How to Port phoneME™ Advanced Software to Google Android, iPhone, OpenMoko, LiMO, and More

Hinkmond Wong, Senior Staff Engineer, Sun Microsystems, Inc.

TS-6304

Java

Sun microsystems

- Learn how to port an open source Java™ Platform, Micro Edition (Java ME) implementation to new mobile platforms such as Google Android, iPhone, OpenMoko, LiMO, and more.

GOAL

Agenda

- **Introduction to phoneME™ software**
- Building phoneME Advanced software
- Porting Layers of phoneME Advanced software
- Porting Core Layer (threads, IO, and networking)
- Porting Graphics Layers (Personal Basis and Personal Profiles)
- Example Cases
- Sample Code
- Testing Port
- Summary

Agenda

- Introduction to phoneME software
- **Building phoneME Advanced software**
- Porting Layers of phoneME Advanced software
- Porting Core Layer (threads, IO, and networking)
- Porting Graphics Layers (Personal Basis and Personal Profiles)
- Example Cases
- Sample Code
- Testing Port
- Summary

Introduction to phoneME Software

- Project phoneME software is the open source version of Java ME technology
- Subversion repository on java.net
  - https://phoneme.dev.java.net/source/browse/phoneme/
- Supported by email lists, forums, and wiki pages
- Two stack:
  - phoneME Advance software: Java ME CDC based technology
  - phoneME Feature software: Java ME CLDC/MIDP based technology
- Dual Licensed
  - GPL version 2
  - Sun Commercial License

