Tackling Java ME Device Fragmentation: Orange and Sun Collaboration

Martin Wrigley

Head of Partner's Technology, Orange Partners Operations

Orange/France Telecom

http://www.orangepartner.com/

Rhian Sugden

Account Manager, OEM Software Sales

Sun Microsystems, Inc.

http://java.sun.com/

Nir Vazana

Technical Leader, Engineering Services

Sun Microsystems, Inc.

http://java.sun.com/

TS-5051

Goal

Reduce number of versions for your mobile application. See how Sun and Orange think it should be done.

Agenda

Overview

Design Approach

Case Studies

The Sun and Orange Collaboration

Q&A

Agenda

Overview

- **Fragmentation—operator's view**
- **Problem parameters**
- **Possible solutions**

Design Approach

Case Studies

The Sun and Orange Collaboration

Q&A

Fragmentation—Operator's View

Problem dimensions

- Problem scope
  - Some numbers and stats
- Too many SKUs to manage
- Too many apps to test and sign
- Uneconomical to deploy
- Un-friendly for users: "Can I get it too?"

Fragmentation—Operator's View

Impact on operator

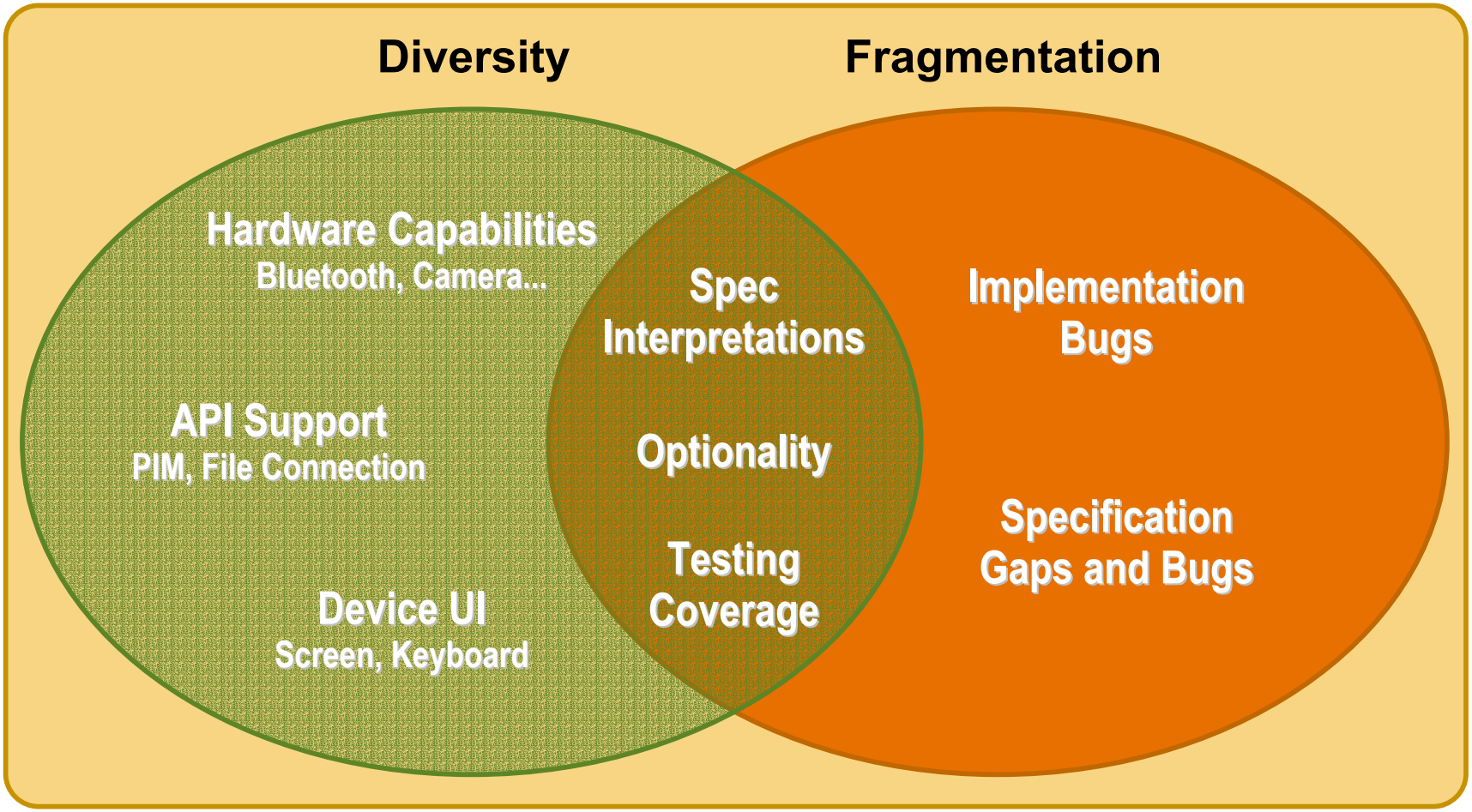
- Impractical to deploy to wide number of handsets
  - End up with just games for top 20
  - Artificial restriction of addressable market
  - Becomes uneconomical to address as a mass market
- Operators will lose interest!

