**FLÁVIO AUGUSTO SIODONI XIMENES**

# DESENVOLVIMENTO DE UM GERADOR DE GRÁFICOS DE BARRA E PIZZA PARA CELULAR: NANOCHART

**BATATAIS**
**2010**

**FLÁVIO AUGUSTO SIODONI XIMENES**

# DESENVOLVIMENTO DE UM GERADOR DE GRÁFICOS DE BARRA E PIZZA PARA CELULAR: NANOCHART

**Monografia apresentada ao Centro Universitário Claretiano como parte dos requisitos para obtenção do título de Especialista em Desenvolvimento de Projetos em Java com Banco de Dados.**

**Orientador: Prof. Dr. Rodrigo Plotze**

**BATATAIS**
**2010**

**FLÁVIO AUGUSTO SIODONI XIMENES**

**Monografia apresentada ao Centro Universitário Claretiano como parte dos requisitos para obtenção do título de Especialista em Desenvolvimento de Projetos em Java com Banco de Dados.**
**Orientador: Prof. Dr. Rodrigo Plotze**

# DESENVOLVIMENTO DE UM GERADOR DE GRÁFICOS DE BARRA E PIZZA PARA CELULAR: NANOCHART

**Orientador(a): Prof. Dr. Rodrigo Plotze**

**Examinador(a):**

_____

**Examinador(a):**

_____

**Batatais, 17 de Abril de 2010.**

*Dedico este trabalho à minha família, amigos e em especial a minha companheira Tawana, por terem entendido minha ausência durante o período de dedicação a esse trabalho.*

**Resumo**

Baseado em uma acelerada expansão no mercado de aparelhos celulares, espera-se que as empresas mostrem interesse em desenvolvimento de aplicações móveis, sendo assim a proposta desse trabalho é oferecer ao usuário desenvolvedor, uma biblioteca responsável em fornecer um gráfico de barras ou pizza com valores positivos, chamado de NanoChart, para que o mesmo possa utilizar em suas aplicações móveis. Dentre as várias tecnologias atualmente disponíveis no mercado para o desenvolvimento de aplicações móveis, a plataforma Java Micro Edition (J2ME) foi escolhida para esse trabalho, pois a mesma foi desenvolvida para a utilização em aparelhos com poucos recursos de processamento, vídeo e memória, sendo muito frequente seu uso em aparelhos celulares. A linguagem Java não fornece em suas bibliotecas nativas ferramentas para a geração de gráficos de forma simples e direta, para isso foram utilizados na construção da interface os métodos da classe Canvas e Graphics, que possibilitaram o desenvolvimento da biblioteca. A mesma possui leiaute dividido em abas, onde a primeira aba apresenta o gráfico e a segunda apresenta as informações de legenda; foi também possível a implementação de suporte a telas sensíveis ao toque para a navegação das abas, além do suporte à navegação pelo teclado. O NanoChart ainda fornece mais dois recursos, sendo o primeiro deles a interface Cor que oferece ao usuário desenvolvedor códigos hexadecimais das cores no padrão RGB (red, green, blue) para que o mesmo os utilize em suas aplicações; o segundo recurso é uma validação de três itens, sendo eles: a existência de valores negativos informados ao gráfico, a comparação entre a igualdade da quantia de valores e a quantia de legendas passadas ao gráfico e a existência de mais de dez valores fornecidos ao gráfico. O NanoChart é uma biblioteca com código fonte aberto, poderá ser modificada por qualquer pessoa que tenha interesse em melhorar ou expandir esse projeto.

Palavras-chave: Java. Desenvolvimento. Gráfico. Aplicação Móvel. Telefone Celular.

**Abstract**

Based on a rapid expansion in handset market, is expected that companies show interest in developing mobile applications, thus the purpose of this study is to provide the user developer, a library meant to provide a bar graph or pie with positive values, called NanoChart, so that it can use in their mobile applications. Among the various technologies currently available in the market for the development of mobile applications, Java Micro Edition (J2ME) platform was chosen for this work, because it has been developed for use in devices with limited processing resources, video and memory, being very frequent use in cell phones. The Java language does not provide in their native libraries, tools for generating graphics in a simple mode and direct, for that were used in the construction of the interface methods of the Canvas and Graphics, which enabled the development of library. The same layout has divided into tabs, where the first tab displays the graph and the second presents the caption information; was also possible to implement support for touch screens for navigation tabs, well as support for keyboard navigation. The NanoChart also provides two additional resources, being the first of the interface Cor which provides the developer user hexadecimal codes of colors in the standard RGB (red, green, blue) for the same use them in your applications; the second feature is a validation of three items, and they: the existence of negative values reported to the chart, comparison between the amount of equal value and the amount of legends reported to the chart and the existence of more than ten figures provided to the chart. The NanoChart is a library with open source, may be modified by anyone who is interested in upgrading or expanding this project.

Keywords: Java. Development. Chart. Mobile Application. Cell Phone.

# ÍNDICE DE FIGURAS

# SUMÁRIO

# 1 – INTRODUÇÃO

Este trabalho tem como principal objetivo o desenvolvimento de uma biblioteca geradora de gráficos para dispositivos móveis. Com base em uma expansão do mercado de aparelhos celulares é esperado que as empresas demonstrem interesse em desenvolvimento móvel.

A proposta principal é entregar ao desenvolvedor um gráfico de barras ou pizza com valores positivos para que o mesmo o utilize em suas aplicações móveis.

Para isso será utilizada a linguagem de programação Java para o desenvolvimento do projeto, a escolha dessa tecnologia é justificada pelo grande suporte da linguagem na construção de aplicações móveis, principalmente por conta dos emuladores e das bibliotecas presentes no Java Micro Edition.

## 1.1 – Justificativa

A cada dia a venda de aparelhos celulares aumenta, e cada vez mais são exigidos novos recursos para esses dispositivos. Segundo o site G1 apud ANATEL“[...] o número de linhas celulares ativas em julho no Brasil cresceu 1,45 por cento ante junho, superando a barreira dos 160 milhões segundo dados divulgados pela Agência Nacional de Telecomunicações[...]”.

Tomando como base um cenário em constante expansão, torna-se viável o desenvolvimento da biblioteca para elaboração de gráficos, pois com o constante crescimento, a exigência por novas tecnologias se torna uma constante e o desenvolvimento de aplicações móveis nas empresas se torna cada vez mais comum.

Gerar gráficos de barra e pizzas em aplicações móveis é uma tarefa difícil, pois, a linguagem Java não fornece nenhuma classe que faça isso nativamente. A biblioteca NanoChart surge para suprir essa dificuldade do desenvolvedor.

## 1.2 – Objetivos

O objetivo do trabalho é desenvolver uma biblioteca geradora de gráficos de barra e pizza com valores positivos, denominada de NanoChart para uso em aplicações móveis. Essa biblioteca deverá conter métodos para a geração de gráficos de forma rápida e eficiente.

## 1.3 – Organização do trabalho

O trabalho está divido em 3 capítulos principais. O capítulo 2 apresentará as principais tecnologias para desenvolvimento móvel. O capítulo 3 apresentará as particularidades do desenvolvimento de uma interface utilizando a classe Canvas presente no Java ME. O capítulo 4 apresentará como o trabalho está estruturado e organizado, também mostrará um exemplo de utilização da biblioteca desenvolvida.

# 2 – TECNOLOGIAS PARA DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS

No início, os celulares serviam somente para a comunicação de voz, com o passar do tempo esses aparelhos foram se tornando mais poderosos e com mais recursos.

O efeito disso foi o surgimento de várias tecnologias para desenvolvimento, dentre as principais, conhecidas ou mais utilizadas estão:

- SuperWaba;
- Android;
- Symbian;
- Java ME.

## 2.1 – SuperWaba

SuperWaba é uma linguagem de programação que surgiu no início de 2000, influenciada pelo Java e pelo Waba, roda em aparelhos móveis com os sistemas operacionais PalmOs, Windows e SymbianOS, é executada em uma maquina virtual desenvolvida para cada sistema operacional específico, possui sintaxe parecida com a do Java.

