Porting BlackBerry applications and services to S60 platform

Introduction

BlackBerry devices are developed by Research In Motion(RIM). While porting your application from BlackBerry to Nokia S60 one should make use of Java or I would say Java ME.

It is also possible to port from BlackBerry Java to S60 and Symbian C++, but as the original runtime (BlackBerry) is Java based we are taking this path into consideration at first.

But do keep in mind that Java Virtual Machine implementation does change from BlackBerry to Nokia and with that changes the availability of some APIs, mainly the Java ME Optional Packages.

In fact, even though Blackberry JVM supports full CLDC and MIDP, RIM adds several libraries for specific functions, like GUI, application notification mechanism, cryptography, data storage, Phone API and others. Some have a counterpart available in standard Java ME, mainly by means of additional Optional Packages but some are not available.

Besides, some standard Java ME JSRs are finalized by JCP (Java Community Process) but have not been implemented in a single device by any manufacturer so far, for example, JSR-253 MTA (Mobiile Telephony API).

Therefore, most standard Java ME applications can be rather easily installed on a Blackberry, while the reciprocity is not true, mainly if the original BlackBerry application has been developed using the BlackBerry specific APIs. So porting from RIM Java to standard Java ME is somehow different from porting from standard Java ME to RIM Java.

Anyway, differences in screen sizes and input methods and keys will always exist, even when using the same Java ME code on both, some problems or tweaks might need to be addressed or performed.

Standard Java ME is also available on BlackBerry but it is very limited regarding access to device's capabilities, key handling and other requirements.

Application Development

First thing which any one should check before going ahead with application development using Java, is to know about the supported APIs. That means checking the supported Configurations, Profiles and Optional Packages of Java ME as well as the manufacturer specific APIs available, mainly when taking this path of porting from BlackBerry to Java for S60.

(C) Copyright Nokia 2013. All rights reserved.

Java ME Configuration
Depending on different devices of BlackBerry and Nokia configuration supported are *CLDC 1.0 & CLDC 1.1*

Java ME Profile
In addition to basic *MIDP 2.x* platform BlackBerry contains *BlackBerry x.x.x* platform. *JTWI* platform is implemented in some of BlackBerry devices but mostly on all Nokia devices.

Supported Optional Packages
There are lot of additional JSRs which Nokia supports (they vary from device to device, for details check the link), for example *JSR 75 FileConnection and PIM API*, *JSR 82 Java™ APIs for Bluetooth 1.1*, *JSR 205 Wireless Messaging API 2.0* many more.

Some are implemented by BlackBerry devices as well but some are not, or the BlackBerry platform has a specific API that maps to a standard JSR.

So when porting from BlackBerry to Java ME is important to identify the APIs being used by the original application on BlackBerry platform and map them to standard Java ME JSR or Nokia specific API (Nokia always tries to use the standard ones, one exception is the Nokia UI API).

For details about what all J2ME packages BlackBerry and Nokia contain you can have a look at the devices specification for both the original (source) BlackBerry device and the new (target) Nokia device when porting. Besides, it is very important to check the APIs in detail, using the JavaDocs available for each API supported by both devices.

If your application is using APIs specified in CLDC and MIDP then you can directly port your application to Nokia device without changing your code.

But in case you are using some specific packages of BlackBerry or any Optional Package then you need to do an equivalent implementation of those classes for your Nokia application.

Installable
Nokia - JAD , JAR

BlackBerry - ALX or JAD, COD (It is generated from JAR using the rapc tool available on BlackBerry SDK). One JAR may result in many COD files.

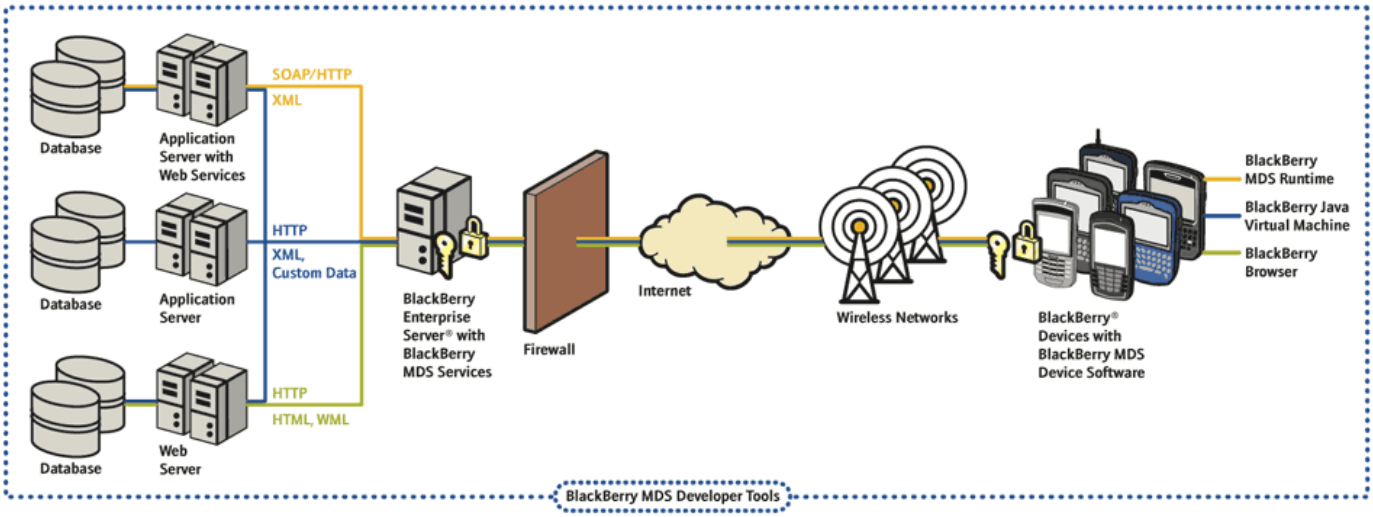
Signing
Nokia supports: VeriSign, Thawte, Java Verifed (UTI) Certificate (No Developer signing).

