CMER
Centre for Mobile Education and Research

Objective

The objective to this lab is to:

1. Get you started with Blackberry devices
2. Develop a simple application on the Blackberry device

Experiment 1 Download and install the BlackBerry JDE (Java Development Environment).

In this experiment, you will download and install the BlackBerry® Java™ Development Environment (JDE) which is a fully integrated development environment and simulation tool for building Java 2, Micro Edition (J2ME™) applications for Java-based BlackBerry devices.

To download the JDE, do the following tasks:

1. Go to the URL http://na.blackberry.com/eng/developers/downloads/jde.jsp
2. Look at the "Downloads" section and click on the latest version of "Blackberry JDE".
3. Fill out the requirement fields, confirm the information and click on the "next" button
4. Download and install "Blackberry Java Development Environment"
5. In addition to JDE, it includes some other components required for development of third party applications for BlackBerry devices as follows:
    - RAPC compiler: it compiles .java and .jar files into .cod files that can be run in the BlackBerry device simulator or loaded onto a BlackBerry device
    - JavaLoader: it is a tool which allows applications to be added or updated on a BlackBerry device from a command prompt
    - Java Debug Wire Protocol (JDWP) proxy: it enables the debugging of applications using third party IDEs

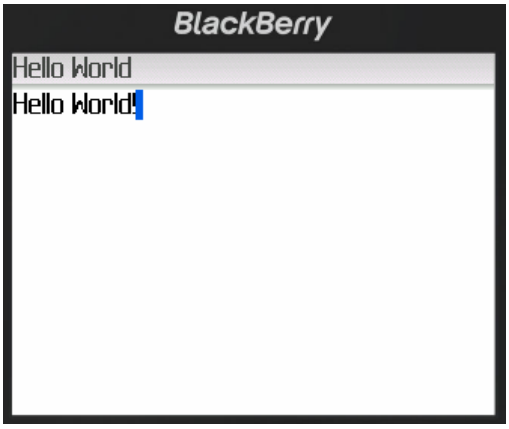
Experiment 2 Explore the JDE

In this experiment you will use the code of an existing application (HelloWorld) located in the "sample" directory of RIM's JDE installation directory, to explore the JDE.

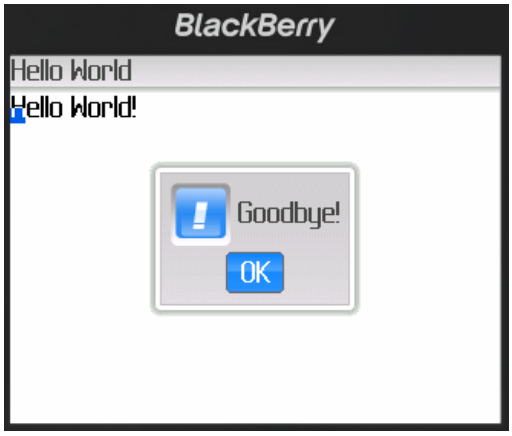
To start the experiment, do the following:

1. Launch the JDE installed on the machine.
2. Open the "HelloWorld.java" by going to File -> Open -> ..\samples\com\rim\samples\device\helloworld\HelloWorld.java.
3. Go to Build menu, click on "Build" item to compile your application. Alternately, the user can simply press "F7" as an accelerator.
4. Go to "Debug" menu and click on the "Go" item or alternately press "F5" as an accelerator.
5. Using arrow keys navigate icons on Blackberry device screen, select "Applications" icon and press "Enter" key.
6. Using arrow keys navigate the applications icons on the Blackberry device's screen, select "HelloWorld" icon and press "Enter" key.

   You will see the following window:



7. Press "Esc" button, you will see the following dialog:



- Explore the "HelloWorld.java" source code. [**4 marks**]

Exercise 1 Make the first Blackberry application

In this exercise you will write a simple BlackBerry application that displays a simple text message.

To get started, do the following steps:

1. Create a new workspace
   - Click on "File" menu and select "New..." item.
   - Click on "Workspaces" tab. There are two tasks needed to be done: choosing a workspace name and browsing a location for your project.
   - Your workspace will be saved with the ".jdw" extension.

