Módulo 5

Desenvolvimento de Aplicações Móveis

Lição 2

Como Começar

Versão 1.0 - Set/2007

Autor
xxx

Equipe
Rommel Feria
John Paul Petines

Necessidades para os Exercícios

Sistemas Operacionais Suportados

NetBeans IDE 5.5 para os seguintes sistemas operacionais:

- Microsoft Windows XP Profissional SP2 ou superior
- Mac OS X 10.4.5 ou superior
- Red Hat Fedora Core 3
- Solaris™ 10 Operating System (SPARC® e x86/x64 Platform Edition)

NetBeans Enterprise Pack, poderá ser executado nas seguintes plataformas:

- Microsoft Windows 2000 Profissional SP4
- Solaris™ 8 OS (SPARC e x86/x64 Platform Edition) e Solaris 9 OS (SPARC e x86/x64 Platform Edition)
- Várias outras distribuições Linux

Configuração Mínima de Hardware

Nota: IDE NetBeans com resolução de tela em 1024x768 pixel

Sistema Operacional	Processador	Memória	HD Livre
Microsoft Windows	500 MHz Intel Pentium III workstation ou equivalente	512 MB	850 MB
Linux	500 MHz Intel Pentium III workstation ou equivalente	512 MB	450 MB
Solaris OS (SPARC)	UltraSPARC II 450 MHz	512 MB	450 MB
Solaris OS (x86/x64 Platform Edition)	AMD Opteron 100 Série 1.8 GHz	512 MB	450 MB
Mac OS X	PowerPC G4	512 MB	450 MB

Configuração Recomendada de Hardware

Sistema Operacional	Processador	Memória	HD Livre
Microsoft Windows	1.4 GHz Intel Pentium III workstation ou equivalente	1 GB	1 GB
Linux	1.4 GHz Intel Pentium III workstation ou equivalente	1 GB	850 MB
Solaris OS (SPARC)	UltraSPARC IIIi 1 GHz	1 GB	850 MB
Solaris OS (x86/x64 Platform Edition)	AMD Opteron 100 Series 1.8 GHz	1 GB	850 MB
Mac OS X	PowerPC G5	1 GB	850 MB

Requerimentos de Software

NetBeans Enterprise Pack 5.5 executando sobre Java 2 Platform Standard Edition Development Kit 5.0 ou superior (JDK 5.0, versão 1.5.0_01 ou superior), contemplando a Java Runtime Environment, ferramentas de desenvolvimento para compilar, depurar, e executar aplicações escritas em linguagem Java. Sun Java System Application Server Platform Edition 9.

- Para **Solaris**, **Windows**, e **Linux**, os arquivos da JDK podem ser obtidos para sua plataforma em http://java.sun.com/j2se/1.5.0/download.html
- Para **Mac OS X**, Java 2 Plataform Standard Edition (J2SE) 5.0 Release 4, pode ser obtida diretamente da Apple's Developer Connection, no endereço: http://developer.apple.com/java (é necessário registrar o download da JDK).

Para mais informações: http://www.netbeans.org/community/releases/55/relnotes.html

Colaboradores que auxiliaram no processo de tradução e revisão

Aécio Júnior
Alexandre Mori
Alexis da Rocha Silva
Allan Souza Nunes
Allan Wojcik da Silva
Anderson Moreira Paiva
Andre Neves de Amorim
Angelo de Oliveira
Antonio Jose R. Alves Ramos
Aurélio Soares Neto
Bruno da Silva Bonfim
Carlos Fernando Gonçalves
Denis Mitsuo Nakasaki
Fábio Bombonato
Fabrício Ribeiro Brigagão
Francisco das Chagas
Frederico Dubiel
Herivelto Gabriel dos Santos
Jacqueline Susann Barbosa
João Vianney Barrozo Costa
Kefreen Ryenz Batista Lacerda
Kleberth Bezerra G. dos Santos
Leandro Silva de Morais
Leonardo Ribas Segala
Lucas Vinícius Bibiano Thomé
Luciana Rocha de Oliveira
Luiz Fernandes de Oliveira Junior
Marco Aurélio Martins Bessa
Maria Carolina Ferreira da Silva
Massimiliano Giroldi
Mauro Cardoso Mortoni
Paulo Afonso Corrêa
Paulo Oliveira Sampaio Reis
Pedro Henrique Pereira de Andrade
Ronie Dotzlaw
Seire Pareja
Sergio Terzella
Vanessa dos Santos Almeida
Robson Alves Macêdo