Problem Parameters

A technical view at fragmentation

- Why are devices different?
  - Technological differences
  - Consumers are different
  - Price range
  - Implementation bugs
- When are differences “bad”?

Diversity vs. Fragmentation



A Technical View at Fragmentation

Main fragmentation areas

- Platform (MIDP version, CPU, language)
- Screen parameters (size, colour depth)
- Input methods (layout, touchscreen, keycodes)
- Memory (heap and persistent, RMS capabilities)
- Multimedia support (codecs, mixing)
- Connectivity (Bluetooth, IR, number of connections)

Source: http://developers.sun.com/techtopics/mobility/reference/techart/design_guidelines/overview.html

Possible Solutions

A technical view at fragmentation

- Mandate a single Java™ Platform, Micro Edition (Java ME platform) implementation for devices
- “Tighten” device specifications
  - Java Specification Request (JSR) 248 MSA
- Create “aim low” applications
- Fragment applications

- Consider a different design approach

Source: http://developers.sun.com/techtopics/mobility/reference/techart/design_guidelines/overview.html

Agenda

Overview

Design Approach

- **Traditional vs. suggested approach**
- **Picking the right solution**
- **Optimisation considerations**

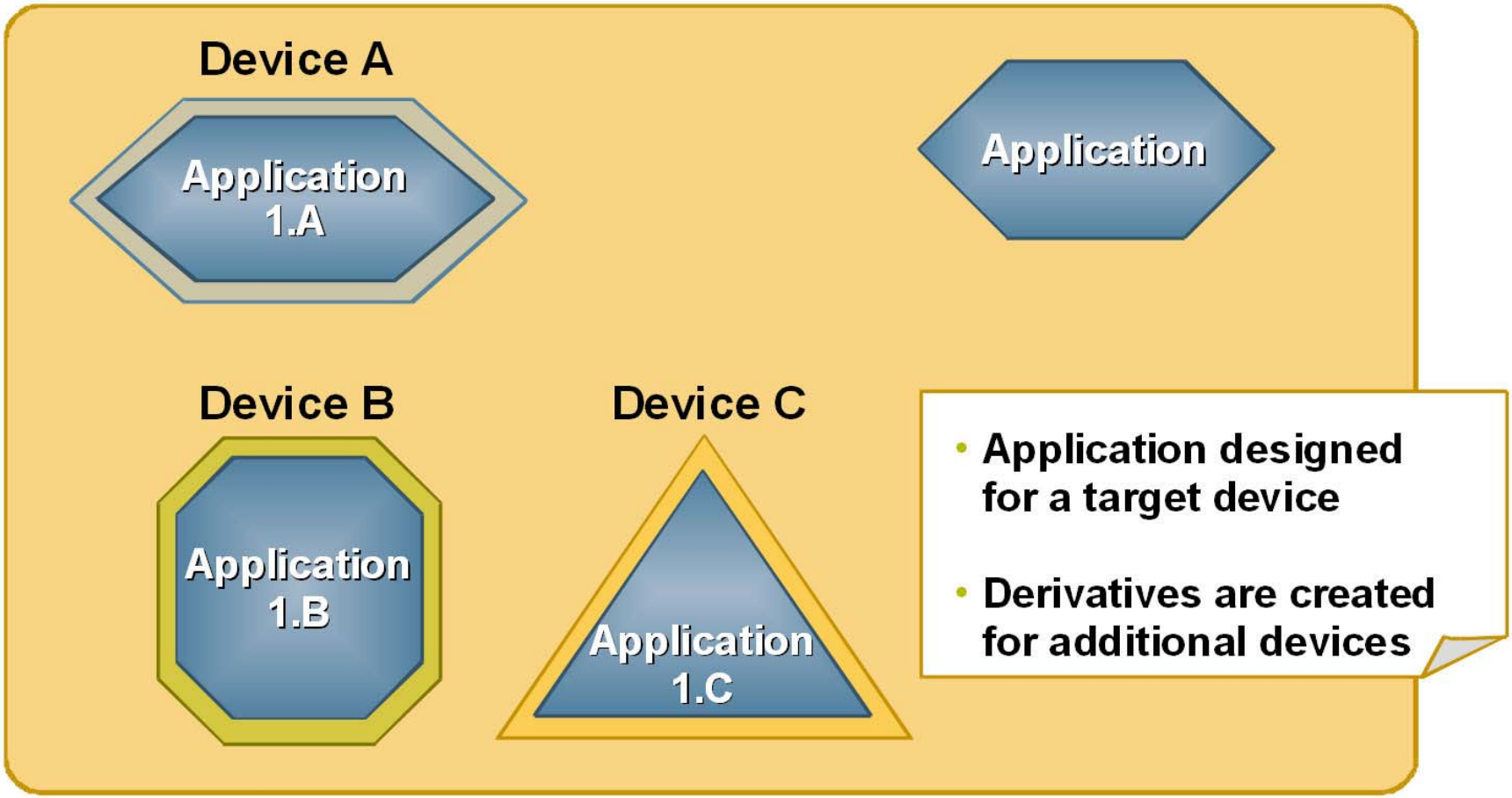
Case Studies

The Sun and Orange Collaboration

Q&A

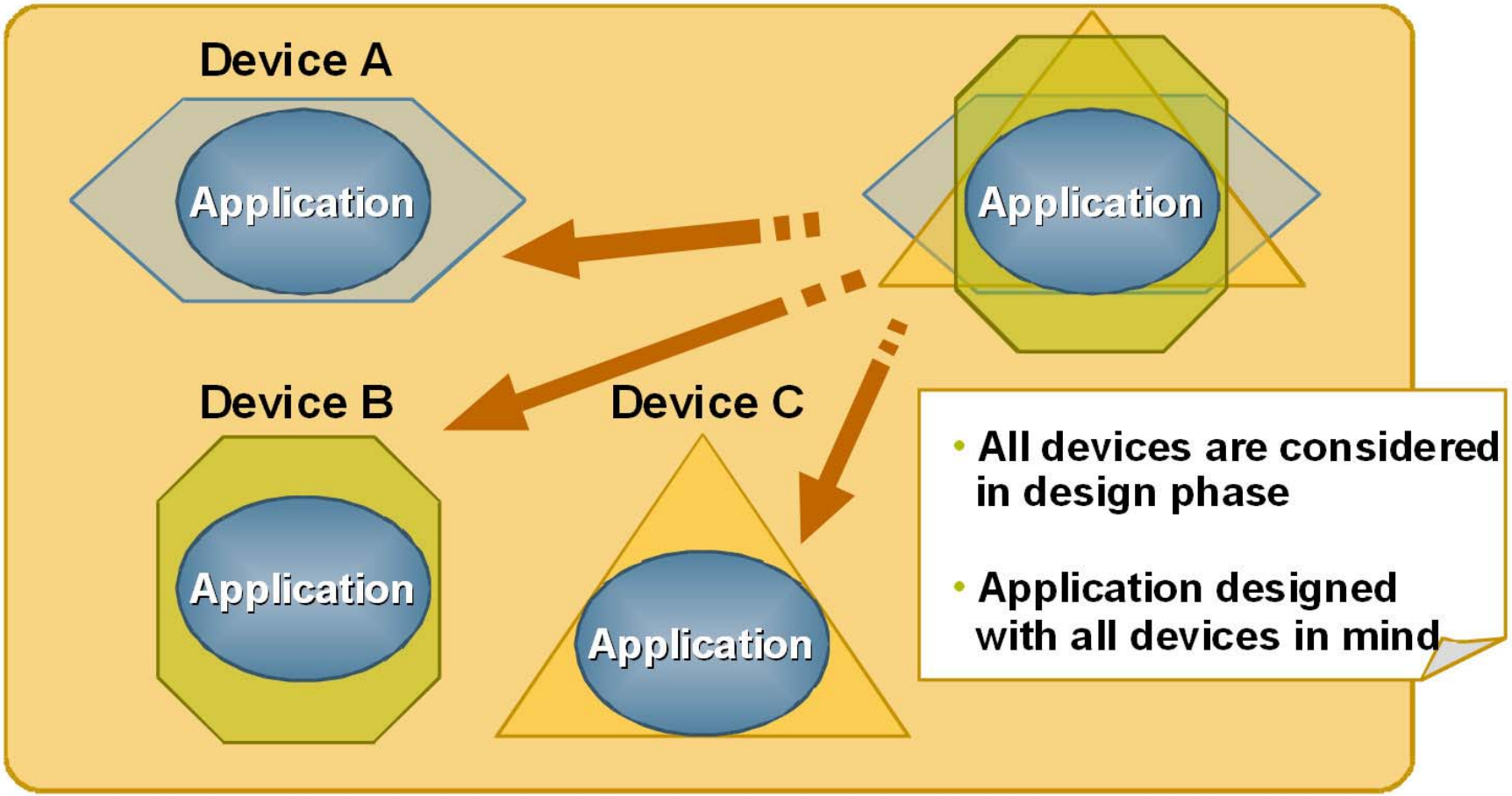
The Traditional Approach

Write once, debug anywhere



The Suggested Approach

Design for a device range



De-Fragmentation Considerations

Selecting the right solution