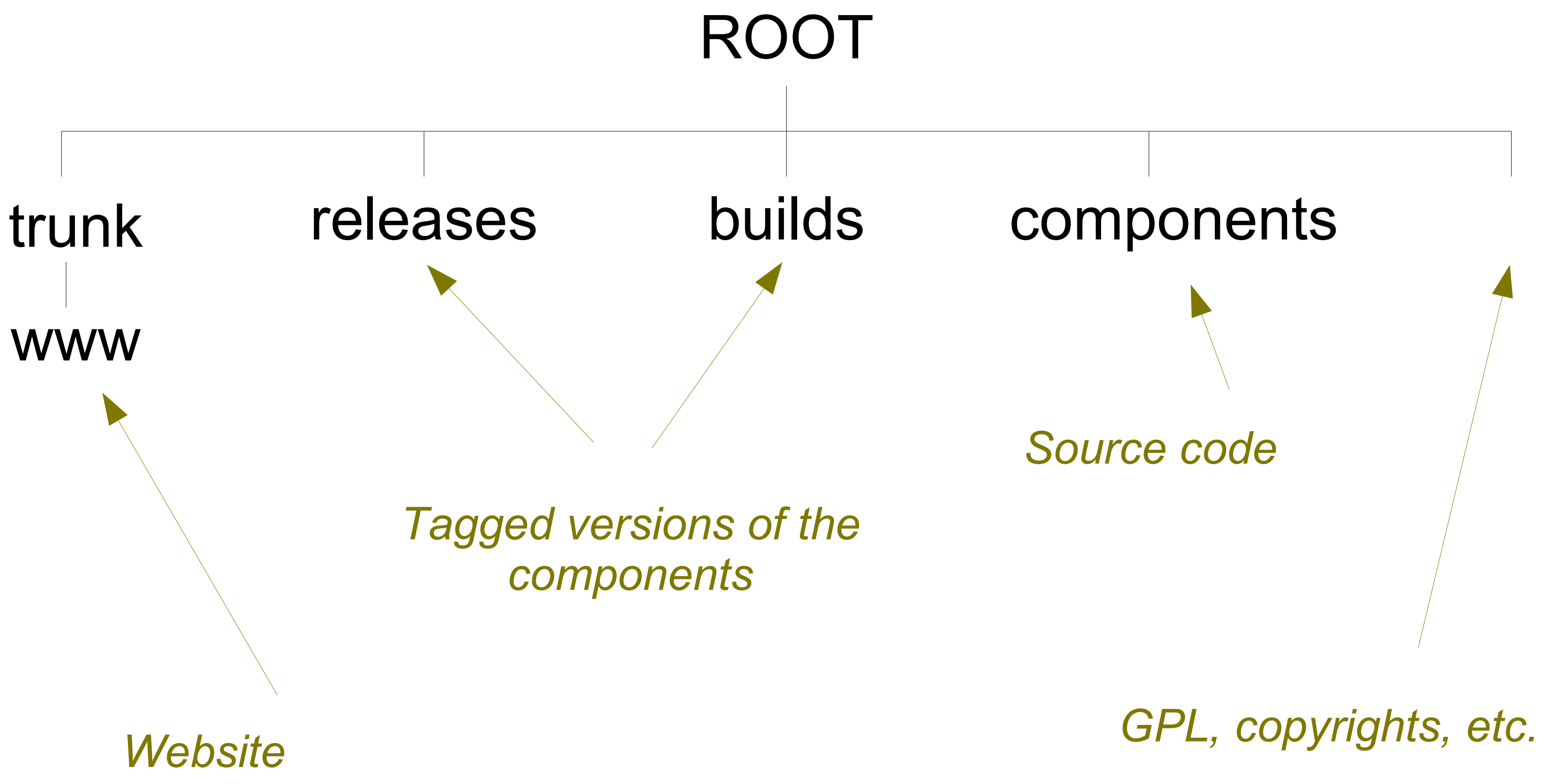
Agenda

- **Introduction to phoneME software**
- **Building phoneME Advanced software**
- Porting Layers of phoneME Advanced software
- Porting Core Layer (threads, IO, and networking)
- Porting Graphics Layers (Personal Basis and Personal Profiles)
- Example Cases
- Sample Code
- Testing Port
- Summary

