# A10 – Writing Your First Application

for BlackBerry

## Contents

# A10 – Writing Your First Application

This tutorial will show you how to write a basic application for BlackBerry devices. The application will display the simple "Hello World" message on the screen.

To be able to do so it is required to have the following installed on your system:

- Sun JDK
- Eclipse SDK,
- BlackBerry JDE Plug-in for Eclipse and
- BlackBerry JDE Component Packs 4.3 – 4.7

If you need help setting up please look at A1 – Setting up Tools tutorial.

If you are ready, launch the Eclipse, and go to your Workbench (Figure 1).

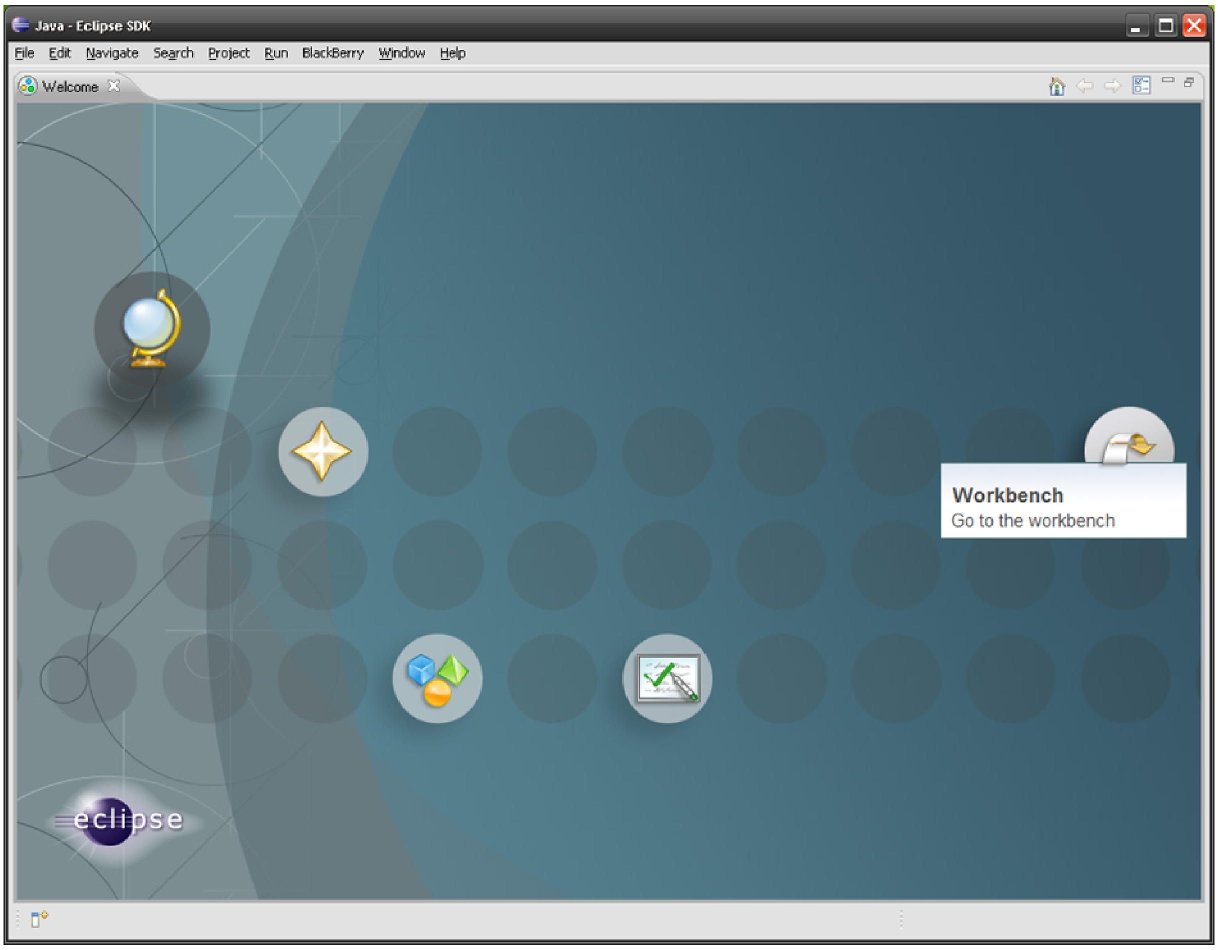

Java - Eclipse SDK
File Edit Navigate Search Project Run BlackBerry Window Help
Welcome
Workbench
Go to the workbench
eclipse

