Create Appealing Cross-device Applications for Mobile Devices with Java ME and LWUIT

11, 12 Dec. 2009

Pune, India

IndicThreads.com Conference On Java Technology

Biswajit Sarkar

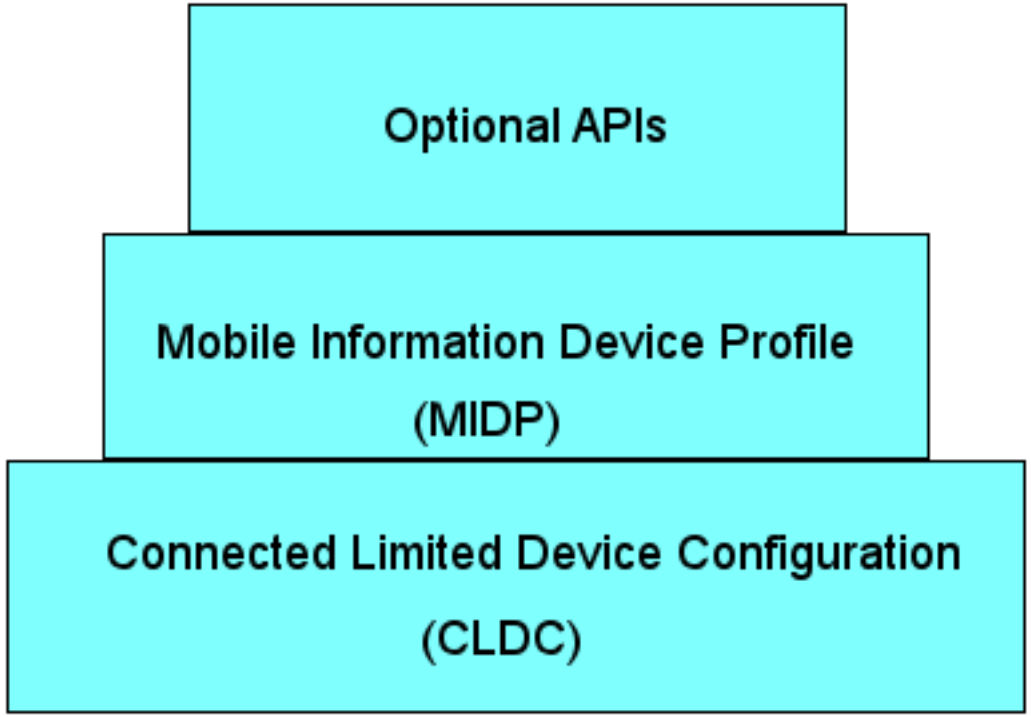
Agenda

- Java ME basics with focus on CLDC/MIDP
- Java ME application development
- A simple demo application
- Fragmentation
- Non-uniform UI
- LWUIT
- Resources

