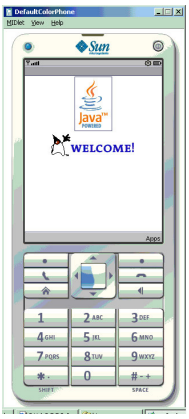
Programação para Celular com Java

Objetivo:

- Capacitar o aluno a desenvolver aplicações com o *Sun Java Wireless Toolkit for CLDC*.



Sumário

Prof. Vida
vida@utfpr.edu.br

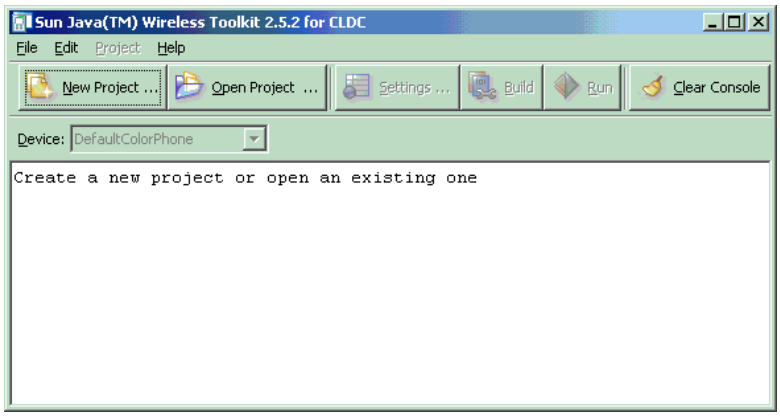
1. Introdução:

MIDlet suites são organizadas em projetos. O resultado de um projeto é um Midlet Suite.

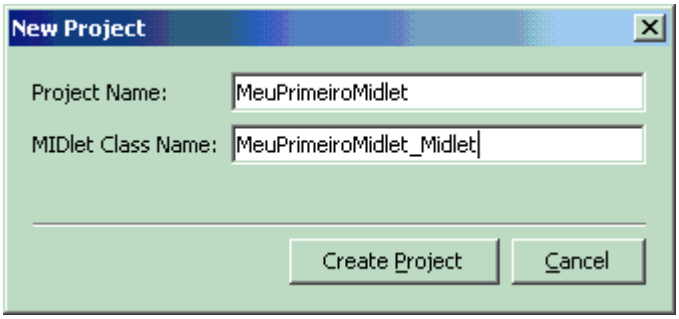
Um projeto contém todos os arquivos necessários para a construção de um MIDlet suite.

2. Como criar um novo Projeto?

1. Na interface do WTK2.5.2, clique no botão **`New Project`**.

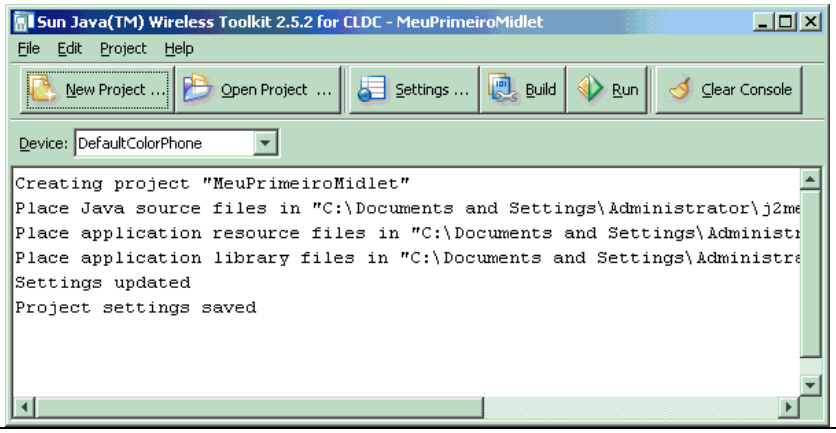


2. Na Janela que é apresentada informe o nome do projeto e o nome da classe, conforme ilustrado pela figura a seguir.



3. Uma janela de *Settings for Projects* é apresentada. Sem modificar nada, clique no botão **OK** dessa janela.

Mensagens serão apresentadas no console informando o local de armazenamento dos códigos, conforme ilustra a figura a seguir.



Prof. Vida
vida@utfpr.edu.br

3. Ciclo Simples de Desenvolvimento

O ciclo de desenvolvimento de uma aplicação é formado por:

1. **Edit:** Editar e salvar o código fonte.
2. **Build:** Compilação e pré-verificação do código Java.
3. **Run:** Execução das classes compiladas em um emulador.