- Consider application requirements
  - Performance, persistent/heap memory requirements, API support
- Evaluate flexibility
  - Is your application pushing the limits of the device?
- Analyse device capabilities

- **Select your solution!**

De-Fragmentation Considerations

Performance considerations

- The trade-off of flexibility vs.
  - Memory
  - CPU time
  - Feature set
  - Java Archive (JAR) file size
- Device categories

Agenda

Overview

Design Approach

Case Studies

- **Language**
- **Key input**
- **Screen size**
- **HTTP redirection**

The Sun and Orange Collaboration

Q&A

De-Fragmenting Languages

The challenge

- Support various languages and regions
- Aspects of localisation
  - Text messages
  - Formatting policy (date/time/numeric quantities)
  - String collation—lexicographic sorting

Source: https://www28.cplan.com/javaone05_93_1/session_details.jsp?isid=269828&ilocation_id=93-1&ilanguage=english

MIDP and CLDC Support for Localization

- java.util package:
  - Calendar, Date, and TimeZone classes
- **Use:** `microedition.encoding, microedition.locale`
- User-defined Java Application Descriptor (JAD) file attributes
  - `MIDlet.getAppProperty()`
- Use resource files from the MIDlet suite JAR file
  - `Class.getResourceAsStream`
- **Retrieve classes dynamically**
  - `Class.forName(className)`

Source: https://www28.cplan.com/javaone05_93_1/session_details.jsp?isid=269828&ilocation_id=93-1&ilanguage=english

Solution Overview

- Query device parameters
- Retrieve localised information from:
  - JAD file
  - JAR file
