

Embedded Java IDE

Embedded Java IDE in current version is a plugin for NetBeans 8 framework.

Embedded Java project

Embedded Java project is an extension of conventional NetBeans C language project, compatible upside down. We can see additional folder “Java Files”, which contains Java subproject with Java sources.

Also we can open Embedded Java projects in NetBeans IDE without plugin, in this case Java subfolder will be represented as ordinary file tree and during debugging all Java structures will be represented as their C counterparts.

Embedded Java project

The screenshot displays the NetBeans IDE environment for an Embedded Java project named "LWUIT". The interface is divided into three main sections:

- Project Explorer (Left):** Shows the project structure. The "Source Packages" section lists various packages under "com.sun.lwuit", including "animations", "events", "geom", "impl", "layouts", "list", "plaf", "spinner", "table", "tree", "util", "device", "io", "lang", "text", "util", and "javax.com". The "Libraries" section shows "Linker Files" and "Source Files".
- Code Editor (Center):** Displays the source code for the "Calendar.java" file. The code defines a "Calendar" class with various static and instance variables, including month and day arrays, and methods for creating and managing the calendar. The code is as follows:

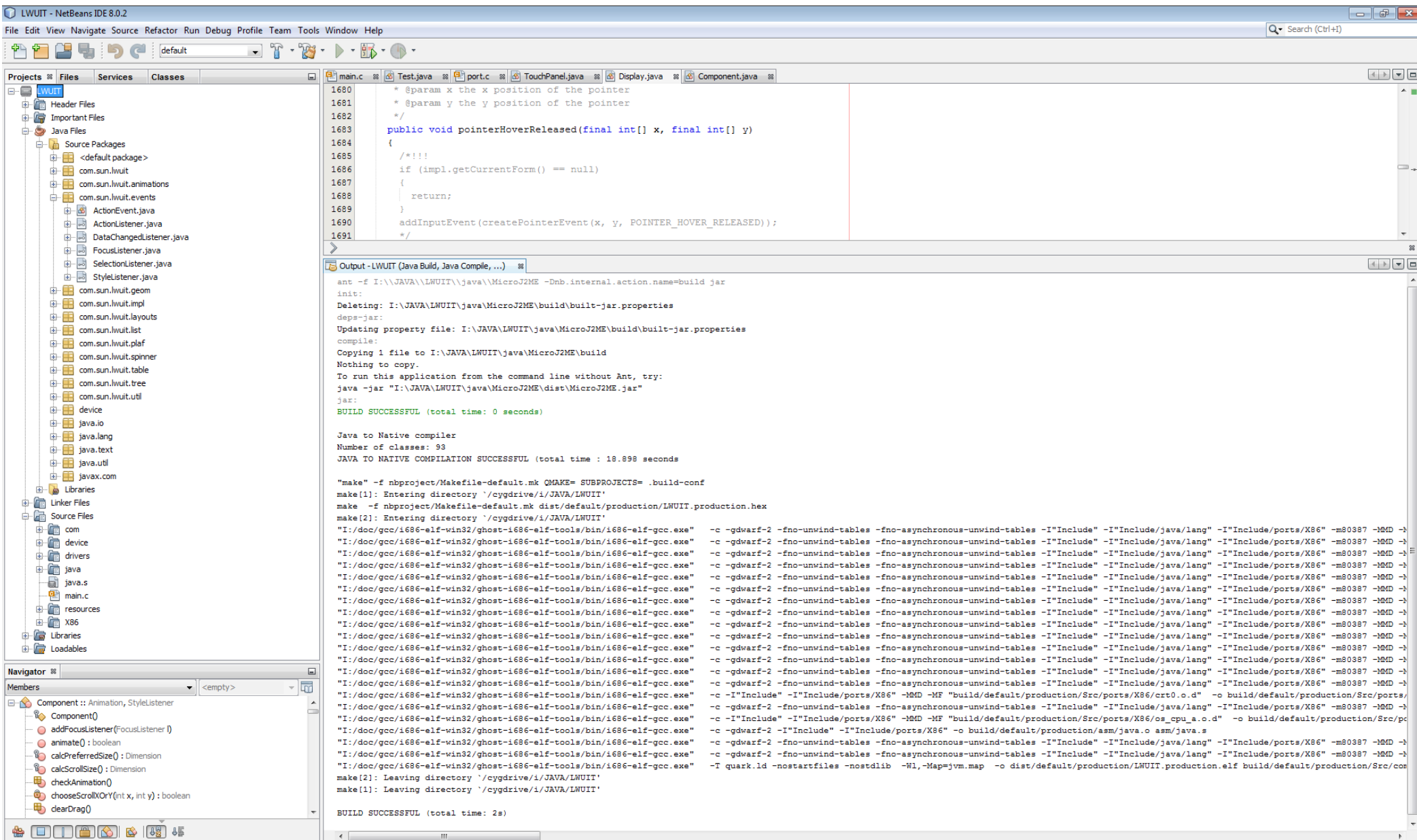
```
55 private ComboBox month;
56 private ComboBox year;
57 private MonthView mv;
58 private static final String[] MONTHS = new String[]
59 {
60     "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
61 };
62 private static final String[] DAYS =
63 {
64     "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
65 };
66 private static final String[] LABELS =
67 {
68     "Su", "M", "Tu", "W", "Th", "F", "Sa"
69 };
70 static final long MINUTE = 1000 * 60;
71 static final long HOUR = MINUTE * 60;
72 static final long DAY = HOUR * 24;
73 static final long WEEK = DAY * 7;
74 private EventDispatcher dispatcher = new EventDispatcher();
75 private EventDispatcher dataChangeListeners = new EventDispatcher();
76 private long[] dates = new long[42];
77 private boolean changesSelectedDateEnabled = true;
78 private TimeZone tmz;
79
80 /**
81  * Creates a new instance of Calendar set to the given date based on time
82  * since epoch (the java.util.Date convention)
83  *
84  * @param time time since epoch
85  */
86 public Calendar(long time)
87 {
88     this(time, java.util.TimeZone.getDefault());
89 }
90
91 /**
92  * Constructs a calendar with the current date and time
93  */
94 public Calendar()
95 {
96     this(System.currentTimeMillis());
97 }
98
99 /**
100  * Creates a new instance of Calendar set to the given date based on time
101  * since epoch (the java.util.Date convention)
102  *
103  * @param time time since epoch
104  * @param tmz a reference timezone
105  */
106 public Calendar(long time, TimeZone tmz)
107 {
108     super(new BorderLayout());
109     this.tmz = tmz;
```
- Members Window (Bottom):** Shows the members of the "Calendar" class, including methods like "Calendar(long time, TimeZone tmz)", "Calendar()", "Calendar(long time)", "addActionListener(ActionListener l)", "addDataChangeListener(DataChangeListener l)", "componentChanged()", and "createDay() : Button".