Surgiu com o intuito de facilitar o desenvolvimento de aplicações móveis pois "[...] quando Guilherme Campos Hazan (Guich) percebeu que o Waba (linguagem, máquina virtual e SDK para PalmOS e Windows CE) era simples demais para criar uma aplicação financeira [...]" PEREIRA, o mesmo começou a acrescentar novas funcionalidades, dessa forma a linguagem passou a ser distribuída com o nome de SuperWaba, já que as alterações feitas não foram incorporadas na versão oficial do Waba.

## 2.2 – Android

"O Android é um empilhamento de sistemas para dispositivos móveis, que inclui um sistema operacional, middleware e aplicações chave. O SDK Android fornece as ferramentas e APIs necessárias para começar a desenvolver aplicações

na plataforma Android usando a linguagem de programação Java” [1] ANDROID (tradução própria).

Ainda é um sistema pouco encontrado no Brasil, visto que ainda pouquíssimos aparelhos dispõe desse sistema.

## 2.3 – Symbian

Symbian é um sistema operacional que foi desenvolvido por um consórcio de empresas sendo elas: Panasonic, Siemens, Nokia e Sony Ericsson. Atualmente pertence a Nokia, pois a mesma possui quase todas as ações. “SymbianOS é um sistema operacional muito versátil, permite o desenvolvimento de aplicativos em diversas linguagens como: Symbian C/C++, Java ME, FlashLite, Perl, Python, Ruby, Lua e Acelarometer” CIENCIASDACOMPUTACAO.

## 2.4 – Java ME

“Escreva uma vez, rode em qualquer lugar" [2], com essa filosofia em mente a Sun lançava em a plataforma J2SE (Java 2 Standard Edition), em seguida a mesma seria ampliada para conceber aplicações de pequeno e grande porte, foi então que surgiu a plataforma J2ME (Java 2 Micro Edition), desenvolvida especialmente para ser utilizada em aparelhos móveis.

Atualmente existem três plataformas Java para desenvolvimento sendo elas, Standard Edition (J2SE), desenvolvida para ser utilizada em computadores pessoais, Enterprise Edition (J2EE) desenvolvida para ser utilizadas em servidores e Micro Edition (J2ME), desenvolvida para aparelhos com poucos recursos, de processamento, vídeo e memória, podendo ser celulares, pagers, PDA's, televisores e etc. O Java ME está presente em muitos aparelhos celulares, o que torna seu uso para o desenvolvimento desse projeto viável.

“A tecnologia Java ME é baseada em três elementos: uma configuração fornece o conjunto mais básico das bibliotecas e recursos da máquina virtual para

[1] Trecho original: Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

[2] Trecho original: Write Once, Run Anywhere ™

uma ampla gama de dispositivos; um perfil é um conjunto de APIs que suportam uma gama de dispositivos, e um pacote opcional para um conjunto de APIs de tecnologia específica“ [3] SUN (tradução própria).

Com a passar do tempo a tecnologia foi em dividida em duas configurações bases, uma para atender somente dispositivos com poucos recursos e outra para dispositivos mais avançados como smartphones por exemplo. A configuração para os dispositivos mais simples é chamada de CLDC (Connected Limited Device Configuration) e para os com mais capacidade CDC (Connected Device Configuration).

“A figura abaixo representa uma visão geral dos componentes da tecnologia Java ME e como se relaciona com as outras tecnologias Java” [4] SUN (tradução própria).

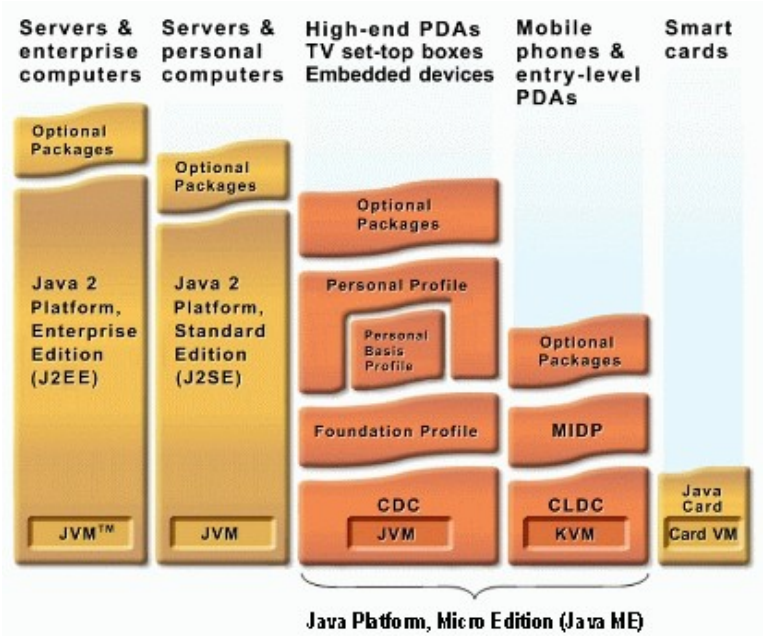


Figura 1 - Plataforma Java (http://java.sun.com/javame/technology/index.jsp)

“Para tratar dessa ampla variação de recursos e para dar mais flexibilidade à medida que a tecnologia muda, a Sun introduziu o conceito de perfil na plataforma J2ME. Um perfil é uma extensão [...] de uma configuração. […] O MIDP (Mobile

[3] Trecho original: The Java ME technology is based on three elements; a configuration provides the most basic set of libraries and virtual machine capabilities for a broad range of devices, a profile is a set of APIs that support a narrower range of devices, and
an optional package is a set of technology-specific APIs.

[4] Trecho original: The figure below represents an overview of the components of Java ME technology and how it relates to the other Java Technologies.

Information Device Profile) [...], define as APIs para componentes, entrada e tratamento de eventos de interface com o usuário, armazenamento persistente, interligação em rede e cronômetros, levando em consideração as limitações de tela e memória dos dispositivos móveis" MUCHOW (2004 p. 4 e 5). A figura 2 mostra a interligação entre o CLDC e o MIDP.

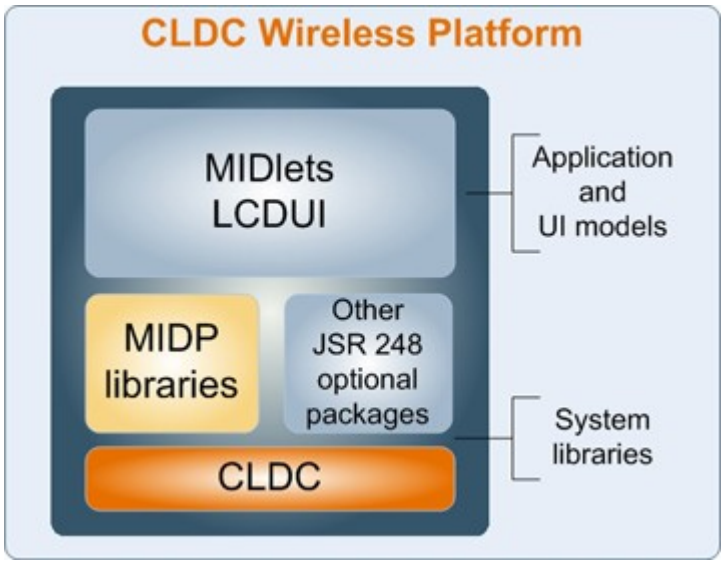


Figura 2 - CLDC e MIDP (http://java.sun.com/javame/technology/index.jsp)

# 3 – DESENVOLVIMENTO DA INTERFACE

Neste capítulo será apresentado os conceitos fundamentais a respeito do desenvolvimento de interfaces, para isso será demonstrado como utilizar a classe Canvas e Graphics nesse desenvolvimento.

Segundo VALENTE (2004 p. 17) apud NORMAN (2002), “interface é aquilo que serve de conexão entre dois modelos, entre duas visões: o modelo mental do usuário em relação ao sistema e o modelo de programa, construído pelos engenheiros de software. Toda interface tem dois lados, um para cada modelo conectado a ela”.

Em uma aplicação móvel espera-se que a interface seja a mais simples e usual possível, pois devido a limitação de recursos a usabilidade torna-se um ponto chave para o sucesso ou o fracasso do produto.