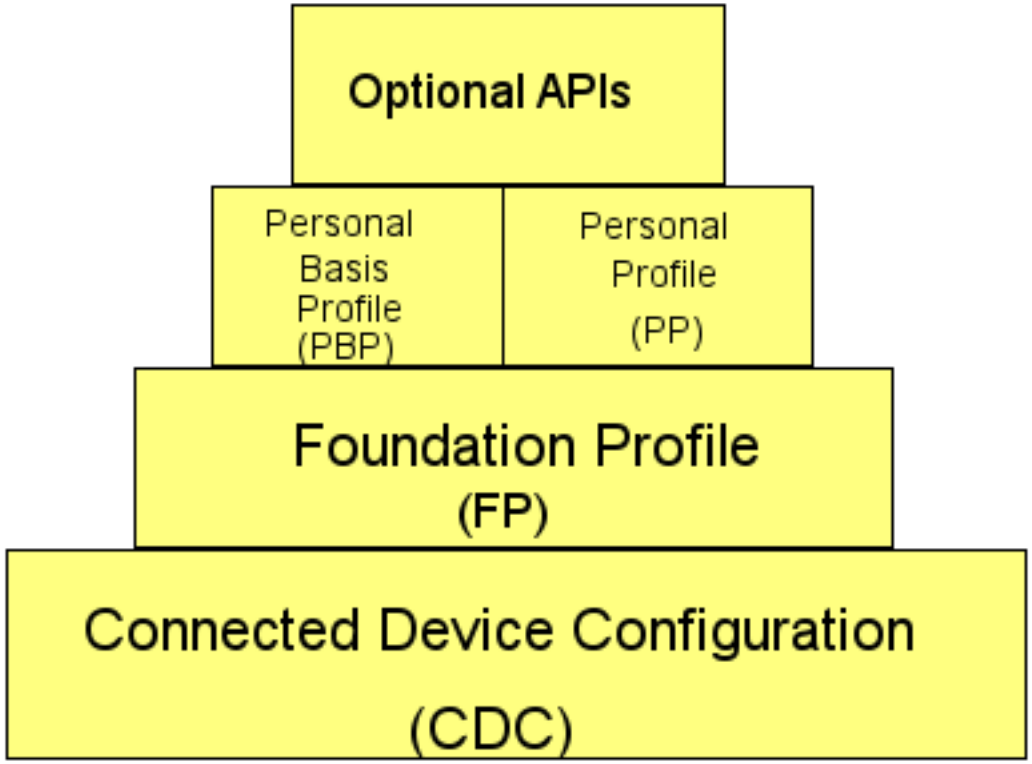
Java ME – an Introduction

- Highly portable – supported on 2 billion+ phones
- Works on many devices such as:
  - Phones
  - Personal Digital Assistants (PDAs)
  - Set-top boxes
- Version based on Java ME Personal Basis Profile for Blu-ray Disc players (BD-J)
- Robust security
- Built-in networking protocols
- Continuously growing

Java ME for Devices with Limited Resources



Java ME for Less Resource Constrained Devices



CLDC 1.1 (JSR 139)

- At least 160 KB of non-volatile memory such as ROM (memory budget 192 KB for FP support)
- At least 32 KB of volatile memory
- Defines a subset of J2SE
- The RI implementation based on JVM optimized for small devices – known as KVM
- Defines off-device preverification

MIDP 2.0 (JSR 118)

- At least 256 KB ROM for MIDP implementation
- At least 128 KB VRWM for runtime heap
- At least 8 KB NVRWM for persistent data (RMS)
- Screen size at least 96x54 pixels
- Two-way wireless networking – limited bandwidth, possibly intermittent
- Input through one-handed/two-handed keyboard or touchscreen
- Ability to play tones via hardware or software

CLDC 1.1 Packages

- java.lang
- java.lang.ref
- java.io
- java.util
- javax.microedition.io

MIDP 2.0 Packages

- javax.microedition.lcdui
- javax.microedition.lcdui.game
- javax.microedition.media
- javax.microedition.media.control
- javax.microedition.midlet
- javax.microedition.pki
- javax.microedition.rms

CLDC & MIDP – Some Missing Items

- Access to native methods
- Reflection
- Classloader – accessible only to Application Manager
- Object finalization