Project Build

Then we can build entire project. In this figure we can see all stages of building process:

- Build Java subproject
- Java to Native (X86 assembler) compilation
- Build Native (C & asm language subproject)

Project Build



Simulator or Board connection

After build, simulator of X86 (or, later, real board with Curie) can be connected. *Current version supports simulator, but has all interfaces to JTAG debugging.*

Program loaded to simulator and running example.

We can see a form with Calendar widget from LWUIT, which contains different internal widgets, such as labels, buttons and combo boxes.

LCD simulates real display 320 to 240 pixels, which is driven by ILI9325 controller, connected to processor I/O bus.

Board or Simulator connection

MicroJ2ME - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

PC: 0x482F9 381.3/694.0MB

Projects Files Services Classes

com.sun.lwuit

Calendar.java

```
73 static final long WEEK = DAY * 7;
74 private EventDispatcher dispatcher = new EventDispatcher();
75 private EventDispatcher dataChangeListeners = new EventDispatcher();
76 private long[] dates = new long[42];
77 private boolean changesSelectedDateEnabled = true;
78 private TimeZone tmz;
79
80 /**
81  * Creates a new instance of Calendar set to the given date based on time
82  * since epoch (the java.util.Date convention)
83  *
84  * @param time time since epoch
85  */
86 public Calendar(long time)
87 {
88     this(time, java.util.TimeZone.getDefault());
89 }
90
91 /**
92  * Constructs a calendar with the current date and time
93  */
94 public Calendar()
95 {
96     this(System.currentTimeMillis());
97 }
98
99 /**
100  * Creates a new instance of Calendar set to the given date based on time
101  * since epoch (the java.util.Date convention)
102  *
103  * @param time time since epoch
104  * @param tmz a reference timezone
105  */
106 public Calendar(long time, TimeZone tmz)
107 {
108     super(new BorderLayout());
109     this.tmz = tmz;
110     setUIID("Calendar");
111     Container upper = new Container(new FlowLayout(Component.CENTER));
112     month = new ComboBox();
113     year = new ComboBox();
114     mv = new MonthView(time);
115
116     Vector months = new Vector();
117     for (int i = 0; i < MONTHS.length; i++)
118     {
119         months.addElement("" + getLocalizedMonth(i));
120     }
121     ListModel monthsModel = new DefaultListModel(months);
122     int selected = months.indexOf(getLocalizedMonth(mv.getMonth()));
123     month.setModel(monthsModel);
124     month.setSelectedIndex(selected);
125     month.addActionListener(mv);
126
127     java.util.Calendar cal = java.util.Calendar.getInstance(tmz);
128     cal.setTime(new java.util.Date(time));
```