Pois ainda, segundo VALENTE (2004 p. 22) apud PREECE (2002) “os objetivos da experiência do usuário são simples: a experiência deve trazer satisfação; a experiência deve ser agradável; a qualquer momento, o usuário deve ter ajuda para a tarefa que deseja executar; o programa deve motivar o seu uso; o programa deve ser esteticamente agradável; a experiencia de utilização do programa deve se recompensadora”.

Pensando nesses pressupostos, por se tratar do desenvolvimento de uma biblioteca, onde o usuário principal é um programador, alguns pontos ainda devem ser observados: a documentação deve estar completa, os exemplos devem estar adequados a realidade do profissional e a interface deve possuir todos os recursos necessários.

## 3.1 – A Classe Canvas

Para que um pintor possa trabalhar em um quadro, o mesmo necessita de várias ferramentas, dentre elas as principais estão, a tela, o pincel, a tinta, a régua e o compasso.

Partindo dessa analogia podemos pensar na classe Canvas como sendo a tela do pintor, através dela temos acesso ao tamanho da tela do aparelho, e podemos saber a altura e a largura usando os métodos getHeight e getWidth

respectivamente. “A classe Canvas também fornece métodos para tratamento de eventos de baixo nível” MUCHOW (2004 p. 199).

A classe Graphics nos fornece as ferramentas para a execução do trabalho, o pincel tem sua correspondência no método drawLine, a escolha das cores será representado pelo método setColor, o desenho de formas geométricas somente contornadas são representadas pelos métodos drawArc, drawRect, drawRoundRect e para formas preenchidas por cores os métodos, fillArc, fillRect, fillRoundRect e fillTriangle. A seguir serão descritos as particularidades de cada ferramenta.

### 3.2 – Sistema de coordenadas

Segundo KEOGH (2003 p. 215 tradução própria) “a tela é dividida em uma rede virtual em que cada célula representa um pixel [...]. A coordenada x representa a coluna, e a coordenada y representa a célula da linha. A primeira célula localizada no canto superior esquerdo da grade tem a coordenada local de 0, 0, onde o primeiro zero é a coordenada x do zero e outra é a coordenada y” [5].

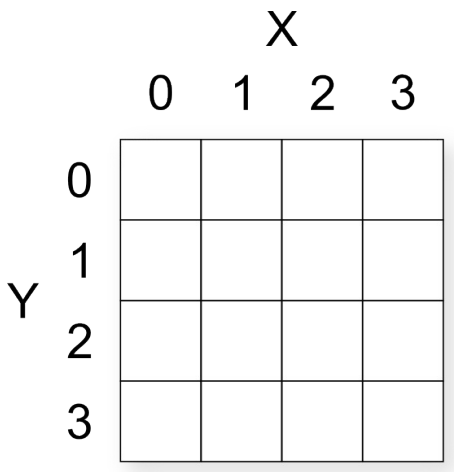


Figura 3 - Sistemas de coordenadas.

O entendimento do sistema de coordenadas é muito importante, pois para a construção de uma interface personalizada é preciso que se saiba por exemplo a

[5] Trecho original: “The canvas is divided into a virtual grid in which each cell represents one pixel [...]. The x coordinate represents the column, and the y coordinate represents the cell’s row. The first cell located in the upper-left corner of the grid has the coordinate location of 0, 0, where the first zero is the x coordinate and the other zero is the y coordinate”.

altura e a largura da tela do aparelho, pois não será possível trabalhar com posições absolutas já que existem diversos tamanhos de telas disponíveis no mercado atual.

### 3.3 – Desenhando linhas

Para um pintor desenhar uma linha em uma tela ele irá precisar de um pincel, no Java ME sua correspondência é o método drawLine, onde será necessário informar a posição inicial e final de x e y. “A espessura de uma linha tem sempre um pixel de largura” MUCHOW (2004 p. 233).

Na construção da interface do NanoChart o método drawLine foi utilizado por exemplo, no método desenhaEixo, esse método é responsável por desenhar as linhas dos eixos X e Y do gráfico de barras.

```
private void desenhaEixo(Graphics g) {
    ...
    //Contornos
    ...
    g.drawLine(inicioLargura, inicioAltura - 1, fimLargura - 3, inicioAltura - 1);
    g.drawLine(fimLargura - 3, inicioAltura, fimLargura - 3, fimAltura);

    //Eixo X
    ...
    g.drawLine(inicioLargura - distCol + 1, fimAltura, fimLargura, fimAltura);
    g.drawLine(inicioLargura - distCol + 1, fimAltura + 1, fimLargura, fimAltura + 1);

    //Eixo Y
    g.drawLine(inicioLargura, inicioAltura - distCol, inicioLargura, fimAltura + distCol);
    g.drawLine(inicioLargura+1, inicioAltura-distCol, inicioLargura+1, fimAltura + distCol);
    ...
}
```

Figura 4 - Parte do método desenhaEixo.

### 3.4 – Desenhando polígonos

Desenhar polígonos exige uma certa habilidade, pois será necessário uma régua, ou um compasso dependendo do caso, no Java ME essa tarefa é facilitada pelos métodos drawArc, drawRect, drawRoundRect, fillArc, fillRect, fillRoundRect.

Com esses métodos podemos desenhar somente os contornos das formas (drawArc, drawRect, drawRoundRect) ou o preenchimento das mesmas (fillArc, fillRect, fillRoundRect e fillTriangle). Abaixo será mostrado a forma de utilização de alguns desses métodos na construção da interface do NanoChart.

### 3.4.1 – Desenhando arcos

"O desenho de um arco começa pela especificação da caixa de contorno (isso é as dimensões externas de uma caixa 'imaginária' que conterá o arco)." MUCHOW (2004 p. 234). Essa caixa imaginária é definida pelos atributos X, Y, altura e largura do método drawArc ou fillArc.

Para a representação do gráfico de pizzas foi utilizado os métodos, fillArc, para o desenho das "fatias" e o método drawArc para o desenho do "contorno" da pizza. Abaixo segue parte do código fonte do método desenhaPizza presente no projeto.

```
private void desenhaPizza(Graphics g) {
    ...
        g.fillArc(inicioLargura, inicioAltura, fimLargura - inicioLargura,
                  fimLargura - inicioLargura, anguloAcumulado, valorAngulo[i]);
    ...
    g.drawArc(inicioLargura, inicioAltura, fimLargura - inicioLargura,
              fimLargura - inicioLargura, 0, 360);
    ...
}
```

Figura 5 - Parte do método desenhaPizza.

### 3.4.2 – Desenhando retângulos

"Desenhar um retângulo é tão simples como desenhar uma linha reta" [6] TOPLEY (2003 p. 149 tradução própria). "Os retângulos podem ter cantos retos ou arredondados e ser desenhados como contorno ou preenchidos" MUCHOW (2004 p. 238).

Os métodos responsáveis por desenhar retângulos com cantos retos são o drawRect e o fillRect, ambos foram utilizados para a representação do gráfico de barras no método desenhaColuna como pode ser visto abaixo.

```
private void desenhaColuna(Graphics g) {
    ...
        g.fillRect(acumulado, inicioAltura, larguraColuna, fimAltura - inicioAltura);
        g.drawRect(acumulado, inicioAltura, larguraColuna, fimAltura - inicioAltura);
        g.fillRect(acumulado, inicioAltura, larguraColuna + 1,
                   getTamMaxColuna() - getQtdePixelColuna(valorInt[i]));
    ...
}
```

Figura 6 - Parte do método desenhaColuna.

[6] Trecho original: "Drawing a rectangle is just as easy as drawing a straight line".