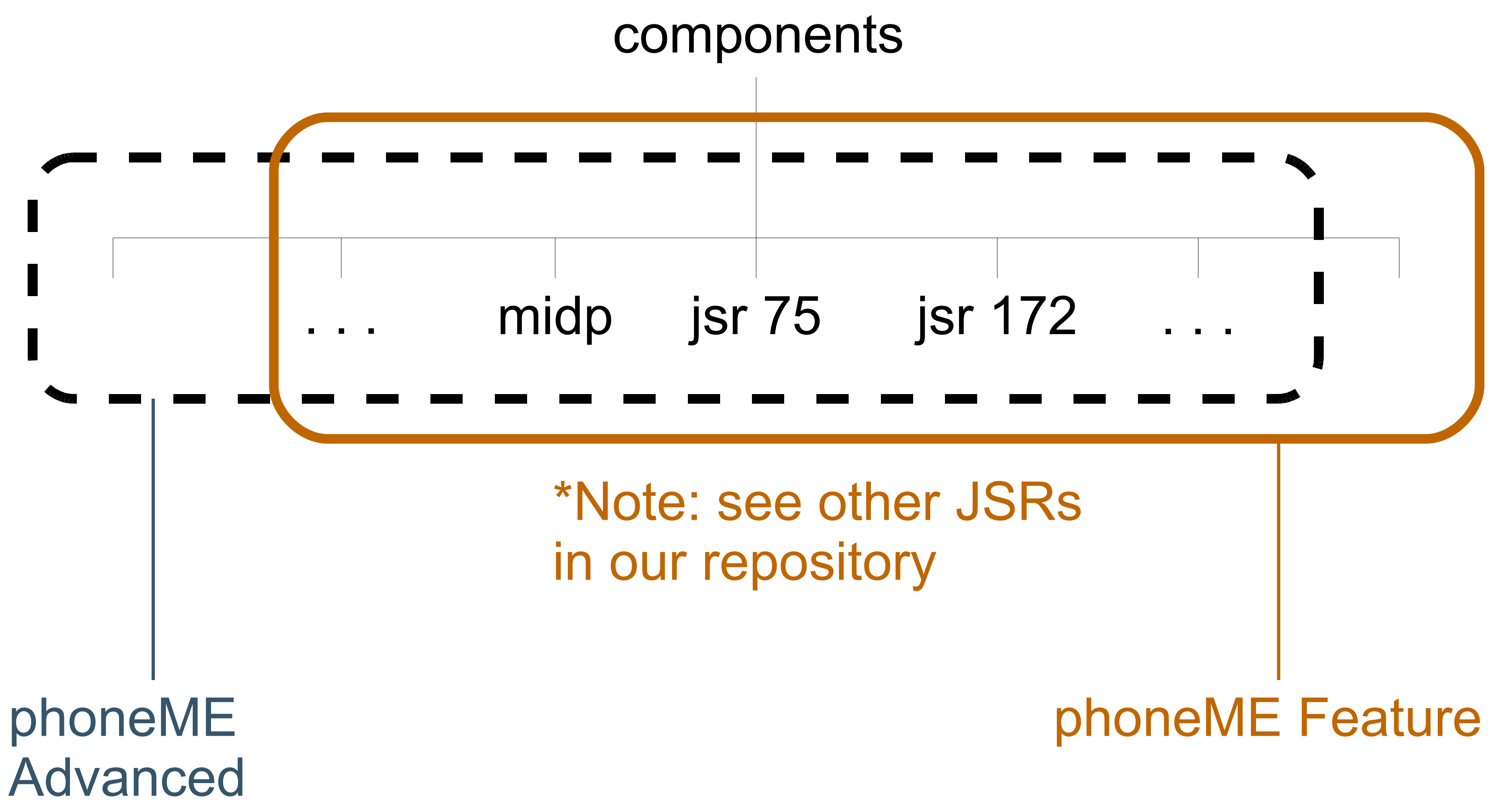
Building phoneME Advanced Software

- Source code at java.net
- Code bundles
  - https://phoneme.dev.java.net/downloads_page.html
- Direct SVN repository access
  - https://phoneme.dev.java.net/svn/phoneme
- Trunks exist for each component
  - Warning: many branches exist for each component

Building phoneME Advanced Software



Building phoneME Advanced Software



Building phoneME Advanced Software

- Download
  - Using "svn co" command
  - Checkout the needed components
- Build (Tools Required):
  - GNU make (version 3.81)
  - gcc (version 3.x or 4.x)
  - ant (version 1.6.5)
- Getting Started Guide:
  - https://phoneme.dev.java.net/content/phoneme_advanced_r2.html

Building phoneME Advanced Software

```
svn co
https://phoneme.dev.java.net/svn/phoneme/components/cdc/tr
unk cdc

svn co
https://phoneme.dev.java.net/svn/phoneme/components/tools/
trunk tools

cd cdc/build/linux-x86-generic

make J2ME_CLASSLIB=foundation
```