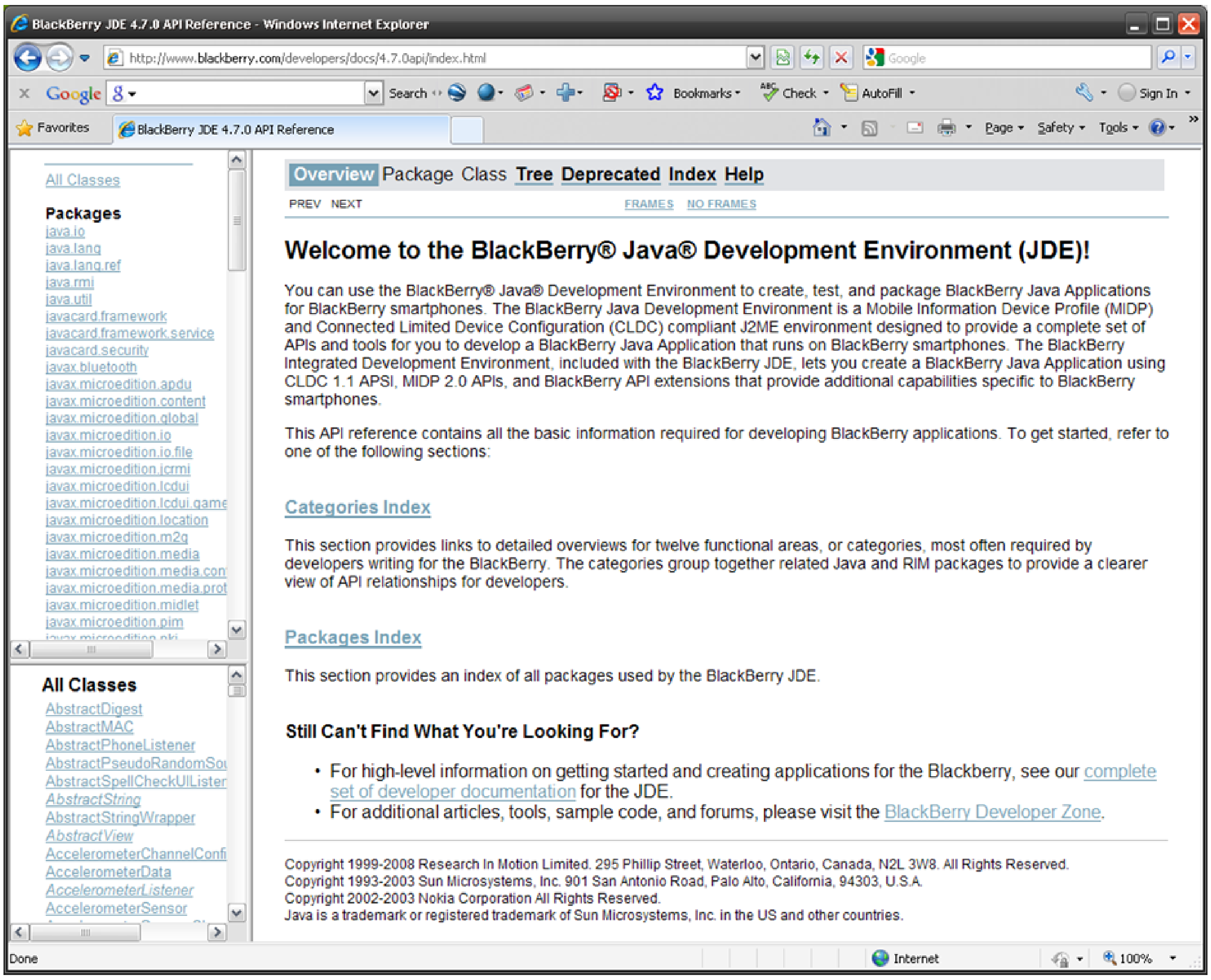
Figure 1

## Introduction

In this tutorial I will show you the following:

- How to setup and configure new BlackBerry project,
- How to create classes,
- Some details about UiApplication and MainScreen,
- How to write a Hello World application and
- How to run your application in simulator.

Please note that you can always find more information about the APIs we are using in BlackBerry API reference document, which is the part of any JDE Component Pack (Figure 2). You can find it on your computer under Start / Programs / Research in Motion / BlackBerry JDE 4.x.


BlackBerry JDE 4.7.0 API Reference - Windows Internet Explorer
http://www.blackberry.com/developers/docs/4.7.0api/index.html
Google
Search
Bookmarks
Check
AutoFill
Sign In
Favorites
BlackBerry JDE 4.7.0 API Reference
Page
Safety
Tools
Overview Package Class Tree Deprecated Index Help
PREV NEXT
FRAMES NO FRAMES
Done
Internet
100%

All Classes

**Packages**
java.io
java.lang
java.lang.ref
java.rmi
java.util
javacard.framework
javacard.framework.service
javacard.security
javax.bluetooth
javax.microedition.apdu
javax.microedition.content
javax.microedition.global
javax.microedition.io
javax.microedition.io.file
javax.microedition.jcrmi
javax.microedition.lcdui
javax.microedition.lcdui.game
javax.microedition.location
javax.microedition.m2g
javax.microedition.media
javax.microedition.media.con
javax.microedition.media.prot
javax.microedition.midlet
javax.microedition.pim

**All Classes**
AbstractDigest
AbstractMAC
AbstractPhoneListener
AbstractPseudoRandomSou
AbstractSpellCheckUIListen
*AbstractString*
AbstractStringWrapper
*AbstractView*
AccelerometerChannelConfi
AccelerometerData
*AccelerometerListener*
AccelerometerSensor

**Welcome to the BlackBerry® Java® Development Environment (JDE)!**

You can use the BlackBerry® Java® Development Environment to create, test, and package BlackBerry Java Applications for BlackBerry smartphones. The BlackBerry Java Development Environment is a Mobile Information Device Profile (MIDP) and Connected Limited Device Configuration (CLDC) compliant J2ME environment designed to provide a complete set of APIs and tools for you to develop a BlackBerry Java Application that runs on BlackBerry smartphones. The BlackBerry Integrated Development Environment, included with the BlackBerry JDE, lets you create a BlackBerry Java Application using CLDC 1.1 APSI, MIDP 2.0 APIs, and BlackBerry API extensions that provide additional capabilities specific to BlackBerry smartphones.

This API reference contains all the basic information required for developing BlackBerry applications. To get started, refer to one of the following sections:

Categories Index

This section provides links to detailed overviews for twelve functional areas, or categories, most often required by developers writing for the BlackBerry. The categories group together related Java and RIM packages to provide a clearer view of API relationships for developers.

Packages Index

This section provides an index of all packages used by the BlackBerry JDE.

**Still Can't Find What You're Looking For?**

- For high-level information on getting started and creating applications for the Blackberry, see our complete set of developer documentation for the JDE.
- For additional articles, tools, sample code, and forums, please visit the BlackBerry Developer Zone.

Copyright 1999-2008 Research In Motion Limited. 295 Phillip Street, Waterloo, Ontario, Canada, N2L 3W8. All Rights Reserved.
Copyright 1993-2003 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California, 94303, U.S.A.
Copyright 2002-2003 Nokia Corporation All Rights Reserved.
Java is a trademark or registered trademark of Sun Microsystems, Inc. in the US and other countries.