Auxiliadores especiais

Revisão Geral do texto para os seguintes Países:

- **Brasil** – Tiago Flach
- **Guiné Bissau** – Alfredo Cá, Bunene Sisse e Buon Olossato Quebi – ONG Asas de Socorro

Coordenação do DFJUG

- **Daniel deOliveira** – JUGLeader responsável pelos acordos de parcerias
- **Luci Campos -** Idealizadora do DFJUG responsável pelo apoio social
- **Fernando Anselmo -** Coordenador responsável pelo processo de tradução e revisão, disponibilização dos materiais e inserção de novos módulos
- **Rodrigo Nunes -** Coordenador responsável pela parte multimídia
- **Sérgio Gomes Veloso -** Coordenador responsável pelo ambiente JEDI™ (Moodle)

Agradecimento Especial

John Paul Petines – Criador da Iniciativa JEDI™
Rommel Feria – Criador da Iniciativa JEDI™

1. Objetivos

Nesta lição, escreveremos e construiremos aplicações Java ME utilizando o emulador e empacotador de aplicações. A IDE que utilizaremos será o NetBeans 5.5 (www.netbeans.org) com o pacote adicional denominado *NetBeans Mobility Pack 5.5* (pode ser obtido no mesmo site).

Uma IDE (*Integrated Development Environment*) é um ambiente de programação com um construtor de GUI, um editor de código ou texto, um compilador e/ou um interpretador e um depurador. No nosso caso, o NetBeans vem também com um emulador de dispositivos. Isto permite ver como o programa parecerá num dispositivo real.

Ao final desta lição, o estudante será capaz de:

- Entender a estrutura de um *MIDlet*
- Criar um projeto *Mobile* no NetBeans
- Criar e executar um *MIDlet* no NetBeans

2. "Hello, world!" *MIDlet*

Vimos na lição anterior o ciclo de vida de um *MIDlet*. A vida de um *MIDlet* inicia quando ele é criado pelo *Application Management System* (AMS) do dispositivo. Inicialmente, ele fica no estado "pausado".

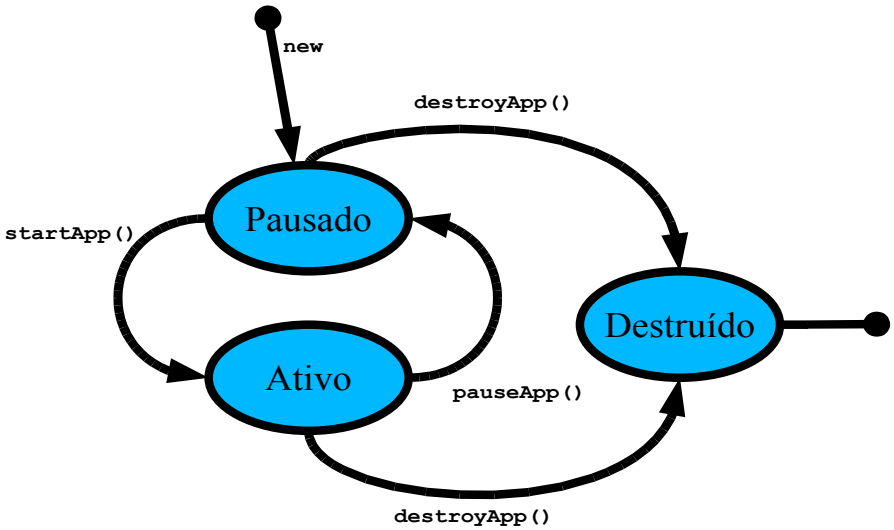

Figura 1: Ciclo de Vida do MIDlet

Para ser capaz de criar um *MIDlet*, devemos criar uma subclasse da classe *MIDlet* do pacote *javax.microedition.midlet*. Deve-se sobrescrever ou implementar os métodos: *startApp*(), *destroyApp*() e *pauseApp*(). Estes são os métodos esperados pelo AMS para se executar e controlar o *MIDlet*.