O desenho de retângulos com cantos arredondados fica a cargo dos métodos drawRoundRect e fillRoundRect, ambos foram utilizados para o desenvolvimento da interface do NanoChart, no método desenhaTab, são utilizados para o desenho das abas “Gráfico” e “Dados”, onde a primeira mostra o gráfico e a segunda mostra as informações de legenda do gráfico, nesse mesmo método são guardados as informações x, y, largura e altura das abas para que seja usado no suporte as telas sensíveis ao toque (será melhor explicado no tópico 3.6 desse trabalho). Abaixo segue parte do código onde foram utilizados os métodos citados.

```
private void desenhaTab(Graphics g) {
    ...
        g.fillRoundRect(largTab + 1, tamTab, largTab - 3, altTab, angTab, angTab);
    ...
        g.fillRoundRect(1, tamTab, largTab, altTab, angTab, angTab);
    ...
    g.drawRoundRect(1, tamTab, largTab, altTab, angTab, angTab);
    g.drawRoundRect(largTab + 1, tamTab, largTab - 3, altTab, angTab, angTab);
    ...
    g.drawRect(0, altTab * 2, largura - 1, altura);
    ...
    g.fillRect(1, (altTab * 2) + 1, largura - 2, altura);
    ...
}
```

Figura 7 - Parte do método desenhaTab.

### 3.5 – Desenhando textos

Escrever não é uma tarefa simples, principalmente no Java ME, pois “o suporte a fonte [...] diminuiu significativamente comparando ao J2SE” MUCHOW (2004 p. 242). Para se obter uma instancia da classe Font é necessário invocar o método estático getFont, “existem três atributos associados a um objeto Font: o tipo, o estilo e o tamanho” MUCHOW (2004 p. 242). Esses atributos estão presentes na classe Font e podem ser vistos na figura 8.

O métodos para desenho de texto são, drawChar, drawChars e drawString, na construção da interface do NanoChart foi utilizado o método drawString para a representação das legendas do gráfico na aba “Dados”, a utilização do método pode ser conferida na parte do código presente na figura 9.

| Atributo | Descrição | Valor constante |
|---|---|---|
| FACE_SYSTEM | Caracteres de sistema | 0 |
| FACE_MONOSPACE | Caracteres monoespacejados | 32 |
| FACE_PROPORTIONAL | Caracteres proporcionais | 64 |
| STYLE_PLAIN | Caracteres normais | 0 |
| STYLE_BOLD | Caracteres em negrito | 1 |
| STYLE_ITALIC | Caracteres em itálico | 2 |
| STYLE_UNDERLINED | Caracteres sublinhados | 4 |
| SIZE_SMALL | Caracteres pequenos | 8 |
| SIZE_MEDIUM | Caracteres médios | 0 |
| SIZE_LARGE | Caracteres grandes | 16 |

Figura 8 - Atributos de Font MUCHOW (2004 p. 242 e 243).

```
private void desenhaLegenda(Graphics g) {
    g.setFont(fontePeq);
    ...
    for (int i = 0; i < tamValor; i++) {
        ...
        g.drawString(rotulo[i] + " - " + getValorLegenda(i),
                     tamLegenda * 2, acumulado, Graphics.LEFT | Graphics.TOP);
        ...
    }
}
```

Figura 9 - Parte do método desenhaTab.

### 3.6 – Suporte a touch-screen

"Alguns dispositivos, principalmente PDAs, podem suportar um ponteiro. A popular plataforma Palm, por exemplo, baseia-se em torno da utilização de uma caneta e uma tela sensível ao toque" [7] LI e KNUDSEN (2005 p. 249 tradução própria).

Partindo desse pressuposto o NanoChart possibilita em sua interface a navegação das abas através do uso do teclado ou através do uso da tecnologia touch-screen. Para que isso se torne possível é preciso armazenar as posições das abas conforme foi descrito no tópico 3.4.2 desse trabalho, dessa forma somente foi implementado o método pointerPressed para a manipulação da navegação em abas. A figura 10 mostra parte do código do método onde são armazenadas as posições das abas, na figura 11 será mostrado o código responsável por navegar pelas abas através do toque na tela do aparelho.

[7] Trecho original: "Some devices, particularly PDAs, may support a pointer. The popular Palm platform, for example, is based around the use of a stylus and a touch-sensitive screen".

```
private void desenhaTab(Graphics g) {
    ...
    //Guardando as posições para a utlização no método pointerPressed.
    //Tab Gráfico
    posTab[0] = 1;                      // X
    posTab[1] = tamTab;                 // Y
    posTab[2] = posTab[0] + largTab;    // largura
    posTab[3] = posTab[1] + altTab;     // altura

    //Tab Dados
    posTab[4] = largTab + 1;            // X
    posTab[5] = tamTab;                 // Y
    posTab[6] = posTab[2] + largTab - 3; // largura
    posTab[7] = posTab[5] + altTab;     // altura
    ...
}
```

Figura 10 - Parte do método desenhaTab.

```
protected void pointerPressed(int x, int y) {
    if (hasPointerEvents()) {
        if (( x >= posTab[0] && x <= posTab[2])
          && (y >= posTab[1] && y <= posTab[3])) {
            grafico = true;
            repaint();
        } else if (( x >= posTab[4] && x <= posTab[6])
                && (y >= posTab[5] && y <= posTab[7])) {
            grafico = false;
            repaint();
        }
    }
}
```

Figura 11 - Método pointerPressed.

## 3.7 – Considerações finais

O capitulo 3 apresentou as particularidades da construção de uma interface utilizando a classe Canvas, um trabalho dispendioso mais que no final apresenta um resultado satisfatório. Foram demonstrados os principais métodos utilizados para a elaboração do NanoChart, demonstrando com partes do código o resultado final dessa utilização.

# 4 – DESENVOLVIMENTO DO PROJETO

Neste capítulo será apresentado a maneira como o projeto foi desenvolvido e organizado, demonstrando a estrutura, suas principais funcionalidades e o resultado obtido após o desenvolvimento. A estrutura do projeto está ilustrada na figura 12.

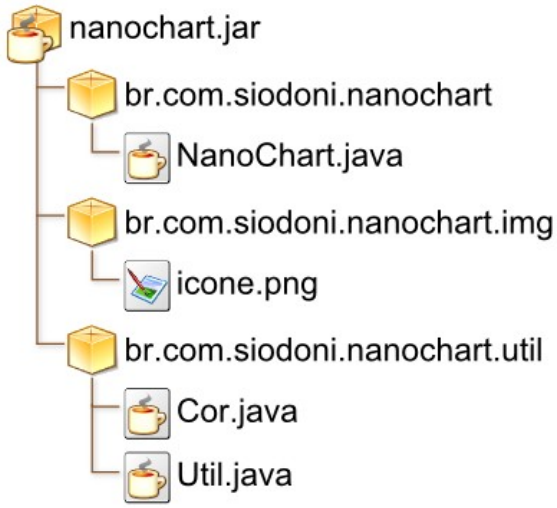


Figura 12 - Estrutura do projeto NanoChart.

## 4.1 – Estrutura / Organização

O projeto NanoChart foi organizado em 3 pacotes como pode ser visto na figura 12, todos estão dentro da estrutura de diretórios br/com/siodoni/nanochart, o último nível ainda foi subdivido para comportar os pacotes img e util. Segue abaixo uma descrição dos pacotes, das classes e dos arquivos que o compõe.

- br.com.siodoni.nanochart: nesse pacote está presente a classe NanoChart.java, a mesma é responsável por fornecer o gráfico de barra e pizza para o usuário;
- br.com.siodoni.nanochart.img: nesse pacote está presente o arquivo icone.png. Esse arquivo pode ser associado como ícone da MIDlet que utilizará o NanoChart;
- br.com.siodoni.nanochart.util: nesse pacote está presente a classe Util.java essa classe possui os métodos doubleParaInt e floatParaInt, os mesmos são utilizados na construção da interface do gráfico, pois para o calculo da área

do gráfico devem ser utilizados valores inteiros. Ainda nesse pacote está presente a interface Cor.java, a mesma fornece os códigos das cores para que usuário possa utilizar na elaboração do gráfico.