3.1- A edição do código fonte.

Crie no seu editor de códigos a seguir:

3.2- Código do Exemplo

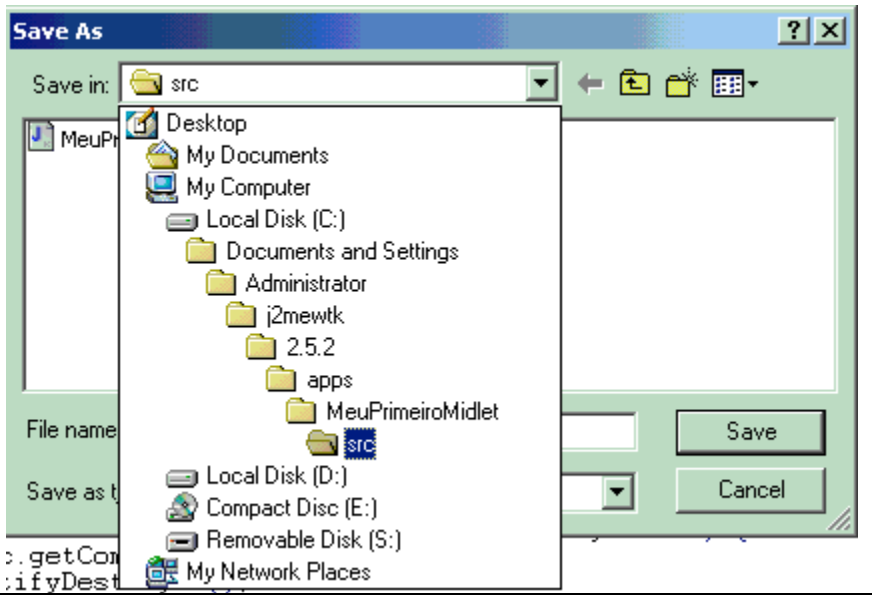
```
/////////////// arquivo MeuPrimeiroMidlet_Midlet.java
import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;
public class MeuPrimeiroMidlet_Midlet
    extends MIDlet implements CommandListener
{
 private Form janela;
 private Display display;
 public void startApp()
 { display = Display.getDisplay(this);
   janela = new Form("MeuMidlet");
   janela.append("Ola Mundo!");
   Command exitCommand = new Command("Exit"
                       , Command.EXIT, 0);
    janela.addCommand(exitCommand);
    janela.setCommandListener(this);
    display.setCurrent(janela);
  }

 public void pauseApp () {}
 public void destroyApp(boolean unconditional){}

 public void commandAction(Command c, Displayable s)
 { if (c.getCommandType() == Command.EXIT)
      notifyDestroyed();
 }
}
```

/ Observação Importante: O código Java exemplificado não deverá ser compilado através da interface de edição.* A compilação do código será realizada no WTK2.5.2. **/*

3.3- Aonde salvar o arquivo digitado?

Esse arquivo deve ser salvo no diretório `workdir\apps\MeuPrimeiroMidlet\src\`.

Se foi utilizado a instalação padrão, o diretório a ser instalado será o ilustrado na figura a seguir.



3.4- Análise do código

A API microedition é um conjunto de pacotes exclusivos. Esse conjunto é formado pelos seguintes pacotes:

- `javax.microedition.midlet.*` : define as aplicações e a interação entre a aplicação e o ambiente de execução.
- `javax.microedition.lcdui.*` : Fornece classes para implementação de interface gráfica.
- `javax.microedition.lcdui.game.*` : Fornece classes para o desenvolvimento de jogos.
- `javax.microedition.rms.*` : Fornece classes para implementar o armazenamento de dados no dispositivo.
- `javax.microedition.pki.*` : Permite autenticar a comunicação segura.

