

# Java ME & NetBeans Mobility



*Petr Suchomel*  
Architect, NetBeans Mobility  
Sun Microsystems

# Agenda

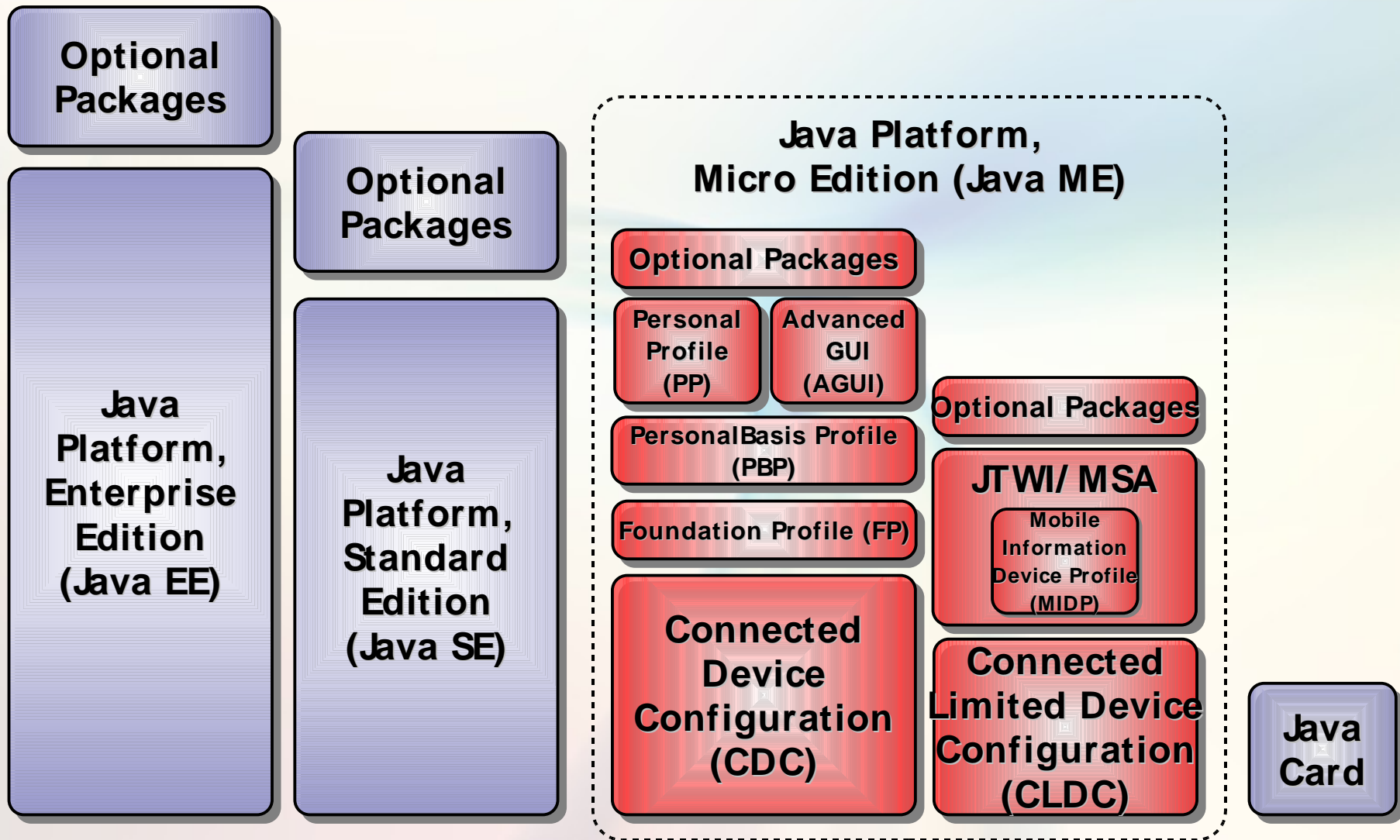
---

- Java ME introduction
- Java ME applications
- NetBeans Mobility Edition
- Power of advanced features
- Demos, demos, demos
- Q/A

# Java ME Brief Introduction

- A Java platform for consumer and embedded devices
- Defines configurations, profiles and optional packages
- Mobile phones usually use CLDC configuration and MIDP profile
- Devices using CDC configuration are coming to the market – Phones, Printers, IPTV, Blu-ray players

