the POWER of JAVA™

"The Incredible Shrinking Application": Making Desktop Applications Mobile With CDC

Bartley Calder

Nandini Ramani

Sun Microsystems

TS-1053

Goal of This Talk

Learn how to migrate desktop Java^{TM} technology-based applications to mobile devices with the CDC stack

Agenda

Java SE Platform/JSR 209 API Differences

- AWT/Swing
- Java 2D^{TM} API/Image IO

Migrating Java SE Applications to JSR 209

- Xlet Design Philosophy
- Sun Tools and Environment
- Code Examples
- Demo

Lessons Learned

- Planning for Migration

Wrap Up

Agenda

Java SE Platform/JSR 209 API Differences

- AWT/Swing
- Java 2D^{TM} API/Image IO

Migrating Java SE Applications to JSR 209

- Xlet Design Philosophy
- Sun Tools and Environment
- Code Examples
- Demo

Lessons Learned

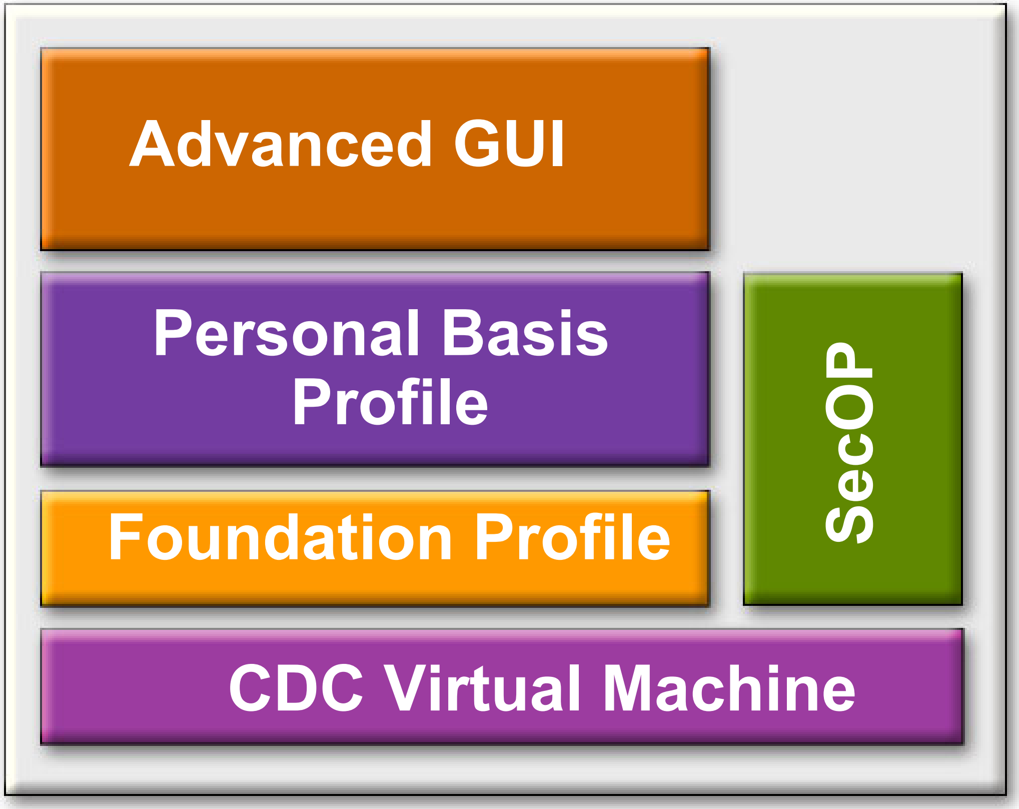
- Planning for Migration

Wrap Up

Platform Background

- Today's discussion is based on JSR 209 and supporting components
  - Connected Device Configuration 1.1 (JSR 219)
  - Foundation Profile 1.1 (JSR 218)
  - Personal Basis Profile (JSR 217)
  - Advanced Graphics and User Interface (JSR 209)
- These specification are proper subsets of Java Standard Edition (Java SE)