Calendar

Nov 2015

Su	M	Tu	W	Th	F	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5
6	7	8	9	10	11	12

Calendar - Navigator

Members

Calendar :: Container

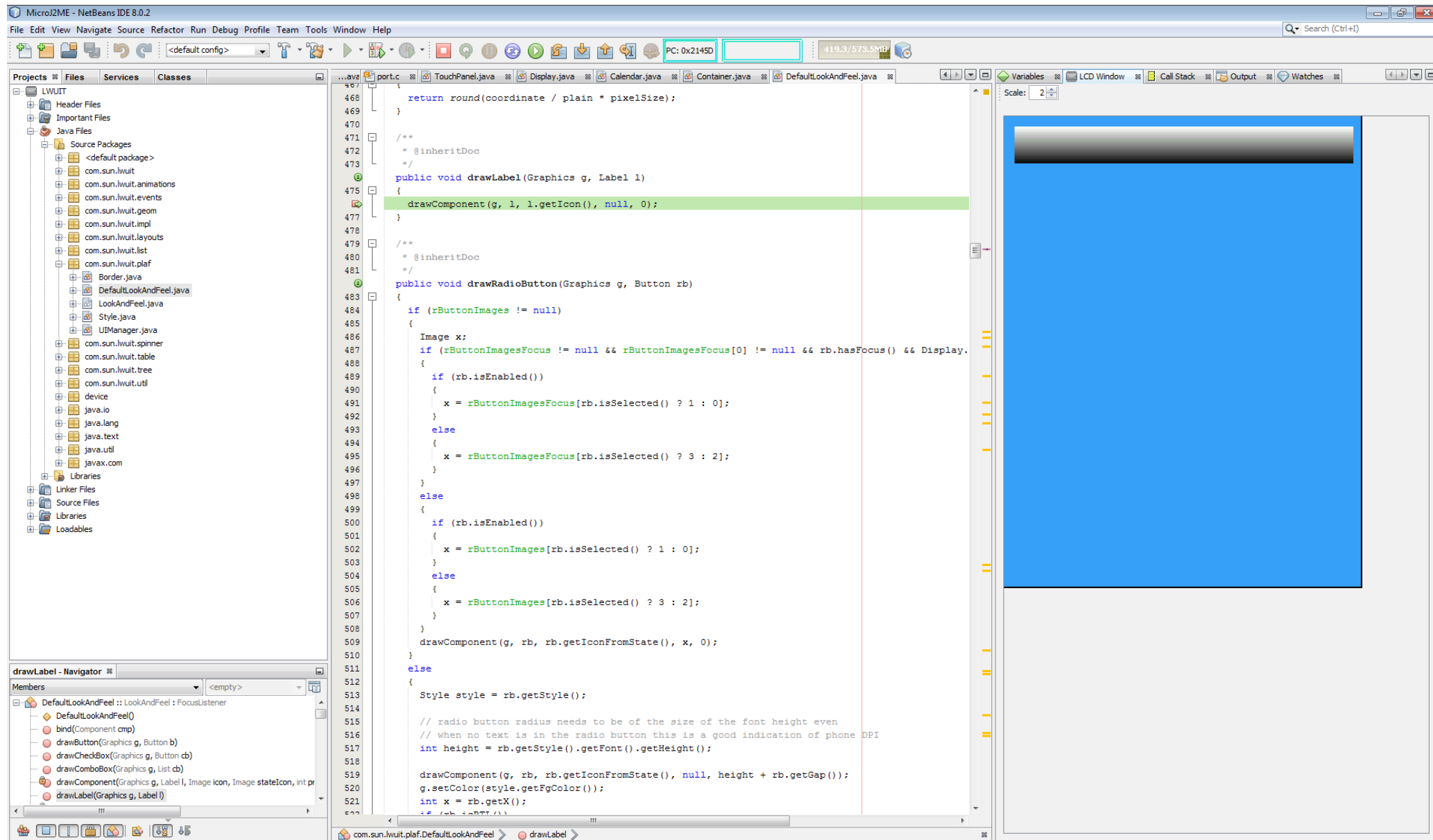
- Calendar(long time, TimeZone tmz)
- Calendar()
- Calendar(long time)
- addActionListener(ActionListener l)
- addDataChangeListener(DataChangeListener l)
- componentChanged()
- createDay() : Button
- createDayTitle(int day) : Label

Stop on breakpoint

Now we can set a breakpoint and restart application.