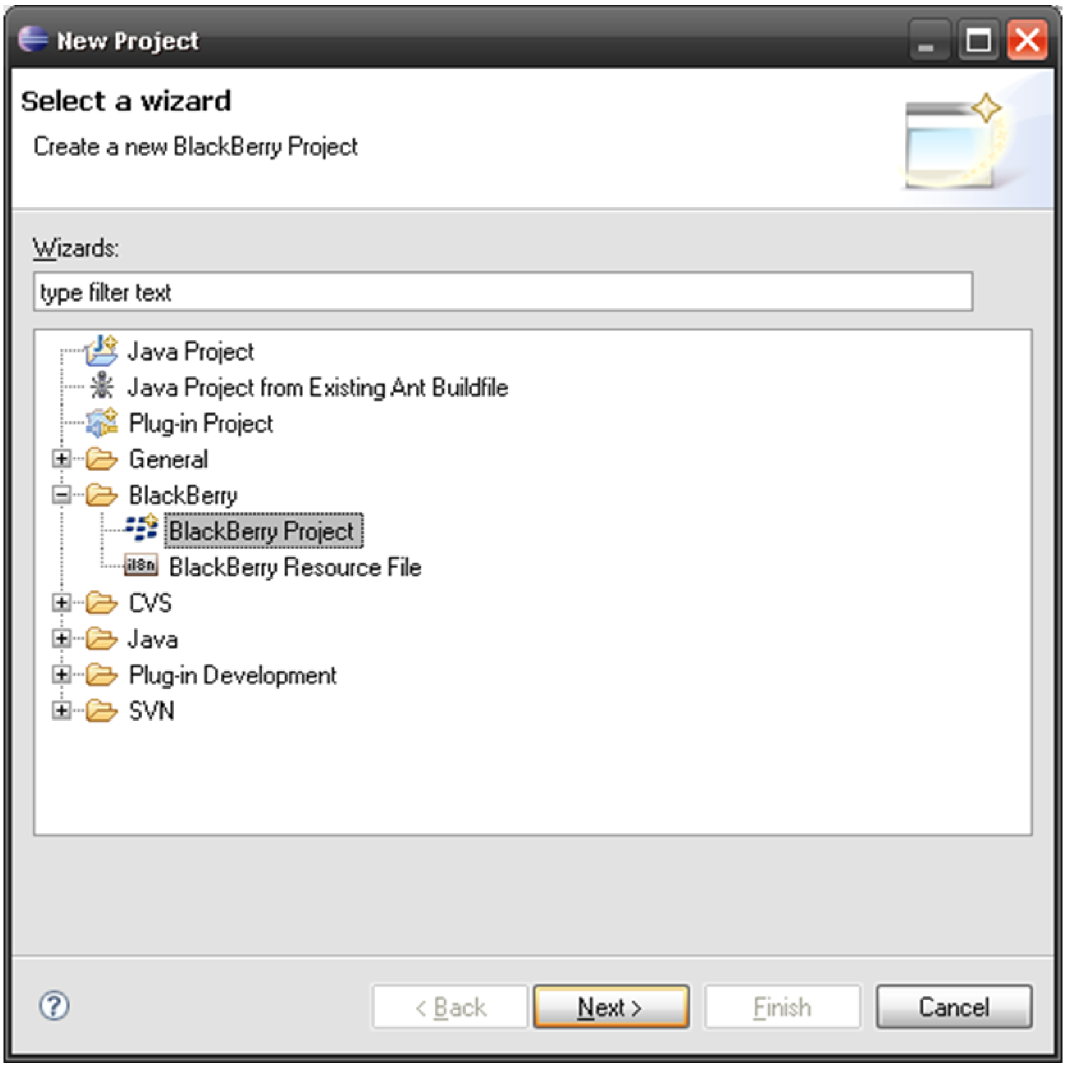
Figure 2

# Development

## Setting up New BlackBerry Project

To setup your new BlackBerry project:

1. Click on File/New/Project menu.
2. Select BlackBerry project (Figure 3).



New Project

Select a wizard

Create a new BlackBerry Project

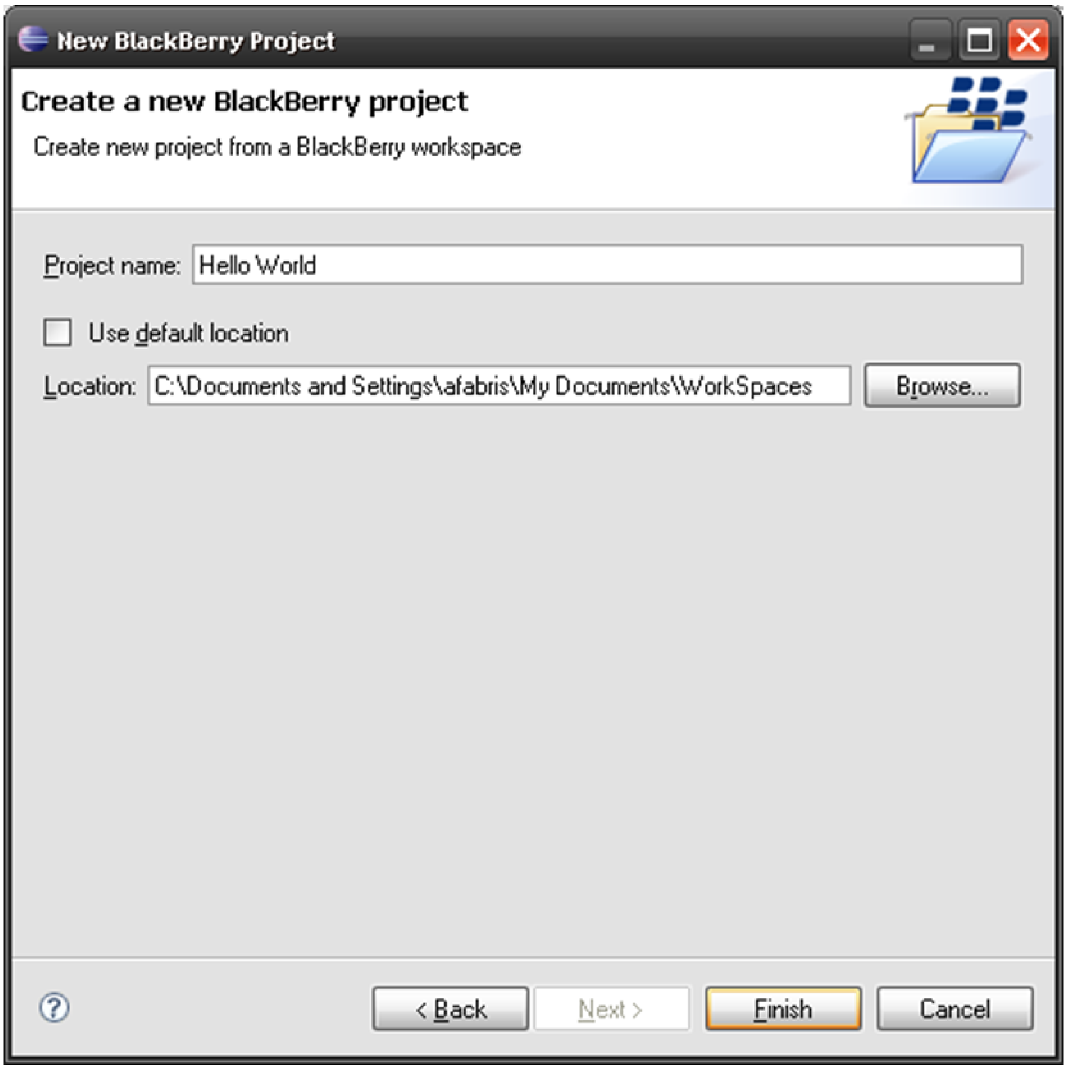
Wizards:

type filter text

- Java Project
- Java Project from Existing Ant Buildfile
- Plug-in Project
- General
- BlackBerry
  - BlackBerry Project
  - BlackBerry Resource File
- CVS
- Java
- Plug-in Development
- SVN