Optional APIS - Some Examples

Personal Information

- JSR 75 - PIM and File
- JSR 179 - Location

Optional APIS - Some Examples

Communication

- JSR 120 - SMS Messaging
- JSR 205 - MMS Messaging
- JSR 82 - Bluetooth & OBEX

Optional APIS - Some Examples

Graphics

- JSR 226 - SVG
- JSR 184 - 3D Graphics

Commerce

- JSR 229 -- Payment

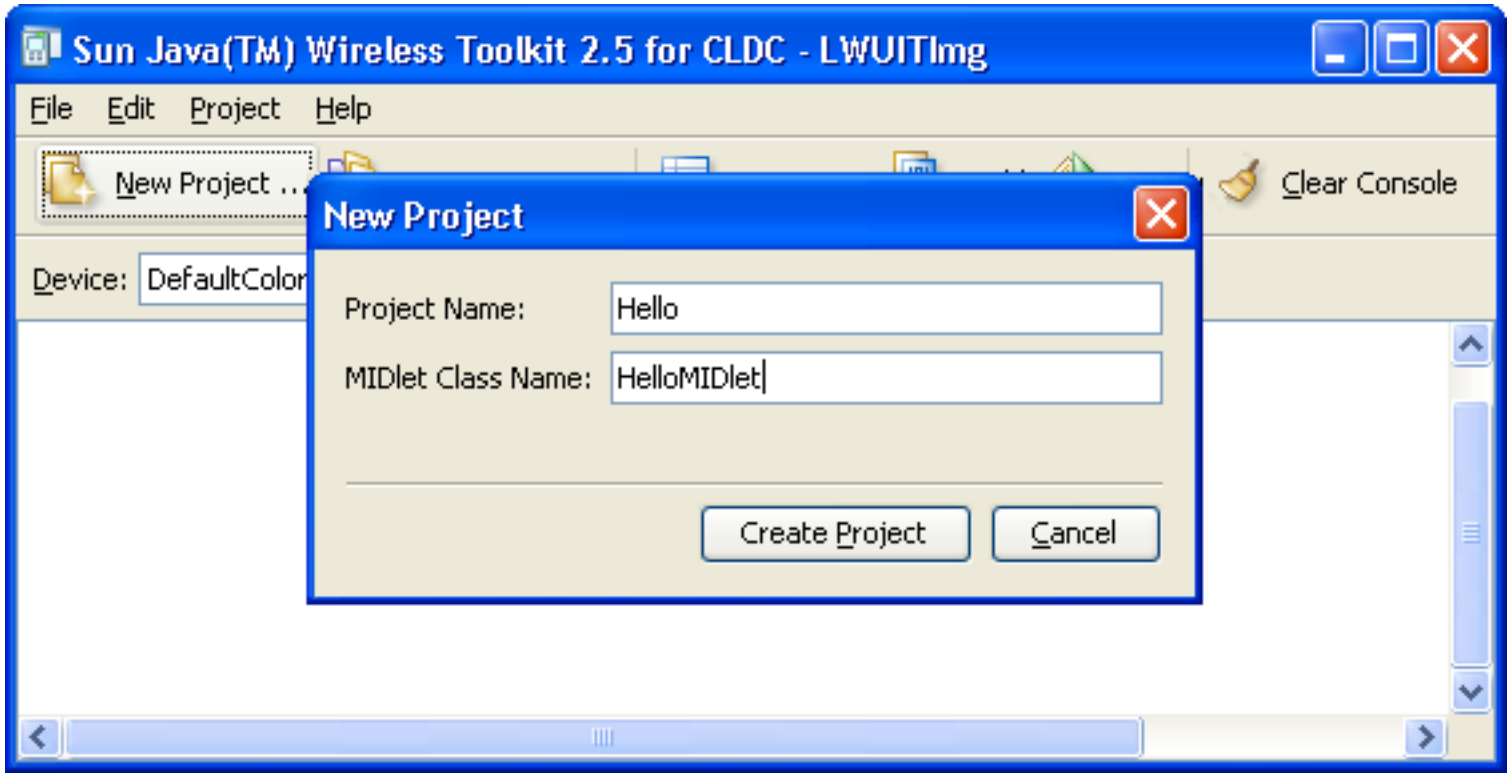
Application Development Steps

- Write code
- Compile
- Preverify
- Package
- Deploy

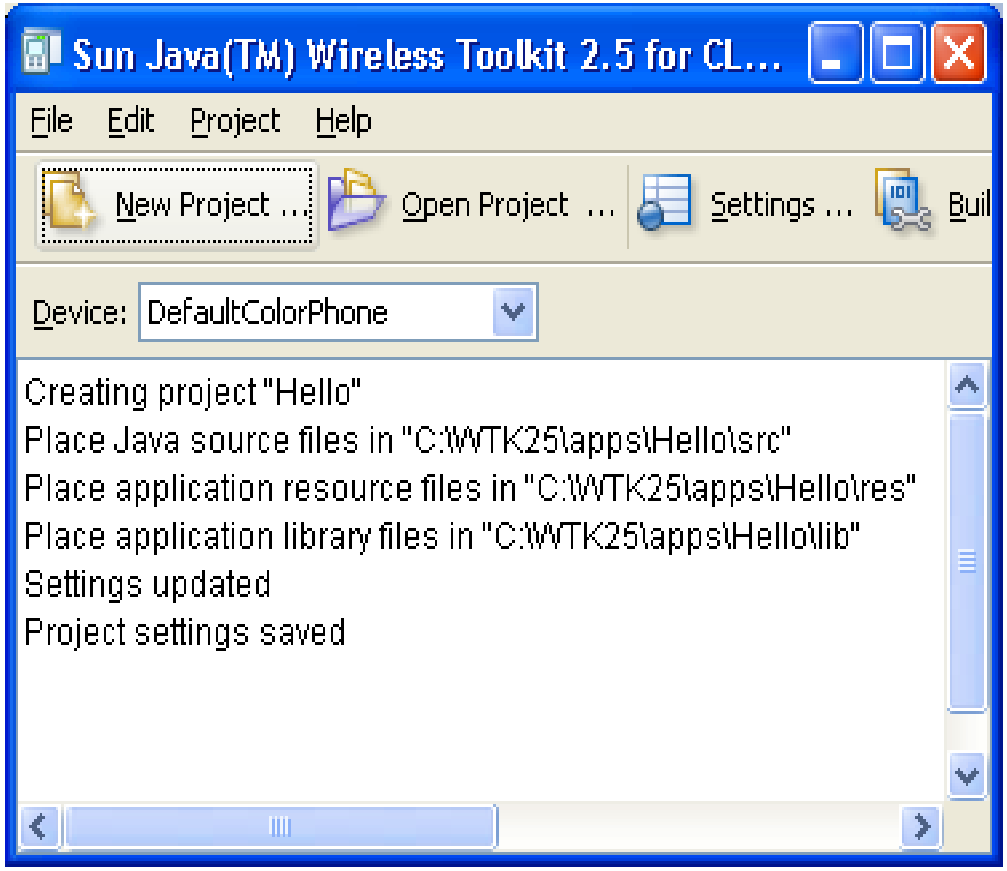
Development Tools

- Netbeans
- Eclipse
- Java ME Platform SDK
- Sun Java Wireless Toolkit for CLDC
- Many other SDKs/WTKs

Creating a Project



Project Created



MIDlets

- MIDP Applications are called MIDlets
- Multiple applications (MIDlets) can be packaged into a MIDlet suite
- Application entry point extends MIDlet class
- Application Manager initiates MIDlet

MIDlet Life Cycle

- (Constructor)
- StartApp
- PauseApp
- DestroyApp
- (NotifyDestroyed)

MIDlet States

A Demo MIDlet - Imports

```
import javax.microedition.lcdui.*;

import javax.microedition.midlet.MIDlet;
```

A Demo MIDlet – Declarations

```
public class HelloMIDlet extends MIDlet
  implements CommandListener
{
    private Command CMD_EXIT;
    private Display display;
    private TextBox textBox;
    .
    .
```

A Demo MIDlet - Constructor

```
public HelloMIDlet()
{
    CMD_EXIT = new
        Command("Exit", Command.EXIT, 1);
    display = Display.getDisplay(this);
    textBox = new TextBox("Hello", "Hello World
        of Java ME", 40, 0);
    textBox.addCommand(CMD_EXIT);
    textBox.setCommandListener(this);
}
```

A Demo MIDlet – Starting the Application

```
protected void startApp()
{
    display.setCurrent(textBox);
}
```