Diferentemente de um programa Java típico onde o método *main*() é executado automaticamente somente uma única vez na vida de um programa, o método *startApp*() pode ser chamado mais de uma vez durante o ciclo de vida do *MIDlet*. Então, não se deve colocar códigos de inicialização única no método *startApp*(). Para isso, deve ser criado um construtor para o *MIDlet* e as inicializações devem ser feitas nele.

Eis o código de nosso primeiro programa MIDP:

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

public class HelloMidlet extends MIDlet implements CommandListener {
    Display display;
    Command exitCommand = new Command("Exit", Command.EXIT, 1);
    Alert helloAlert;

    public HelloMidlet(){
        helloAlert = new Alert(
            "Hello MIDlet", "Hello, world!",
            null, AlertType.INFO
        );
        helloAlert.setTimeout(Alert.FOREVER);
        helloAlert.addCommand(exitCommand);
        helloAlert.setCommandListener(this);
    }
    public void startApp() {
        if (display == null){
            display = Display.getDisplay(this);
        }
        display.setCurrent(helloAlert);
```

```
    }
    public void pauseApp() {
    }
    public void destroyApp(boolean unconditional) {
    }
    public void commandAction(Command c, Displayable d){
        if (c == exitCommand){
            destroyApp(true);
            notifyDestroyed(); // Exit
        }
    }
}
```

A seguir, dissecaremos o primeiro *MIDlet*, focando em cada linha de código:

```
public class HelloMidlet extends MIDlet implements CommandListener {
```

Como dito anteriormente, é criada uma subclasse de *MIDlet* para criar nosso programa *MIDP*. Nessa linha, foi criada uma subclasse de *MIDlet* estendendo-a e dando o nome *HelloMidlet*. Além disso, implementaremos a interface *CommandListener*. Veremos sua função mais a frente.

```
    Display display;
    Command exitCommand = new Command("Exit", Command.EXIT, 1);
    Alert helloAlert;
```

Estes são os atributos do *MIDlet*. O objeto *Display* (existe somente um *display* associado por *MIDlet*) será utilizado para desenhar na tela do aparelho. O objeto *exitCommand* é um comando que poderemos inserir na aplicação, por exemplo, para que possamos finalizar o programa. Se não fosse colocado qualquer comando de saída, não haveria forma de finalizar um *MIDlet* de forma normal. Este será montado da seguinte forma:

```
    public HelloMidlet(){
        helloAlert = new Alert(
            "Hello MIDlet", "Hello, world!",
            null, AlertType.INFO
        );
        helloAlert.setTimeout(Alert.FOREVER);
        helloAlert.addCommand(exitCommand);
        helloAlert.setCommandListener(this);
    }
```

O construtor inicializa o objeto *Alert*. A classe *Alert* será discutida nas próximas lições. O método *addCommand*() do objeto *Alert* mostra um comando "Exit" na tela. O método *setCommandListener*() avisa o sistema para passar todos os eventos de comando para o *MIDlet*.

```
public class HelloMidlet extends MIDlet implements CommandListener {
```

O código "implements CommandListener" serve para controlar o pressionamento das teclas e comandos, de forma que o programa seja capaz de manipular eventos de "command". Se a classe implementar a interface CommandListener, deve ser criado o método commandAction(), como se segue:

```
    public void commandAction(Command c, Displayable d){
        if (c == exitCommand){
            destroyApp(true);
            notifyDestroyed(); // Exit
        }
    }
```

Usamos o *commandAction*() somente para as requisições de saída. Finalizamos a nossa classe utilizando o método *notifyDestroyed*() se o comando "Exit" for enviado.

```
    public void startApp() {
        if (display == null){
            display = Display.getDisplay(this);
        }
        display.setCurrent(helloAlert);
```

```
    }
```

Este é o ponto de entrada da nossa classe, uma vez que está pronta para ser iniciada pelo AMS. Lembre-se que o método *startApp*() pode ser incluído mais de uma vez no ciclo de vida do *MIDlet*. Se um *MIDlet* é pausado (*paused*), por exemplo, quando ocorre uma chamada telefônica, neste momento entramos em estado de pausa (*pauseApp*). Após a chamada o AMS pode retornar o nosso programa e executar novamente o método *startApp*(). O método estático *setCurrent*(), da classe *Display*, informa ao sistema que queremos que o objeto alerta seja mostrado na tela. Podemos iniciar o objeto da exibição chamando o método estático *getDisplay*() da mesma classe.

O Netbeans cria automaticamente um *Java Application Descriptor* (JAD) para o nosso projeto. E insere o arquivo JAD na pasta "dist" localizada na pasta do projeto. Este é um exemplo de arquivo JAD criado pelo Netbeans:

```
MIDlet-1: HelloMidlet, , HelloMidlet
MIDlet-Jar-Size: 1415
MIDlet-Jar-URL: ProjectHello.jar
MIDlet-Name: ProjectHello
MIDlet-Vendor: Vendor
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.0
```

Agora estamos prontos para compilar e rodar o nosso primeiro *MIDlet*.

3. Utilizando *Netbeans* e o *Mobility Pack*

O pré-requisito para esta lição, é o *Netbeans 5.5* e o *Mobility Pack* que devem ser instalados em seu computador. Ambos são arquivos executáveis bastando ao aluno seguir as telas guias para efetuar corretamente a instalação.

Passo 1: Iniciar criando um novo Projeto (Opção File | New Project...).

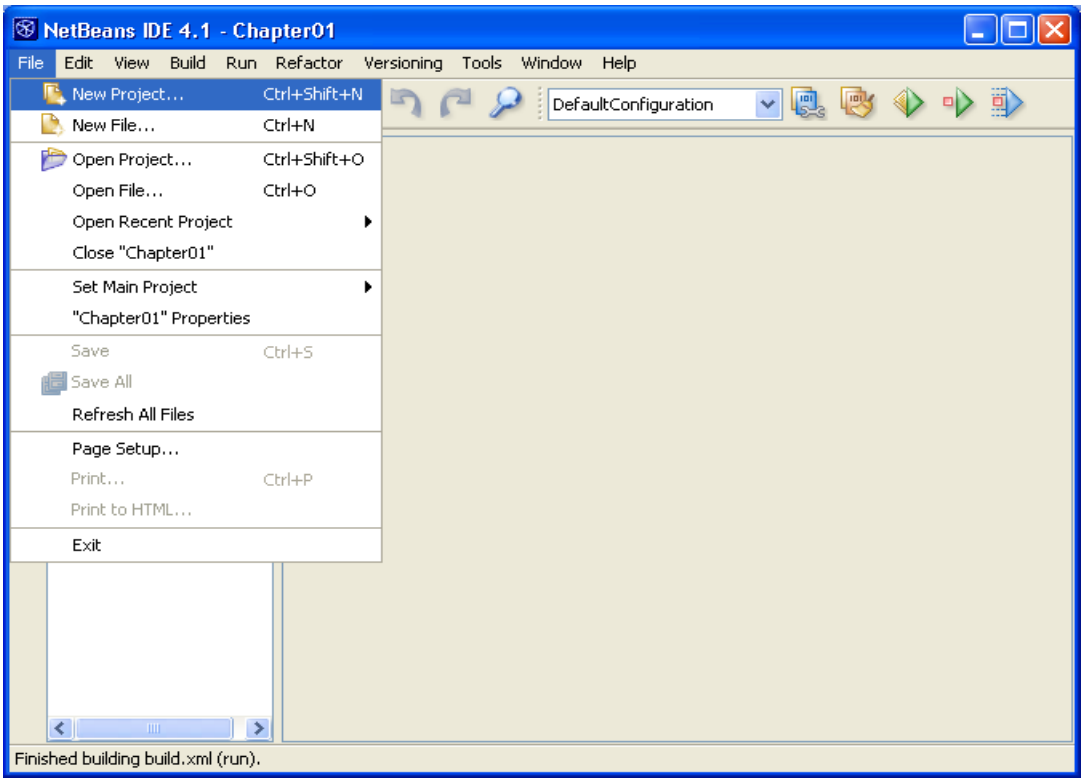

Figura 2: Menu File do NetBeans