# Java ME Brief Introduction



# Java ME Applications

---

- Application Lifecycle
  - > CLDC/MIDP – MIDlet
  - > CDC – Xlet, Main or Applet
- MIDlet
  - > Managed by AMS, similar to Applet but with some differences
- Xlet
  - > Similar to MIDlet, allows communication between Xlets
- Applet and Main are same models as Java SE application model

# Differencias & Limitations

- Using libraries
  - > CLDC/MIDP – can to use user libraries, all libraries should be merged before preverification
  - > CDC – usage of user libraries is defined by underlying platform
- Build process
  - > CLDC platform requires process called preverification (stack map verification) during build
- Limitations
  - > CLDC does not have reflection, custom classloaders

# Tools - NetBeans Mobility

- Until NetBeans 6.0 add on – Mobility Pack
- Starting NetBeans 6.0 included in NetBeans Mobility Edition or Full Edition with support for both CLDC and CDC devices
- Includes Sun Java Wireless Toolkit
- Rich set of examples
- Over 1.000.000 cumulative downloads
- Open source project - <http://mobility.netbeans.org>
  - > Platform for Java ME tools

# Tools - NetBeans Mobility

- Ant based build system
  - > With specific extensions for Java ME/CLDC and CDC
  - > Project can be built outside IDE
  - > Extensible – user defined targets
- Bundled Sun Java Wireless Toolkit 2.5.2
- Support for 3-rd party emulators
  - > Automatic detection
  - > Nokia, Sony Ericsson, Motorola, Siemens and others
  - > Special / custom platforms can be added as well
- Advanced testing support



# Customer Pain Points

- Increasing Developer Productivity
  - > Decreasing Complexity
  - > Enabling Interoperability
  - > 44% of mobile devs target 11+ devices for each app
  - > Maintaining Costs
- Using the tools
  - > NetBeans Mobility Pack
- By further standardization
  - > Java Technology for Wireless Industry (JTWI, JSR 185)
  - > Mobile Service Architecture (MSA, JSR 248, 249)

# Visual Designer for MIDP

- New, powerful Visual Designer for Java ME / MIDP
- Flow editor
  - > Design your application flow using a visual tool
- Screen editor
  - > Visually edit the individual screens of the application
- Source editor
  - > Add/Edit application business logic
- Custom components
  - > Splash screen, Wait screen, Table item, SVG support

# Drag & Drop w/ Visual Mobile Designer

The image displays the NetBeans Visual Mobile Designer interface. On the left is a mobile device emulator showing a Java ME application with the text "Hello Hello, World!" and a choice group containing a "Choice Element". The bottom of the screen has an "Exit" button.

The central area is a visual programming diagram with the following components and connections:

- Mobile Device** (MIDlet): Contains "Started" and "Resumed" events.
- form1** (Form): Contains an "exitCommand1" command.
- list** (List): Contains "Form", "Alert", and "Exit" elements.
- alert** (Alert): A yellow warning icon.
- method** (Method): Contains a "Condition Result" with "True" and "False" options.

Connections: "Started" and "Resumed" events are connected to the "list" component. The "list" component is connected to the "form1" component. The "form1" component is connected to the "alert" component. The "method" component is connected to the "alert" component.

On the right is the **Palette** with the following categories:

- Displayables**: Alert, Form, List, Text Box, Login Screen, Splash Screen, Wait Screen, File Browser, PIM Browser, SMS Composer.
- Commands**: Back Command, Cancel Command, Exit Command.

At the bottom right, a preview of the **form1** component shows a list of items: "stringItem", "Test", "stringItem1", and "<null>".

# Application Porting Features

- Helps with porting of applications on real devices
  - > Devices exist in many variations – screen size, multimedia with different codecs support
- Application configuration
  - > Every configuration represents one set of files
  - > Settings can be modified for each configuration
- Commenting preprocessor
  - > Comments out non-active blocks
  - > Integrated with editor, highlighting and code completion
- Build selected or all configurations in one step

# Application Porting

The screenshot displays the NetBeans IDE interface for a project named "MobileApplication7". The left sidebar shows a project tree with folders for "build", "dist", "K800", "Nokia", "NokiaS40", "NokiaS60", "nbproject", "resources", "128x128", and "src". The "NokiaS40" folder is expanded, showing "MobileApplication7.jar" and "MobileApplication7.jad".