Figure 3

3. Click Next.
4. Chose Project name and location (Figure 4).


New BlackBerry Project

Create a new BlackBerry project

Create new project from a BlackBerry workspace

Project name: Hello World

Use default location ☐

Location: C:\Documents and Settings\afabris\My Documents\WorkSpaces

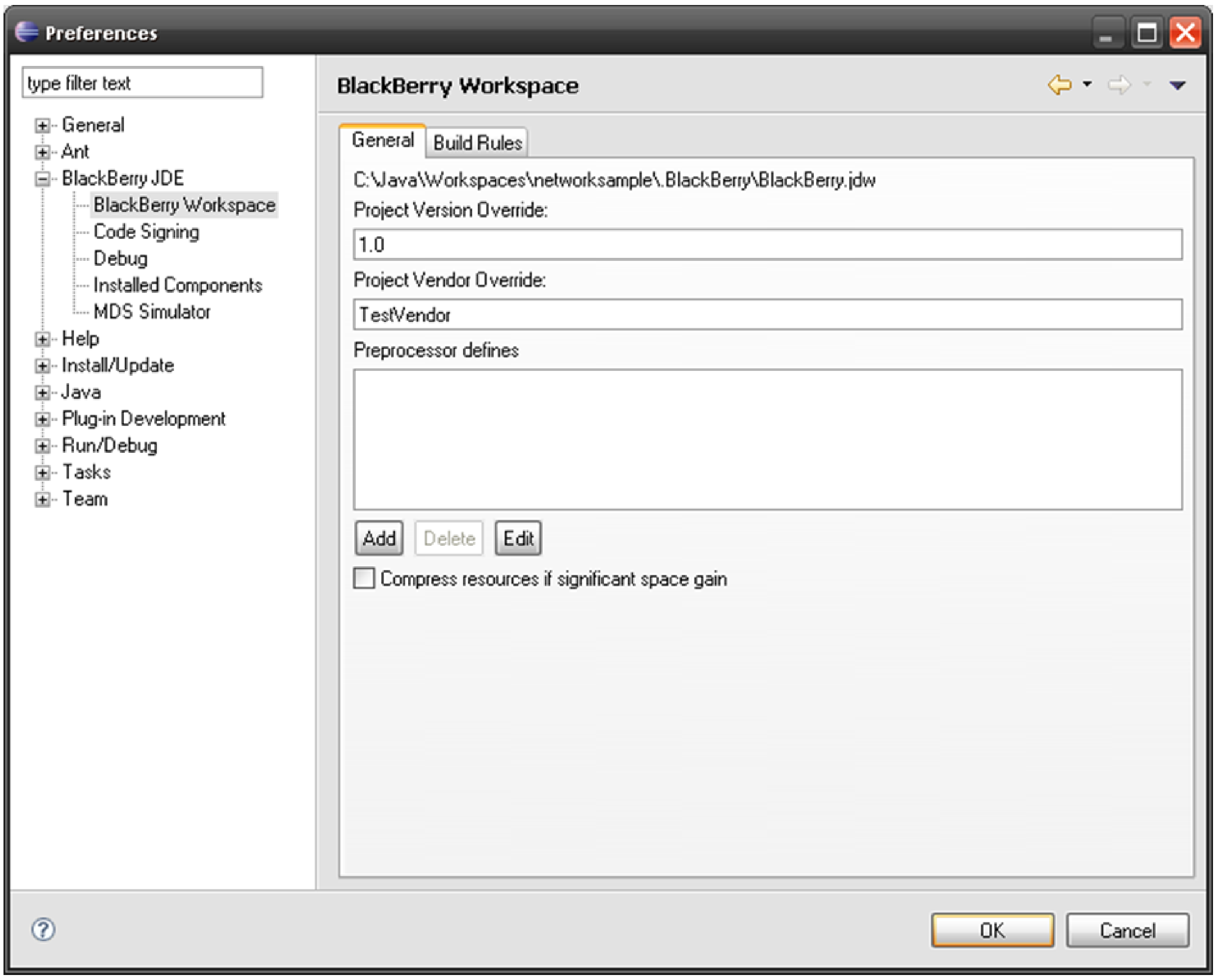
Figure 4

5. Enter the project name, i.e. “Hello World“.
6. Select your location or use a default one to store your project.
7. Click Finish.

## Configuring Your New BlackBerry Project

To configure your new BlackBerry project:

1. Click on BlackBerry/Configure Blackberry Workspace.
2. Click on the BlackBerry Workspace and insert your Vendor and Version data (Figure 5).



C:\Java\Workspaces\networksample\.BlackBerry\BlackBerry.jdw

Project Version Override: 1.0

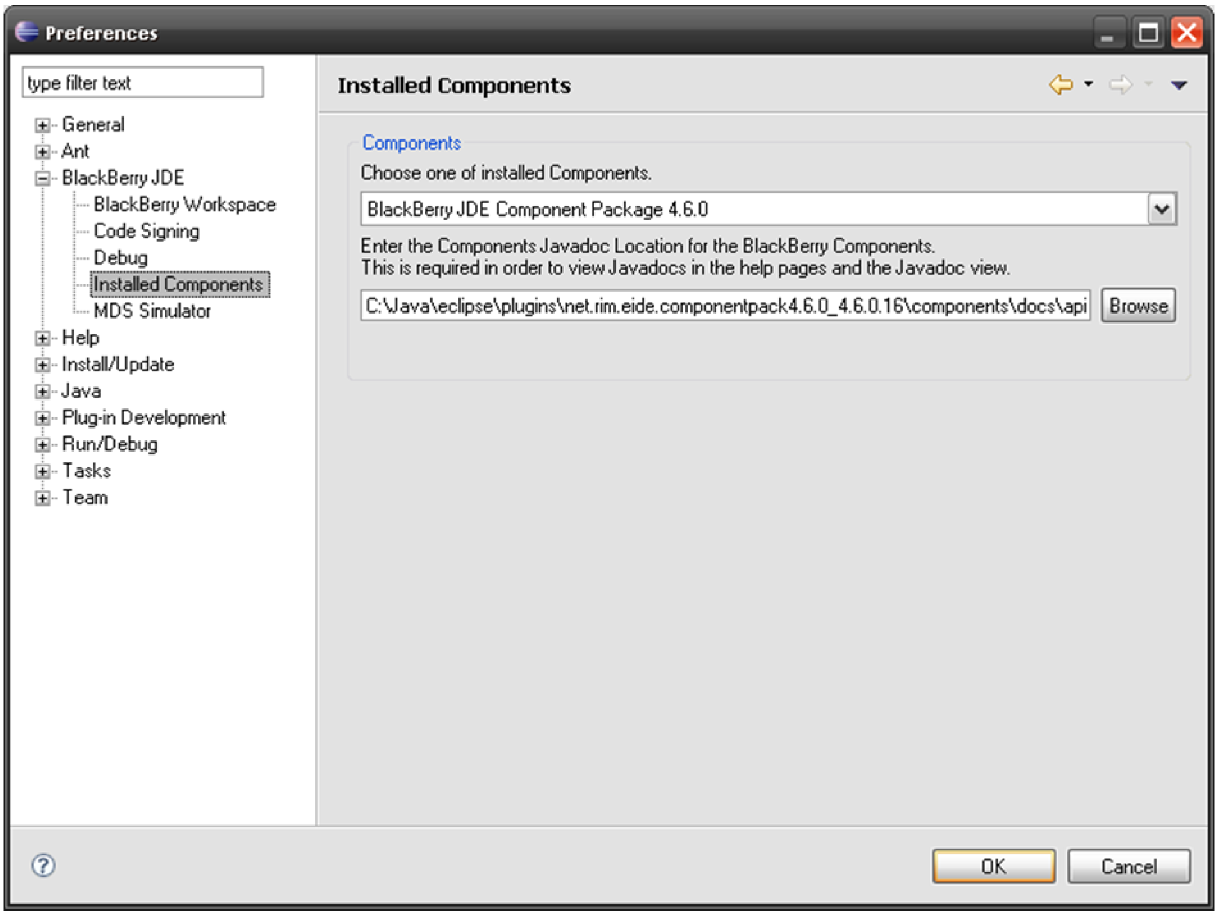
Project Vendor Override: TestVendor

Preprocessor defines

☐ Compress resources if significant space gain

Figure 5

3. Here you can change a number of different settings. Let's just enter version number 1.0 and vendor "TestVendor".
4. From BlackBerry JDE, select Installed Components



Installed Components

Components

Choose one of installed Components.