Passo 2: Selecionar a Categoria "Mobile"

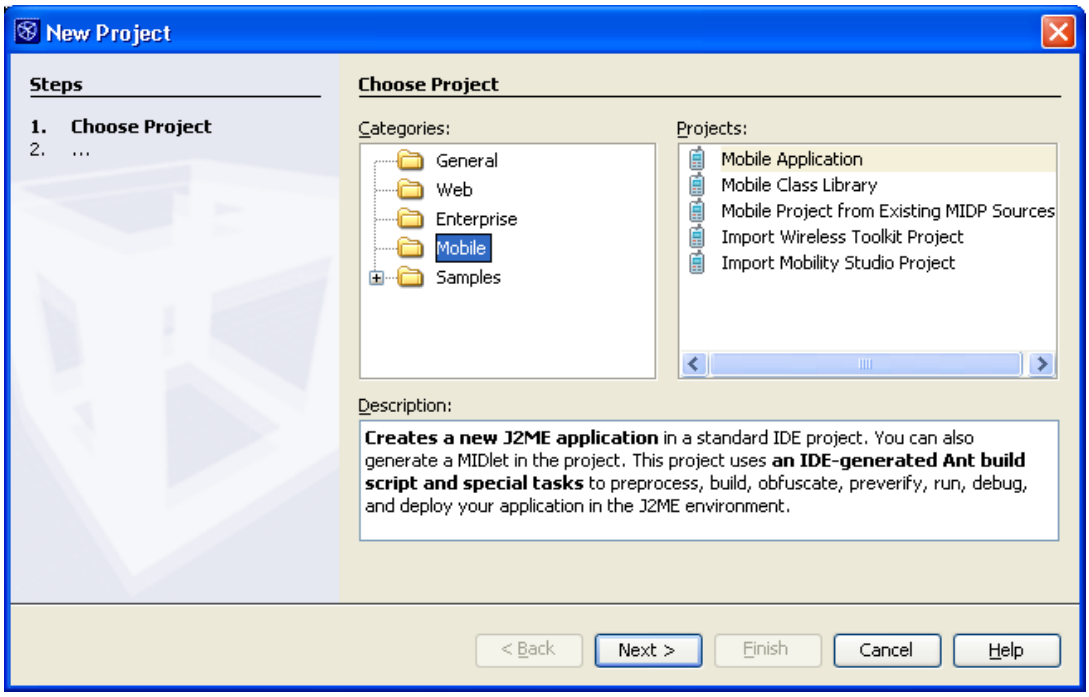

Figura 3: Janela New Project – Selecionando a categoria

Passo 3: Selecionar "Mobile Application" e pressionar o botão Next

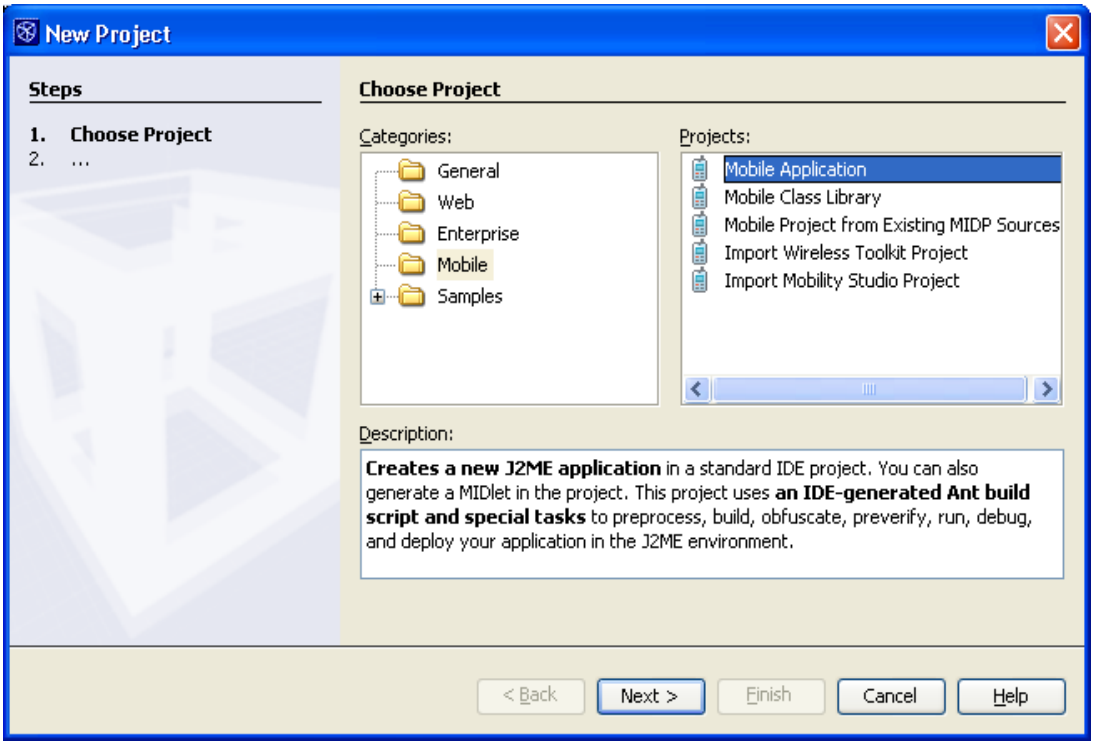

Figura 4: Janela New Project – Selecionando o tipo de projeto

Passo 4:Nomear o projeto e especificar sua localização