## 4.2 – Funcionalidades

As funcionalidades no NanoChart são:

- Agilidade no desenvolvimento, pois a biblioteca entrega ao desenvolvedor um gráfico de barras ou de pizza de forma simples e direta;
- Leiaute dividido em abas, dessa forma se tem um aproveitamento maior da tela do aparelho, pois as informações ficam separadas, o que torna a interface mais agradável;
- Suporte a telas sensíveis ao toque, com essa funcionalidade o NanoChart se torna mais acessível, pois já existem no mercado aparelhos com essa tecnologia;
- Fornecimento de cores para a elaboração do gráfico, essa ferramenta auxilia o desenvolvedor na utilização da biblioteca, pois fornece ao mesmo cores para que possa ser utilizado na elaboração das barras ou nas fatias da pizza;
- Validação de informações, o NanoChart faz três validações antes de fornecer o gráfico ao desenvolvedor, a primeira é se existem valores negativos passados ao gráfico, a segunda é se existem mais de dez valores passados ao gráfico e a ultima é se a quantidade de valores é igual a quantidade de legendas para o gráfico, essas validações podem ser vistas na figura 13.

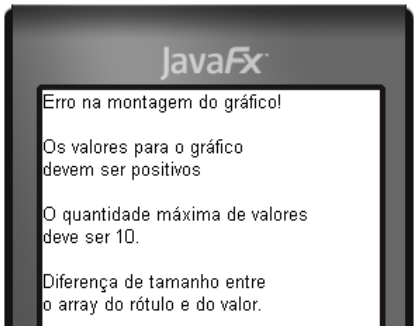


Figura 13 - Validações NanoChart.

### 4.3 – Testes / Utilização

Para exemplificar o uso da biblioteca NanoChart em uma aplicação móvel será utilizado a IDE NetBens (http://www.netbeans.org). Primeiro é necessário baixar o arquivo da biblioteca no seguinte endereço:

http://nanochart.googlecode.com/files/nanochart.jar

Após feito o download deverá ser criado um novo projeto conforme figura 14. Deverá ser escolhido a categoria “Java ME” e o tipo de projeto “Aplicativo móvel”.

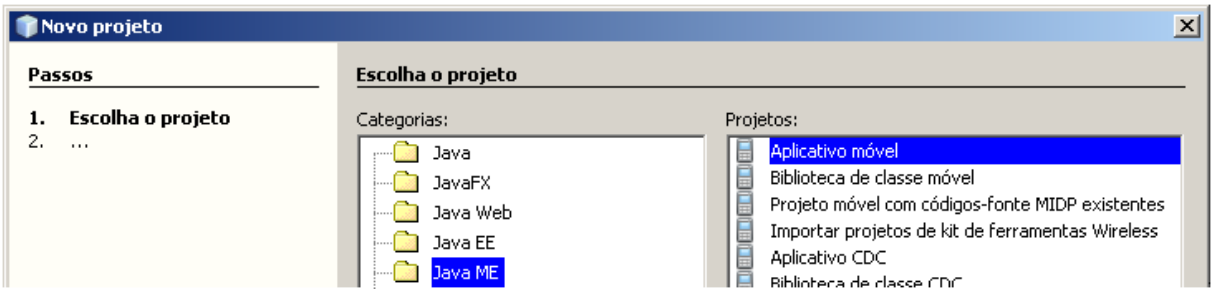


Figura 14 - Criação de um novo projeto.

A próxima etapa é a escolha do nome da aplicação como pode ser visto na figura 15. No exemplo foi escolhido o nome “ExemploNanoChart”, note que a opção “Criar MIDlet Olá” deverá estar desmarcada.

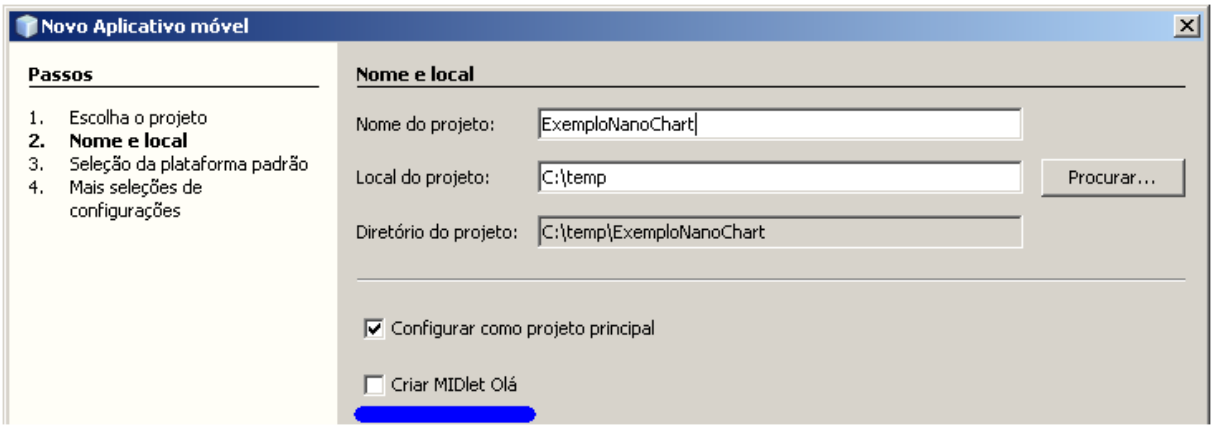


Figura 15 - Escolha do nome do projeto.

Em seguida deverá ser escolhida a plataforma do emulador, o dispositivo, a configuração do dispositivo e o perfil do dispositivo. Nesse exemplo foi utilizado a

plataforma “Java (TM) Plataform Micro Edition SDK 3.0”, o dispositivo “DefaultFxTouchPhone1”, a configuração do dispositivo “CLDC-1.1” e o perfil do dispositivo “MIDP-2.0”, conforme pode ser visto na figura 16.

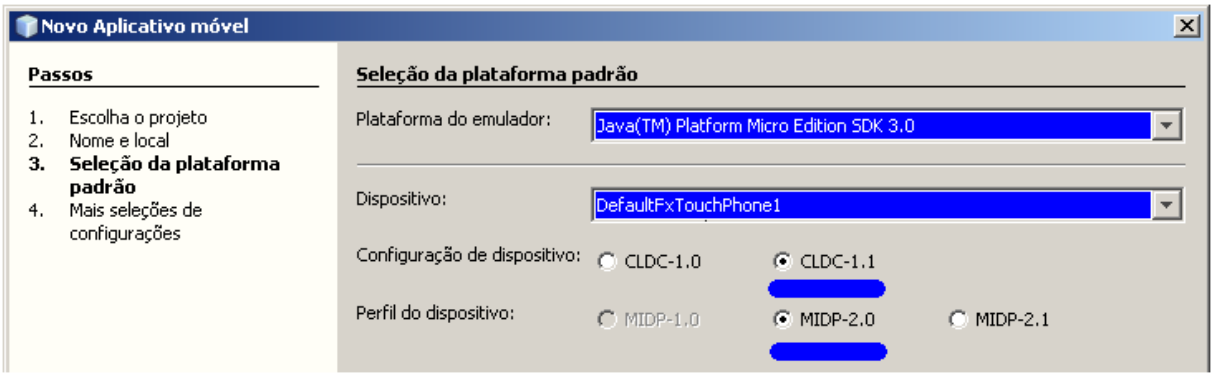


Figura 16 - Escolha da plataforma, dispositivo, configuração e perfil.

A próxima etapa é a escolha de outras configurações do projeto. Nesse exemplo essa etapa foi ignorada, clique somente em “Finalizar” para criar o projeto.

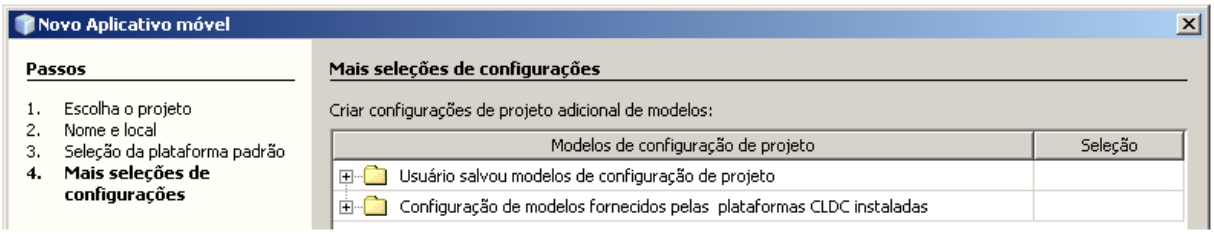


Figura 17 - Mais configurações do projeto.