CDC STACK


Advanced GUI
Personal Basis Profile
Foundation Profile
SecOP
CDC Virtual Machine

AWT API Differences

Java SE Platform

- Multiple Windows
- Applets/main()
- Lightweight and Heavyweight AWT Components

JSR 209/PBP

- Single Window
- Xlets/main()
- Lightweight AWT Components

Swing API Differences

Java SE Platform

- Widget set suited for desktop
- JFrame
- JToolBar
- JWindow

JSR 209

- Widget set suitable for small devices
- No JFrame
- No JToolBar
- No JWindow
- No FileChooser
- No ColorChooser

Java 2D Differences

Java SE Platform

- All BufferedImage types
- Allows user created Rasters
- Doubles, Floats, Ints
- Porter-Duff compositing rules (All of them!)

JSR 209

- One BufferedImage type
- Possible platform dependent types
- No user created Rasters
- Platform dependent types may have custom DataBuffers (not Java language arrays)
- No double types
- AlphaComposite (SRC, SRC_OVER, CLEAR)
- Subsetted LookupOp
- All RenderedImages are BufferedImages

Image IO Differences

Java SE Platform

- Support for SPI
- Support for Metadata
- GIF writer in version 5
- Supports Tiling

JSR 209

- No SPI support; rely on factory method
- Imageio.getImageReader()
- No Metadata support
- No GIF writer
- No Tiling

Agenda

Java SE Platform/JSR 209 API Differences

- AWT/Swing
- Java 2D^{TM} API/Image IO

Migrating Java SE Applications to JSR 209

- Xlet Design Philosophy
- Sun Tools and Environment
- Code Examples
- Demo

Lessons Learned

- Planning for Migration

Wrap Up

Xlet Design Philosophy

- Xlet execution occurs in a managed environment
  - Usually downloaded
  - Security permissions regulate access to full system
  - Display capabilities controlled through root container
- Two-way communication with Xlet manager
  - Manager communicates to Xlet via Xlet interface
  - Xlet communicates Manager via XletContext
- XletContext also provides environment information to Xlet

main()/Xlet Differences

main()

- Environment vars via args[]
- Multiple presentation strategies
- System gives no indication of intended app state

Xlet

- Environment vars via XletContext
- Root container provided
- Tightly defined application lifecycle

Xlet API

```
public class MyXlet implements Xlet {
    public MyXlet() {    }

    /* Initialization here, not in constructor, create UI */
    public void initXlet(XletContext context)
        throws XletStateChangeException { }

    /* Xlet will be started here, display UI*/
    public void startXlet() throws XletStateChangeException{}

    /* pause the Xlet */
    public void pauseXlet() { }

    /* Clean up*/
    public void destroyXlet(boolean unconditional)throws
        XletStateChangeException { }
}
```

XletContext

```
public class XletContext {

 public static final String ARGS; // used to get initial
                                  // arguments of an Xlet

 public ClassLoader getClassLoader(); // base class loader

 public Container getContainer(); // Xlet root container

 public Object getXletProperty(String key); // property access

 public void notifyDestroyed(); // moving to Destroyed state

 public void notifyPaused(); // moving to the Paused state

 public void resumeRequest(); // request to become active
}
```

Things to Remember about Xlets

- Initialize data and build UI in **`initXlet()`**
  - **`XletContext.getContainer`** returns root container
  - Construct your UI and **`add()`** to the root container
  - Wait for **`startXlet()`** to make your UI visible
- When **`startXlet()`** is called ‘your on!’
  - Start background threads
  - Expect user input
- If **`pauseXlet()`** is called ‘get small’
  - Stop threads
  - Release memory if possible