Desmarcar a opção "*Create Hello MIDlet*", criaremos nosso *MIDlet* em próximas lições

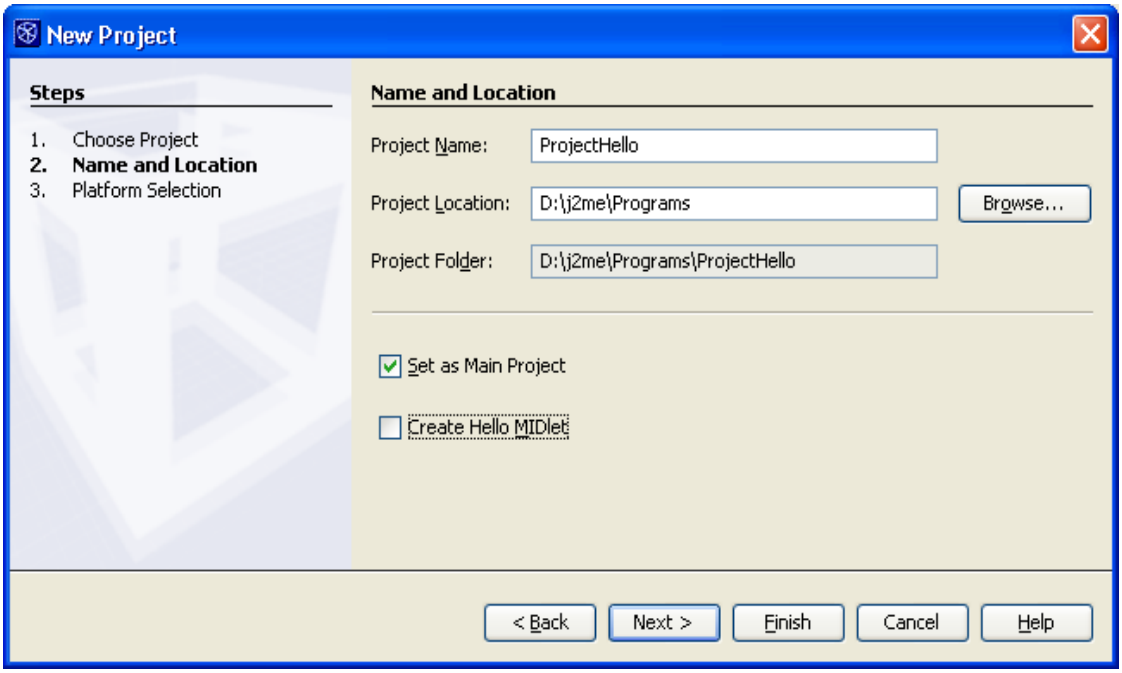

Figura 5: Janela New Project – Selecionando o nome e localização do projeto

Passo 5: Selecionar a configuração do projeto (opcional)

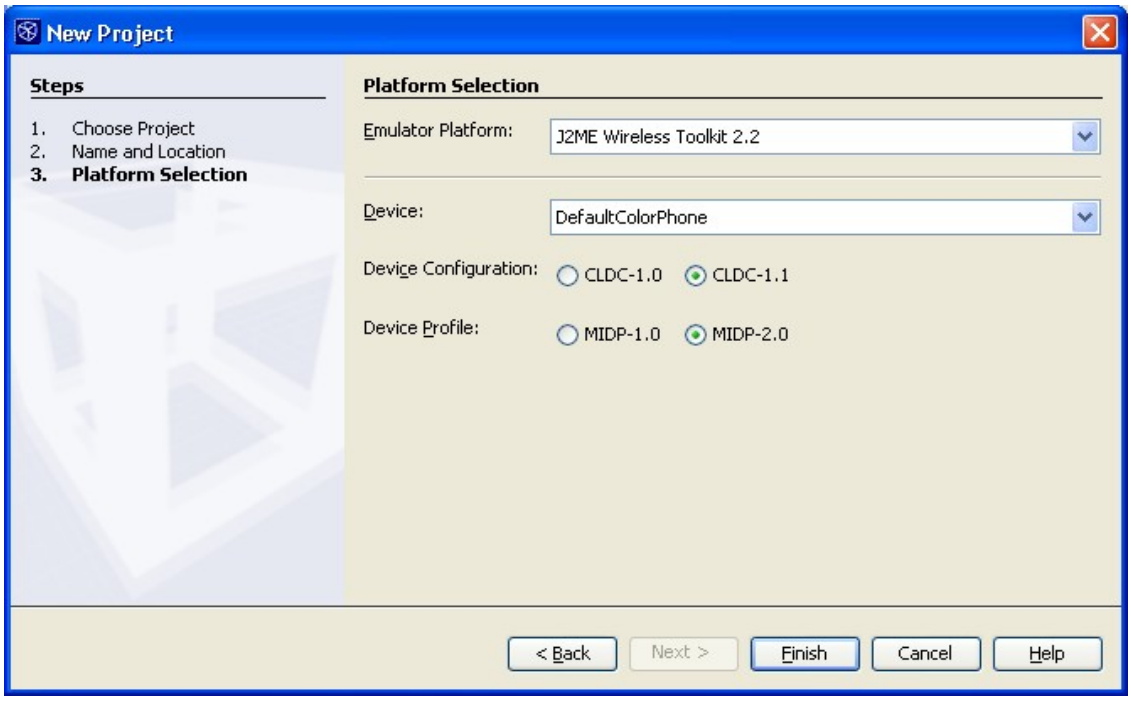

Figura 6: Janela New Project – Selecionando a configuração do projeto