BlackBerry JDE Component Package 4.6.0

Enter the Components Javadoc Location for the BlackBerry Components.
This is required in order to view Javadocs in the help pages and the Javadoc view.

C:\Java\eclipse\plugins\net.rim.eide.componentpack4.6.0_4.6.0.16\components\docs\api

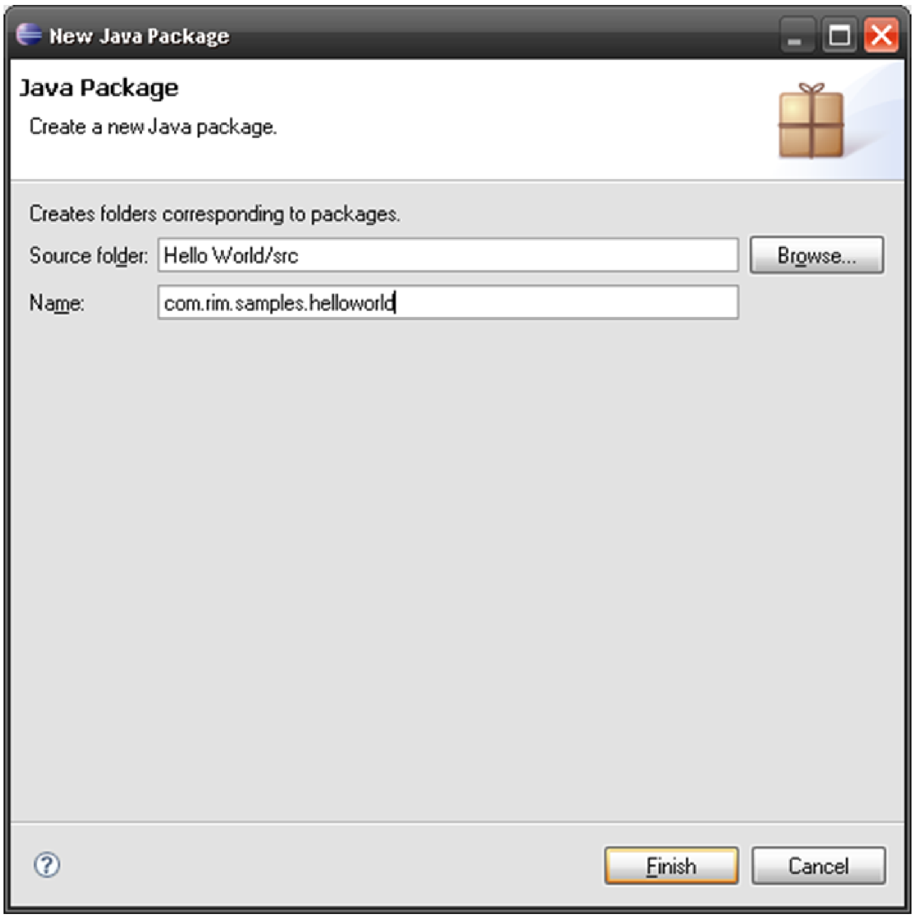
Figure 6

5. Chose component package – 4.6.0 (Figure 6).
6. Click OK.

## Creating HelloWorld Class

To start developing our application after setting up and configuring our Workspace, we need to create a new HelloWorld Class:

1. Click on File/New/Package (Figure 7).
2. Enter the package path i.e. com.rim.samples.helloworld.


New Java Package

Java Package

Create a new Java package.

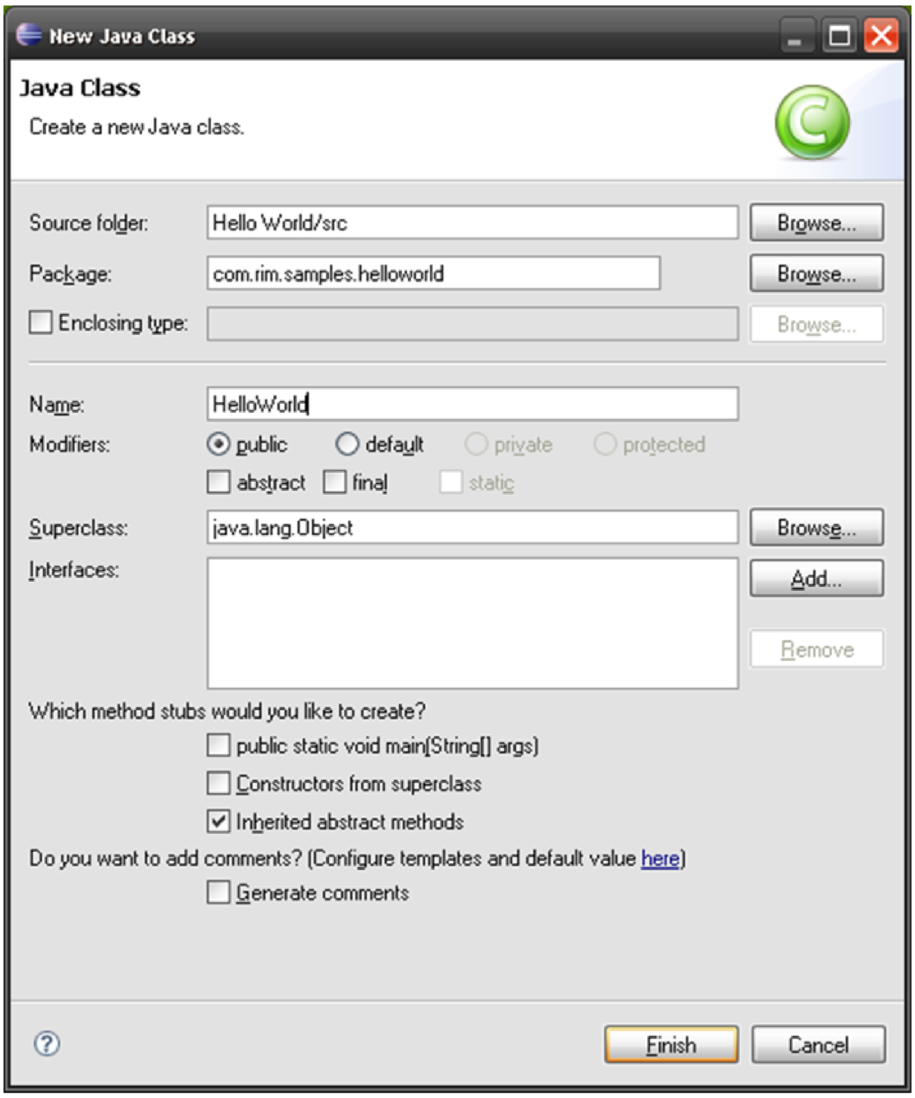
Creates folders corresponding to packages.

