

诺基亚 JAVA 用户界面 API 声音范例：曲调

中文版本 1.0

2003 年 1 月 6 日

目录

| | | |
|-----|----------------------------|----|
| 1 | 简介 | 4 |
| 1.1 | 目的 | 4 |
| 1.2 | 参考文献 | 4 |
| 2 | TonesMIDlet..... | 5 |
| 2.1 | 用户界面 | 5 |
| 2.2 | 设计 | 6 |
| 2.3 | TonesMIDlet.java..... | 7 |
| 2.4 | TonesCanvas.java..... | 8 |
| 2.5 | TonePlayer.java..... | 13 |
| 2.6 | NokiaTonePlayer.java | 14 |

修订纪录

| | | |
|-------------|----------|------------|
| 2002年3月13日 | 版本 0.9.3 | 文档在诺基亚论坛发布 |
| 2002年11月18日 | 版本 1.0 | 文档更新 |
| 2003年1月6日 | 中文版本 1.0 | |

诺基亚用户界面 API 声音范例：曲调
中文版本 1.0

声明：

本文档中的信息基于其现有状况，不存在任何保证，包括销售保证、适用某一特殊用途的保证，或从任何建议、规范或范例中衍生出来的保证。此外，本文档中提供的并非最终信息，在其最终发布前会做较大改动。本文档仅用作信息通报。

诺基亚公司不承担所有因实施本文档中所表述的信息而产生的相关责任，包括侵犯任何知识产权的责任。诺基亚公司并不保证或认为使用这些信息不会构成对相应知识产权的侵犯。

诺基亚公司保留不预先通知而随时修改此规范的权力。

本文档中显示的手机用户界面图像仅作演示之用，并不代表任何实际设备。

“诺基亚”和“诺基亚以人为本”是诺基亚公司的注册商标。

Java 和所有基于 Java 的标志是 Sun 微系统有限公司的商标或注册商标。

在此提到的其它产品和公司名称可能是其所有者的商标或商业名称。

授权许可：

仅授予下载或打印本规范作个人用途的权力。除此之外，不存在对其它任何知识产权的授权许可。

1 简介

1.1 目的

本文介绍了一个名为“Tones”的 MIDlet 范例，该范例使用了诺基亚 Java 用户界面 API 中的声音特性，使用户们能在其电话的键盘上演奏曲调。可以从诺基亚论坛网站 (<http://www.forum.nokia.com>) 上下载诺基亚用户界面 API 的 JavaDoc 文档。

本文假设您熟悉 Java 编程，同时也对基本的 MIDP 编程有所了解，比如，您已经读过诺基亚论坛中的文章《MIDP 编程简介》 [MIDPPROG]。

读完这篇文章以后，你将更好地理解开发有声 MIDlets 的过程，其中包括：

- 用诺基亚用户界面 API 演奏简单的曲调
- 开发能支持一个或多个供应商 API 的 MIDlet，即使那些 API 不可用却仍可运行该 MIDlet

1.2 参考文献

MIDPPROG MIDP 编程简介

诺基亚论坛，2002 年
<http://www.forum.nokia.com>

2 TONESMIDLET

2.1 用户界面

该 MIDlet 只有一个屏幕：



图 1：TonesMIDlet 的用户界面

此屏幕显示了一个漂亮的“音高谱号”和五线谱（水平线）。当用户按下‘#’号以外的某个数字键时就会奏出音调声，并持续到这个键被释放为止。按‘#’号键表示高音调（上图屏幕显示的是正在演奏高音曲调）。注意，音调‘B’和‘E’并不能被提升到高音（这对应于钢琴键盘上黑色键的间隙）。