BlackBerry supports: RIM Certificate (Developer signing is available with charge) and the only provider is RIM.

Tools
- Wireless tool kit(WTK)
- Nokia S60 SDK

Services

From data communication point of view one most important think to consider in case of BlackBerry is Mobile Data System (MDS). All communication of BlackBerry device is diverted through this, picture below will give you a better overview of this:



There are some parameters required to set while initializing communication from an application build for BlackBerry device. But since no such communication architecture is used by Nokia those parameters can be avoided for communication.

Even the emulators of BlackBerry need MDS installed in your system to do communication. But when porting to S60 there is no need to worry about such scheme as the BlackBerry code using the Http(s)Connection is tranparent and will not be different when running on S60, MDS is a proxy (mediator) but does not influence the coding directly.

Some points to be considered while Porting

1. MDS is not required in case of Nokia communication architecture
2. Take advantage of additional JSRs on Nokia handsets
3. Writing code according to Java coding standard itself helps a lot while porting.

(C) Copyright Nokia 2013. All rights reserved.

The following elements probably will require work:

- Graphical interface - if using RIM components, since those are not present in MIDP. Most have equivalent componentes in standar Java ME
- Storage - since RIM provides some powerful mechanisms for data storage
- Cryptography - which can be achieved via libraries like Bouncy Castle or JSR-177 (SATSA)
- Bluetooth - it is handled in a very different way on Blackberry. The user must pair the device with the phone before the application sees it. Moreover, Bluetooth library on Blackberry used to be not the JSR 82, and is still not on some device
- In the same way, access to PIM is specific since RIM provides extra informations. However, newer BlackBerry devices use the standard JSR-75, so porting will be easier.

Classes with similar purpose but different Name

There are many classes which do similar work but has different name in both platform like Form(J2ME) and MainScreen(Blackberry). Below we have some examples:

Error creating thumbnail: Image type not supported

Change in Coding Strategy

BlackBerry API extensions versus Standard Java ME APIs

When developing applications for BlackBerry, you can use the standard Java ME APIs as usual or the BlackBerry API extensions. The vast majority of BlackBerry devices have some standard Java JSRs but almost all the advanced features of BlackBerry devices are acessible only when using a BlackBerry specific API.

So, when porting BlackBerry applications to S60, it is almost sure that you will need to know the differences and specifics of those BlackBerry specific APIs to map them to standard Java ME APIs in Nokia S60 devices.

It is important to emphasize that Nokia devices support the standard Java ME APIs in a better and more complete way than BlackBerry devices, mainly regarding robustness, the number of standard APIs implemented and JSR coverage (optional packages) as well as access to native features and hardware integration.

The code samples below will show some differences regarding the classes, interfaces, methdos and other details one must take into consideration when porting from BlackBerry to S60 Java.

Examples - Application Structure

This is a simple “Hello World” application on BlackBerry using the RIM specific APIs. There are several differences regarding normal MIDlet development.

So, withour further ado, let us talk about the differences and points we must consider when porting.

First, the main class representing an application is called UIApplication when using the BlackBerry runtime (instead of MIDlet class of standard MIDP). It is possible to use MIDlet as well but as cited above, the support is very limited and you will not be able to access the trackwhell or the trackball and other BlackBerry keys, for example, just as a sample of the limitations.

Also, when extending the UIApplication class, your application will use a main() method as normal Java SE class instead of the MIDlet callback methods that are controlled by the AMS/JAM software. Besides, you must signal the application to enter the event dispatcher loop so that it can start listening to UI events.

BlackBerryApp

Invalid language.