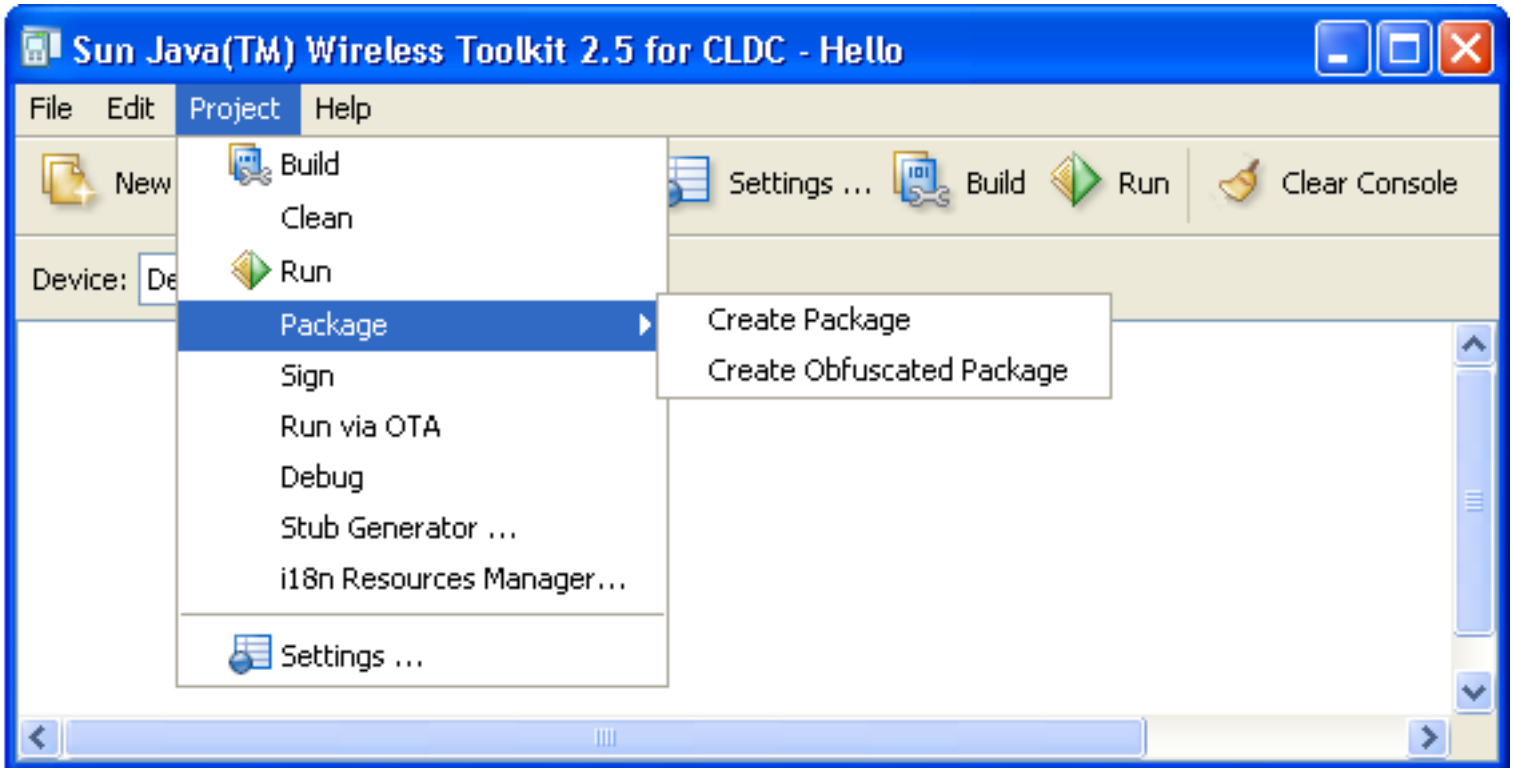
A Demo MIDlet – Command Response

```
public void commandAction(Command c,
  Displayable d)
{
    if (c == CMD_EXIT)
    {
        notifyDestroyed();
    }
}
```

A Demo MIDlet - Other Methods

```
protected void pauseApp()
{
}

protected void destroyApp(boolean unconditional)
{
}
```