“Exit”命令终止该 MIDlet 的执行。（在上面的模拟器中，仅当您按下某一个功能键时，才显示出各种命令）。

2.2 设计

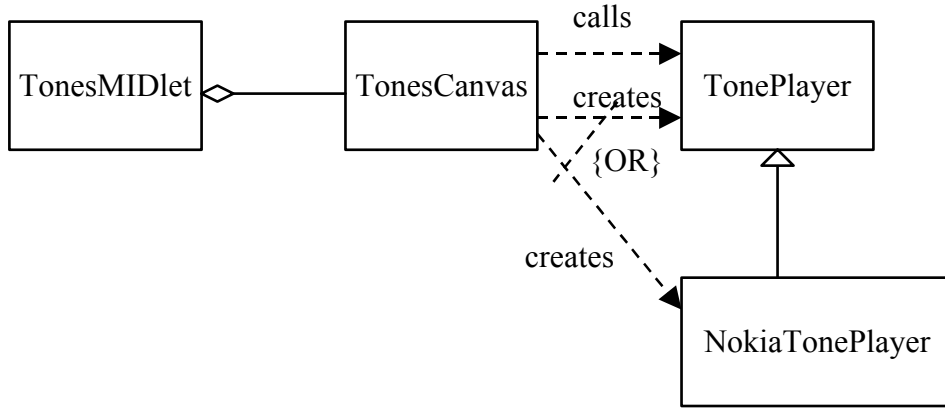


图 2：TonesMIDlet 的类图

TonesMIDlet 创建一个 TonesCanvas，它提供了 MIDlet 的各种用户界面。

根据手机是否支持诺基亚 Java 用户界面 API，TonesCanvas 或者创建 NokiaTonePlayer，或者创建 TonePlayer，以实现曲调的演奏。之后，演奏各种曲调通过调用 TonePlayer 类中定义的各种方法，这些方法在 NokiaTonePlayer 类中被重载。（TonePlayer 类中被重载的相关方法实际上不作任何操作）。

2.3 TonesMIDlet.java

TonesMIDlet 处理 MIDlet 应用框架中状态模型的回调，而 TonesCanvas 负责用户界面的管理。

```
package example.tones;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class TonesMIDlet
    extends MIDlet
{
    private final TonesCanvas canvas;

    public TonesMIDlet()
    {
        canvas = new TonesCanvas(this);
    }

    public void startApp()
    {
        Display.getDisplay(this).setCurrent(canvas);
    }

    public void pauseApp()
    {
        canvas.stopNote();
    }

    public void destroyApp(boolean unconditional)
    {
        canvas.stopNote();
    }

    void exitRequested()
    {
        destroyApp(false);
        notifyDestroyed();
    }
}
```

2.4 TonesCanvas.java

这个类为 TonesMIDlet 提供了用户界面。它绘制如图 1 所示的用户界面。电话键盘上的 1, 2, 3, 4, 5, 6, 7, 8, 9, * 和 0 键分别代表 C, D, E, F, G, A, B, C, D, E 和 F 音调。(第一个音调被认为是中音 C, 虽然实际上它是一个高八度音, 这是因为诺基亚 Java 用户界面 API 并不保证支持象中音 C 那样低的音调)。电话键#决定演奏的是自然音调还是高音调。如果演奏的是高音调, 演奏时在它们之后会显示出高音符号。音调 E 和 B 用于演奏自然音调, 即使正在演奏的其他音调为高音。

最后一个方法 makeTonePlayer 是一个类工厂方法, 它根据移动电话是否支持诺基亚 Java 用户界面 API 而创建一个 NokiaTonePlayer 实例, 或 TonePlayer 实例。用此方法可以支持其他供应商的各种声音 API。所以这个 MIDlet 能在许多不同厂商的电话上创作声音。

```
package example.tones;

import javax.microedition.lcdui.*;

class TonesCanvas
    extends Canvas
    implements CommandListener
{
    // Constants and frequencies for tone playing
    private static final int SILENT = -1; // rogue value for table index
    private static int frequencies[][] =
    {
        { 523, 554},           // C, C#
        { 587, 622},           // D, D#
        { 659, 659},           // E, E (there is no E#)
        { 698, 740},           // F, F#
        { 784, 831},           // G, G#
        { 880, 932},           // A, A#
        { 988, 988},           // B, B (there is no B#)
        {1047, 1109},          // C, C#
        {1175, 1245},          // D, D#
        {1319, 1319},          // E, E (there is no E#)
        {1397, 1480}           // F, F#
    };
    private static final int NATURAL = 0;
    private static final int SHARP = 1;

    // Constants for screen layout. Top of clef is highest point; bottom
    // of middle C note is lowest point. Clef will be drawn at screen left,
    // note at screen centre, possible sharp to left of note
    private static final int CLEF_G_LINE_Y = 30; // clef curls around 'G' line
    private static final int NOTE_WIDTH = 7;
    private static final int NOTE_HEIGHT = 7;
    private static final int SHARP_WIDTH = 5;
    private static final int LINE_SPACING = 6;
    private static final int TOP_LINE_Y = CLEF_G_LINE_Y - 3 * LINE_SPACING;
    private static final int MIDDLE_C_Y = TOP_LINE_Y + 5 * LINE_SPACING;
```