You need to specify a language like this: `<source lang="html4strict">...</source>`

Supported languages for syntax highlighting:

4cs, 6502acme, 6502kickass, 6502tasm, 68000devpac, abap, actionscript, actionscript3, ada, algol68, apache, applescript, apt_sources, asm, asp, autoconf, autohotkey, autoit, avisynth, awk, bascomavr, bash, basic4gl, bf, bibtex, blitzbasic, bnf, boo, c, c_loadrunner, c_mac, caddcl, cadlisp, cfdg, cfm, chaiscript, cil, clojure, cmake, cobol, coffeescript, cpp, cpp-qt, csharp, css, cuesheet, d, dcs, delphi, diff, div, dos, dot, e, ecmascript, eiffel, email, epc, erlang, euphoria, f1, falcon, fo, fortran, freebasic, fsharp, gambas, gdb, genero, genie, gettext, glsl, gml,

(C) Copyright Nokia 2013. All rights reserved.

gnuplot, go, groovy, gwbasic, haskell, hicest, hq9plus, html4strict, html5, icon, idl, ini, inno, intercal, io, j, java, java5, javascript, jquery, kixtart, klonec, klonecpp, latex, lb, lisp, llvm, locobasic, logtalk, lolcode, lotusformulas, lotusscript, lscript, lsl2, lua, m68k, magiksf, make, mapbasic, matlab, mirc, mmix, modula2, modula3, mpasm, mxml, mysql, newlisp, nsis, oberon2, objc, objeck, ocaml, ocaml-brief, oobas, oracle11, oracle8, oxygene, oz, pascal, pcre, per, perl, perl6, pf, php, php-brief, pic16, pike, pixelbender, pli, plsql, postgresql, povray, powerbuilder, powershell, proftpd, progress, prolog, properties, providex, purebasic, pycon, python, q, qbasic, rails, rebol, reg, robots, rpmspec, rsplus, ruby, sas, scala, scheme, scilab, sdlbasic, smalltalk, smarty, sql, systemverilog, tcl, teraterm, text, thinbasic, tsql, typoscript, unicon, uscript, vala, vb, vbnet, verilog, vhdl, vim, visualfoxpro, visualprolog, whitespace, whois, winbatch, xbasic, xml, xorg_conf, xpp, yaml, z80, zxbasic

```
package bberryexample;

public class BlackBerryApp extends UiApplication {

    public BlackBerryApp() {
        showScreen(new ScrSplash(this));
    }

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

}
```

So, below it is easy to see the differences. Your Java app on S60 will extend MIDlet, implement the standard callback methods (startApp, pauseApp, destroyApp), change the showScreen() method call to setCurrent() method call, that is available on Display class.

JavaMEApp

Invalid language.

You need to specify a language like this: `<source lang="html4strict">...</source>`

Supported languages for syntax highlighting:

4cs, 6502acme, 6502kickass, 6502tasm, 68000devpac, abap, actionscript, actionscript3, ada, algol68, apache, applescript, apt_sources, asm, asp, autoconf, autohotkey, autoit, avisynth, awk, bascomavr, bash, basic4gl, bf, bibtex, blitzbasic, bnf, boo, c, c_loadrunner, c_mac, caddcl, cadlisp, cfdg, cfm, chaiscript, cil, clojure, cmake, cobol, coffeescript, cpp, cpp-qt, csharp, css, cuesheet, d, dcs, delphi, diff, div, dos, dot, e, ecmascript, eiffel, email, epc, erlang, euphoria, f1, falcon, fo, fortran, freebasic, fsharp, gambas, gdb, genero, genie, gettext, glsl, gml, gnuplot, go, groovy, gwbasic, haskell, hicest, hq9plus, html4strict, html5, icon, idl, ini, inno, intercal, io, j, java, java5, javascript, jquery, kixtart, klonec, klonecpp, latex, lb, lisp, llvm, locobasic, logtalk, lolcode, lotusformulas, lotusscript, lscript, lsl2, lua, m68k, magiksf, make, mapbasic, matlab, mirc, mmix, modula2, modula3, mpasm, mxml, mysql, newlisp, nsis, oberon2, objc, objeck, ocaml, ocaml-brief, oobas, oracle11, oracle8, oxygene, oz, pascal, pcre, per, perl, perl6, pf, php, php-brief, pic16, pike, pixelbender, pli, plsql, postgresql, povray, powerbuilder, powershell, proftpd, progress, prolog, properties, providex, purebasic, pycon, python, q, qbasic, rails, rebol, reg, robots, rpmspec, rsplus, ruby, sas, scala, scheme, scilab, sdlbasic, smalltalk, smarty, sql, systemverilog, tcl, teraterm, text, thinbasic, tsql, typoscript, unicon, uscript, vala, vb, vbnet, verilog, vhdl, vim, visualfoxpro, visualprolog, whitespace, whois, winbatch, xbasic, xml, xorg_conf, xpp, yaml, z80, zxbasic

```
package midletexample;

public class JavaMEApp extends MIDlet {

    Display display;

    public void startApp() {
      display = Display.getDisplay(this);
      ...
    }
    public void pauseApp() {...}
    public void destroyApp(boolean unconditional) {...}

}
```

Examples - User Inteface (low-level and high-level) and Event-handlers

When developing an application for both BlackBerry and S60, you can opt for using high-level GUI components or low-level ones.