Feito todas as etapas acima o projeto estará criado, a próxima etapa é criar a MIDlet que utilizará a biblioteca NanoChart. Clique com o botão direito do mouse no pacote onde a MIDlet deverá ser criada. Escolha a opção “Novo” depois em “Outro...”, conforme pode ser visto na figura 18.

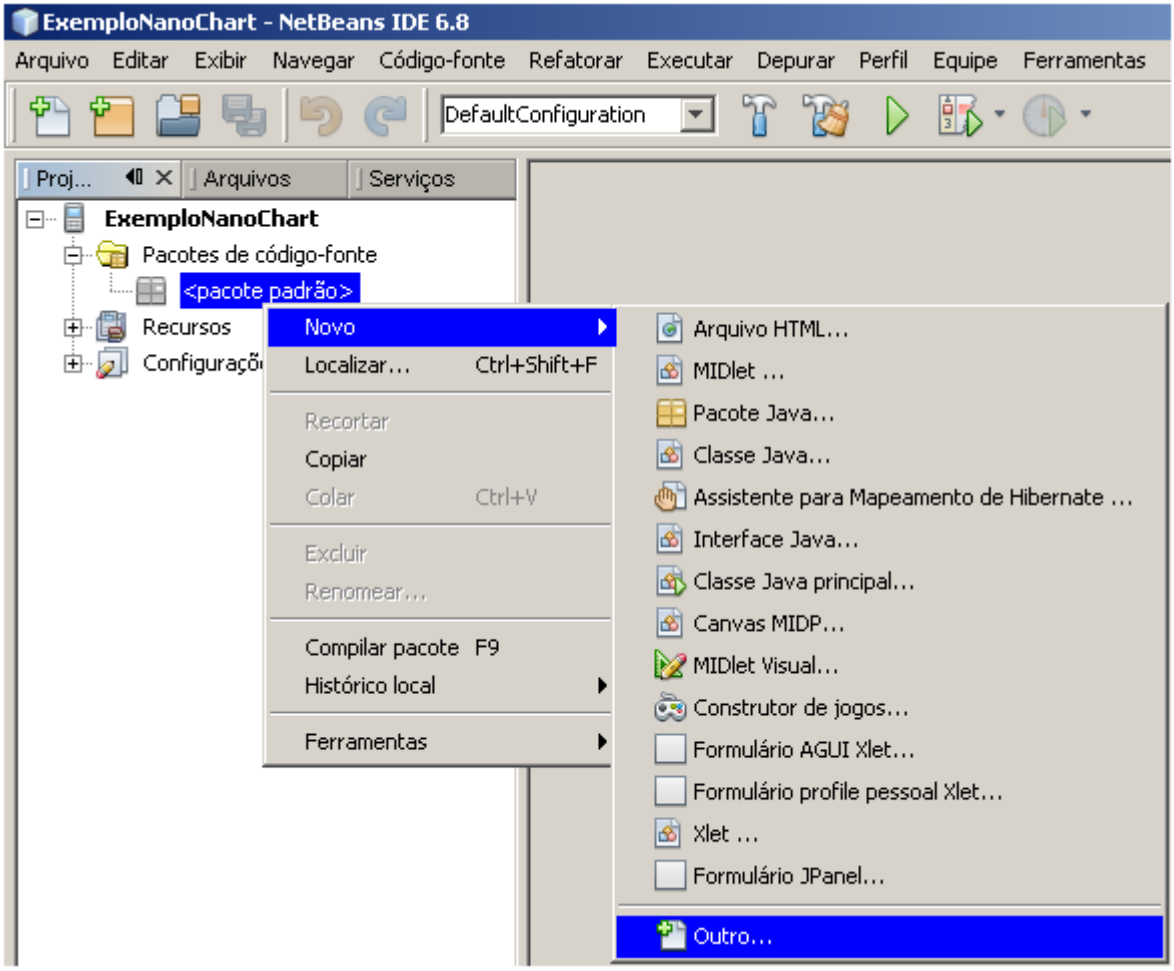


Figura 18 - Criando a MIDlet.

A seguir será mostrado a tela de escolha do tipo de arquivo conforme figura 19, escolha a categoria “MIDP” e o tipo de arquivo “MIDlet”.

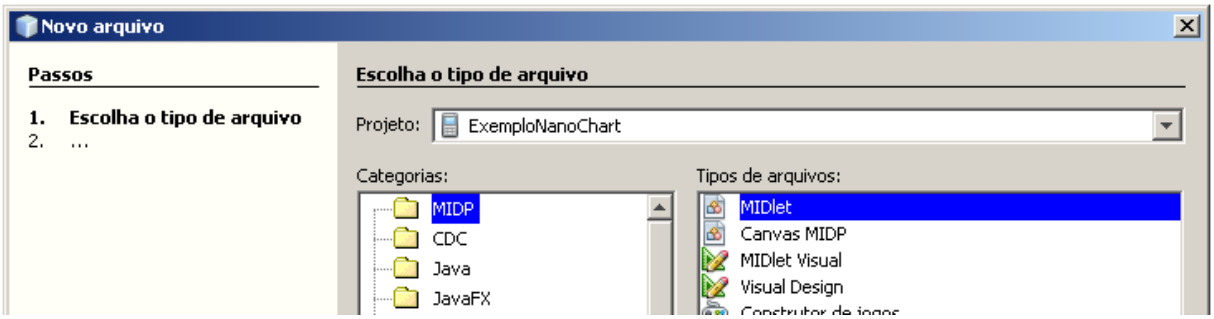


Figura 19 - Escolhendo o tipo de arquivo.

Em seguida será necessário indicar o nome do MIDlet e da classe MIDP, no exemplo foi utilizado o nome Midlet para ambas as situações, conforme figura 20.

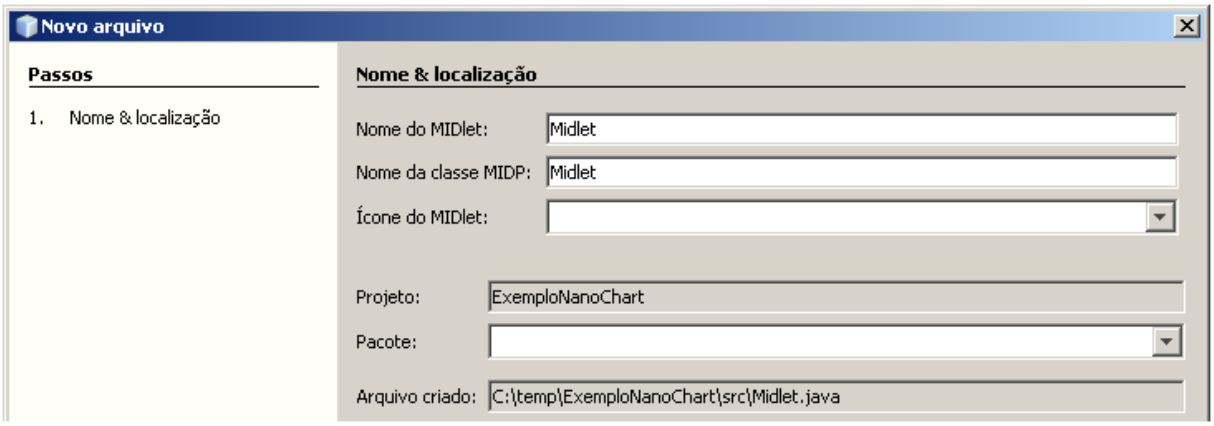


Figura 20 - Escolhendo o nome da classe e da MIDlet.

A seguir deverá ser importada a biblioteca do NanoChart para o projeto, para isso clique com o botão direito sobre o nome do projeto e escolha a opção “Propriedades”, como pode ser visto na figura 21.