Building the Application



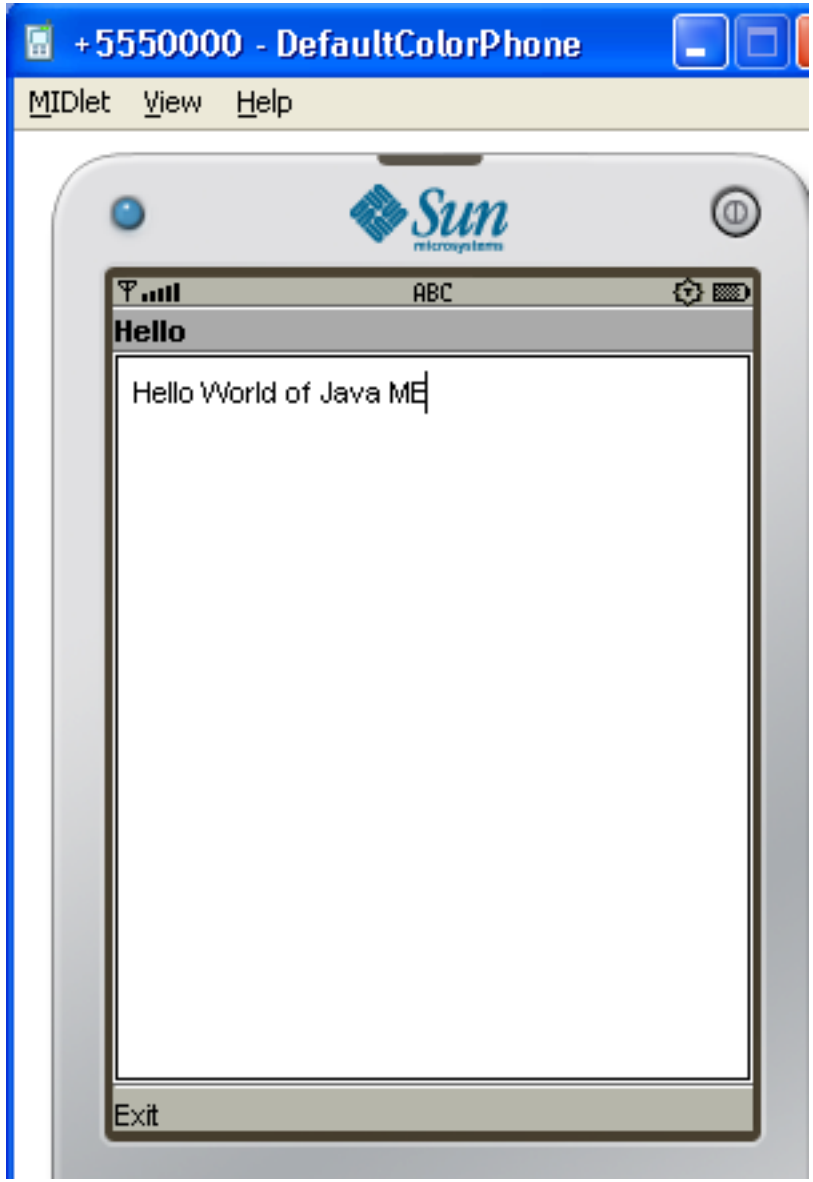
Deployment

Direct

- Data Cable
- Bluetooth

Through Network

- Over the Air (OTA)


+5550000 - DefaultColorPhone
MIDlet
View
Help
Sun
microsystems
ABC
Hello
Hello World of Java ME
Exit

Fragmentation

- Makes it difficult to write multi-platform apps
- Caused by differences between implementations
- Different devices have different API sets
- Some phone manufacturers use proprietary APIs
- Different security policies imposed by manufacturers

API Standardization a Possible Solution for Fragmentation

Mobile Service Architecture (MSA)

"The MSA Specification is a Java architecture definition that describes the essential Java client components of an end-to-end wireless environment."

Mandatory JSRs - MSA Subset

- JSR 118 - MIDP
- JSR 139 - CLDC
- JSR 226 - SVG
- JSR 205 - Messaging
- JSR 184 - 3D Graphics
- JSR 135 - Mobile Media
- JSR 75 - PIM & File

Other Mandatory JSRs

- JSR 238 – Internationalization
- JSR 234 – Multimedia Supplement
- JSR 229 – Payment
- JSR 211 – Content Handler
- JSR 180 – SIP
- JSR 172 – Web Services

Conditionally Mandatory JSRs

- JSR 82 – Bluetooth & OBEX (part of MSA Subset)
- JSR 177 – Security & Trust
- JSR 179 – Location

Lack of UI Uniformity

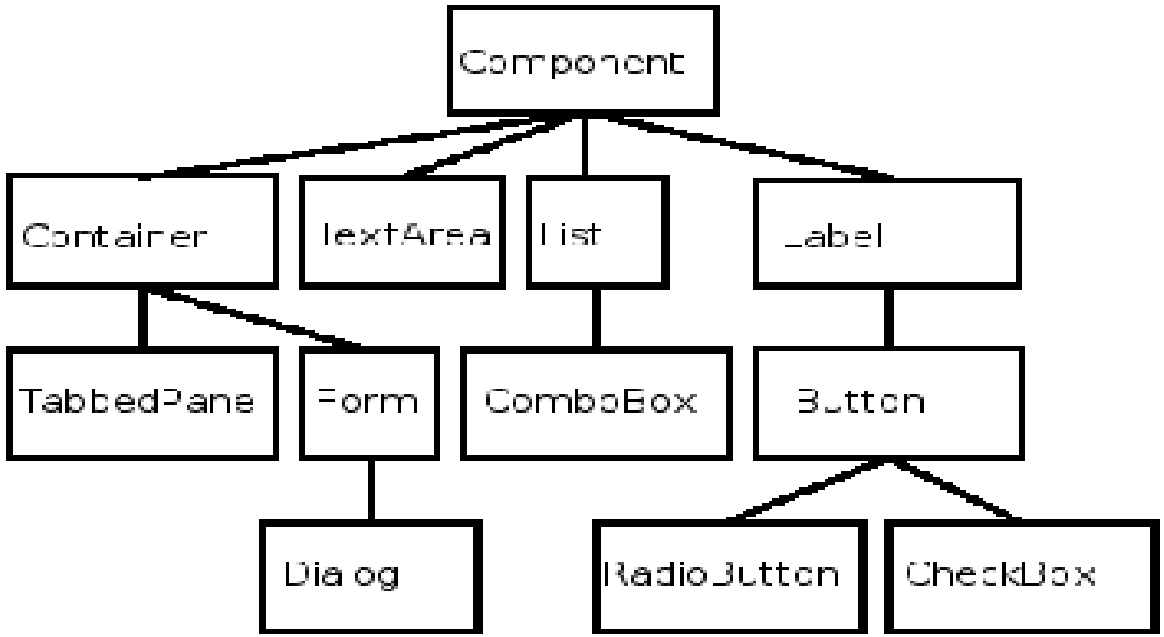
- Different display styles on devices
- Native UI components have diverse look-and-feel
- Command placements differ from one device to another
- Menu structures dictated by native operating environment