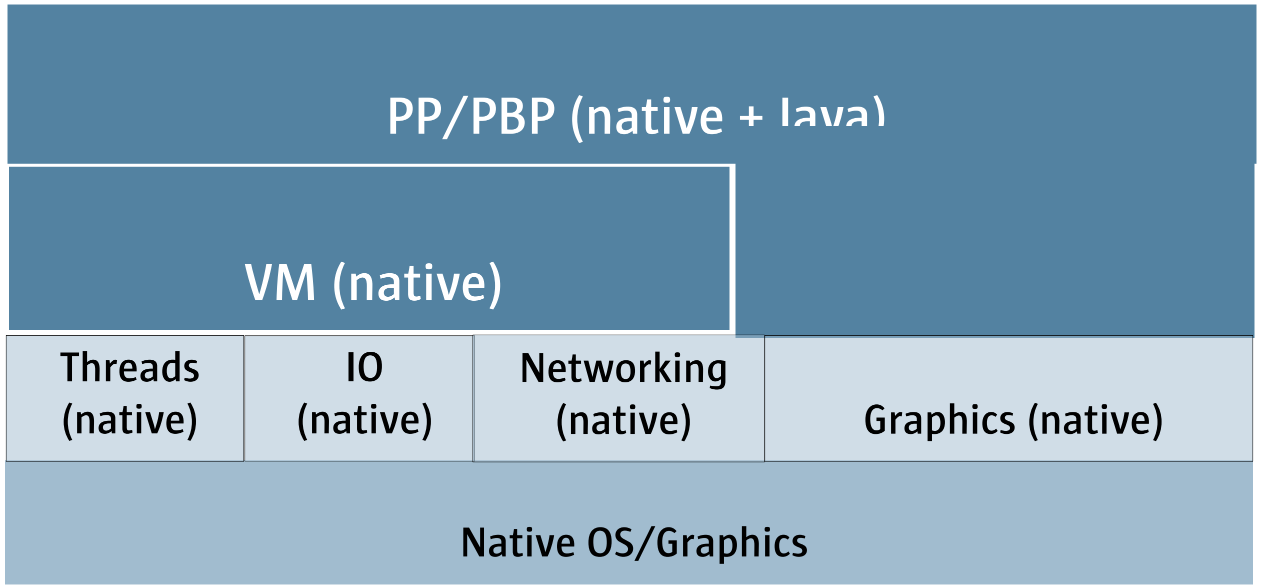
Agenda

- **Introduction to phoneME software**
- **Building phoneME Advanced software**
- Porting Layers of phoneME Advanced software
- Porting Core Layer (threads, IO, and networking)
- Porting Graphics Layers (Personal Basis and Personal Profiles)
- Example Cases
- Sample Code
- Testing Port
- Summary

Porting Layers of phoneME Advanced Software

- Core Layer
  - Threads
  - IO
  - Networking
- Graphics Layer
  - Personal Profile: Proper Subset of Abstract Windowing Toolkit (AWT) from Java Platform, Standard Edition, (Java SE) version 1.4.2
  - Personal Basis Profile: Proper Subset of drawing primitives from Java SE version 1.4.2 platform

Porting Layers of phoneME Advanced Software



Agenda

- **Introduction to phoneME software**
- **Building phoneME Advanced software**
- Porting Layers of phoneME Advanced software
- Porting Core Layer (threads, IO, and networking)
- Porting Graphics Layers (Personal Basis and Personal Profiles)
- Example Cases
- Sample Code
- Testing Port
- Summary

Porting Core Layer

- *Threads*
  - *POSIX standard*
  - *Follow existing port*
  - *Examples:*
    - *cdc/src/linux/javavm/runtime/threads_md.c*
    - *cdc/src/solaris/javavm/runtime/threads_md.c*
    - *cdc/src/darwin/javavm/runtime/threads_md.c*
    - *cdc/src/vxworks/javavm/runtime/threads_md.c*
    - *cdc/src/win32/javavm/runtime/threads_md.c*
- *Map to native POSIX threads calls in machine dependent (*_md.c) file*

Porting Core Layer

- *IO*
  - *Common standard for open, seek, read, close, etc. among popular OS's*
  - *Follow existing port*
  - *Examples:*
    - *cdc/src/linux/javavm/runtime/io_md.c*
    - *cdc/src/solaris/javavm/runtime/io_md.c*
    - *cdc/src/darwin/javavm/runtime/io_md.c*
    - *cdc/src/vxworks/javavm/runtime/io_md.c*
    - *cdc/src/win32/javavm/runtime/io_md.c*
- *Map to native I/O calls in machine dependent (*_md.c) file*

Porting Core Layer

- *Networking*
  - *Common standard for connect, send, receive, timeout, etc. among popular OS's*
  - *Follow existing port*
  - *Examples:*
    - *cdc/src/linux/javavm/runtime/net_md.c*
    - *cdc/src/darwin/javavm/runtime/net_md.c*
    - *cdc/src/win32/javavm/runtime/net_md.c*
- *Map to native networking calls in machine dependent (*_md.c) file*

Porting Core Layer

- *Rest of Core Layer*
  - *Host Porting Interface (HPI)*
  - *See cdc/src/share/javavm/include/porting/*
  - `ansi/` `globals.h` `jni.h` `sync.h` `vm-defs.h`
  - `defs.h` `int.h` `linker.h` `system.h`
  - `doubleword.h` `io.h` `memory.h` `threads.h`
  - `endianness.h` `java_props.h` `net.h` `time.h`
  - `float.h` `jit/` `path.h` `timezone.h`
- *Use (*_md.c) files form an existing port (linux, darwin [MacOS], win32) as a guide*