Regarding BlackBerry, if the original project has used the low-level approach, the GUI classes might have extend the net.rim.device.api.ui.Screen class (or any of its subclasses such as FullScreen) and implemented its callback method for painting, that is the protected void paint(Graphics g) method.

From an event-handling standpoint, the class probably has implemented the listener interface of choice (depending on the type of event the original developer wanted to listen to. It is almost the same as the event handling approach used by Java Desktop GUI framweworks like Java AWT and Java Swing. Below our class implements the KeyListener interface and overrides its methods (keyDown, keyUp, etc).

MenuScreen - BlackBerry low-level API

Invalid language.

You need to specify a language like this: `<source lang="html4strict">...</source>`

Supported languages for syntax highlighting:

4cs, 6502acme, 6502kickass, 6502tasm, 68000devpac, abap, actionscript, actionscript3, ada, algol68, apache, applescript, apt_sources, asm,

(C) Copyright Nokia 2013. All rights reserved.

```
asp, autoconf, autohotkey, autoit, avisynth, awk, bascomavr, bash, basic4gl, bf, bibtex, blitzbasic, bnf, boo, c, c_loadrunner, c_mac, caddcl,
cadlisp, cfdg, cfm, chaiscript, cil, clojure, cmake, cobol, coffeescript, cpp, cpp-qt, csharp, css, cuesheet, d, dcs, delphi, diff, div, dos, dot,
e, ecmascript, eiffel, email, epc, erlang, euphoria, f1, falcon, fo, fortran, freebasic, fsharp, gambas, gdb, genero, genie, gettext, glsl, gml,
gnuplot, go, groovy, gwbasic, haskell, hicest, hq9plus, html4strict, html5, icon, idl, ini, inno, intercal, io, j, java, java5, javascript,
jquery, kixtart, klonec, klonecpp, latex, lb, lisp, llvm, locobasic, logtalk, lolcode, lotusformulas, lotusscript, lscript, lsl2, lua, m68k,
magiksf, make, mapbasic, matlab, mirc, mmix, modula2, modula3, mpasm, mxml, mysql, newlisp, nsis, oberon2, objc, objeck, ocaml, ocaml-brief,
oobas, oracle11, oracle8, oxygene, oz, pascal, pcre, per, perl, perl6, pf, php, php-brief, pic16, pike, pixelbender, pli, plsql, postgresql,
povray, powerbuilder, powershell, proftpd, progress, prolog, properties, providex, purebasic, pycon, python, q, qbasic, rails, rebol, reg,
robots, rpmspec, rsplus, ruby, sas, scala, scheme, scilab, sdlbasic, smalltalk, smarty, sql, systemverilog, tcl, teraterm, text, thinbasic,
tsql, typoscript, unicon, uscript, vala, vb, vbnet, verilog, vhdl, vim, visualfoxpro, visualprolog, whitespace, whois, winbatch, xbasic, xml,
xorg_conf, xpp, yaml, z80, zxbasic
```

```
package bberryexample;

public class MenuScreen extends FullScreen implements KeyListener {

    protected void paint(Graphics g) {
        g.drawBitmap(0, 0, 320, 240, imgScr, 0, 0);
        if (index == 0)
            g.drawBitmap(0, 0, bt[index].getWidth(), bt[index].getHeight(), bt[index], 0, 0);
        else if (index == 1)
            g.drawBitmap(0, 0, bt[index].getWidth(), bt[index].getHeight(), bt[index], 0, 0);
        else if (index == 2)
            g.drawBitmap(0, 0, bt[index].getWidth(), bt[index].getHeight(), bt[index], 0, 0);
        g.setFont(Font.getDefault().derive(Font.BOLD, 11));
        g.setColor(0x00999999);
        g.drawText("Select an option.", 5, 224);
    }

    boolean keyChar(char key, int status, int time) {...}

    boolean keyDown(int keycode, int time) {...}

    boolean keyRepeat(int keycode, int time) {...}

    boolean keyStatus(int keycode, int time) {...}

    boolean keyUp(int keycode, int time) {...}

}
```

If the original BlackBerry project has used the high-level approach, the GUI classes might have extended the net.rim.device.api.ui.container.MainScreen class. In this case the painting will be made by the internal implementation of this high-level component as usual.

From an event-handling standpoint, the class probably has implemented the listener interface of choice as above or it has delegated the event handling to a standard high-level component, depending on the desired effect.

MenuScreen - BlackBerry high-level API

Invalid language.