Program stops on breakpoint during display form painting.

Stop on breakpoint



Disassembler Window

Now we can switch to Disassembler Window and see X86 assembler code (*this is code with maximum debugging capabilities, not for evaluation of compiler optimizations*).

In Disassembler Window we can toggle breakpoints of two different types – Java/C Source line breakpoints and Address breakpoints, system automatically choose which type of breakpoint will be toggled. If disassembly line has mapping from Java/C source, then Source line breakpoint will be added/removed, otherwise – Address breakpoint, which can be seen only in Disassembler of Breakpoints window

Disassembler Window

NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

default PC: 0x32890 601.9/842.5MB

Projects Files Services

- LWUIT
 - Header Files
 - com
 - device
 - drivers
 - java
 - X86
 - Important Files
 - Java Files
 - Source Packages
 - <default package>
 - com.sun.lwuit
 - Button.java
 - ButtonGroup.java
 - Calendar.java
 - CheckBox.java
 - ComboBox.java
 - Command.java
 - Component.java
 - Container.java
 - Dialog.java
 - Display.java
 - Font.java
 - Form.java
 - Graphics.java
 - Image.java
 - Label.java
 - List.java
 - MenuBar.java
 - Painter.java
 - RGBImage.java
 - RadioButton.java
 - RunnableWrapper.java
 - Slider.java
 - TabbedPane.java
 - Tabs.java
 - TextArea.java
 - TextField.java
 - VirtualKeyboard.java
 - com.sun.lwuit.animations
 - com.sun.lwuit.events
 - com.sun.lwuit.geom
 - com.sun.lwuit.impl
 - com.sun.lwuit.layouts
 - com.sun.lwuit.list

main.c Test.java port.c TouchPanel.java Disassembly Display.java Calendar.java Graphics.c Container.java...

```
1 // public Calendar(long time)
2 // {
3   com.sun.lwuit.Calendar.Calendar:
4   00032884 55          push    ebp
5   00032885 89 E5      mov     ebp, esp
6   00032887 53          push    ebx
7   //      this(time, java.util.TimeZone.getDefault());
8   00032888 E8 AC 22 00 00 call    java_util_TimeZone_getDefault: +8876
9   0003288D 89 C3      mov     ebx, eax
10  0003288F 53          push    ebx
11  00032890 FF 75 10    push    [ebp + 16]
12  00032893 FF 75 0C    push    [ebp + 12]
13  00032896 FF 75 08    push    [ebp + 8]
14  00032899 E8 28 00 00 00 call    com$sun$lwuit$Calendar$Calendar$3: +40
15  0003289E 83 C4 10    addl    esp, 0x10
16  //      }
17  000328A1 5B          pop     ebx
18  000328A2 5D          pop     ebp
19  000328A3 C3          ret
20  //
21  // /**
22  //  * Constructs a calendar with the current date and time
23  //  */
24  // public Calendar()
25  // {
26  //   com.sun.lwuit.Calendar.Calendar:
27  000328A4 C8 08 00 00 enter    0x8, 0x0
28  //      this(System.currentTimeMillis());
29  000328A8 E8 FB 28 FD FF call    java_lang_System_currentTimeMillis: -186117
30  000328AD 89 45 F8    mov     [ebp - 8], eax
31  000328B0 89 55 FC    mov     [ebp - 4], edx
32  000328B3 FF 75 FC    push    [ebp - 4]
33  000328B6 FF 75 F8    push    [ebp - 8]
34  000328B9 FF 75 08    push    [ebp + 8]
35  000328BC E8 C3 FF FF FF call    com_sun_lwuit_Calendar_Calendar_J_: -61
36  000328C1 83 C4 0C    addl    esp, 0xC
37  //      }
38  000328C4 C9          leave
39  000328C5 C3          ret
40  //
41  // /**
42  //  * Creates a new instance of Calendar set to the given date based on time
43  //  * since epoch (the java.util.Date convention)
44  //  *
45  //  * @param time time since epoch
46  //  * @param tmz a reference timezone
47  //  */
48  // public Calendar(long time, TimeZone tmz)
49  // {
50  //   com.sun.lwuit.Calendar.Calendar:
51  000328C6 C8 0C 00 00 enter    0xC, 0x0
52  000328CA 53          push    ebx
53  000328CB 56          push    esi
54  000328CC 57          push    edi
55  //      super(new BorderLayout());
56  000328CD 68 3E C2 03 00 pushl    0x3C23E
57  000328D2 E8 10 23 FD FF call    newObject: -187632
```

Debugger Console

LWUIT (Java Build, Java Compile, ...)