The main workspace is divided into several panes:

- Project Configuration:** A dropdown menu is open, showing "NokiaS40" as the selected configuration. Other options include "DefaultConfiguration" and "Add Configuration...".
- Category:** A tree view showing various configuration categories such as "General", "Platform", "Abilities", "Application Descriptor", "Attributes", "MIDlets", "Push Registry", "API Permissions", "Build", "Sources Filtering", "Compiling", and "Libraries & Resource".
- Bundled Libraries and Resources:** A list of libraries and resources, including "C:\projects\MobileApplication7\resources\128x128" and "NetBeans MIDP Components".
- Add Configuration Dialog:** A modal dialog box is open, titled "Add Configuration". It has a "Use Configuration Template" dropdown set to "Sony\_Ericsson\_K800i\_template" and a "New Configuration Name" text field containing "K800".
- Source Editor:** The main code editor shows Java code with conditional compilation directives for different devices:

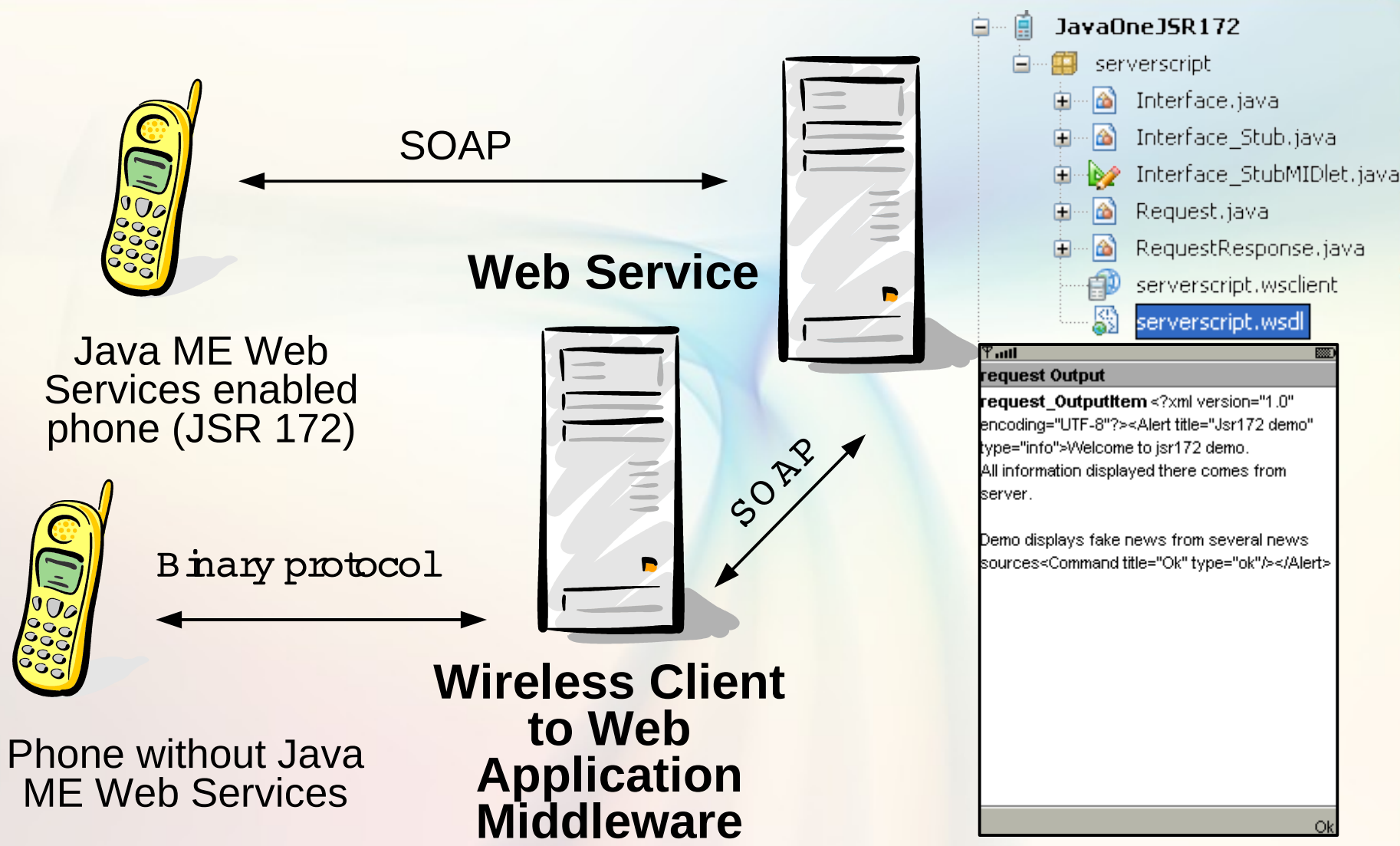
```
StringItem("Hello", "Hello, World!");
code here
#ifdef Nokia
    helloStringItem = new StringItem("Hello", "Hello, Nokia!");
#elifdef K800
    helloStringItem = new StringItem("Hello", "Hello, K800!");
#endif
```