You need to specify a language like this: `<source lang="html4strict">...</source>`

Supported languages for syntax highlighting:

```
4cs, 6502acme, 6502kickass, 6502tasm, 68000devpac, abap, actionscript, actionscript3, ada, algol68, apache, applescript, apt_sources, asm,
asp, autoconf, autohotkey, autoit, avisynth, awk, bascomavr, bash, basic4gl, bf, bibtex, blitzbasic, bnf, boo, c, c_loadrunner, c_mac, caddcl,
cadlisp, cfdg, cfm, chaiscript, cil, clojure, cmake, cobol, coffeescript, cpp, cpp-qt, csharp, css, cuesheet, d, dcs, delphi, diff, div, dos, dot,
e, ecmascript, eiffel, email, epc, erlang, euphoria, f1, falcon, fo, fortran, freebasic, fsharp, gambas, gdb, genero, genie, gettext, glsl, gml,
gnuplot, go, groovy, gwbasic, haskell, hicest, hq9plus, html4strict, html5, icon, idl, ini, inno, intercal, io, j, java, java5, javascript,
jquery, kixtart, klonec, klonecpp, latex, lb, lisp, llvm, locobasic, logtalk, lolcode, lotusformulas, lotusscript, lscript, lsl2, lua, m68k,
magiksf, make, mapbasic, matlab, mirc, mmix, modula2, modula3, mpasm, mxml, mysql, newlisp, nsis, oberon2, objc, objeck, ocaml, ocaml-brief,
oobas, oracle11, oracle8, oxygene, oz, pascal, pcre, per, perl, perl6, pf, php, php-brief, pic16, pike, pixelbender, pli, plsql, postgresql,
povray, powerbuilder, powershell, proftpd, progress, prolog, properties, providex, purebasic, pycon, python, q, qbasic, rails, rebol, reg,
robots, rpmspec, rsplus, ruby, sas, scala, scheme, scilab, sdlbasic, smalltalk, smarty, sql, systemverilog, tcl, teraterm, text, thinbasic,
tsql, typoscript, unicon, uscript, vala, vb, vbnet, verilog, vhdl, vim, visualfoxpro, visualprolog, whitespace, whois, winbatch, xbasic, xml,
xorg_conf, xpp, yaml, z80, zxbasic
```

```
package bberryexample;

public class MenuScreen extends MainScreen implements KeyListener {

    MenuScreen () {
      setTitle(new LabelField("Nokia Developer - Main", DrawStyle.ELLIPSIS | USE_ALL_WIDTH));
      Manager vfm = new VerticalFieldManager(FIELD_HCENTER);
      Field title = new LabelField("Nokia Developer - BlackBerry Client to be ported to S60", FIELD_HCENTER);
      Field uri = new LabelField("http://www.developer.nokia.com/", FIELD_HCENTER);
      vfm.add(title);
      vfm.add(uri);
      Field okButton = new ButtonField("OK", FIELD_HCENTER | ButtonField.CONSUME_CLICK);
      okButton.setChangeListener(new KeyListener(this));
      vfm.add(okButton);
      add(vfm);
  }

    boolean keyChar(char key, int status, int time) {...}

    boolean keyDown(int keycode, int time) {...}

    boolean keyRepeat(int keycode, int time) {...}

    boolean keyStatus(int keycode, int time) {...}
```


(C) Copyright Nokia 2013. All rights reserved.

```
    boolean keyUp(int keycode, int time) {...}

}
```

So, here are the tips to port the BlackBerry GUIs above to standard Java ME GUIs, regarding both low-level and high-level APIs.

If using the low-level API, your new class must extend the javax.microedition.lcdui.Canvas instead of FullScreen, for example.

Besides, it must implement the public void paint(Graphics g) method as well as the key event handlers, such as keyPressed, keyReleased, etc.

At last, it is good to implement the showNotify and hideNotify methods to provide proper release of resources and acquire them again when the Canvas is sent to background and back to foreground, replacing the onClose, onDisplay or onUndisplay methods or any other display related method used on the original BlackBerry screen.

MenuScreen - Java ME low-level API

Invalid language.

You need to specify a language like this: `<source lang="html4strict">...</source>`

Supported languages for syntax highlighting:

4cs, 6502acme, 6502kickass, 6502tasm, 68000devpac, abap, actionscript, actionscript3, ada, algol68, apache, applescript, apt_sources, asm, asp, autoconf, autohotkey, autoit, avisynth, awk, bascomavr, bash, basic4gl, bf, bibtex, blitzbasic, bnf, boo, c, c_loadrunner, c_mac, caddcl, cadlisp, cfdg, cfm, chaiscript, cil, clojure, cmake, cobol, coffeescript, cpp, cpp-qt, csharp, css, cuesheet, d, dcs, delphi, diff, div, dos, dot, e, ecmascript, eiffel, email, epc, erlang, euphoria, f1, falcon, fo, fortran, freebasic, fsharp, gambas, gdb, genero, genie, gettext, glsl, gml, gnuplot, go, groovy, gwbasic, haskell, hicest, hq9plus, html4strict, html5, icon, idl, ini, inno, intercal, io, j, java, java5, javascript, jquery, kixtart, klonec, klonecpp, latex, lb, lisp, llvm, locobasic, logtalk, lolcode, lotusformulas, lotusscript, lscript, lsl2, lua, m68k, magiksf, make, mapbasic, matlab, mirc, mmix, modula2, modula3, mpasm, mxml, mysql, newlisp, nsis, oberon2, objc, objeck, ocaml, ocaml-brief, oobas, oracle11, oracle8, oxygene, oz, pascal, pcre, per, perl, perl6, pf, php, php-brief, pic16, pike, pixelbender, pli, plsql, postgresql, povray, powerbuilder, powershell, proftpd, progress, prolog, properties, providex, purebasic, pycon, python, q, qbasic, rails, rebol, reg, robots, rpmspec, rsplus, ruby, sas, scala, scheme, scilab, sdlbasic, smalltalk, smarty, sql, systemverilog, tcl, teraterm, text, thinbasic, tsql, typoscript, unicon, uscript, vala, vb, vbnet, verilog, vhdl, vim, visualfoxpro, visualprolog, whitespace, whois, winbatch, xbasic, xml, xorg_conf, xpp, yaml, z80, zxbasic

```
public class MenuScreen extends Canvas {

       public void paint(Graphics g) {
        if (!hasFocus)
            return;

        g.setColor(0xffffff);
        g.fillRect(0, 0, getWidth(), getHeight());

        StringBuffer hiName = new StringBuffer();
        g.drawImage(bckGrd, 0, 0, Graphics.LEFT|Graphics.TOP);
        g.drawImage(menuImg, 0, 0, Graphics.LEFT|Graphics.TOP);
        ...
        g.setColor(0xFF6600);
        g.setFont(font);
        g.drawString(getStatus(), 95, 90, Graphics.LEFT|Graphics.TOP);

        g.drawString(freightF, 15, 115, Graphics.LEFT|Graphics.TOP);

        g.drawString(mid.doFormatTel(tel), 15, 130, Graphics.LEFT|Graphics.TOP);
      }

      public void keyPressed(int kc) {

        if(kc == Canvas.KEY_STAR) mid.doQuit();
        else if(kc == Canvas.KEY_NUM1) mid.doInvoice();
        else if (kc == Canvas.KEY_NUM2) mid.doDownload();
        else if (kc == Canvas.KEY_NUM3) mid.doSettings();
        else if (kc == Canvas.KEY_NUM4) mid.doStatus();
        ...
      }

      protected void hideNotify(){...}
      protected void showNotify(){...}

}
```

If using the high-level API, your new class must extend the javax.microedition.lcdui.Form class instead of MainScreen, for example.

Besides, as the high-level components are not painted by your code but the native implementation, there is not method you must override to do so.

Regarding the event handling code, your class will implement the corresponding even handler interface to provide the event handling logic that is appropriate to the type of event you want to receive, as usual. In MIDP high-level interface examples of such interfaces are CommandListener, ItemStateListener and a few others. After implementing the interface you must override the relevant interface method and provide your logic to handle the event accordingly.

At last, almost the same strategy as before it is good practice to provide proper release of resources and acquire them again when the Form is sent to background and back to foreground, now using the public boolean isShown() method of Form class.

MenuScreen - Java ME high-level API

Invalid language.

You need to specify a language like this: `<source lang="html4strict">...</source>`

(C) Copyright Nokia 2013. All rights reserved.

Supported languages for syntax highlighting:

4cs, 6502acme, 6502kickass, 6502tasm, 68000devpac, abap, actionscript, actionscript3, ada, algol68, apache, applescript, apt_sources, asm, asp, autoconf, autohotkey, autoit, avisynth, awk, bascomavr, bash, basic4gl, bf, bibtex, blitzbasic, bnf, boo, c, c_loadrunner, c_mac, caddcl, cadlisp, cfdg, cfm, chaiscript, cil, clojure, cmake, cobol, coffeescript, cpp, cpp-qt, csharp, css, cuesheet, d, dcs, delphi, diff, div, dos, dot, e, ecmascript, eiffel, email, epc, erlang, euphoria, f1, falcon, fo, fortran, freebasic, fsharp, gambas, gdb, genero, genie, gettext, glsl, gml, gnuplot, go, groovy, gwbasic, haskell, hicest, hq9plus, html4strict, html5, icon, idl, ini, inno, intercal, io, j, java, java5, javascript, jquery, kixtart, klonec, klonecpp, latex, lb, lisp, llvm, locobasic, logtalk, lolcode, lotusformulas, lotusscript, lscript, lsl2, lua, m68k, magiksf, make, mapbasic, matlab, mirc, mmix, modula2, modula3, mpasm, mxml, mysql, newlisp, nsis, oberon2, objc, objeck, ocaml, ocaml-brief, oobas, oracle11, oracle8, oxygene, oz, pascal, pcre, per, perl, perl6, pf, php, php-brief, pic16, pike, pixelbender, pli, plsql, postgresql, povray, powerbuilder, powershell, proftpd, progress, prolog, properties, providex, purebasic, pycon, python, q, qbasic, rails, rebol, reg, robots, rpmspec, rsplus, ruby, sas, scala, scheme, scilab, sdlbasic, smalltalk, smarty, sql, systemverilog, tcl, teraterm, text, thinbasic, tsql, typoscript, unicon, uscript, vala, vb, vbnet, verilog, vhdl, vim, visualfoxpro, visualprolog, whitespace, whois, winbatch, xbasic, xml, xorg_conf, xpp, yaml, z80, zxbasic

```
public class MenuScreen extends Form
    implements CommandListener {

    MenuScreen (YourMIDlet midlet,
                HttpPoster httpPoster)
    {
        super("Menu");
        this.midlet = midlet;
        this.httpPoster = httpPoster;

        usernameField = new TextField("User", null, 16, TextField.ANY);
        append(usernameField);
        passwordField = new TextField("Password", null, 8, TextField.PASSWORD);
        append(passwordField);
        quitCommand = new Command("Quit", Command.EXIT, 2);
        addCommand(quitCommand);
        loginCommand = new Command("Login", Command.SCREEN, 2);
        addCommand(loginCommand);

        setCommandListener(this);
    }

    public void commandAction(Command c, Displayable d)
    {
        if (c == quitCommand)
        {
            midlet.loginScreenQuit();
        }
        ...
    }

}
```