Launching
User program running
User program stopped
User program running
User program stopped
User program running
User program stopped
User program running
User program stopped
User program running
User program stopped

Processor registers window

Processor registers window contains nothing special. For convenience it has a tree representation of different register groups (processor peripherals will be also included).

Processor registers window

The screenshot displays the NetBeans IDE 8.0.2 interface with the 'Registers' window open. The main editor shows assembly code for two methods: `drawLabel` and `drawRadioButton`. The 'Registers' window on the right is divided into 'Core Registers' and 'Control Registers'.

Core Registers:

Register	Value	Format
EAX	0003a400	Hex
EBX	0020047c	Hex
ECX	00200448	Hex
EDX	000004df	Hex
EDI	00000023	Hex
ESI	00000019	Hex
EBP	003ffe3c	Hex
ESP	003ffe34	Hex
EIP	00021463	Hex
EFLAGS	00000246	Hex
AC	0	Hex
VM	0	Hex
RF	0	Hex
NT	0	Hex
IPOL	0	Hex
OF	0	Hex
DF	0	Hex
IF	1	Hex
TF	0	Hex
SF	0	Hex
ZF	1	Hex
AF	0	Hex
PF	1	Hex
CF	0	Hex

Control Registers:

Register	Value	Format
CR0	60000011	Hex
CR1	00000000	Hex
CR2	00000000	Hex
CR3	00000000	Hex
CR4	00000000	Hex

Assembly Code (Main Editor):

```
1 // public void drawLabel(Graphics g, Label l)
2 // {
3   com.sun.lwuit.plaf.DefaultLookAndFeel.drawLabel:
4   00021459 55          push     ebp
5   0002145A 89 E5       mov     ebp, esp
6   0002145C 53          push     ebx
7   // drawComponent(g, l, l.getIcon(), null, 0):
8   0002145D 8B 45 10    mov     eax, [ebp + 16]
9   00021460 50          push     eax
10  00021461 8B 00       mov     eax, [eax]
11  00021463 FF 90 E4 03 00 00 call    [eax + 996]
12  00021469 59          pop      ecx
13  0002146A 89 C3       mov     ebx, eax
14  0002146C 6A 00       push    0x0
15  0002146E 6A 00       push    0x0
16  00021470 53          push     ebx
17  00021471 FF 75 10    push    [ebp + 16]
18  00021474 FF 75 0C    push    [ebp + 12]
19  00021477 FF 75 08    push    [ebp + 8]
20  0002147A E8 2A 1C 00 00 call    com_sun_lwuit_plaf_DefaultLookAnd
21  0002147F 83 C4 18    addl    esp, 0x18
22  // }
23  00021482 5B          pop      ebx
24  00021483 5D          pop      ebp
25  00021484 C3          ret
26 //
27 // /**
28 //  * @inheritDoc
29 //  */
30 // public void drawRadioButton(Graphics g, Button rb)
31 // {
32   com.sun.lwuit.plaf.DefaultLookAndFeel.drawRadioButton:
33   00021485 C8 08 00 00 enter    0x8, 0x0
34   00021489 53          push     ebx
35   0002148A 56          push     esi
36   0002148B 57          push     edi
37   // if (rButtonImages != null)
38   0002148C 8B 45 08    mov     eax, [ebp + 8]
39   0002148F 8B 80 90 00 00 00 mov     eax, [eax + 144]
40   00021495 83 F8 00    cmpl    eax, 0x0
41   00021498 0F 84 A2 01 00 00 jz      +418
42   // {
43   // Image x;
44   // if (rButtonImagesFocus != null && rButtonImagesFocus[0] != null && rb
45   0002149E 8B 45 08    mov     eax, [ebp + 8]
46   000214A1 8B 80 98 00 00 00 mov     eax, [eax + 152]
47   000214A7 83 F8 00    cmpl    eax, 0x0
48   000214AA 0F 84 DE 00 00 00 jz      +222
49   000214B0 8B 45 08    mov     eax, [ebp + 8]
50   000214B3 8B 80 98 00 00 00 mov     eax, [eax + 152]
51   000214B9 8B 40 0C    mov     eax, [eax + 12]
52   000214BC 83 F8 00    cmpl    eax, 0x0
53   000214BF 0F 84 C9 00 00 00 jz      +201
54   000214C5 8B 45 10    mov     eax, [ebp + 16]
55   000214C8 50          push     eax
56   000214C9 8B 00       mov     eax, [eax]
```

Runtime Heap Window

Some useful window, which represent runtime Heap.

- Green color – free memory blocks.
- Black color – dynamically allocated Java objects
- Blue color – special allocated regions (for example, processor thread stacks).