Source folder: Hello World/src

Name: com.rim.samples.helloworld

Figure 7

3. Click on Finish button.
4. Click on File/New/Class (Figure 8)



Java Class

Create a new Java class.

Source folder: Hello World/src

Package: com.rim.samples.helloworld

☐ Enclosing type:

Name: HelloWorld

Modifiers: [✓ public] default private protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Figure 8

5. Check the source folder and package. It should be Hello World/src and com.rim.samples.helloworld respectively.
6. Enter HelloWorld as the name and click Finish.

## UiApplication

Any BlackBerry application that provides a user interface must extend the UiApplication class.

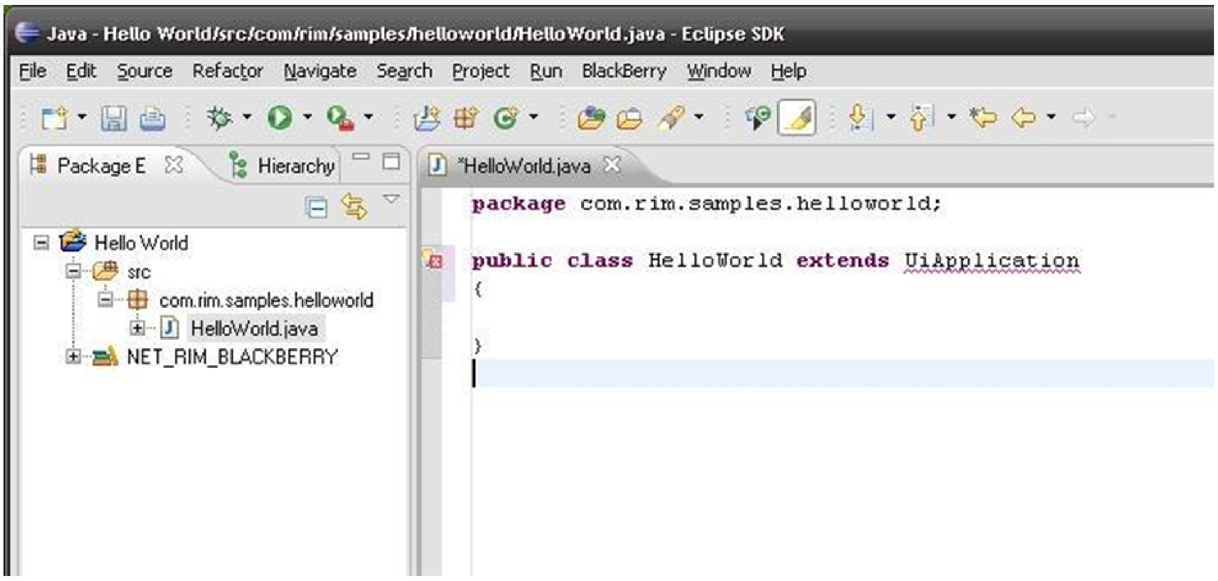
A UI application maintains a stack of Screen objects. As it pushes screens onto the stack, it draws them on top of any other screens already on the stack. When the application pops a screen off the stack, it redraws the underlying screens as necessary. Only the screen on the top of the stack receives input events.

To implement a UiApplication, the HelloWorld class we just created must extend the UiApplication class:

1. Type `extends UiApplication` after `public class HelloWorld` (Figure 9).
2. We need to import the `net.rim.device.api.ui` package. To the left of the `public class HelloWorld` line is a light bulb icon with red cross. Click on it and then click on Import 'UiApplication' (Figure 10).
3. You can also type:

   ```
   import net.rim.device.api.ui.UiApplication;
   ```

but the above approach saves you a bit of time.

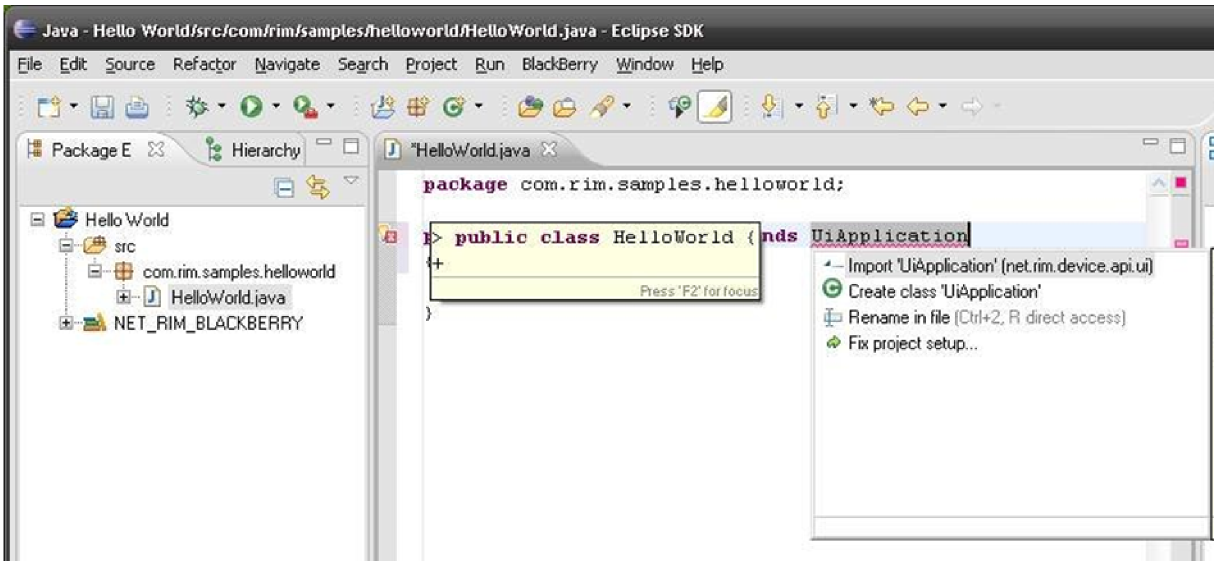

Java - Hello World/src/com/rim/samples/helloworld/HelloWorld.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run BlackBerry Window Help
Package E
Hierarchy
*HelloWorld.java
Hello World
src
com.rim.samples.helloworld
HelloWorld.java
NET_RIM_BLACKBERRY

```
package com.rim.samples.helloworld;

public class HelloWorld extends UiApplication
{

}
```

Figure 9


Java - Hello World/src/com/rim/samples/helloworld/HelloWorld.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run BlackBerry Window Help
Package E
Hierarchy
Hello World
src
com.rim.samples.helloworld
HelloWorld.java
NET_RIM_BLACKBERRY
*HelloWorld.java
Press 'F2' for focus
Import 'UiApplication' (net.rim.device.api.ui)
Create class 'UiApplication'
Rename in file (Ctrl+2, R direct access)
Fix project setup...