  - Class files

Code Sample—1

Query device parameters

```
// The current locale of the device
System.getProperty("microedition.locale");
// The default character encoding name
```

- `System.getProperty("microedition.encoding");`

Read Messages From JAD File

- Adding user-defined attributes to the JAD file
- One attribute per application string per locale
- Attribute name represents the locale context
- Pros and cons
  - Very simple, does not require extra coding
  - Easy for translators
  - This approach might not be useful for large amount of resources
  - Performance might be affected due to reading of large JAD file
  - This approach addresses only string resources

Code Sample—Messages in JAD File

```
{
    String locale = System.getProperty
                    ("microedition.locale");

    //Get User-Defined attribute from JAD
    String exitStr =
      getAppProperty("hello" + "-" + locale);

    Command exitCommand = new Command(exitStr,
                                      Command.EXIT, 1);
    }

    Attributes in the JAD file
    hello-en-US: hello
    hello-fr-FR: bonjour
```

Read Messages From JAR File

- Text files that contain localized attributes
- Define one file for each locale
- File name contains the locale (e.g., en-US.txt)
- Resource files are bundled in JAR file
- Pros and cons
  - Easy integration with translator work
  - Separate translators can work independently
  - Need to create a stream parser
  - No StringTokeniser
  - Needs to hold the resources in internal data structure
  - Increases startup time