If we click on some black region, we can see some information of allocated object (address, size and Java Class of this object)

Runtime Heap Window

MicroJ2ME - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

PC: 0x32888 562.5/728.5MB

Projects Files Services

main.c Test.java port.c TouchPanel.java Heap Display.java Calendar.java Graphics.c Container.java DefaultLookAndFeel.java C...

LWUIT

- Header Files
 - com
 - device
 - drivers
 - java
 - X86
- Important Files
- Java Files
 - Source Packages
 - <default package>
 - com.sun.lwuit
 - com.sun.lwuit.animations
 - com.sun.lwuit.events
 - com.sun.lwuit.geom
 - com.sun.lwuit.impl
 - com.sun.lwuit.layouts
 - com.sun.lwuit.list
 - com.sun.lwuit.plaf
 - Border.java
 - DefaultLookAndFeel.java
 - LookAndFeel.java
 - Style.java
 - UIManager.java
 - com.sun.lwuit.spinner
 - com.sun.lwuit.table
 - com.sun.lwuit.tree
 - com.sun.lwuit.util
 - device
 - java.io
 - java.lang
 - java.text
 - java.util
 - javax.com
 - Libraries
 - Linker Files
 - Source Files
 - Libraries
 - Loadables

Navigator

Members

- numColors() : int
- onEditingComplete(Component c, String text)
- playBuiltinSound(String soundIdentifier)
- playDialogSound(int type)
- pointerDragged(int[] x, int[] y)
- pointerHover(int[] x, int[] y)
- pointerHoverPressed(int[] x, int[] y)
- pointerHoverReleased(int[] x, int[] y)

Runtime Heap Window

Total bytes : 71680

Free bytes : 60860

Allocated objects : 212

Selected object :

Class : java.lang.String

Address : 0x201610

Size : 24

Debugging Console

LWUIT (Clean, Build) | LWUIT (Java Build, Java Compile, ...) | Debugge...

Launching

User program running

User program stopped

User program running

User program stopped

Runtime Status

Another useful information about runtime status. We can see how many object of which Java class are allocated in this execution point.

Runtime Status

NetBeans IDE 8.0.2 - LWUIT - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

default

PC: 0x32825

495.7/670.5 MB

Projects Files Services

types.h

X86

Important Files

Java Files

Source Packages

<default package>

com.sun.lwuit

com.sun.lwuit.animations

com.sun.lwuit.events

com.sun.lwuit.geom

com.sun.lwuit.impl

com.sun.lwuit.layouts

com.sun.lwuit.list

com.sun.lwuit.plaf

Border.java

DefaultLookAndFeel.java

LookAndFeel.java

Style.java

UIManager.java

com.sun.lwuit.spinner

com.sun.lwuit.table

com.sun.lwuit.tree

com.sun.lwuit.util

device

java.io

java.lang

java.text

java.util

javax.com

Libraries

Linker Files

Source Files

com

device

drivers

java

lang

except.c

gc.c

native.c

string.c

support.c

Thread.c

main() - Navigator

Limited code assistance (no associated project)

CINUSE_BIT

CINUSE_BIT_NUM

CLEAR_CINUSE

CLEAR_PINUSE

CLEAR_SIZE

CODE_SIZE

CODE_START

COLOR_BIT0

COLOR_BIT0_NUM

COLOR_BIT1

Class Name

Instances [%]

Instances

char[]		53 (25%)
java.lang.StringBuffer		26 (12.3%)
java.lang.String		26 (12.3%)
java.lang.Integer		26 (12.3%)
java.lang.Object[]		25 (11.8%)
int[]		13 (6.1%)
java.util.Hashtable		6 (2.8%)
java.lang.String[]		5 (2.4%)
com.sun.lwuit.plaf.Style		4 (1.9%)
java.lang.Byte		4 (1.9%)
java.util.Vector		3 (1.4%)
java.lang.Object		3 (1.4%)
com.sun.lwuit.plaf.Border		3 (1.4%)
java.lang.Boolean		2 (0.9%)
com.sun.lwuit.Graphics		2 (0.9%)
java.util.Date		2 (0.9%)
device.TouchPanel		1 (0.5%)
com.sun.lwuit.Display		1 (0.5%)
com.sun.lwuit.plaf.UIManager		1 (0.5%)
com.sun.lwuit.plaf.DefaultLookAndFeel		1 (0.5%)
com.sun.lwuit.Display.LWUITImplementation		1 (0.5%)
java.util.CalendarImpl		1 (0.5%)
boolean[]		1 (0.5%)
java.util.SimpleTimeZone		1 (0.5%)
com.sun.lwuit.Calendar		1 (0.5%)
device.DMI		0 (0%)
com.sun.lwuit.plaf.LookAndFeel		0 (0%)
java.lang.Number		0 (0%)
java.util.Calendar		0 (0%)
java.util.TimeZone		0 (0%)
com.sun.lwuit.Container		0 (0%)
com.sun.lwuit.Component		0 (0%)

Debugger Console

LWUIT (Java Build, Java Compile, ...)