API Mappings

Below we have the APIs in categories, considering the original functionality in the orginal BlackBerry APIs and the corresponding APIs in S60. When porting from BlackBerry to S60, just check to find out the similar API available on S60.

User Interfaces

- RIM
  - net.rim.device.api.ui
  - javax.microedition.lcdui
  - javax.microedition.lcdui.game
  - javax.microedition.midlet
  - net.rim.device.api.ui
  - net.rim.device.api.ui.autotext
  - net.rim.device.api.ui.component
  - net.rim.device.api.ui.container
  - net.rim.device.api.ui.decor
  - net.rim.device.api.ui.text
  - javax.microedition.m2g
  - org.w3c.dom
  - org.w3c.dom.events
  - org.w3c.dom.svg
  - net.rim.plazmic.mediaengine
  - net.rim.plazmic.mediaengine.io
  - net.rim.blackberry.api.spellcheck
  - net.rim.device.api.browser.field
- S60
  - javax.microedition.lcdui
  - javax.microedition.lcdui.game
  - javax.microedition.midlet
  - javax.microedition.m2g
  - org.w3c.dom
  - org.w3c.dom.events
  - org.w3c.dom.svg

(C) Copyright Nokia 2013. All rights reserved.

- com.nokia.mid.ui
- org.eclipse.ercp.swt.mobile
- org.eclipse.swt
- org.eclipse.swt.browser
- org.eclipse.swt.dnd
- org.eclipse.swt.events
- org.eclipse.swt.graphics
- org.eclipse.swt.internal
- org.eclipse.swt.layout
- org.eclipse.swt.widgets

Location

- RIM
  - net.rim.device.api.lbs
  - javax.microedition.location
  - net.rim.blackberry.api.maps
  - net.rim.device.api.gps

- S60
  - javax.microedition.location

Multimedia

- RIM
  - javax.microedition.media
  - javax.microedition.media.control
  - javax.microedition.media.protocol
  - net.rim.device.api.control

- S60
  - javax.microedition.media
  - javax.microedition.media.control
  - javax.microedition.media.protocol
  - com.nokia.mid.sound

Security

- RIM
  - javax.microedition.pki
  - net.rim.device.api.crypto
  - net.rim.device.api.crypto.asn1
  - net.rim.device.api.crypto.certificate
  - net.rim.device.api.crypto.certificate.status
  - net.rim.device.api.crypto.certificate.wtls
  - net.rim.device.api.crypto.certificate.x509
  - net.rim.device.api.crypto.cms
  - net.rim.device.api.crypto.encoder
  - net.rim.device.api.crypto.keystore
  - net.rim.device.api.crypto.oid
  - net.rim.device.api.crypto.tls
  - net.rim.device.api.crypto.tls.ssl30
  - net.rim.device.api.crypto.tls.tls10
  - net.rim.device.api.crypto.tls.wtls20
- S60
  - javax.microedition.apdu
  - java.rmi
  - javacard.framework
  - javacard.framework.service
  - javacard.security
  - javax.microedition.jcrmi
  - javax.microedition.pki
  - javax.microedition.securityservice
  - java.security
  - java.security.spec
  - javax.crypto
  - javax.crypto.spec

Persistent Storage

(C) Copyright Nokia 2013. All rights reserved.