Read Messages From Class File

- Java class files contain localised resources
- Classes are compiled and packaged in JAR file
- Design your own version of Java Platform, Standard Edition (Java SE platform) Resource Bundles
- Resource files are bundled in JAR file
- Pros and cons
  - No stream handling
  - No file parsing
  - This approach addresses all I10n resources, not only strings
  - Difficult to integrate translation work
  - Large footprint on stack/JAR file size

Code Sample—Messages in Class File

```
public abstract class ResourceBundle {
//Gets a resource bundle using the specified base name and
locale
    public static final ResourceBundle getBundle
                (String baseName, String locale){
        String className = baseName + "_" + locale;
        Class c = Class.forName(className);
        ResourceBundle bundle =
                (ResourceBundle)c.newInstance();
        return bundle;
    }
//Gets an object for the given key from this resource
bundle
    protected abstract Object handleGetObject(String key);
}
```

Code Sample—Messages in Class File

```
public class Resources_fr_FR extends ResourceBundle {

    private Hashtable resources = new Hashtable();

    public Resources_fr_FR() {
        resources.put("hello", "bonjour");
        resources.put("bye", "au revoir");
        ...
    }

    protected Object handleGetObject(String key){
        return resources.get(key);
    }
}
```

Code Sample—Messages in Class File

```
{
    //The current locale of the device
    String locale = System.getProperty
                        ("microedition.locale");
    ResouceBundle bundle = ResourceBundle.getBundle
                        ("Resources", locale);
    String exitStr = (String)
                    bundle.handleGetObject("bye");
Command exitCommand = new Command(exitStr,
                                  Command.EXIT, 1);
}
```

De-Fragmenting Key Assignment

The challenge

- On different devices game key events are assigned to different keys
- Soft keys are assigned different key codes

Source: http://developers.sun.com/techtopics/mobility/reference/techart/design_guidelines/key_assignment.html

De-Fragmenting Key Assignment

Possible solutions

- Use MIDP abstraction: `Canvas.getGameAction()`
  - **Will support multiple key pads**
  - **Small footprint solution**
  - **Does not resolve soft keys allocation for customised UI implementation**
- **Use JAD file to map keys to actions**
  - **Requires device research**
  - **Requires large JAD file/JAD file management**
  - **Resolves soft key implementation**

Source: http://developers.sun.com/techtopics/mobility/reference/techart/design_guidelines/key_assignment.html

Code Sample—1

Generating a keyCode data base

```
Class myClass{
    public final static int SOFT_LEFT=-6;
    ...
// This code can be called anywhere in the code
 //Get User-Defined attribute from JAD
int upKeyCode = 0;
int softLeftKeyCode = 0;

String codeStr;
try {
    codeStr = getAppProperty("DEVICE-UP-KEY");
    upKeyCode = Integer.parseInt(codeStr);
    codeStr = getAppProperty("DEVICE-LEFT-SOFTKEY");
    softLeftKeyCode = Integer.parseInt(codeStr);
} catch (NumberFormatException nfe) {
...
```

Code Sample—2

```
// This method returns the assigned keyCode
    public int myGetGameAction(int keyCode) {
        if (keyCode == upKeyCode) {
            return Canvas.UP;
        }
        ...
        if (keyCode == softLeftKeyCode) {
        return myClass.SOFT_LEFT;
        }


    ...
    }
    ...
```

Source: http://developers.sun.com/techtopics/mobility/reference/techart/design_guidelines/key_assignment.html

De-Fragmenting Screen Size

The challenge

- Different devices have different screen sizes
- MIDP does not support image scaling—requires different JAR file for every screen size

De-Fragmenting Screen Size

Possible solutions

- Query device screen size
- Use MIDP 2.0 tiling mechanism
  - Limited to simple background imaging
  - Does not solve character image scaling
- Re-scale images to appropriate size
  - Use `javax.microedition.lcdui.Graphics` ***clipping options***
  - Performance considerations
  - Memory consumption consideration

Source: http://developers.sun.com/techtopics/mobility/reference/techart/design_guidelines/image_resizing.html

Code Sample—Querying Screen Size

```
/* This code can be called anywhere in the application.
 * recommended to use during startApp
 * int width and int height are class members*/

    Canvas dummyCanvas = new Canvas()

   // asuming application will run in full screen
   dummyCanvas.setFullScreenMode(true);

   // get the dimensions of the screen:

   width = dummyCanvas.getWidth ();
  height = dummyCanvas.getHeight();
```