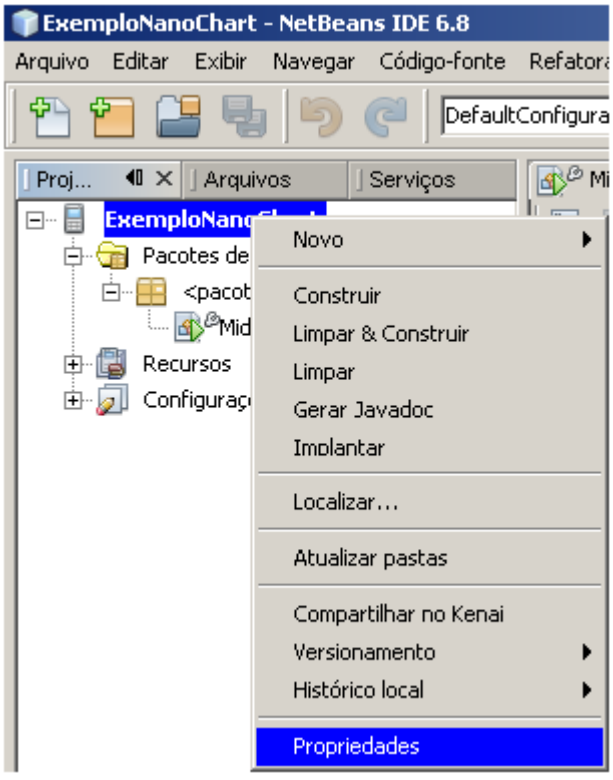


Figura 21 - Propriedades do projeto.

Feito isso navegue até a categoria “Bibliotecas e Recursos”, e clique na opção “Adicionar jar/zip” conforme figura 22.

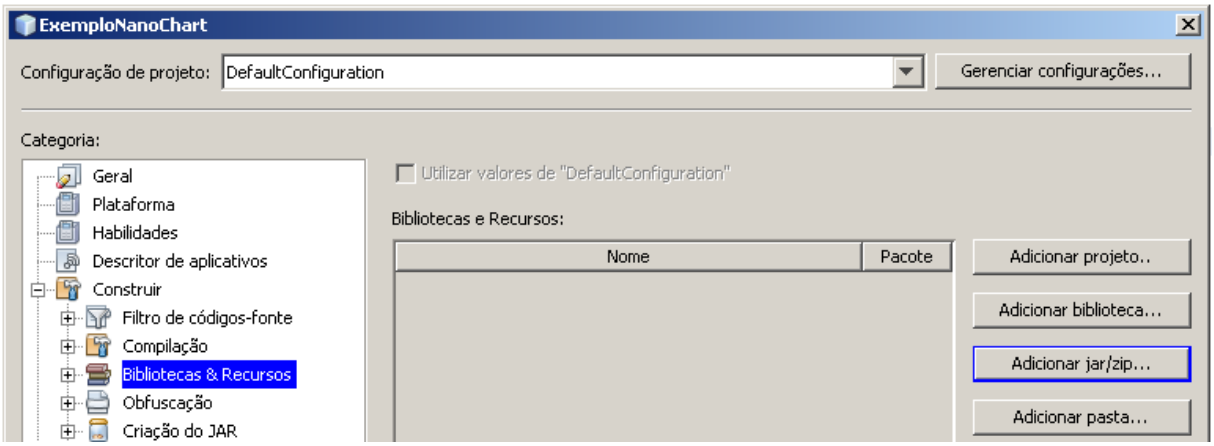


Figura 22 - Incluindo a biblioteca no projeto.

Escolha o arquivo “nanochart.jar” e clique no botão abrir, após feito isso o projeto deverá estar com a estrutura conforme a figura 23.

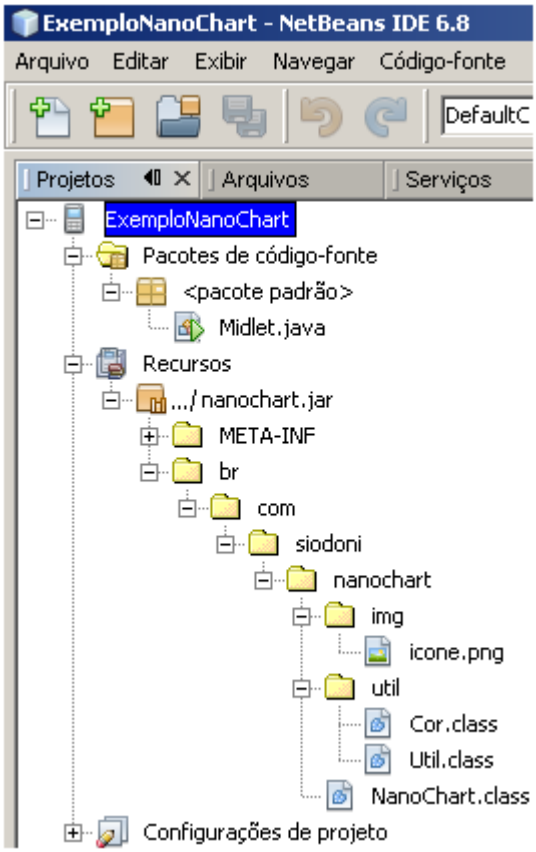


Figura 23 - Estrutura do projeto de exemplo.

A última etapa antes de executar o projeto é escrever a MIDlet de exemplo, o código dessa classe pode ser visto na figura 24.

```
import br.com.siodoni.nanochart.util.Cor;
import br.com.siodoni.nanochart.NanoChart;
import javax.microedition.midlet.MIDlet;
import java.util.Random;
import javax.microedition.lcdui.Display;

public class Midlet extends MIDlet implements Cor {

    public void startApp() {
        //Atributo responsavel por gerar valores aleatórios para o exemplo.
        Random random = new Random();

        //Array de cores que será utilizada para a montagem do gráfico.
        int cor[] = {VERMELHO2, VERDE2, AMARELO2, AZUL2, ROXO2,
                     VERMELHO3, VERDE3, AMARELO3, AZUL3, ROXO1};

        //Array de valores que será utilizado para a montagem do gráfico de exemplo.
        int valor[] = new int[10];

        //Array de rotulos que será utilizado para a montagem
        //das legendas do gráfico de exemplo.
        String rotulo[] = new String[valor.length];

        //Título do gráfico de exemplo.
        String titulo = "Exemplo de gráfico aleatório";

        //Tipo de gráfico utilizado. Podendo ser GRAFICO_BARRA ou GRAFICO_PIZZA.
        int tpGrafico = NanoChart.GRAFICO_BARRA;

        //Montando de forma aleatória os valores e as legendas.
        for (int i = 0; i < valor.length; i++) {
            valor[i] = random.nextInt(50) + 1;
            rotulo[i] = "valor aleatório";
        }

        //Instanciando o atributo nanoChart de acordo com as informações fornecidas.
        NanoChart nanoChart = new NanoChart(cor, valor, rotulo, titulo, tpGrafico);

        //Solicitando ao Display que mostre o grafico de
        //acordo com as informações fornecidas.
        Display.getDisplay(this).setCurrent(nanoChart);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
        notifyDestroyed();
    }
}
```

Figura 24 - Código da MIDlet de exemplo.

Após ter escrito a MIDlet de exemplo o projeto poderá ser executado o resultado da execução pode ser visto na figura 25.