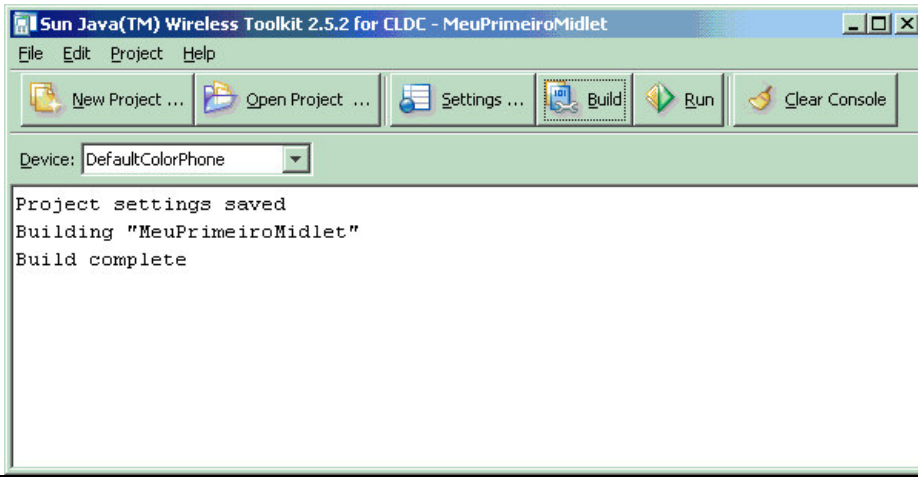
O Midlet está inicialmente no estado carregado. Quando a aplicação é iniciada pelo método `startApp()`, o Midlet muda o seu estado para ativo. O método `pauseApp()` dá uma pausa na execução e o `destroyApp()` termina a sua execução. A definição desses três métodos é obrigatoria.

4. Build

O próximo passo é chamar o compilador através do botão Build do WTK2.5.2.

Ao clicar no botão Build da interface do WTK2.5.2, o resultado da

compilação é apresentada no console, conforme ilustra a figura a seguir.



4.1- Resultado da Execução

Observação

A interface gráfica para este tipo de dispositivo é bastante limitada quando comparada com uma aplicação Desktop. Tipicamente, a tela possui 95x125 ou 128x128 pixels. Só é possível utilizar poucos elementos visuais simples.

Programação para Celular com Java – Interface Gráfica

Objetivo:
Capacitar o aluno a
- Criar múltiplos formulários
- Desenvolver elementos navegadores
- Inserir alguns componentes gráficos em uma aplicação
- Adicionar interatividade na aplicação.

---

5. A Biblioteca javax.microedition.lcdui.*.

Essa biblioteca é responsável por disponibilizar os seguintes componentes:

- `javax.microedition.`AlertType
- `javax.microedition.`Command
- `javax.microedition.`Display
- `javax.microedition.`Displayable
  - `javax.microedition.`Canvas
  - `javax.microedition.`Screen
    - `javax.microedition.`Alert
    - `javax.microedition.`Form
    - `javax.microedition.`List
    - `javax.microedition.`TextBox
- `javax.microedition.`Font
- `javax.microedition.`Graphics
- `javax.microedition.`Image
- `javax.microedition.`Item
  - `javax.microedition.`ChoiceGorup
  - `javax.microedition.`CustomItem
  - `javax.microedition.`DateField
  - `javax.microedition.`Gauge
  - `javax.microedition.`ImageItem
  - `javax.microedition.`Spacer
  - `javax.microedition.`StringItem
  - `javax.microedition.`TextField
- `javax.microedition.`Ticker

Prof. Vida
vida@utfpr.edu.br

5.1 As classes `javax.microedition.Display` e `javax.microedition.Displayable`

Uma instância dessa classe representa a tela de exibição. Pode-se utilizar apenas um único objeto Display. O Display representao o hardware enquanto que a Displayable é uma informação que pode ser apresentada no Display.

6. A classe javax.microedition.lcdui.Command

Essa classe permite adicionar funcionalidade aos objetos exibidos. Os comandos podem ser dos tipos listados na Tabela a seguir.

BACK	Retorna à tela anterior
CANCEL	Cancela uma operação
EXIT	Encerra a aplicação
HELP	Chama por ajuda
ITEM	Utilizado junto com o elemento do tipo choice para determinar um item da tela
OK	Resposta positiva.
STOP	Interrompe um processo de execução
SCREEN	expecifica que um comando está associado à tela.

6.1 O construtor

Uma instância de Command deve ser inicializada através da definição de um nome, tipo e prioridade. Por exemplo,

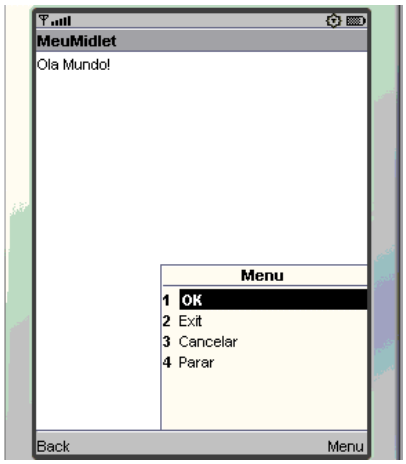
```
Command sair = new Command("Exit",Command.EXIT, 0);
   Command ok = new Command("OK",Command.OK, 1);
Command back = new Command("Back",Command.BACK, 1);
```