诺基亚用户界面 API 声音范例：曲调

中文版本 1.0

```
private static final int LAYOUT_HEIGHT = MIDDLE_C_Y + NOTE_HEIGHT / 2 + 1;
private static final int SHARP_OFFSET = SHARP_WIDTH + 2; // 2 for spacing
private static final int LEGER_LINE_WIDTH = NOTE_WIDTH + 4;

private final TonesMIDlet parent;
private final Image trebleImg;
private final Image noteImg;
private final Image sharpImg;
private final Command exitCommand;
private final TonePlayer tonePlayer;

private int note = SILENT;
private int currentKey;
private boolean sharp = false;

TonesCanvas(TonesMIDlet parent)
{
    this.parent = parent;
    trebleImg = createImage("/treble.png");
    noteImg = createImage("/note.png");
    sharpImg = createImage("/sharp.png");

    // create a NokiaTonePlayer if Nokia UI API is available,
    // otherwise a dummy TonePlayer
    tonePlayer = makeTonePlayer();

    exitCommand = new Command("Exit", Command.EXIT, 1);
    addCommand(exitCommand);

    setCommandListener(this);
}

public void paint(Graphics g)
{
    int width = getWidth();
    int height = getHeight();
    g.setColor(0x00FFFFFF); // white
    g.fillRect(0, 0, width, height);

    int topOffset = (getHeight() - LAYOUT_HEIGHT) / 2;

    // draw treble clef, 1 pixel in from left edge
    g.drawImage(trebleImg, 1, topOffset, Graphics.TOP | Graphics.LEFT);

    if (note != SILENT)
    {
        // draw note
        drawNote(g, topOffset, width);
    }

    drawLines(g, topOffset, width);
}

// draw the five horizontal lines of the music score
private void drawLines(Graphics g, int topOffset, int width)
```

诺基亚用户界面 API 声音范例：曲调

中文版本 1.0

```
{
    g.setColor(0x00000000); // black
    for (int i = 0; i < 5; ++i)
    {
        int y = topOffset + TOP_LINE_Y + LINE_SPACING * i;
        g.drawLine(0, y, width-1, y);
    }
}

private void drawNote(Graphics g, int offset, int width)
{
    int y = offset + MIDDLE_C_Y - note * LINE_SPACING / 2;
    int x = width / 2;
    g.drawImage(noteImg, x, y, Graphics.HCENTER | Graphics.VCENTER);

    // this next bit should be made more smart if it ever
    // needs to be more general
    if (note == 0) // C should have a 'leger line' through it
    {
        g.setColor(0x00000000); // black
        g.drawLine(x-LEGER_LINE_WIDTH/2, y, x+LEGER_LINE_WIDTH/2, y);
    }

    // is sharp set, and can this note be sharp (E & B can't)
    if (sharp &&
        (frequencies[note][NATURAL] != frequencies[note][SHARP]))
    {
        g.drawImage(sharpImg, x - SHARP_OFFSET, y,
            Graphics.HCENTER | Graphics.VCENTER);
    }
}

public void commandAction(Command c, Displayable d)
{
    if (c == exitCommand)
    {
        parent.exitRequested();
    }
}

public void keyPressed(int keyCode)
{
    if (keyCode == Canvas.KEY_POUND)
    {
        sharp = !sharp; // toggle sharp state
    }
    else
    {
        switch (keyCode)
        {
            case Canvas.KEY_NUM1:
                note = 0; // middle C
                break;
            case Canvas.KEY_NUM2:
                note = 1; // D
        }
    }
}
```

```
        break;
    case Canvas.KEY_NUM3:
        note = 2; // E
        break;
    case Canvas.KEY_NUM4:
        note = 3; // F
        break;
    case Canvas.KEY_NUM5:
        note = 4; // G
        break;
    case Canvas.KEY_NUM6:
        note = 5; // A
        break;
    case Canvas.KEY_NUM7:
        note = 6; // B
        break;
    case Canvas.KEY_NUM8:
        note = 7; // C
        break;
    case Canvas.KEY_NUM9:
        note = 8; // D
        break;
    case Canvas.KEY_STAR:
        note = 9; // E
        break;
    case Canvas.KEY_NUM0:
        note = 10; // F
        break;
    default:
        note = SILENT;
        break;
    }

    if (note != SILENT)
    {
        startNote();
    }
    currentKey = keyCode;
}
repaint();
}

public void keyReleased(int keyCode)
{
    // This check deals with the possibility that we have this sequence:
    // key A pressed; ...; key B pressed; key A released
    // The user will expect that the tone for key B continues playing.
    if (keyCode == currentKey)
    {
        note = SILENT;
        stopNote();
    }
    repaint();
}

private void startNote()
```