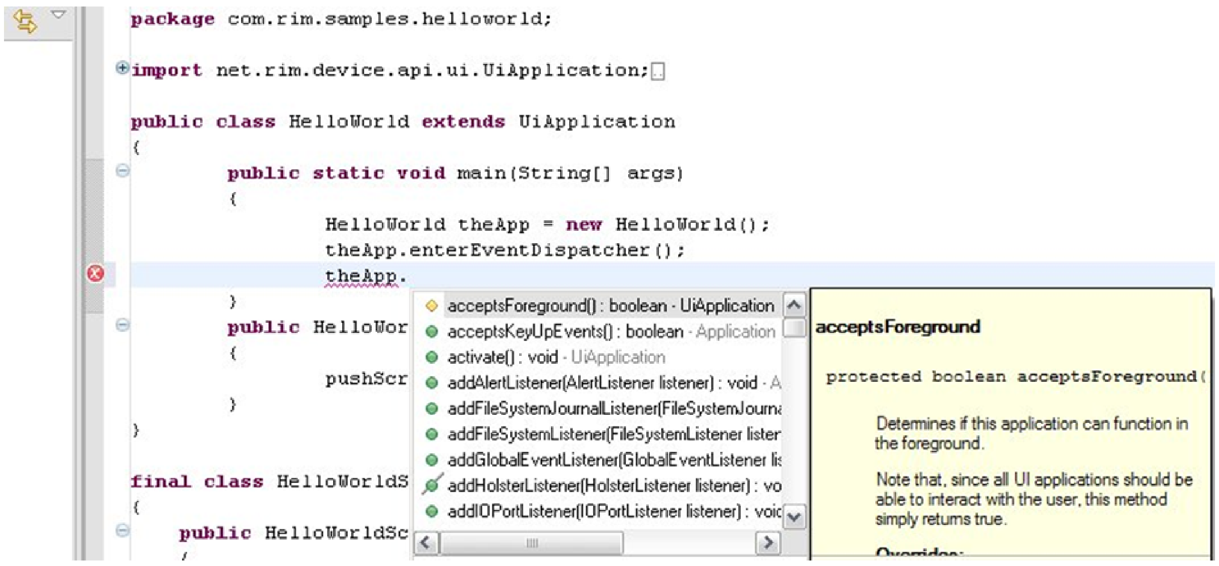
```
package com.rim.samples.helloworld;

public class HelloWorld extends UiApplication
}
```

Figure 10

4. Class HelloWorld must have one method:

- `main(String[] args)` which is the entry point into our application. We need to create an instance of our application and start it by getting it on the event thread. To get more information about any methods used in this tutorial (i.e. enterEventDispatcher) please check the API reference document. Also, when you type in the Eclipse editor, you will get tips, such as when you type theApp and press . (dot) after a second or so a list of options will come up with tips about what each of the options do (Figure 11).



```
package com.rim.samples.helloworld;

import net.rim.device.api.ui.UiApplication;

public class HelloWorld extends UiApplication
{
        public static void main(String[] args)
        {
                HelloWorld theApp = new HelloWorld();
                theApp.enterEventDispatcher();
                theApp.
        }
        public HelloWor
        {
                pushScr
        }
}

final class HelloWorldS
{
    public HelloWorldSc
```

```
acceptsForeground() : boolean - UiApplication
acceptsKeyUpEvents() : boolean - Application
activate() : void - UiApplication
addAlertListener(AlertListener listener) : void - A
addFileSystemJournalListener(FileSystemJourna
addFileSystemListener(FileSystemListener lister
addGlobalEventListener(GlobalEventListener lis
addHolsterListener(HolsterListener listener) : vo
addIOPortListener(IOPortListener listener) : voic
```

**acceptsForeground**

`protected boolean acceptsForeground(`

Determines if this application can function in the foreground.

Note that, since all UI applications should be able to interact with the user, this method simply returns true.

Overrides:

Figure 11

5. In the constructor we will create a new Screen object and display it (Figure 12).

```
package com.rim.samples.helloworld;

import net.rim.device.api.ui.UiApplication;

public class HelloWorld extends UiApplication
{
        public static void main(String[] args)
        {
                HelloWorld theApp = new HelloWorld();
                theApp.enterEventDispatcher();
        }
        public HelloWorld()
        {
                //display a new screen
                pushScreen(new HelloWorldScreen());
        }
}
```

Figure 12

## MainScreen

MainScreen is a class which provides us with a working area where we can display all the data from our application (In this case the “Hello World” message). It has title section, separator, and main scrollable section.

1. We will add a HelloWorldScreen class which extends MainScreen:
```
final class HelloWorldScreen extends MainScreen
```

2. You will need to add an import:
```
import net.rim.device.api.ui.container.MainScreen;
```

You can type the line or click on light bulb icon as described before.

3. In the constructor of this class we will create `LabelField title`, to label our Application. Set the title and add a RichTextField with our message “Hello World!”. We also need to remember to import necessary packages as well as call MainScreen’s constructor (Figure 13).

```
public HelloWorldScreen()
    {
            super();
            LabelField title = new LabelField("HelloWorld Sample",
                 LabelField.ELLIPSIS | LabelField.USE_ALL_WIDTH);
            setTitle(title);
            add(new RichTextField("Hello World!"));
    }
```

**Figure 13**

4. The imports we have to add are:

```
import net.rim.device.api.ui.component.LabelField;
import net.rim.device.api.ui.component.RichTextField;
```

5. We will add a small dialog which will appear when the user wants to exit the application. To do so we need to overwrite the onClose() method of the HelloWorldScreen class:

```
public boolean onClose()
{
      Dialog.alert("Goodbye!");
      System.exit(0);
      return true;
}
```

6. Again we need to import the component so we add:

   ```
   import net.rim.device.api.ui.component.Dialog;
   ```

7. We can also use more generic approach to import all the components instead of importing one by one:

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

8. And with that we have finished our application. The complete code is shown on (Figure 14).

```
package com.rim.samples.helloworld;

import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.MainScreen;

public class HelloWorld extends UiApplication
{
        public static void main(String[] args)
        {
                HelloWorld theApp = new HelloWorld();
                theApp.enterEventDispatcher();
        }
        public HelloWorld()
        {
                pushScreen(new HelloWorldScreen());
        }
}

final class HelloWorldScreen extends MainScreen
{
      public HelloWorldScreen()
    {
            super();
            LabelField title = new LabelField("HelloWorld Sample",
                 LabelField.ELLIPSIS | LabelField.USE_ALL_WIDTH);
            setTitle(title);
            add(new RichTextField("Hello World!"));
    }

      public boolean onClose()
    {
        Dialog.alert("Goodbye!");
        System.exit(0);
        return true;
    }
}
```

Figure 14

## Running Application in the Simulator

Running the application is quite easy:

1. Click on Run/Run or the green shortcut icon on the toolbar.
2. You can also choose to click Run/Debug, which will allow you to debug your application, but it also takes longer to load.
3. When you get the simulator (Figure 15) find and start your application from Downloads folder.
4. When you run the application you should see our Hello World message (Figure 16) .
5. And when you click on exit button you will get the “Goodbye” dialog (Figure 17).
6. To exit the simulator, just close its window.