6.2 Exemplo

No Midlet desenvolvido anteriormente, modifique o código inserindo novas opções de comando, conforme ilustra a figura a seguir.

```
12
13      Command exitCommand = new Command("Exit", Command.EXIT, 1);
14      Command ok = new Command("OK",Command.OK, 0);
15      Command back = new Command("Back",Command.BACK, 2);
16      Command parar = new Command("Parar",Command.STOP,3 );
17      Command cancelar = new Command("Cancelar",Command.CANCEL,0 );
18
19      janela.addCommand(exitCommand);
20      janela.addCommand(ok);
21      janela.addCommand(back);
22      janela.addCommand(parar);
23      janela.addCommand(cancelar);
24      janela.setCommandListener(this);
25
26      display.setCurrent(janela);
27    }
28    public void pauseApp () {}
29
```

O resultado da modificação é ilustrado na figura a seguir.



6.3 Exercício.

No Midlet desenvolvido até o presente momento, altere o código conforme ilustrado a seguir. Qual a ordem dos elementos a serem apresentado no Menu?

```
//////////////////////////////  arquivo MeuPrimeiroMidlet_Midlet.java
import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;

public class MeuPrimeiroMidlet_Midlet
    extends MIDlet
    implements CommandListener {
    private Form janela;
    private Display display;

  public void startApp() {
    display = Display.getDisplay(this);

    janela = new Form("MeuMidlet");
    janela.append("Ola Mundo!");

    Command exitCommand = new Command("Exit", Command.OK, 1);
    Command ok          = new Command("OK",Command.OK, 0);
    Command back        = new Command("Back",Command.OK, 2);
    Command parar       = new Command("Parar",Command.OK,3 );
    Command cancelar    = new Command("Cancelar",Command.OK,4 );
    Command valorPi     = new Command("Valor de Pi", Command.OK,5);

    janela.addCommand(exitCommand);
    janela.addCommand(ok);
    janela.addCommand(back);
```

```
    janela.addCommand(parar);
    janela.addCommand(cancelar);
    janela.addCommand(valorPi);
    janela.setCommandListener(this);

    display.setCurrent(janela);
  }
  public void pauseApp () {}

  public void destroyApp(boolean unconditional) {}

  public void commandAction(Command c, Displayable s) {
    if (c.getLabel() == "Exit")
      notifyDestroyed();
    else janela.append("\n"+c.getLabel());

    if (c.getLabel() == "Valor de Pi")
      janela.append("="+ Math.PI);
  }
}
```

7. A Classe javax.microedition.lcdui.Screen

A classe Screen permite a inserção de objetos gráficos.

Tipos de Tela:

A classe Screen admite quatro tipos de tela, a saber:

- Form
- Alert
- List
- TextBox

7.1 Tela Form

Form é uma tela que pode conter diversos tipos de elementos gráficos, tais como imagens, campos de edição e textos. O exemplo estudado até o presente momento utilizou apenas a Tela Form.

Construtor	O construtor do Form deve ser chamado com a seguinte estrutura: • **Form**(String title) : Cria um formulário vazio. • **Form**(String title, Item[] items) : Cria um formulário com alguns itens.

Exemplo de construtor:

```
Form janela = new Form("MeuMidlet");
```

Outro exemplo de Construtor:

```
 Item[] lista = new Item[3];
 lista[0] = new StringItem(
            "Opção 1","",Item.PLAIN);
 lista[1]= new StringItem(
            "Opção 2","",Item.PLAIN);
 lista[2]= new StringItem(
            "Opção 3","",Item.PLAIN);

 janela = new Form("MeuMidlet",lista);
```

CommandListener

Um form pode conter comandos de ações. Exemplo de definição de CommandListener:

```
janela.setCommandListener(this);
```

Métodos

int append(Item i)	Adiciona um item ao formulário
void deleteAll( )	Deleta todos os itens do formulário
int getHeight( )	Obtém a largura da tela
int getWidth ( )	Obtém o comprimento da tela
void insert(int p, Item i)	Insere um item noformulário

7.2 Tela TextBox

A classe TextBox é um tipo de tela.