As a result, virtually impossible to create device-independent and uniform look-and-feel

Light Weight User Interface Toolkit (LWUIT) for UI Uniformity and Sophistication

- Platform independent UIs
- Wide range of components
- New functionalities
- Swing like architecture
- Bundled library
- Works with CLDC 1.1 and MIDP 2.0

LWUIT -- Components



Adapted from : Developer's Guide, Lightweight UI Toolkit (Sun Microsystems, Inc.)

LWUIT - Functionalities

- Animation
- Transitions – both 2D and 3D
- Visual styling
- Theming
- Localization
- Logging
- Special effects
- Elaborate layouts
- SVG support
- Design tool – the LWUIT Designer

LWUIT - Swing-like Architecture

- Lightweight
- Modified M-V-C
- Customizable
- Handles and encapsulates all UI threading through its main thread – EDT
- Events and paint calls serialized through EDT


Hello LWUIT
Hello World of LWUIT
Exit

Form – Constituent Parts

- Title Bar
- Contentpane
- Menu Bar

Title Bar and Form Style – First Approach

```
//create a font
Font font = Font.createSystemFont
      (Font.FACE_PROPORTIONAL,
      Font.STYLE_BOLD,Font.SIZE_LARGE);

//set text color for title
demoForm.getTitleStyle().setFgColor(0);

//set font style for title
demoForm.getTitleStyle().setFont(font);

//set background color for the title bar and form
demoForm.getTitleStyle().setBgColor(0xff8040);
demoForm.getStyle().setBgColor(0x656974);
```