Launching

User program running

User program stopped

User program running

User program stopped

[Class Name Filter]

Variables and Watches Windows

Variables and Watches Windows contain nothing special. System will show variable according with current execution point. If we stopped on Java source, all variables and watches will be Java types and classes. If we stopped on C source, there will be native C structures or native counterparts to Java classes.

Variables and Watches Windows

NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

default PC: 0x3283E 422.0/754.5MB

Projects Files Services

main.c Test.java port.c TouchPanel.java Display.java Calendar.java Container.java DefaultLookAndFeel...

```
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
```

```

};
private static final String[] DAYS =
{
    "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
};
private static final String[] LABELS =
{
    "Su", "M", "Tu", "W", "Th", "F", "Sa"
};
static final long MINUTE = 1000 * 60;
static final long HOUR = MINUTE * 60;
static final long DAY = HOUR * 24;
static final long WEEK = DAY * 7;
private EventDispatcher dispatcher = new EventDispatcher();
private EventDispatcher dataChangeListeners = new EventDispatcher();
private long[] dates = new long[42];
private boolean changesSelectedDateEnabled = true;
private TimeZone tmz;

/**
 * Creates a new instance of Calendar set to the given date based on time
 * since epoch (the java.util.Date convention)
 *
 * @param time time since epoch
 */
public Calendar(long time)
{
    this(time, java.util.TimeZone.getDefault());
}

/**
 * Constructs a calendar with the current date and time
 */
public Calendar()
{
    this(System.currentTimeMillis());
}

/**
 * Creates a new instance of Calendar set to the given date based on time
 * since epoch (the java.util.Date convention)
 *
 * @param time time since epoch
 * @param tmz a reference timezone
 */
public Calendar(long time, TimeZone tmz)
{
    super(new BorderLayout());
    this.tmz = tmz;
    setUIID("Calendar");
    Container upper = new Container(new FlowLayout(Component.CENTER));
    month = new ComboBox();
    year = new ComboBox();
    mv = new MonthView(time);

```

Calendar - Navigator

Members

Calendar :: Container

- Calendar(long time, TimeZone tmz)
- Calendar()
- Calendar(long time)
- addActionListener(ActionListener l)
- addDataChangeListener(DataChangeListener)
- componentChanged()
- createDay(): Button

Variables

Name	Type	Value
this	com.sun.lwuit.Calendar	
selectText	java.lang.String	"Select"
value	char[]	(length = 16571)
offset	int	12628
count	int	6
alwaysTensile	boolean	false
tensileLength	int	-1
nextFocusDown	com.sun.lwuit.Component	null
nextFocusUp	com.sun.lwuit.Component	null
enabled	boolean	true
nextFocusRight	com.sun.lwuit.Component	null
nextFocusLeft	com.sun.lwuit.Component	null
name	java.lang.String	null
hasLead	boolean	false
grabsPointerEvents	boolean	false
tensileDragEnabled	boolean	true
tensileHighlightEnabled	boolean	false
tensileHighlightIntensity	int	0
tactileTouch	boolean	true
visible	boolean	true
cellRenderer	boolean	false
bounds	com.sun.lwuit.geom.Rectangle	
painterBounds	com.sun.lwuit.geom.Rectangle	null
scrollX	int	0
scrollY	int	0
sizeRequestedByUser	boolean	false
preferredSize	com.sun.lwuit.geom.Dimension	null
scrollSizeRequestedByUser	boolean	false
scrollSize	com.sun.lwuit.geom.Dimension	null
unSelectedStyle	com.sun.lwuit.plaf.Style	null
pressedStyle	com.sun.lwuit.plaf.Style	null
selectedStyle	com.sun.lwuit.plaf.Style	null
disabledStyle	com.sun.lwuit.plaf.Style	null
parent	com.sun.lwuit.Container	null
focused	boolean	false
focusListeners	com.sun.lwuit.util.EventDispatcher	
handlesInput	boolean	false
shouldCalcPreferredSize	boolean	true
shouldCalcScrollSize	boolean	true
focusable	boolean	false
isScrollVisible	boolean	true
repaintPending	boolean	false
snapToGrid	boolean	false
hideInPortrait	boolean	false
scrollOpacity	int	255
scrollOpacityChangeSpeed	int	5

Native Methods Debugging

We stopped on Java native method implementation. Call Stack Window reflects this situation: it shows Java methods and also native methods and C functions.