Code Sample—Resizing the Image

```
int srcWidth = src.getWidth();
int srcHeight = src.getHeight();
Image tmp = Image.createImage(screenWidth,srcHeight);
Graphics g = tmp.getGraphics();
int ratio = (srcWidth << 16) / screenWidth;
int pos = ratio/2;

//Horizontal Resize

for (int x = 0; x < screenWidth; x++) {
//public void setClip(int x, int y, int width, int height)
    g.setClip(x, 0, 1, srcHeight);
    g.drawImage(src, x - (pos >> 16), 0, Graphics.LEFT
|                                   Graphics.TOP);
   pos += ratio;
}
```

Code Sample—Resizing the Image

```
Image resizedImage = Image.createImage(screenWidth,
                                       screenHeight);
g = resizedImage.getGraphics();
ratio = (srcHeight << 16) / screenHeight;
pos = ratio/2;

//Vertical resize

for (int y = 0; y < screenHeight; y++) {
    g.setClip(0, y, screenWidth, 1);
    g.drawImage(tmp, 0, y - (pos >> 16), Graphics.LEFT |
                                    Graphics.TOP);
    pos += ratio;
}
```

De-Fragmenting Screen Size

Solution considerations

- Memory/performance trade off
  - Call once at first startup and save resized images to memory
- How to categorise devices?
- How to decide which image to use?

Source: http://developers.sun.com/techtopics/mobility/reference/techart/design_guidelines/image_resizing.html

De-Fragmenting HTTP 302 Code

The challenge

- Different devices handle the HTTP 302 code in different ways
  - Implementation returns 302 code to the MIDlet
    - “Expected” behavior
  - Implementation handles the redirection
    - Returns 302 but redirect address is set to null
    - When the redirection will be complete a Code 200 is received
  - Redirect address is inside the response content
    - Due to bad parsing of HTTP headers

Solution approach—build robust code to handle all cases

- **Wait after 302 code is received**
- **Parse response content for redirection**

http://developers.sun.com/techtopics/mobility/reference/techart/design_guidelines/http_redirection.html

DEMO

Verifying solutions on NetBeans™ Mobility Pack for CLDC

Screen size

Agenda

Overview

Design Approach

Case Studies

The Orange and Sun Collaboration

- **Background**
- **Outcomes**
- **How can you help?**

Q&A

Sun and Orange Collaboration

Background

- Goals
  - Reduce the number of application derivatives submitted to Orange
  - Provide short-term solution as part of a long-term quality improvement program
- Time frames
  - 18 months to date

Sun and Orange Collaboration

- Audience: Developers, operators and the Java platform eco-system; what's good for one is good for all
- Java platform fragmentation: Identified as root cause of growth restriction
- Action plan
  - Short term: Application guidelines
  - Mid to long term:
    - Influence future standards
    - Raising the bar on quality
- Generating positive change immediately and a brighter future

Releasing to the Community

What have we come up with so far?

- Stay tuned for a press release!

- Sun Developer Website
  - http://developers.sun.com/techtopics/mobility/reference/techart/design_guidelines/overview.html
- Orange partner website
  - http://www.orangepartner.com/site/enuk/develop/initiatives/java_fragmentation/l_fragmentation.jsp
  - http://www.orangepartner.com/site/enuk/develop/initiatives/java_fragmentation/l_java_design_guides_index.jsp

Summary

- Fragmented applications are an unsustainable headache
- Long-term solutions are being explored
- There is no magic solution
  - More sophisticated design and coding
  - Use guile rather than brute force
- Match solution to application requirements
  - Consider trade-offs
- Design for device categories

Q&A

- Martin Wrigley
- Rhian Sugden
- Nir Vazana

Tackling Java ME Device Fragmentation: Orange and Sun Collaboration

Martin Wrigley

Head of Partner's Technology, Orange Partners Operations

Orange/France Telecom

http://www.orangepartner.com/

Rhian Sugden

Account Manager, OEM Software Sales

Sun Microsystems, Inc.

http://java.sun.com/

Nir Vazana

Technical Leader, Engineering Services

Sun Microsystems, Inc.

http://java.sun.com/

TS-5051