# Advanced Features

---

- Client-Server application development
  - > Wireless Connection Bridge (web svcs, EJBs, etc.)
  - > Web Services Client Generator (JSR 172)
- MIDP localization support
- OTA testing
  - > Simulating real mobile devices
- Distributed with ProGuard for code obfuscation and optimization
- Deployment support
- Application signing

# Client-Server Connection Solutions



# SVG-T in NetBeans Mobility Edition

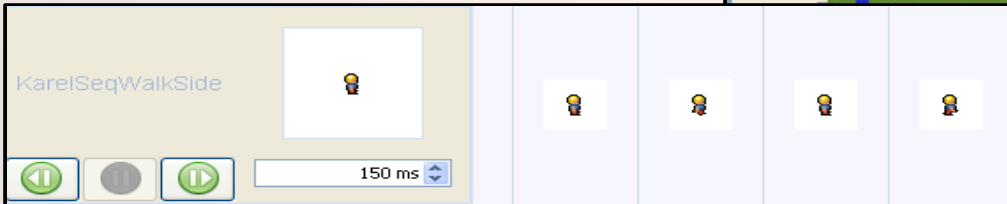
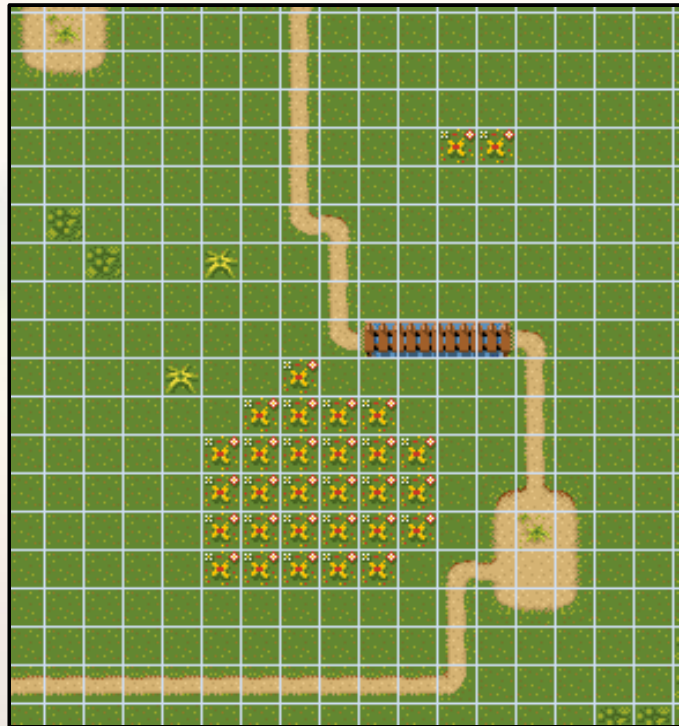
- Support for SVG-T files and development
  - > Scalable Vector Graphics Tiny 1.1 (JSR-226)
- SVG-T composer, viewer and navigator
  - > Explore SVG-T content
  - > Run animations
  - > Basic edition of SVG-T files
- Extended Visual Designer
  - > Use new SVG components to create rich application UI
  - > SVG Menu, SVG Splash Screen, SVG Wait Screen, SVG Image, SVG Animator



# Game Designer Highlights

- Visual Drag'n'Drop application designs
  - > Uses MIDP 2.0 Game API's
- Build individual Tiled Layers and Sprites
  - > Import prepared graphics
- Manipulate overall Scenes layout
  - > By moving individual layers
- Support for Sprite timing
- Simplifies basic blocks building for game developers