Sun Tools

Sun Java Wireless Toolkit or CDC

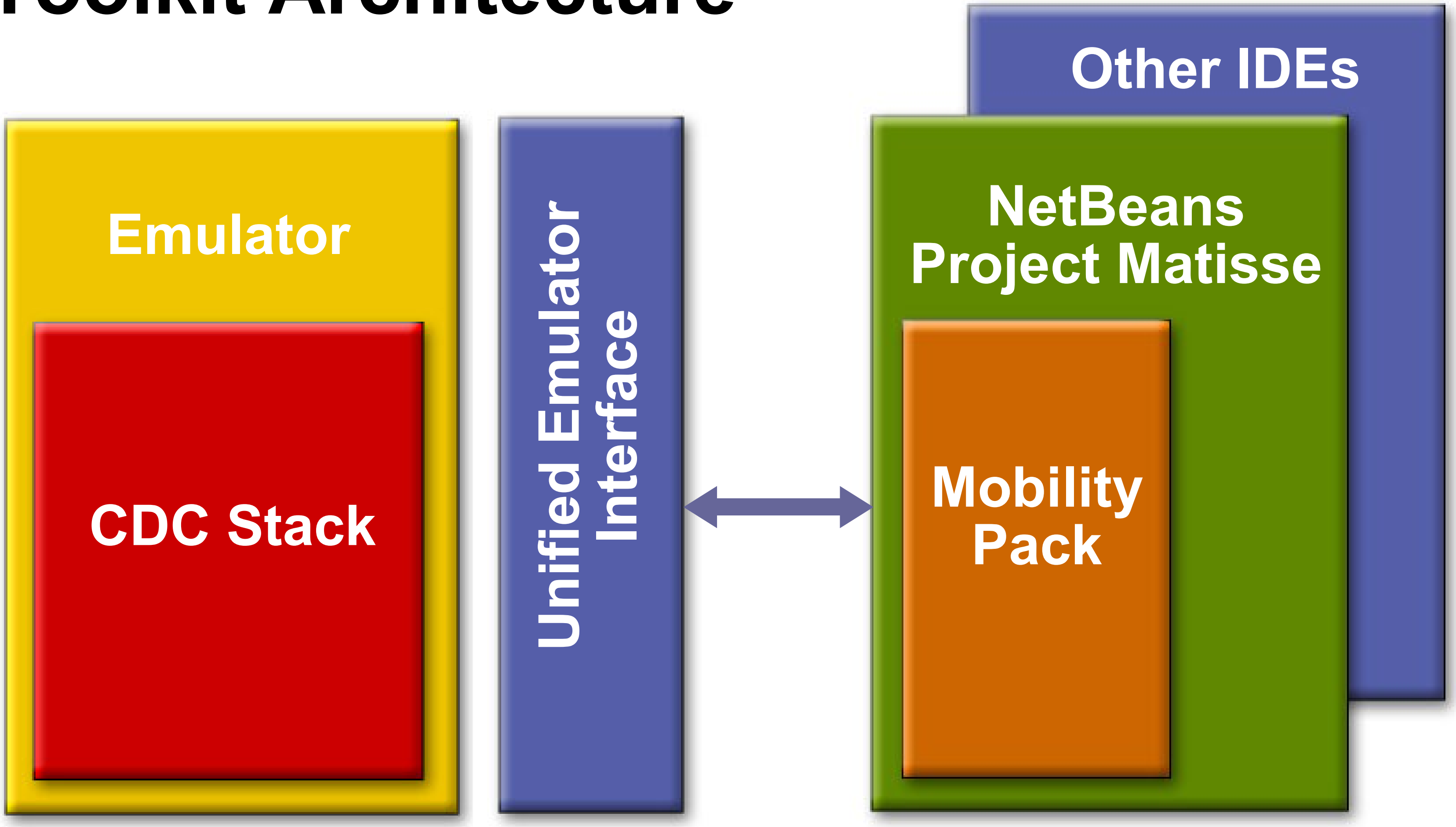
- From the team that brought you the Java 2 Micro Edition (J2ME™) Wireless Toolkit
- Development tools for new generation of the CDC platform
- Device emulation environment
- Same implementation of stack as on device
- Integrable with NetBeans™ IDE

Sun Tools (Cont.)