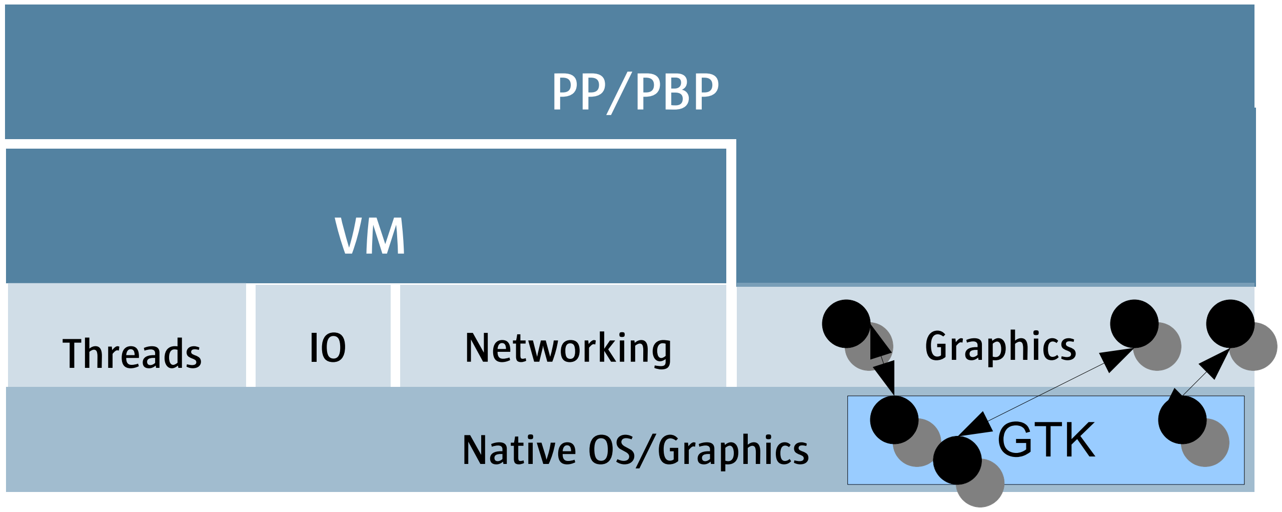
Agenda

- **Introduction to phoneME software**
- **Building phoneME Advanced software**
- Porting Layers of phoneME Advanced software
- Porting Core Layer (threads, IO, and networking)
- Porting Graphics Layers (Personal Basis and Personal Profiles)
- Example Cases
- Sample Code
- Testing Port
- Summary

Porting Graphics Layer

- *Personal Basis Profile*
  - *Basic Graphics Primitives*
  - *Ex. drawArc(), drawLine(), drawOval(), drawPolygon(), drawRect(), drawRoundRect(), fill*(), etc.*
- *Map Graphics components to native Toolkit*
  - *Toolkit, Component, Frame, Window, Graphics Environment, Fonts, Images*
- *Use existing Personal Basis Profile port as a guide*
  - *grep native cdc/src/share/basis/classes/awt/qt/java/awt/QtGraphics**
  - *See: pCopyArea(), pDrawArc(), pDrawLine(), pDrawOval(), pDrawPolygon(), pDrawRect(), pDrawRoundRect()*
  - *Map to native functions*
    - *Ex. in Qt, p.drawArc, p.drawEllipse, p.drawLine, etc.*

