Lab 5: Creating User Interfaces II

Objective

The objective to this lab is to get experience with:

1. BlackBerry User interface APIs
2. Displaying user interface components including Menu Item, Bitmap field and List field.
3. Handling the relevant events in user interface components
4. Develop a Photo Album application using these components

Experiment 1 Develop a Photo Album application using User Interface components

In this experiment, you will learn how to write an application for BlackBerry devices using the BlackBerry User Interface APIs. You will develop a Photo Album application and get experience with Menu, MenuItem, BitmapField and ListField. Figures 1 demonstrate how your application should appear on the screen.

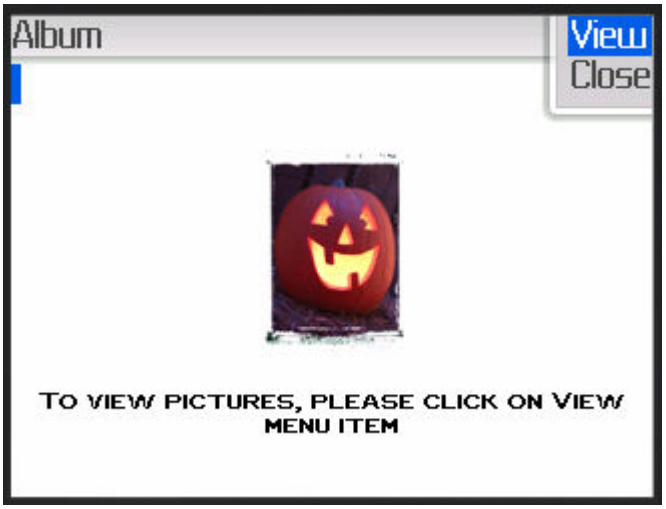

Figure 1- First screen

To start the experiment, do the following steps:

1. Create a class extended from **net.rim.device.api.ui.UiApplication** class. Name it "Album_App".
2. Write the constructor and main() methods and add the required codes to these methods.
3. Create the first screen class and name it "FirstScreen". Figure 1 depicts how it should look like.

```
final class FirstScreen extends MainScreen
   {
        public FirstScreen() {  }
}
```

4. FirstScreen consists of the following UI components: RichTextField for displaying "To view pictures, please click on View menu item", LabelField for the title "Album", Menu and MenuItem for building the menu. You already got experience with RichTextField and LabelField. Thus, create an instance of these elements in the constructor.

```
public FirstScreen()
{
      super();
      LabelField title = new LabelField("Album", LabelField.ELLIPSIS);
      setTitle(title);

      add(new RichTextField(""));
      add(new RichTextField(""));

     /*Create a BitmapField here */

      add(new RichTextField(""));

      Font font = Font.getDefault();
      FontFamily fontFamily[] = FontFamily.getFontFamilies();
      Font font1 = fontFamily[0].getFont(font.getStyle(),12);
      Font fonts[] = new Font[1];
      fonts[0] = font1;

      add( new RichTextField("To view pictures, please click on View menu item" ,
          null, null, fonts,  RichTextField.TEXT_ALIGN_HCENTER));
}
```

5. BitmapField displays a bitmap. In order to show a bitmap, several tasks should be done:
   - First, you must construct a bitmap object. It is done using the method in Bitmap Class as follows:
     ```
     Bitmap myBitmap =  Bitmap.getBitmapResource("pic1.png");
     ```
   - Second, you must construct a new BitmapField instance with provided bitmap in previous step.
     ```
     BitmapField myBitmapField =
                              new BitmapField(myBitmap, BitmapField.FIELD_HCENTER);
     ```
   - Third, you must add the image to the project by right clicking Album_App.java and select the 'Add File to…' and add the image you will be using.
   - Last, add the created BitmapField instance to the screen.
     ```
     add(myBitmapField);
     ```

6. The next step is creating the Menu and MenuItem. For this purpose the following task should be done:
   - Construct a Menu instance:

```
Menu appMenu = new Menu();
```

   - Menu displays a menu in the top right corner of the screen. To choose the selected menu item, you can either click the trackwheel or press ENTER. Depending on what you choose (trackwheel or Enter), you should define a listener interface ("TrackwheelListener" interface for trackwheel events or "FieldChangeListener" interface for keyboard events.). Thus, the next step is implementing a TrackWheelListener interface for the "FirstScreen" class. It is clear that you should override the required methods for this interface.

```
final class FirstScreen extends MainScreen implements TrackwheelListener
{
    public FirstScreen()
    {
        super();
        ……
    }
    public boolean trackwheelClick(int status, int time) {

        Menu appMenu = new Menu();
        return true;
    }

    public boolean trackwheelUnclick( int status, int time ) {
        return false;
    }

    public boolean trackwheelRoll(int amount, int status, int time) {
        return false;
    }

}
```

   - Now, you must create the menuItem objects. In this experiment, we create two menu items: View and Close.

```
private MenuItem _viewItem = new MenuItem("View",  200000, 10) {
    public void run() {
        Dialog.inform("This is the Experiment #1");
    }
};

private MenuItem _closeItem = new MenuItem("Close",  200000, 10) {
    public void run() {
        System.exit(0);
```

```
        }
    };
```

The implementation of *run()* defines the action which occurs when a user click on the menu items. For instance, you can open a dialog.