NetBeans 5.5 IDE Beta

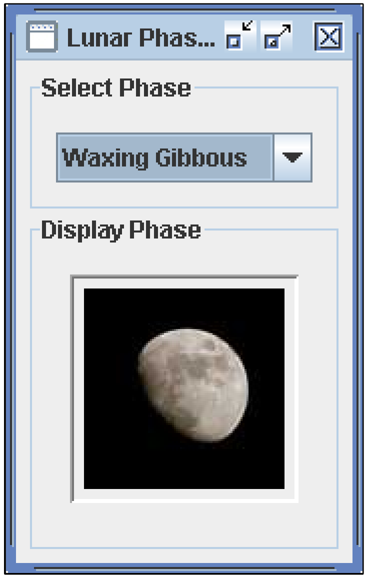
- Seamless integration with IDE
  - CDC is another project type
  - Build, run, debug from IDE, using emulator
- Visual editor for user interfaces
  - Drag-and-drop tool for building UI

Toolkit Architecture



Sample Application

- Lunar Phases demo
- From Swing tutorial
- Single Frame App
- Reads images from files
- Popup menu allows phase selection



Waxing Gibbous

Porting Preparation

- Identify non-UI portions of the application
  - These should port easily from Java SE platform
- Recognize the constraints of the Xlet environment
  - Your application will not create it's root container
  - The screen is likely to be small
  - Your application may not receive pointer input
    - Don't Depend On It
  - You should set the initial focus owner

Our Porting Steps

- Create the UI in NetBeans IDE using NetBeans GUI Builder (formerly code-named Mattise)
  - Create Xlet (CDC Application)
  - Create JPanel Form
- UI supporting code from SE Application to Xlet
  - Event handling code, image reading, etc.
- Link JPanel from Mattise to Xlet's container
  - Connect UI components from JPanel to event handlers
  - Make sure they are visible (public or package private)
  - Alternative: Use builder to call an event handler in Xlet

Xlet.initXlet() From Our Example

```
public void initXlet(XletContext context)
  throws XletStateChangeException {

      this.context = context;
      try {                         // get the root
          root_container = context.getContainer();
      } catch (Exception e) {
          System.err.println("can't get root container");
          System.exit(1);
      }
      my_panel = new LunarPhasesPanel();
      phaseIconLabel = my_panel.phaseIconLabel;

      this.addWidgets();
      root_container.add(my_panel); // set panel to root

  }
```

DEMO

Agenda

Java SE Platform/JSR 209 API Differences

- AWT/Swing
- Java 2D^{TM} API/Image IO

Migrating Java SE Applications to JSR 209

- Xlet Design Philosophy
- Sun Tools and Environment
- Code Examples
- Demo

Lessons Learned

- Planning for Migration

Wrap Up

Lessons Learned (JSR 209→Java SE)

- All applications written to JSR 209 APIs will run on Java SE platform (caveats)
    - Xlets
    - Device keys
    - Soft buttons
    - Traversal and Initial focus
- Usability may be effected
    - Larger screen size
    - Input device considerations
    - Better graphics

Lessons Learned (Java SE→JSR 209)

Java SE Platform to JSR 209

- Single Frame restriction
- Input device constraints
  - Soft buttons
  - Device keys
  - Traversal and Initial focus
- Reduced widget set
- Different user expectation
- Screen resolution

Agenda

Java SE Platform/JSR 209 API Differences

- AWT/Swing
- Java 2D^{TM} API/Image IO

Migrating Java SE Applications to JSR 209

- Xlet Design Philosophy
- Sun Tools and Environment
- Code Examples
- Demo

Lessons Learned

- Planning for Migration

Wrap Up

Summary

- CDC stack provides a Java SE compatible platform for mobile applications
- Migration is simplified with the use of Sun's tool chain
- Some planning may be required
- Be mindful that desktop and consumer devices are different
- CDC provides a rich graphical UI environment

For More Information

Visit

- http://jcp.org
- The specification is available at http://jcp.org/en/jsr/detail?id=209
- http://www.netbeans.org
- http://java.sun.com/products/cdc/
- http://java.sun.com/docs/books/tutorial/uiswing/learn/example5.html

Q&A

the POWER of JAVA™

“The Incredible Shrinking Application”: Making Desktop Applications Mobile With CDC

Bartley Calder

Nandini Ramani

TS-1053