Porting Layers of phoneME Advanced Software

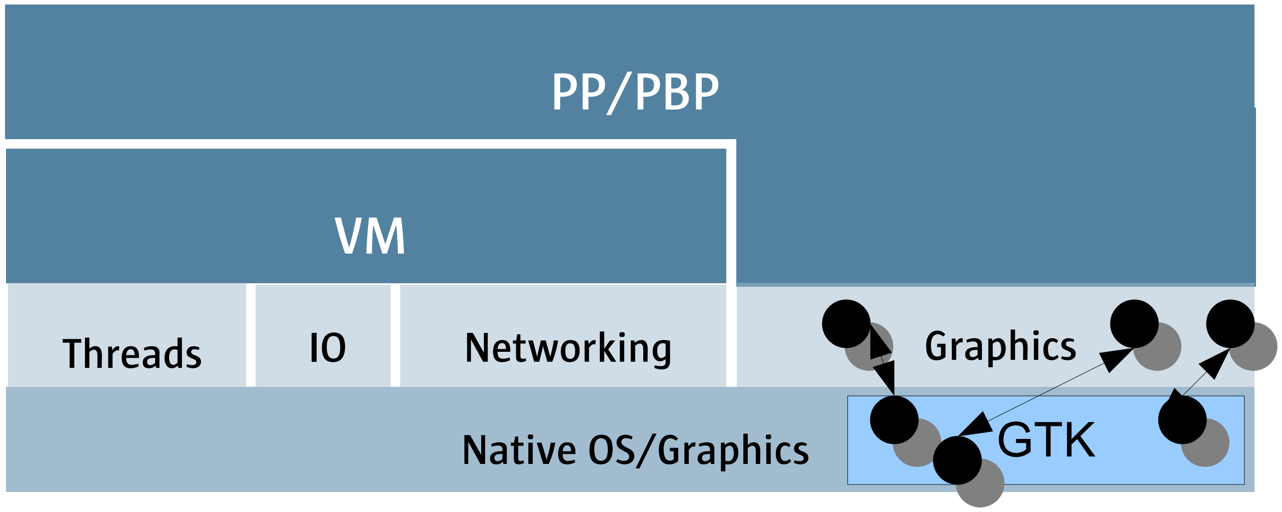


Basic Graphics Primitives

Porting Graphics Layer

- *Personal Profile*
  - *AWT Peer Components (map to native toolkit peers)*
  - *java.awt.*: Button, Dialog, Label, Scrollbar, TextArea, Window, Image, Font, etc.*
- *Map AWT widgets to native Toolkit widgets*
  - *Toolkit, Component, Frame, Window, Graphics Environment, Fonts, Images, Button, Dialog, Scrollbar, TextArea, etc.*
- *Use existing Personal Profile port as a guide*
  - *grep native cdc/src/share/personal/classes/awt/peer_based/sun/awt/qt/**
  - *See: pCopyArea(), pDrawArc(), pDrawLine(), pDrawOval(), pDrawPolygon(), pDrawRect(), pDrawRoundRect()*
  - *Map to native functions*
    - *Ex. in Qt, p.drawArc, p.drawEllipse, p.drawLine, etc.*

Porting Layers of phoneME Advanced Software



AWT Peer Components

Agenda

- **Introduction to phoneME software**
- **Building phoneME Advanced software**
- Porting Layers of phoneME Advanced software
- Porting Core Layer (threads, IO, and networking)
- Porting Graphics Layers (Personal Basis and Personal Profiles)
- Example Cases
- Sample Code
- Testing Port
- Summary

Example Cases: Google Android

- *Google Android*
  - *Start with Linux port of phoneME Advanced software*
  - *Approach as dual stack*
  - *Port phoneME Advanced software to native Linux OS and native GUI Toolkit that will exist on Android emulator host*
  - *Difficult part: Wrap Android Activity (application) to launch Java Virtual Machine (Ex. Compile Java SE wrapper application to launch Java Virtual Machine. Compile it to Dalvik bytecodes)*
- *Reference Android stack on a device*
  - *http://euedge.com/blog/2007/12/06/google-android-runs-on-sharp-zaurus-sl-c760/*
- *Reference running native App from Android emulator*
  - *http://groups.google.com/group/android-developers/browse_thread/thread/f31003bbed8bf7a9/*

Example Cases: Apple iPhone

- *iPhone*
  - *Start with darwin (MacOS) port of phoneME Advanced software (need to link with Objective-C)*
  - *Port phoneME Advanced software to iPhone SDK*
  - *Port core porting layer (threads, IO, and networking) to iPhone MacOS layer*
  - *Port graphics to Core Graphics (Quartz 2D) and UIKit for drawing primitives (Personal Basis Profile)*
- *Reference to iPhone SDK info*
  - *http://developer.apple.com/iPhone/library/navigation/index.html*
- *Reference to iPhone Graphics*
  - *http://developer.apple.com/iPhone/library/referencelibrary/GettingStarted/GS_Graphics_iPhone/index.html*

Example Cases: Other

- *OpenMoko, LiMO, and other Linux based devices*
  - *Start with linux/GTK port of phoneME Advanced software*
  - *Port phoneME Advanced software to specific device Linux distro and GTK*
  - *Port core porting layer (threads, IO, and networking) using existing Linux port*
  - *Port graphics to GTK GUI toolkit*
- *Reference to OpenMoko*
  - *http://wiki.openmoko.org/wiki/Main_Page*
- *Reference to LiMO*
  - *http://wiki.openmoko.org/wiki/OpenmokoFramework*