# Game Designer



# Advanced Testing

---

- Testing is important, but complex issue in Java ME
- Many devices, carriers networks and brandings
- Initial testing can be done using emulator SDK's and JUnit
- Complex testing requires testing on real devices and networks
  - > High cost to maintain devices for development
  - > Requires additional support infrastructure
  - > Can be out of scope for smaller development teams

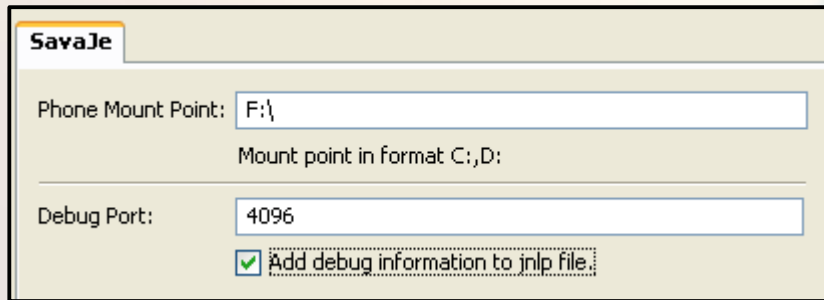
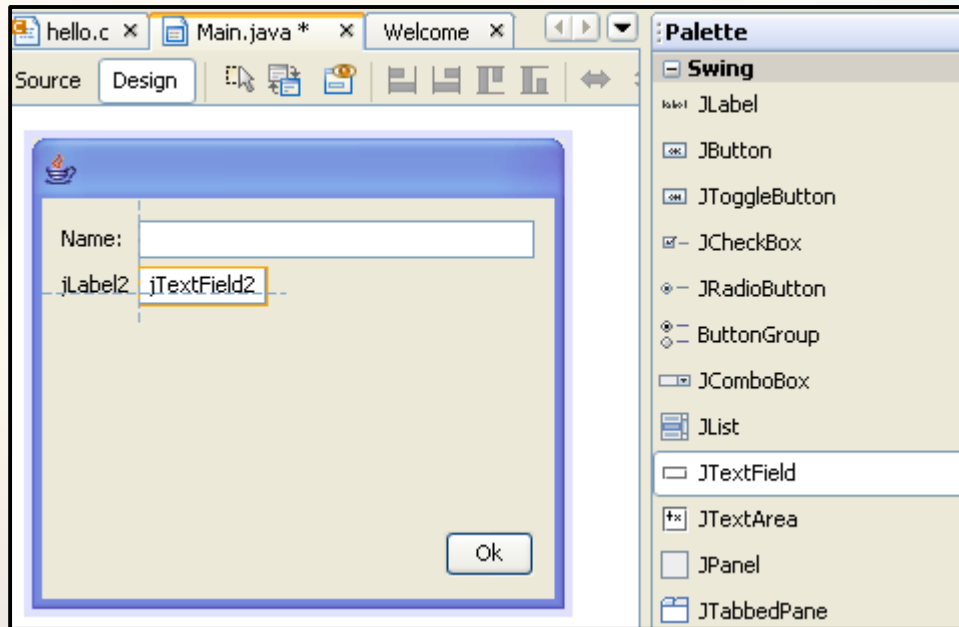
# Testing using DeviceAnywhere

- Unique service allows testing on remote devices
- No need to buy devices for in house testing
- Access to pre-release handsets
- Developer is testing real device in real network
- Advantage for developer who are building apps for third country careers
- Build in support for deployment in NetBeans Mobility Edition

# Support for CDC Configuration

- Starting NetBeans 6.0, CLDC and CDC support is included within one Mobility Edition
- Solution for Java ME CDC application development
- Matisse Visual Designer available for GUI development (Advanced GUI - JSR 209 Swing subset, Personal Profile)
- Supports Sony Ericsson CDC Platform 1, Nokia S80, SavaJe devices, RICOH SDK/J printers, NSIcom CrEme VM support (Windows CE and Windows Mobile Platforms)

# Using Matisse on CDC platform



# Summary

---

- NetBeans Mobility Edition is a complete solution for mobile application development and deployment
- Supports broad set of emulators and devices
- Features for both power users and beginners
- Visual manipulation for both CLDC/MIDP UI and Game API's
- Advanced testing features
- Questions, Requirements, Ideas? - send comments to [users@mobility.netbeans.org](mailto:users@mobility.netbeans.org)

# Questions & Answers

---





# Java ME & NetBeans Mobility



*Petr Suchomel*  
Architect, NetBeans Mobility  
Sun Microsystems