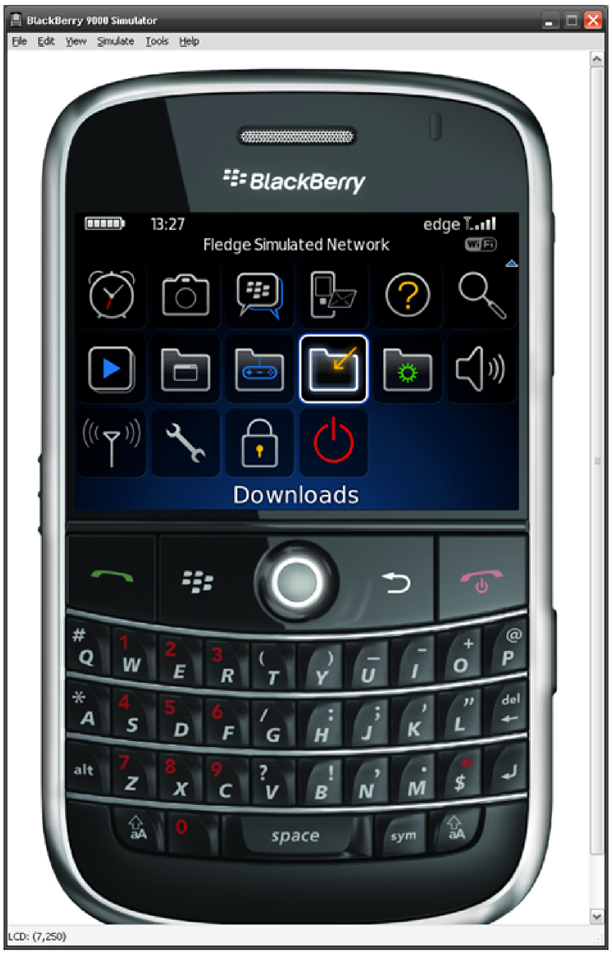
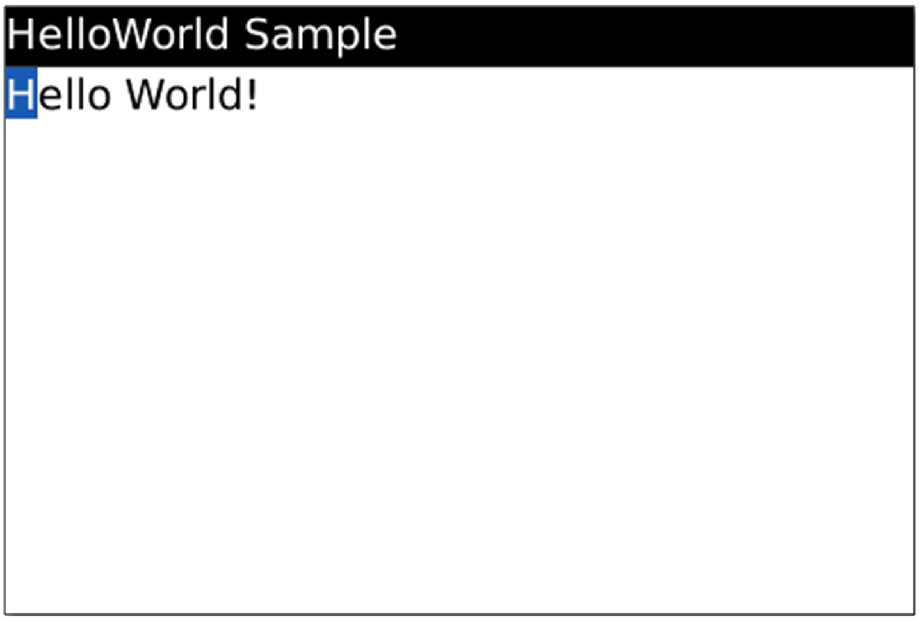
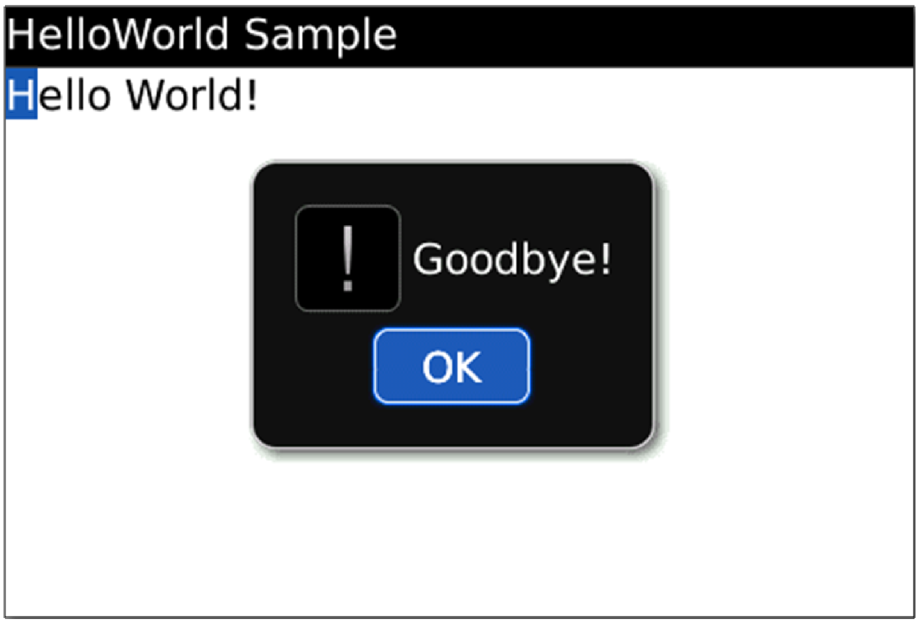


Figure 15


HelloWorld Sample

Hello World!

Figure 16


HelloWorld Sample

Hello World!

Goodbye!

Figure 17

## Other Variations

There are number of things you might want to try with this sample application:

- Try to run it on a different simulator,
- Try to change title and messages,
- Try to add other fields from the `net.rim.device.api.ui.component` package such as the SeparatorField,
- Rename the classes or make separate HelloWorld and HelloWorldScreen class files.

To exit Eclipse, click on File/Exit or just close the window. It will save your projects and when you open it the next time they will be there.

## Links

**BlackBerry Developers Web Site:**

http://na.blackberry.com/eng/developers/

**Developer Video Library:**

- Introduction to BlackBerry Development:

  http://www.blackberry.com/DevMediaLibrary/view.do?name=IntroBlackBerryDev
- Eclipse Installation and Configuration:

  http://www.blackberry.com/DevMediaLibrary/view.do?name=eclipseJDE
- Introduction to BlackBerry Simulators:

  http://www.blackberry.com/DevMediaLibrary/view.do?name=simsintro

**Developer Labs:**

- Writing your first application:

  http://na.blackberry.com/eng/developers/resources/developer_labs.jsp#tab_tab_jde