Agenda

- **Introduction to phoneME software**
- **Building phoneME Advanced software**
- Porting Layers of phoneME Advanced software
- Porting Core Layer (threads, IO, and networking)
- Porting Graphics Layers (Personal Basis and Personal Profiles)
- Example Cases
- Sample Code
- Testing Port
- Summary

Code Sample: IO

```
src/linux/native/java/io/UnixFileSystem_md.c
---
JNIEXPORT jlong JNICALL
Java_java_io_UnixFileSystem_getLastModifiedTime(JNIEnv
*env, jobject this,
                                                jobject
file)
{
    jlong rv = 0;

...
            struct stat sb;
            if (stat(path, &sb) == 0) {
                rv = 1000 * (jlong)sb.st_mtime;
```

Code Sample: Porting Networking

```
src/linux/native/java/net/Inet4AddressImpl_md.c
---
JNIEXPORT jobjectArray JNICALL
Java_java_net_Inet4AddressImpl_lookupAllHostAddr(JNIEnv
*env, jobject this,
                                                jstring
host) {
    const char *hostname;
    jobjectArray ret = 0;
    jclass byteArrayCls;
    struct hostent res, *hp = 0;
    char buf[HENT_BUF_SIZE];

  ...
    /* Try once, with our static buffer. */
#ifdef __GLIBC__
    gethostbyname_r(hostname, &res, buf, sizeof(buf), &hp,
&h_error);
```

Code Sample: Porting Basic Graphics

```
src/share/basis/native/awt/qt/QtImage.cpp
---
JNIEXPORT void JNICALL
Java_java_awt_QtImage_pDrawImage (JNIEnv * env, jclass
cls, jint qtGraphDescDest, jint qtImageDescSrc, jint x,
jint y, jobject bg)
{

...

        p.drawPixmap(x, y, pm);
```

Code Sample: Porting Graphics Widget

```
src/share/personal/native/awt/qt/QtButtonPeer.cc
---

JNIEXPORT void JNICALL
Java_sun_awt_qt_QtButtonPeer_setLabelNative (JNIEnv *env,
jobject thisObj,
jstring label)
{
...
  QString* labelString = awt_convertToQString(env, label);
   ((QpPushButton *)buttonPeer->getWidget())-
>setText(*labelString);
```

Agenda

- **Introduction to phoneME software**
- **Building phoneME Advanced software**
- Porting Layers of phoneME Advanced software
- Porting Core Layer (threads, IO, and networking)
- Porting Graphics Layers (Personal Basis and Personal Profiles)
- Example Cases
- Sample Code
- Testing Port
- Summary

Testing Your Port

- *HelloWorld*
  - *bin/cvm -cp testclasses.zip HelloWorld*
    - *Hello world.*
- *VM tests*
  - *bin/cvm -cp testclasses.zip Test*
    - **CONGRATULATIONS: test Test completed with 411 tests passed and 0 failures*
    - **Output lines starting with a * should be checked for correctness*
    - **They can be compared to src/share/javavm/test/TestExpectedResult*
- *PBP test*
  - *bin/cvm -cp democlasses.jar basis.DemoFrame*
- *PP test*
  - *bin/cvm -cp democlasses.jar personal.DemoFrame*

Demo: How to Port phoneME Advanced Software

Agenda

- **Introduction to phoneME software**
- **Building phoneME Advanced software**
- Porting Layers of phoneME Advanced software
- Porting Core Layer (threads, IO, and networking)
- Porting Graphics Layers (Personal Basis and Personal Profiles)
- Example Cases
- Sample Code
- Testing Port
- Summary

Summary

- Many new and interesting phone platforms exist
- The phoneME Advanced software project allows for open source development of a Java ME technology implementation
- Porting phoneME Advanced software to the new phone platforms can be handled in a methodical way
- Submit your port back to the Java ME Community and participate in our project

For More Information

- See:
  - http://community.java.net/mobileandembedded/
  - https://phoneme.dev.java.net/
  - http://wiki.java.net/bin/view/People/HinkmondWong (updated slides and downloads)

Questions and Answers

- Q & A

THANK YOU

Hinkmond Wong, Senior Staff Engineer, Sun Microsystems, Inc.

TS-6304