- The next step is to add the created MenuItem objects to Menu instance:

```
public boolean trackwheelClick(int status, int time) {

    Menu appMenu = new Menu();
    appMenu.add( _viewItem);
    appMenu.add( _closeItem);
    appMenu.show();

     return true;
}
```

The method *show()* above displays the menu to the user.

- Demo your work to the TA. [**5 marks**]

Listing 1 – Album_App.java:

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;

class Album_App extends net.rim.device.api.ui.UiApplication{

  public Album_App()
  {
    FirstScreen screen  = new FirstScreen();
     pushScreen(screen );
  }

  public static void main(String[] args) {
     Album_App theApp = new Album_App();
     theApp.enterEventDispatcher();
  }


  final class FirstScreen extends MainScreen implements TrackwheelListener
  {
        public FirstScreen()
        {
                super();
```

```
                LabelField title =
                        new LabelField("Album",|  LabelField.USE_ALL_WIDTH);
                setTitle(title);

                add(new RichTextField(""));
                add(new RichTextField(""));
                Bitmap myBitmap =  Bitmap.getBitmapResource("pic1.png");
                BitmapField myBitmapField =
                 new BitmapField(myBitmap, BitmapField.FIELD_HCENTER );

                add(myBitmapField);
                add(new RichTextField(""));

                Font font = Font.getDefault();
                FontFamily fontFamily[] = FontFamily.getFontFamilies();
                Font font1 = fontFamily[0].getFont(font.getStyle(),12);
                Font fonts[] = new Font[1];
                fonts[0] = font1;

                add( new RichTextField("To view pictures, please click on View menu
                item" ,null,null,fonts, RichTextField.TEXT_ALIGN_HCENTER));


            }
            public boolean trackwheelClick(int status, int time) {
                Menu appMenu = new Menu();
                appMenu.add(_viewItem);
              . appMenu.add(_closeItem);
                appMenu.show(); // Display the menu on screen.
                return true;
            }
            public boolean trackwheelUnclick( int status, int time ) {
                return false;
            }
            public boolean trackwheelRoll(int amount, int status, int time) {
                 return false;
            }

            private MenuItem  _viewItem = new MenuItem("View",  200000, 10) {
                        public void run() {
                                Dialog.inform("This is today's message");
                        }
            };

            private MenuItem  _closeItem = new MenuItem("Close",  200000, 10) {
                        public void run() {
                                System.exit(0);
                        }
        };
    }
}
```

Exercise 1 Adding more advance features to your application

In this exercise, you complete your application developed in the Experiment 1 by adding some advance features such as adding ListField and another screen to display a list of photos. The last screen which you add will look like follows:

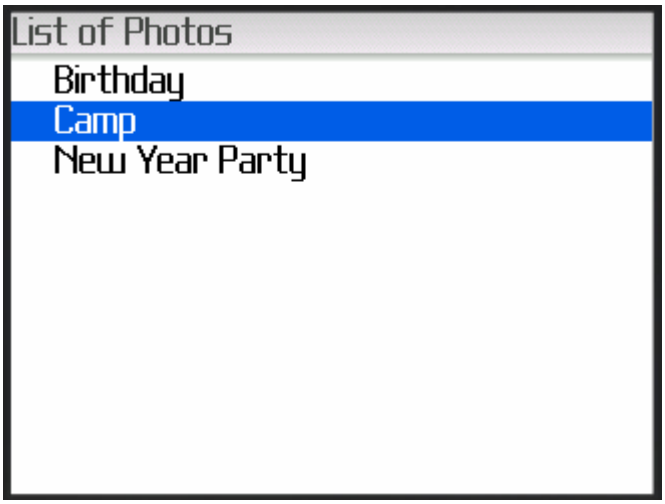

Figure 3- Second screen

To do this exercise, the following processes are suggested:

- Reuse the code in the Experiment 1
- Create another class (called "SecondScreen") extended from "MainScreen" class. This class contains ListField component. In order your screen be able to process the ListField events, you should implement listener interface for this class. The listener interface for ListField is "ListFieldCallback".
- Construct a new ListField instance. Set the callback method for this object. Set the size for this object. Add it to the screen. Note that you need to import the java.util.Vector class.

```
myList = new ListField();
myList.setCallback(this);
myList.setSize(_items.size());
add( myList);
```

_item is a new instance of Vector. It includes the list of items. In this exercise, it includes "Birthday", Camp and "New Year Party".

- Implement the ListFieldCallBack methods. It is necessary to override these methods while you implement the ListFieldCallBack listener interface for the class. The callback methods are as follows:

```
public void drawListRow(ListField list, Graphics graphics, int index, int y, int w) {

        graphics.drawText((String)_items.elementAt(index), 21, y);
}
```

```
public Object get(ListField listField, int index) {
        return  _items.elementAt(index);
}
public int getPreferredWidth(ListField listField) {
        return Graphics.getScreenWidth();
}
public int indexOfList(ListField listField, String prefix, int start) {
       return listField.indexOfList(prefix, start);
}
```

- Demo your work to the TA. [**5 marks**]