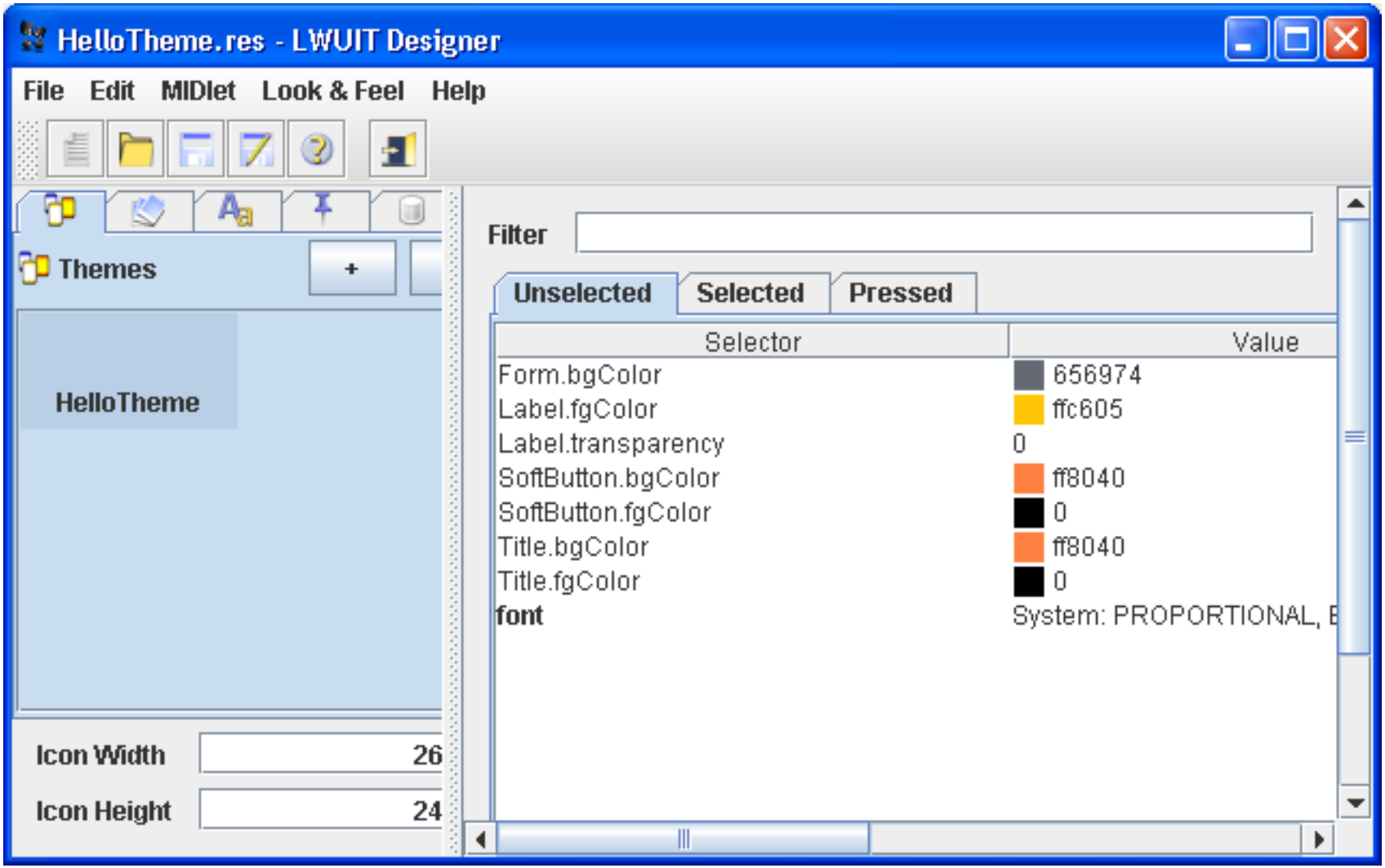
Menu Bar Style - Second Approach

```
//create a new style object
Style menuStyle = new Style();

//set the background color
menuStyle.setBgColor(0xff8040);

//set the text color for menu bar
menuStyle.setFgColor(0);

//set font style for menu bar
menuStyle.setFont(font);

//now install the style for menu bar
demoForm.setSoftButtonStyle(menuStyle);
```

Theming – Third Approach to Styling

- A Theme sets visual attributes for all components of a given type at a single place
- Visual coherence for all screens of an application can be established through a Theme

Creating a Theme



Packaging a Theme

- LWUIT Designer is a utility for creating Themes
- Themes are packaged into Resource (.res) files
- Resource files can also contain
  - Images
  - Bitmap fonts
  - Animations
  - Localization resources
- Resources files are packaged into JARs

LWUIT – Animation and Transitions

- Animation is repeated rendering at a rate determined by the LWUIT environment
- Animated PNG supported
- All components are animation capable
- Transitions define how forms and components are replaced on screen
- A number of transitions built into the library
- Custom transitions can be created easily

LWUIT -- Layouts

- Border
- Box
- Coordinate
- Flow
- Grid
- Group
- Table (part of new component)

LWUIT – Logging

- Records runtime information with time stamp
- Provides data for debugging and on operational parameters
- Four logging levels
  - Debug
  - Info
  - Warning
  - Error
- Logs using RMS or File Connection API (JSR 75)
- Supports pluggable subclass

Staying Current

- Java Application Terminal Alignment Framework (JATAF) – an organization formed by Sun, Orange, Sony Ericsson and Vodafone to address fragmentation issues
- MIDP 3.0 – functional enhancements beyond MIDP 2.0
- MSA v2 (JSR 249) – next revision of JSR 248

Resources

- Kicking Butt with MIDP and MSA: Creating Great Mobile Applications – Jonathan Knudsen (Prentice Hall PTR)
- LWUIT 1.1 for Java ME Developers – Biswajit Sarkar (Packt/Shroff)
- http://www.java.net/articles -- various articles
- http://lwuit.blogspot.com/ -- Shai Almog's blog on Java & LWUIT

Thank You