诺基亚用户界面 API 声音范例：曲调

中文版本 1.0

```
{
    tonePlayer.play(frequencies[note][sharp ? SHARP : NATURAL]);
}

void stopNote()
{
    tonePlayer.stop();
}

private static Image createImage(String filename)
{
    Image image = null;
    try
    {
        image = Image.createImage(filename);
    }
    catch (java.io.IOException e)
    {
        // just let return value be null
    }
    return image;
}

private static TonePlayer makeTonePlayer()
{
    TonePlayer player;

    try
    {
        // This statement throws an exception if no Nokia UI API available
        Class.forName("com.nokia.mid.sound.Sound");
        // If we get here, Nokia UI API is available, so we can safely
        // create a player that uses it. But we use Class.forName rather
        // than 'new' so that there is no link dependency.
        Class clas = Class.forName("example.tones.NokiaTonePlayer");
        player = (TonePlayer) (clas.newInstance());
    }
    catch (Exception e)
    {
        // If no Nokia UI API, then create a dummy tone player
        player = new TonePlayer();
    }

    return player;
}
}
```

2.5 TonePlayer.java

这个类为 MIDlet 演奏曲调提供了一个接口和空实现。如使用由供应商提供的专用 API 的声音特性，必须继承这个类并重载其方法。请参阅§2.6有关使用诺基亚专用声音 API 的范例。

```
package example.tones;

// This is a dummy TonePlayer class which does nothing. Implementations
// for manufacturer APIs should extend this class and override play & stop.
class TonePlayer
{
    TonePlayer()
    {
    }

    void play(int frequency)
    {
    }

    void stop()
    {
    }
}
```

2.6 NokiaTonePlayer.java

这个类使用诺基亚 Java 用户界面 API 的声音特性实现曲调的演奏。当调用相关方法去演奏一个曲调时，它先启动一个两秒钟的曲调，对用户正在演奏的任何曲调来说，这应该足够长了。当调用相关方法去停止演奏曲调时，正在演奏的曲调会被立刻中断。

```
package example.tones;

import com.nokia.mid.sound.*;

// This is an extension of TonePlayer which implements its methods
// using Nokia's UI API.
class NokiaTonePlayer
    extends TonePlayer
{
    private final Sound sound;

    NokiaTonePlayer()
    {
        // We have to provide default values for Sound, though these will not
        // be used as we will call 'init' before using it. Make it silent
        // (frequency = 0) and duration 1ms.
        sound = new Sound(0, 1L);
    }

    // Start playing a tone at given frequency in Hz
    void play(int frequency)
    {
        // Tone will stop after two seconds (2000ms), but that's unlikely
        // to be a problem when people are trying to play music
        sound.init(frequency, 2000L);
        sound.play(1);
    }

    // Stop the currently playing tone (or do nothing, if no tone is playing)
    void stop()
    {
        sound.stop();
    }
}
```