- RIM
  - javax.microedition.io.file
  - javax.microedition.rms
  - net.rim.device.api.system
  - net.rim.device.api.util
  - net.rim.device.api.lowmemory
  - net.rim.device.api.memorycleaner
  - net.rim.device.api.io.file
  - net.rim.device.api.synchronization
- S60
  - javax.microedition.rms
  - javax.microedition.io.file

CLDC and Support Packages

- RIM
  - java.lang
  - java.lang.ref
  - java.util
  - java.rmi
  - javax.microedition.xml.rpc
  - javax.xml.namespace
  - javax.xml.parsers
  - javax.xml.rpc
  - net.rim.device.api.collection
  - net.rim.device.api.collection.util
  - net.rim.device.api.math
  - net.rim.device.api.util
  - net.rim.device.api.xml
  - net.rim.device.api.xml.jaxp
  - net.rim.device.api.xml.parsers
  - org.xml.sax
  - org.xml.sax.helpers
- S60
  - java.io
  - java.lang
  - java.lang.ref
  - java.security
  - java.util
  - javax.microedition.io

Connection Framework (GCF) and I/O

- RIM
  - java.io
  - javax.bluetooth
  - javax.obex
  - javax.microedition.io
  - javax.microedition.apdu
  - javax.microedition.jcrmi
  - javacard.framework
  - javacard.framework.service
  - javacard.security
  - net.rim.device.api.compress
  - net.rim.device.api.io
  - net.rim.device.api.io.http
  - net.rim.device.api.mime
  - net.rim.device.cldc.io.ssl
  - net.rim.device.api.ldap
  - net.rim.device.api.smartcard
  - net.rim.device.api.bluetooth
- S60
  - java.io
  - javax.bluetooth
  - javax.obex
  - javax.microedition.io
  - javax.microedition.apdu
  - javax.microedition.jcrmi
  - javax.microedition.sip

(C) Copyright Nokia 2013. All rights reserved.

Messaging

In Blackbery to send SMS from CDMA device developer need to use DatagramConnection which we do not require in J2ME for s60 plateform.

- RIM
  - net.rim.blackberry.api.mail
  - net.rim.blackberry.api.mail.event
  - net.rim.blackberry.api.messagelist
  - javax.wireless.messaging
  - net.rim.blackberry.api.sms
  - net.rim.device.api.io.DatagramConnectionBase
- S60
  - javax.wireless.messaging

PIM Data (Contacts, Calendar, Tasks)

- RIM
  - javax.microedition.pim
  - net.rim.blackberry.api.pdap
- S60
  - javax.microedition.pim

Content Handling

- RIM
  - javax.microedition.content
  - net.rim.device.api.content
- S60
  - javax.microedition.content

BlackBerry and Nokia runtimes - specific APIs

- RIM
  - net.rim.device.api.system
  - net.rim.device.api.notification
  - net.rim.device.api.servicebook
  - net.rim.device.api.synchronization
  - net.rim.device.api.applicationcontrol
  - net.rim.device.api.itpolicy
  - net.rim.blackberry.api.browser
  - net.rim.device.api.browser.field
  - net.rim.device.api.browser.plugin
  - net.rim.blackberry.api.phone
  - net.rim.blackberry.api.phone.phonelogs
  - net.rim.blackberry.api.blackberrymessenger
  - net.rim.blackberry.api.invoke
  - net.rim.device.api.system
  - net.rim.device.api.browser.plugin
  - net.rim.blackberry.api.homescreen
  - net.rim.blackberry.api.menuitem
  - net.rim.blackberry.api.messagelist
  - net.rim.blackberry.api.options
  - net.rim.blackberry.api.stringpattern
- S60
  - com.nokia.mid.iapinfo

S60 only APIs

- S60 - Mobile Sensor API
  - javax.microedition.sensor
  - javax.microedition.sensor.control
- S60 - Advanced Multimedia Supplements
  - javax.microedition.amms
  - javax.microedition.amms.control
  - javax.microedition.amms.control.audio3d
  - javax.microedition.amms.control.audioeffect
  - javax.microedition.amms.control.camera
  - javax.microedition.amms.control.imageeffect
  - javax.microedition.amms.control.tuner
- S60 - Mobile 3D Graphics
  - javax.microedition.m3g
- S60 - Web-Services (RPC and XML Parser)
  - java.rmi

(C) Copyright Nokia 2013. All rights reserved.

- javax.microedition.xml.rpc
- javax.xml.namespace
- javax.xml.rpc

Inner Class will be converted to new Independent Class

When porting a BlackBerry GUI application, you can see example code where some event handling code is achieved by using static final inner classes.

At the opposite side most of the GUI code sample in Nokia Developer and provided by Sun Microsystems generally has the class file in new Java file and somehow it creates better understanding.

This is due to the fact that when using the native BlackBerry GUI Widgets they use an approach that is similar to Java SE GUI frameworks like AWT or Swing.

So suppose there might be functionality to send sms in inner class, while porting the application you should concentrate to create that functionality in new class or in new method. Anyway it is up to you and your team to define the best way to achieve it using your coding conventions, as defined by your team or company.

Data Storage on device

Blackberry : RMS, FileConnection API

J2ME : RMS, FileConnection API

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.