Objective

The objective of this lab is to get experience with:

1. Listening to event
2. Handling events
3. Develop an event handling application on the Blackberry device

Experiment 1 Handling keystroke events.

In this experiment, you will learn how to handle a keystroke event. Technically, an event is an action detected by an application. For instance, when you press a key or click a mouse button, an event is fired. In reaction to these events, an application provides some listener interfaces. These interfaces receive and respond to the user reaction. In this experiment, we look deeply in the Blackberry API interfaces.

To start the experiment, do the following steps:

1. Create a class extended from **net.rim.device.api.ui.UiApplication** class. This class provides methods for your application to register event listeners and manage UI components.

```
class DisplayEvent extends net.rim.device.api.ui.UiApplication {

}
```

2. In order to handle keystrokes events, the class must implement the KeyListener. To note that, you do not have to implement this listener. You implement it, if you want to override its methods. In this experiment, we want to override the keyChar method.

```
class DisplayEvent extends et.rim.device.api.ui.UiApplication implements KeyListener
{

}
```

3. As experienced in Lab 3, the Blackberry applications start at main(). In main(), two things should be done:
    a. Create a new object for the application
    b. Invoke **enterEventDispatcher().** Using this method the application starts processing the message.

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

4. Define a constructor for your class. It invokes **UiApplication.pushScreen()** to display the main screen which is appeared when your application is started. In addition, you have to add KeyListener() interface to your screen, if you want your screen to listen for the keystrokes events. This is done by **addKeyListener(this)** method.

```
public DisplayEvent(){
      DisplayEventScreen screen = new DisplayEventScreen();
      screen.addKeyListener(this);
      pushScreen(screen);
}
```

5. When you implement an interface, you have to override its methods. In the Step 2, you implement KeyListener interface, then you have to override its methods. It includes the following methods:

```
public boolean keyDown(int keycode, int time) {
   return false;
}
/* Implementation of KeyListener.keyRepeat(). */
public boolean keyRepeat(int keycode, int time) {
   return false;
}
/* Implementation of KeyListener.keyStatus(). */
public boolean keyStatus(int keycode, int time) {
   return false;
}
/* Implementation of KeyListener.keyUp(). */
public boolean keyUp(int keycode, int time) {
   return false;
}
```

```
public boolean keyChar(char key, int status, int time) {
        /* Intercept the ESC key and exit the application. */
        boolean retval = false;
        switch (key) {
           case Characters.ESCAPE:
              Dialog.alert("The ESC key has been pressed");
              System.exit(0);
              retval = true;
              break;
        }
        return retval;
 }
```

6. Define the main screen class. It is extended from **MainScreen** class.

```
final class DisplayEventScreen extends MainScreen {
         public DisplayEventScreen() {
                  super();
                  LabelField title = new LabelField("Display Events
                  Sample", LabelField.ELLIPSIS |
                  LabelField.USE_ALL_WIDTH);
                  setTitle(title);
                  add(new RichTextField("Display Keyboard Events"));
         }
}
```

7. Demo your work to the TA. [**3 marks**]

Listing 1: DisplayEvent.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 DisplayEvent extends net.rim.device.api.ui.UiApplication implements KeyListener{

        public DisplayEvent()
        {
           DisplayEventScreen screen = new DisplayEventScreen();
           screen.addKeyListener(this);
           pushScreen(screen );
        }

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

```
        public boolean keyDown(int keycode, int time) {
            return false;
        }

        /* Implementation of KeyListener.keyRepeat(). */
        public boolean keyRepeat(int keycode, int time) {
            return false;
        }

       /* Implementation of KeyListener.keyStatus(). */
        public boolean keyStatus(int keycode, int time) {
           return false;
        }

        /* Implementation of KeyListener.keyUp(). */
        public boolean keyUp(int keycode, int time) {
           return false;
        }

       public boolean keyChar(char key, int status, int time) {
          /* Intercept the ESC key and exit the application. */
          boolean retval = false;
          switch (key) {
             case Characters.ESCAPE:
                    Dialog.alert("The ESC key has been pressed");
                    System.exit(0);
                     retval = true;
                     break;
             }
             return retval;
          }

      final class DisplayEventScreen extends MainScreen {
          public DisplayEventScreen() {
             super();
              LabelField title = new LabelField("Display Events Sample",
              LabelField.ELLIPSIS |  LabelField.USE_ALL_WIDTH);
             setTitle(title);
             add(new RichTextField("Display Keyboard Events"));

          }
      }
  }
```

Experiment 2

In this experiment, you handle Enter key. It can be done in the same way you have already done for Escape key in the Experiment 1. The difference is that your screen class extended from MainScreen class implements KeyboardListener interface instead of the class extended from UiApplication.

