage("/m2i1.png"),
      Image.createImage("/m2i2.png"),
      Image.createImage("/m2i3.png"),
      Image.createImage("/m2i4.png"),
      Image.createImage("/m2i5.png"),
      Image.createImage("/m2i6.png"),
      Image.createImage("/m2i7.png") };
    Image[] sel = { Image.createImage("/m2i1.png"),
      Image.createImage("/m2i2.png"),
      Image.createImage("/m2i3.png"),
      Image.createImage("/m2i4.png"),
      Image.createImage("/m2i5.png"),
      Image.createImage("/m2i6.png"),
      Image.createImage("/m2i7.png") };
    String[] lab = { "subitem 1", "subitem 2", "subitem 3",
      "subitem 4", "subitem 5", "subitem 6", "subitem 7" };
    subMenu1 = new CategoryBar(unsel, sel, lab);
    subMenu1.setTransitionSupport(true);
    subMenu1.setElementListener(this);
   } catch (Exception ex) {
   }
   try {
    Image[] unsel = { Image.createImage("/m3i1.png"),
      Image.createImage("/m3i2.png"),
      Image.createImage("/m3i3.png"),
      Image.createImage("/m3i4.png"),
      Image.createImage("/m3i5.png") };
    Image[] sel = { Image.createImage("/m3i1.png"),
      Image.createImage("/m3i2.png"),
      Image.createImage("/m3i3.png"),
      Image.createImage("/m3i4.png"),
      Image.createImage("/m3i5.png") };
    String[] lab = { "subitem 1", "subitem 2", "subitem 3",
      "subitem 4", "subitem 5" };
    subMenu2 = new CategoryBar(unsel, sel, lab);
    subMenu2.setTransitionSupport(true);
    subMenu2.setElementListener(this);
   } catch (Exception ex) {
   }
```

Controlling Category bar corresponding to Element selected.

```
 public void notifyElementSelected(CategoryBar c, int element) {
   if (c == menu) {
    switch (element) {
    case 2:
     menu.setVisibility(false);
     subMenu1.setVisibility(true);
     break;
    }
   } else if (c == subMenu1) {
    switch (element) {
    case 0:
     subMenu1.setVisibility(false);
     subMenu2.setVisibility(true);
     break;
    default:
     subMenu1.setVisibility(false);
     menu.setVisibility(true);
     break;
```


(c) Copyright Nokia 2014. All rights reserved.

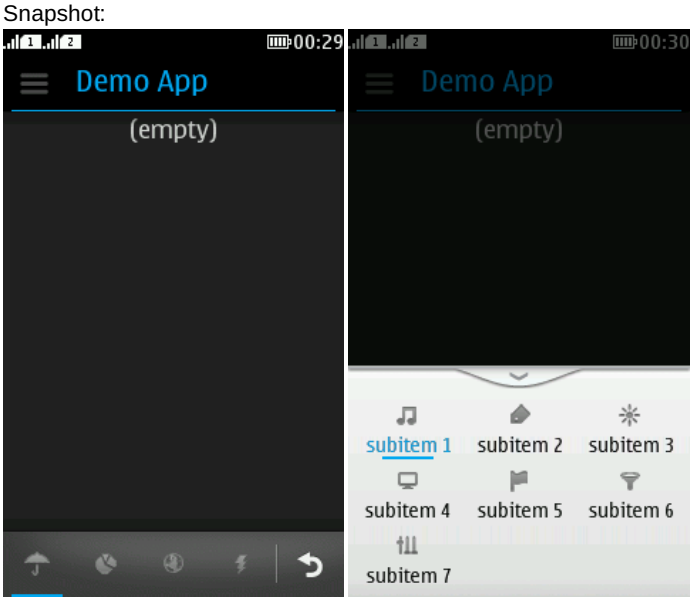
```
    }

   } else if (c == subMenu2) {
    switch (element) {
    default:
     subMenu2.setVisibility(false);
     menu.setVisibility(true);
     break;
    }

   }
 }
```

Tip: You can even make one of sub menu item as a back button to return to previous menu, i.e main menu.

Snapshot:



Source

File:Tabviewsample.zip
File:MenuSubmenu.zip
File:Refreshment.zip

(c) Copyright Nokia 2014. All rights reserved.