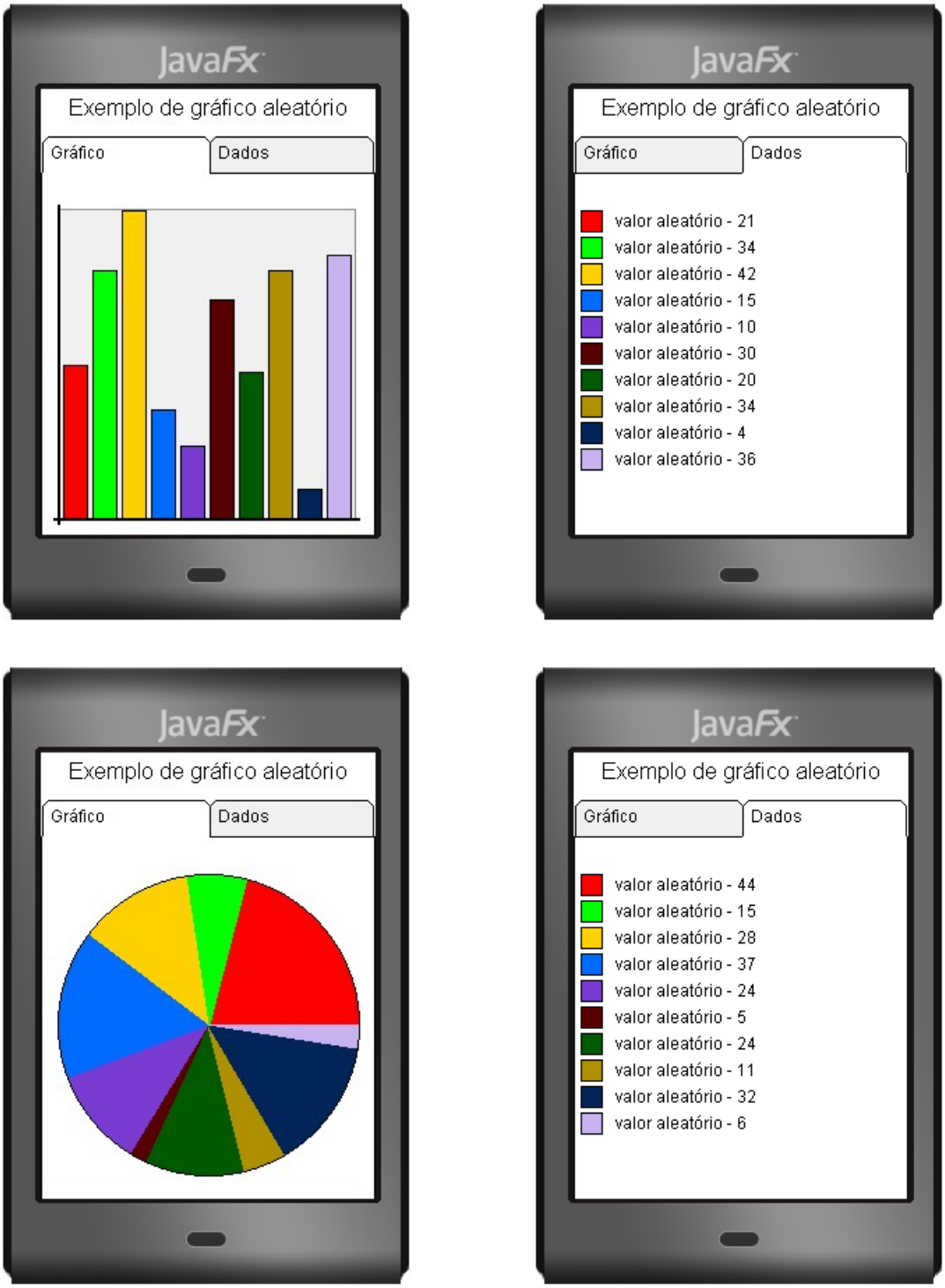


Figura 25 - NanoChart em execução – Aba Gráfico e Dados.

## 4.4 – Discussão

A utilização da biblioteca NanoChart facilita o desenvolvimento de aplicações móveis, pois entrega ao desenvolvedor um gráfico de barras ou pizza pronto, o mesmo só tem que utiliza-lo em sua aplicação sem se preocupar. Porém a biblioteca possui limitações, não foi possível implementar o gráfico de barras recebendo valores negativos, o suporte a fontes em alguns aparelhos é fraco, o que torna a elaboração da aba “Dados” um desafio.

Por se tratar de uma biblioteca com código fonte aberto, qualquer pessoa poderá contribuir para a continuação do projeto, o NanoChart está hospedado na Google e pode ser acessado pelo seguinte endereço:

http://nanochart.googlecode.com

## 4.5 – Considerações finais

O capitulo 4 apresentou a forma como o projeto está estruturado e organizado, demonstrando as principais funcionalidades do NanoChart, também foi apresentado um exemplo prático de utilização biblioteca em uma aplicação.

# 5 – CONCLUSÃO

A crescente expansão no mercado de aparelhos celulares nos últimos anos estimula o desenvolvimento de aplicações móveis, pois, a cada dia surgem novas necessidades. Uma dessas necessidades é a de aplicações móveis mais elaboradas e com mais recursos.

Tendo em base essas premissas tornou-se viável o desenvolvimento da biblioteca NanoChart, que surgiu para suprir as dificuldades que o desenvolvedor encontra ao elaborar suas aplicações móveis.

Para o desenvolvimento do projeto foi utilizado a linguagem de programação Java, sendo que a motivação para o uso dessa tecnologia foi baseada no grande suporte da plataforma na construção de aplicações móveis, devido aos diversos emuladores e bibliotecas presentes na mesma.

O objetivo do projeto NanoChart era fornecer, de forma rápida e eficiente, ao desenvolvedor, gráficos de barras ou pizza de valores positivos para a utilização em suas aplicações móveis.

A interface da biblioteca NanoChart foi construída utilizando-se das classes Canvas e Graphics que forneceram métodos que possibilitaram desenhar linhas, retângulos, círculos e textos, resultando na criação de uma interface personalizada que dispõe de um leiaute dividido em abas de simples visualização, entendimento e navegação, pois foi possível a implementação tanto de suporte a telas sensíveis ao toque quanto a navegação pelo teclado.

O resultado final do trabalho foi a criação da biblioteca NanoChart, cuja a responsabilidade é fornecer ao desenvolvedor, gráficos de barras ou pizza de valores positivos para a utilização em suas aplicações móveis, uma vez que a plataforma Java Micro Edition não fornece nativamente a possibilidade de geração de gráfico em suas bibliotecas.

Com isso o NanoChart torna-se uma biblioteca geradora de gráficos pronta para ser utilizadas em aplicações móveis de qualquer finalidade.

Por ser uma biblioteca com código fonte aberto, torna-se possível sua ampliação por qualquer pessoa que tenha interesse em contribuir para a continuidade e expansão desse projeto.

Novas funcionalidades poderão ser incluídas na biblioteca NanoChart, dentre

as principais sugestões dessas melhorias ou novas implementações estão: a possibilidade da geração do gráfico de barras com valores negativos; a geração de dois novos tipos de gráficos (colunas e linhas) e a possibilidade da utilização do gráfico de barras ou colunas combinado com o gráfico de linhas, formando assim um único gráfico de comparação.

## 6 – REFERÊNCIAS BIBLIOGRÁFICAS

ANDROID. What is Android?. Disponível em: <http://developer.android.com/guide/basics/what-is-android.html>. Acesso em 15 de mar. de 2010.

CIENCIASDACOMPUTACAO. Tecnologias para o desenvolvimento de aplicações móveis. Disponível em: <http://www.cienciasdacomputacao.org/2009/09/tecnologias-para-o-desenvolvimento-de.html>. Acesso em 15 de mar. de 2010.

G1. Internet Summary: Brasil ultrapassa 160 milhões de linhas celulares em uso. Disponível em: <http://g1.globo.com/Noticias/Tecnologia/0,,MUL1273647-6174,00-INTERNET+SUMMARY.html>. Acesso em: 07 set. 2009.

KEOGH, James. J2ME: The Complete Reference. 1.ed. McGraw-Hill: New York, 2003.

LI, SING e KNUDSEN JONATHAN. Beginning J2ME: From Novice to Professional. 3.ed. Apress: New York, 2005.

MUCHOW, JOHN. W.. Core J2ME: Tecnologia e MIDP. 1.ed. Makron Books: São Paulo, 2004.

NORMAN, Donald. A.. The Design of Everyday Things; Basic Books: New York, 2002.

PEREIRA, Rogério. Intodução ao SuperWaba. Disponível em: <http://www.superwaba.org/etc/wm_introducao_ao_SuperWaba.pdf>. Acesso em 15 de mar. de 2010

PREECE, J. et al.. Interaction Design: Beyond Human-Computer Interaction. John Wiley & Sons, 2002.

SUN. Java ME Technology. Disponível em:
<http://java.sun.com/javame/technology/index.jsp>. Acesso em 15 de mar. de 2010.

TOPLEY, K.. J2ME in a Nutshell. O'Reilly, 2002.

VALENTE, Eduardo Cesar. Padrões de Interação e Usabilidade. 2004. 234 p. Tese (Mestrado em Engenharia de Software) – Instituto de Computação, Universidade Estadual de Campinas, Campinas.