2. Create a new project
   - Click on "File" menu and select "New…" item again.
   - Click on the "Projects" tab. There are two tasks needed to be done here: choosing a project name and browsing a location for your project. *To note that when you browse for a location, select the same location created in Step 1.*
   - A dialog will be opened, choose"Always make project active" item and just click on the "Finish" button.

3. Create a new file
   - Click on "File" menu and select "New..." item again.
   - Like to previous steps, select name of a file and browse the location which you created in Step1. Then click on "Ok".
     At this point the IDE creates a barebones Java class with the following code in it, note the import statements you will need:

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

class HelloWorld{

}
```

     Actually, this code does nothing. It is your task to complete the HelloWorld class.
   - Now, you should associate the file to your project created earlier. To do this, simply right click in the file and select the "Insert into project" option. A dialog will be opened. Select the project to which you want to add your file. Then click on "Ok".

4. Develop your application

- In order to provide a Graphical User Interface (GUI) for your application, your class file "HelloWorld" must be extended from net.rim.device.api.ui.UiApplication class. Thus you have:

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

}
```

- Your BlackBerry application is like J2SE applications starting with a main method

```
class HelloWorld extends net.rim.device.api.ui.UiApplication {
        public static void main(String[] args)  {
            HelloWorld instance = new HelloWorld();
            instance.enterEventDispatcher();
        }
        public HelloWorld()  {
            pushScreen(new HelloWorldScreen());
        }
}
```

- The first thing you do in the main method is create an instance of your application by calling its constructor. After calling the constructor, call your new instance's enterEventDispatcher method. This method allows your application to start handling various events that the BlackBerry device may send to the application (e.g., UI-centric events).
- The constructor uses the pushScreen method (a method of UiApplication) to display a screen. You haven't defined the **HelloWorldScreen** class yet. So in the next step you will get to that.
- You can create the new class file HelloWorldScreen as a new file or as a final class in the same file. In this lab, we create it in the same file. The HelloWorldScreen class displays a "Hello World" message to your application user.
- The HelloWorldScreen class extends from the **net.rim.device.api.ui.container.MainScreen** class, giving your simple application consistency with other native BlackBerry applications. It also

provides such features as a default menu with a Close menu item for exiting out of your application.

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

      public static void main(String[] args)    {

            ……..

      }

       public HelloWorld()   {

            pushScreen(new HelloWorldScreen());

    }

}

//================== Second Class ============================
final class HelloWorldScreen extends MainScreen
{
   public HelloWorldScreen()
   {
      super();
   }
}
```

- Add a title to your simple application:

```
……
final class HelloWorldScreen extends MainScreen
{
   public HelloWorldScreen()
   {
               super();

          LabelField applicationTitle = new LabelField("Hello   World");

          setTitle(applicationTitle);

   }
}
```

- Then create a TextField GUI component, and add it to the screen by using the add method you inherited from your ancestor class net.rim.device.api.ui.Screen.

```
……
```

```
final class HelloWorldScreen extends MainScreen
{
   public HelloWorldScreen()
   {
          super();

          LabelField applicationTitle = new LabelField("Hello   World");

          setTitle(applicationTitle);

          RichTextField helloWorldTextField = new RichTextField("Hello
          World!");

           add(helloWorldTextField);

   }
}
```

- You also inherit the **onClose** method from the net.rim.device.api.ui.Screen, which is fired when your screen (HelloWorldScreen) closes. In reaction to the closing event, the application uses the alert method of the **net.rim.device.api.ui.component.Dialog** class to display a popup on the screen stating a message of "Bye World!".

```
……
final class HelloWorldScreen extends MainScreen
{
   public HelloWorldScreen()
   {
        super();

          …….

   }

   public boolean onClose()   {

     Dialog.alert("Bye World!");

     System.exit(0);

     return true;  }

}
```

- Save your file (File>Save menu).
- Build the project by pressing “F7” and then press “F5” to run your first application.
- Demo your work to the TA. [**6 marks**]