Building BlackBerry applications is much like building Java Swing applications. The same general concepts apply, but there also are some very interesting differences, from how list controls work all the way up to how you support multiple entry points in an application.

The design of the BlackBerry means that I need to use a single application entry point to trigger both the given application that I'm testing, as well as the unit tests for that application. In Eclipse, both the application source code and test framework source code need to sit side-by-side in one Eclipse project

1. Open the project properties for the secondary (testing) project.
2. Select **BlackBerry Project Properties**.
3. Select the **Application** tab.
4. From the **Project type:** dropdown, select **Alternate CLDC Application Entry Point**.
5. Ensure that the **Alternate entry point for:** dropdown shows the primary project as being selected.
6. In the field marked **Argument passed to "static public void main(String args[])":**, enter **unittest**

Listing 1: A main application class.

```
import net.rim.device.api.ui.*;
import com.langrsoft.agent.*;
public class AgentApp extends UiApplication {
  public static void main(String args[]) {
    AgentApp app = new AgentApp();
    if (isInTestMode(args))
      app.test();
    else
      app.go();
    app.enterEventDispatcher();
  }
```

```
  private void test() {
    pushScreen(new TestApp(new SampleTest()));
  }
  private static boolean isInTestMode(String[] args) {
    return args.length > 0 && "unittest".equals(args[0]);
  }
  public void go() {
    pushScreen(new MainApp());
  }
}
```

Listing 2: TestApp.

```
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import com.langrsoft.bbtest.*;
public class TestApp extends MainScreen
  implements ResultListener {
  private ListField list = new ListField();
  private TestListFieldModel model = new TestListFieldModel();
  public TestApp(final MicroTestGroup group) {
    list.setCallback(model);
    add(list);
    group.setListener(this);
    new Thread(new Runnable() {
      public void run() {
        group.execute();
      }
    }).start();
```

```
  }
  public void ran(final MicroTest test) {
    UiApplication.getUiApplication().invokeLater(new Runnable() {
      public void run() {
        list.insert(0);
        model.insert(test);
      }
    });
  }
  protected void onObscured() {
    UiApplication.getApplication().requestForeground();
  }
  public boolean keyChar(char key, int status, int time) {
    if (key == Characters.ESCAPE) System.exit(0);
    return false;
  }
}
```

Output :

```
Passing test 2 [SampleTest]
failing test A [SampleTest]
Passing test 1 [SampleTest]
```