Construtor

O construtor pode ser definido conforme o exemplo a seguir:

```
TextBox entrada1 = new TextBox(String title
                               , String text
                               , int maxSize
                               , int constraints)
```

Os parâmetros necessários ao construtor podem ser definidos como sendo:

- `title` – O título do display
- `text` – O conteúdo inicial a ser apresentado.
- `maxSize` – a capacidade máxima em caracteres.
- `constraints` – `ANY, EMAILADDR, NUMERIC, PHONENUMBER, URL DECIMAL e PASSWORD, UNEDITABLE ,SENSITIVE ,NON_PREDICTIVE ,INITIAL_CAPS_WORD ,INITIAL_CAPS_SENTENCE`

Métodos Relacionados

`void delete(int i, int f)`	Deleta caracteres do TextBox
`String getString( )`	Retorna o seu conteúdo
`void insert(String s, int p)`	Insere a string s na posição p

void setMaxSize(int n)	Define o tamanho máximo que o TextBox pode conter
void setTitle(String s)	Dá um título ao TextBox
int size()	Retorna a quantidade de caracteres do TextBox.

Exemplo/Exercício:

Modifique o exemplo anterior, adicionando os novos conceitos apresentados, conforme o código abaixo. O que a aplicação faz?

```
//////////////////////////// arquivo MeuPrimeiroMidlet_Midlet.java
import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;

public class MeuPrimeiroMidlet_Midlet
    extends MIDlet
    implements CommandListener {
    private Form janela;
    private final int tamanho = 3;
    private TextBox[] entrada = new TextBox[tamanho];
    private Item[] lista =      new Item[tamanho];
    private StringItem[] campos = new StringItem[tamanho];
    private Display display;

  public void startApp() {
    display = Display.getDisplay(this);
    campos[0] =  new StringItem("Nome","",Item.PLAIN);
    campos[1] = new StringItem("Idade","",Item.PLAIN);
    campos[2] = new StringItem("Senha","",Item.PLAIN);

    for (int i = 0; i < tamanho; i++)
       lista[i] = campos[i];

    janela = new Form("MeuMidlet",lista);
    janela.append("\nOla Mundo!");

    entrada[0]= new TextBox("Nome",null, 20
                          , TextField.INITIAL_CAPS_WORD);
    entrada[1] = new TextBox("Idade", null, 2, TextField.NUMERIC);
```

```
   entrada[2] = new TextBox("Senha", null, 8
               , TextField.PASSWORD);

   Command exitCommand = new Command("Exit", Command.OK, 1);
   Command nome        = new Command("Nome",Command.OK, 0);
   Command idade       = new Command("Idade",Command.OK, 0);
   Command senha       = new Command("Senha",Command.OK, 0);
   Command back        = new Command("Back",Command.BACK, 2);
   Command parar       = new Command("Parar",Command.OK,3 );
   Command cancelar    = new Command("Cancelar",Command.OK,4 );
   Command valorPi     = new Command("Valor de Pi",Command.OK,5);

   for (int i = 0; i < tamanho; i++) entrada[i].addCommand(back);

   janela.addCommand(exitCommand);
   janela.addCommand(nome);
   janela.addCommand(idade);
   janela.addCommand(senha);
   janela.addCommand(back);
   janela.addCommand(parar);
   janela.addCommand(cancelar);
   janela.addCommand(valorPi);
   janela.setCommandListener(this);
   for (int i = 0; i < tamanho; i++)
      entrada[i].setCommandListener(this);
   display.setCurrent(janela);
}
public void pauseApp () {}

public void destroyApp(boolean unconditional) {}

public void commandAction(Command c, Displayable s) {
   if (c.getLabel() == "Back")
    {  for (int i = 0; i < tamanho; i++)
          campos[i].setText(entrada[i].getString());
     display.setCurrent(janela);
      }
   if (s==janela)
   { if (c.getLabel() == "Exit")
```

```
            notifyDestroyed();
         else  if (c.getLabel() == "Nome")
                    display.setCurrent(entrada[0]);
               else if (c.getLabel() == "Idade")
                         display.setCurrent(entrada[1]);
                    else if (c.getLabel() == "Senha")
                              display.setCurrent(entrada[2]);
                         else janela.append("\n"+c.getLabel());

      if (c.getLabel() == "Valor de Pi")
       janela.append("="+ Math.PI);
    }
  }
}
```

Referência Bibliográfica:

Título:Java para Dispositivos Móveis : desenvolvendo aplicações com J2ME
Ano: 2007
Autor: Thienne M. Johnson
Editora: Novatec Editora
ISBN: 978-85-7522-143-3