Native Methods Debugging

Before entering the native method

The screenshot displays the NetBeans IDE 8.0.2 interface. The main editor window shows the `DefaultLookAndFeel.java` file, specifically the `drawComboBox` method. The code is as follows:

```
587 g.setColor(style.getBgColor());
588 int y = cb.getY();
589 int height = cb.getHeight();
590 int width = comboImageWidth + border;
591 int x = cb.getX();
592 if (cb.isRTL())
593 {
594     x += leftPadding;
595 }
596 else
597 {
598     x += cb.getWidth() - comboImageWidth - rightPadding;
599 }
600
601 if (comboImage != null)
602 {
603     g.drawImage(comboImage, x, y + height / 2 - comboImage.getHeight() / 2);
604 }
605 else
606 {
607     int color = g.getColor();
608     // brighten or darken the color slightly
609     int destColor = findDestColor(color);
610
611     g.fillLinearGradient(g.getColor(), destColor, x, y, width, height, false);
612
613     g.setColor(color);
614     g.drawRect(x, y, width, height - 1);
615
616     width--;
617     height--;
618 }
619
```

The `g.fillLinearGradient` line (611) is highlighted in green. The `Variables` window at the bottom shows the current state of the program:

Name	Type	Value
this	com.sun.lwuit.plaf.DefaultLookAndFeel	
g	com.sun.lwuit.Graphics	
image	com.sun.lwuit.Image	null
currentFont	com.sun.lwuit.Font	
transX	int	36
transY	int	52
clipX1	int	41
clipY1	int	57
clipX2	int	31
clipY2	int	35
translatedX1	int	41
translatedY1	int	57
translatedX2	int	106
translatedY2	int	92
clip_X	int	41
clip_Y	int	57

The `Navigator` window on the left shows the project structure, including the `DefaultLookAndFeel` class. The `Output` window on the right shows the text "Calendar" and "Nov".

Native Methods Debugging

After entering the native method

The screenshot displays the NetBeans IDE interface for LWUIT 8.0.2. The main editor shows the native method `com_sun_lwuit_Graphics_fillLinearGradient` in `Graphics.c`. The code is as follows:

```
1352 com_sun_lwuit_Graphics_setColor_I(this_g, oldColor);
1353 this_g->antiAliased = aa;
1354 }
1355
1356 void com_sun_lwuit_Graphics_fillLinearGradient(struct Hcom_sun_lwuit_Graphics* this_g,
1357 jint startColor, jint endColor, jint x, jint y, jint width, jint height,
1358 jboolean horizontal)
1359 {
1360     int sourceR = startColor >> 16 & 0xff;
1361     int sourceG = startColor >> 8 & 0xff;
1362     int sourceB = startColor & 0xff;
1363     int destR = endColor >> 16 & 0xff;
1364     int destG = endColor >> 8 & 0xff;
1365     int destB = endColor & 0xff;
1366     int oldColor = this_g->rgbColor;
1367     int iter;
1368     if (horizontal)
1369     {
1370         for (iter = 0; iter < width; iter++)
1371         {
1372             updateGradientColor(this_g, sourceR, sourceG, sourceB, destR,
1373                                 destG, destB, width, iter);
1374             com_sun_lwuit_Graphics_drawLine(this_g, x + iter, y, x + iter, y + height);
1375         }
1376     }
1377     else
1378     {
1379         for (iter = 0; iter < height; iter++)
1380         {
1381             updateGradientColor(this_g, sourceR, sourceG, sourceB, destR,
1382                                 destG, destB, height, iter);
1383             com_sun_lwuit_Graphics_drawLine(this_g, x, y + iter, x + width, y + iter);
1384         }
1385     }
1386 }
```

The Variables window shows the following data:

Name	Type	Value
this_g	struct Hcom_sun_lwuit_Graphics*	0x202594
startColor	jint	6734
endColor	jint	13468
x	jint	48
y	jint	5
width	jint	21
height	jint	35
horizontal	jboolean	0
sourceR	int	0
sourceG	int	26
sourceB	int	78
destR	int	0
destG	int	52
destB	int	156
oldColor	int	6734
iter	int	4193336

The right-hand side of the IDE shows a visual representation of the application. It features a blue background with a black header bar containing the text "Calendar". Below the header, a dark blue rectangle displays the text "Nov".

Native Methods Debugging

Then we can open original Java class source from which its native method was implemented and see some methods in current calling sequence.

Two upper methods in calling stack are native methods, which are implemented on C and other methods are Java methods.

Native Methods Debugging