To start this experiment, do the following tasks:

1. Create a class, extend UiApplication class, name it “Exp2”.

```
class Exp2  extends net.rim.device.api.ui.UiApplication {}
```

2. Create the main() and constructor methods. In these methods, write the required codes:

```
class Exp2 extends net.rim.device.api.ui.UiApplication {
        public LoginScreen screen;

       public Exp2()
       {
          ....
       }
       public static void main(String[] args) {
          ....;
       }
}
```

3. Create a screen class extended from MainScreen, name it “LoginScreen”.

```
class Exp2 extends net.rim.device.api.ui.UiApplication {
        public LoginScreen screen;

       public Exp2()
       {
          ....
       }
       public static void main(String[] args) {
          ....;
       }

      final class LoginScreen extends MainScreen {
      }

}
```

4. At this experiment, you implement the KeyListener for the “LoginScreen” class. Thus, you have to override the relevant methods in this class.

```
class Exp2 extends net.rim.device.api.ui.UiApplication {
        public LoginScreen screen;

       public Exp2()   {   ....   }
       public static void main(String[] args) {    ....;   }

      final class LoginScreen extends MainScreen implements  KeyListener
```

```
{
        public boolean keyDown(int keycode, int time) {
            return false;
        }
        /* Implementation of KeyListener.keyRepeat(). */
        public boolean keyRepeat(int keycode, int time) {
            return false;
        }
        /* Implementation of KeyListener.keyStatus(). */
        public boolean keyStatus(int keycode, int time) {
            return false;
        }
        /* Implementation of KeyListener.keyUp(). */
        public boolean keyUp(int keycode, int time) {
            return false;

        }
        public boolean keyChar(char key, int status, int time) {
            /* Intercept the ESC key and exit the application. */
              boolean retval = false;
             switch (key) {
               case Characters.ESCAPE:
                        System.exit(0);
                        retval = true;
                break;

               case Characters.ENTER:
                        retval = true;
               break;
               default:
                         retval = super.keyChar(key,status,time);
              }
             return retval;
        }
}
```

5. Create the constructor for screen class and write the required codes to create a user interface to enter contact information including name, address and phone. When the user presses the Enter key, a dialog box should be demonstrated with this message “Do you want to save the document?”
   The required method to invoke a dialog box is as follows:

```
int response = Dialog.ask(Dialog.D_YES_NO, "Are you sure?", Dialog.NO);
```

6. Demo your work to the TA. [**3 marks**]

Exercise 1 Creating a login page application using BlackBerry APIs.

In this exercise, you reuse the code experienced in Experiment #2 and create a login page. Creating a login page protects your application from illegal accesses. It allows users enter their username and password before they can access to your application.

To get started this experiment, do the following steps:

1. Create two screens with the following specifications:

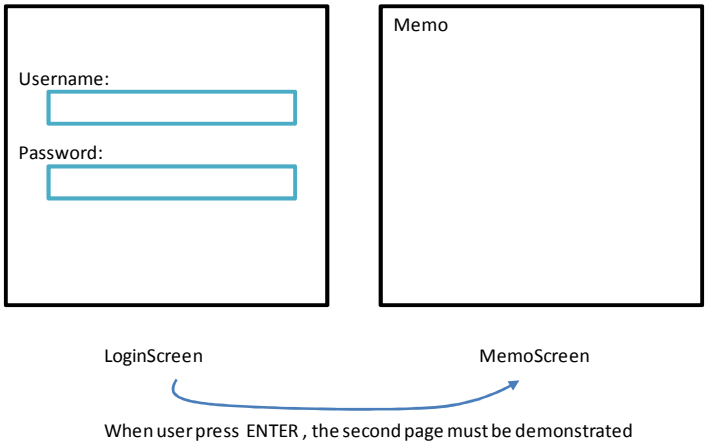

Figure 1- Screen specifications

2. Reuse the code written in the Experiment 2 and complete it based on the specifications mentioned above. To note that when a user press the Enter key, the "MemoScreen" is displayed. (use "popScreen" and "pushScreen").

3. Demo your work to the TA. [**3 marks**]

Exercise 2 Explain the difference between using the listener interface in Experiment 1 and Experiment 2? [**1 marks**]