Etapa 6: Criar um novo *MIDlet*

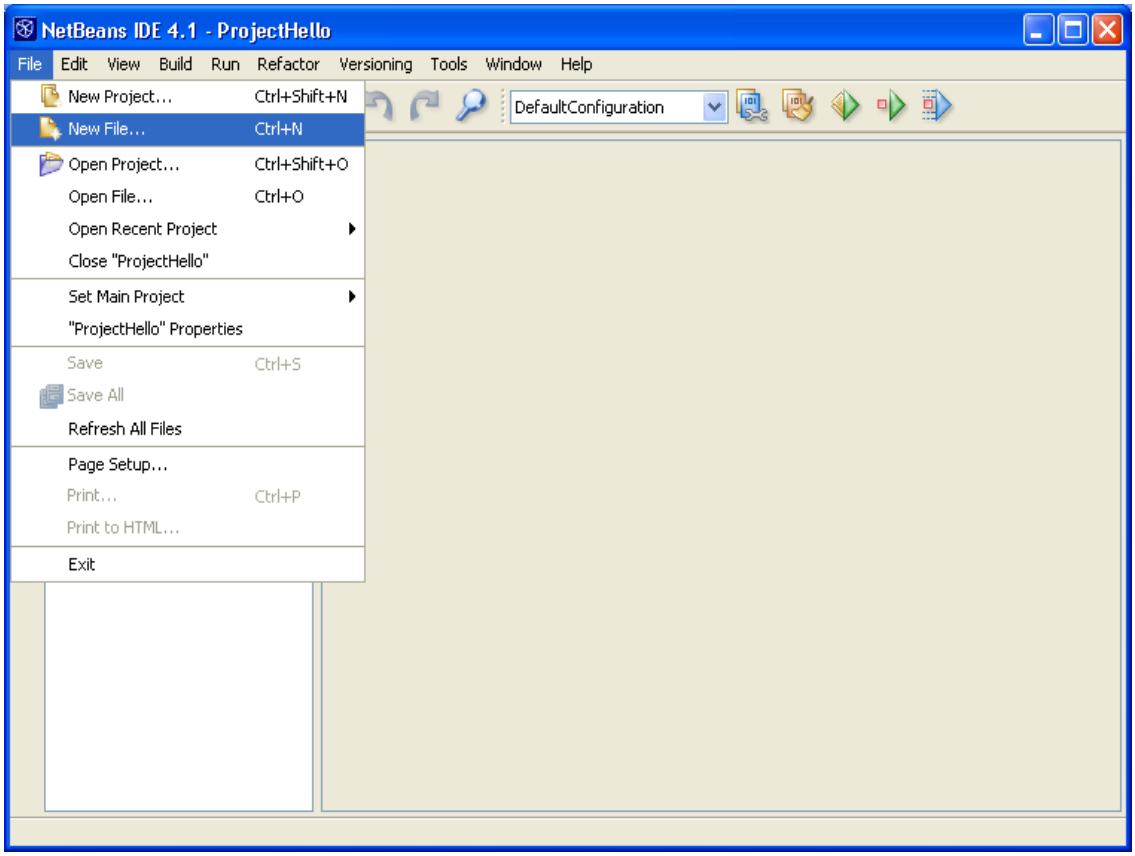

Figura 7: Menu File do NetBeans

Etapa 7: Selecionar na opção *Categories* a opção *MIDP* e em *File Types* a opção *MIDlet*

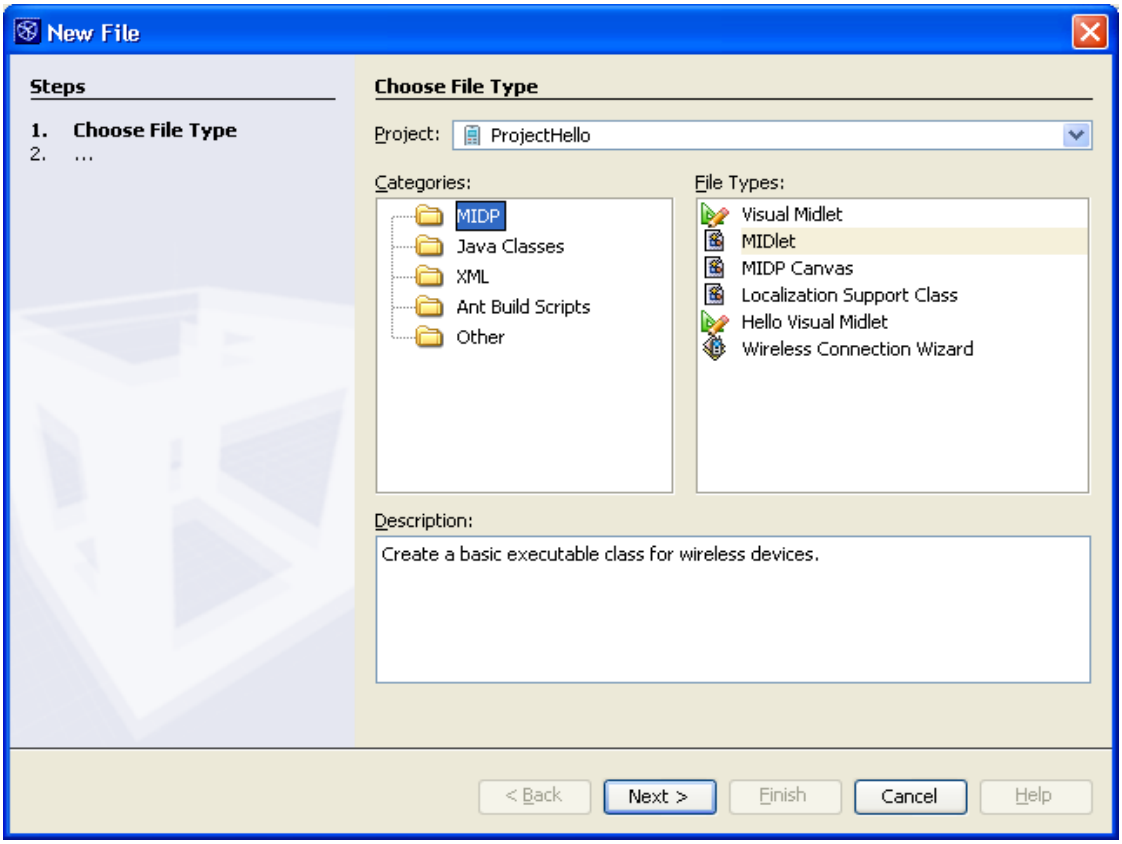

Figura 8: Janela New File – Selecionando o tipo do arquivo

Etapa 8: Informar o nome do *MIDlet* (*HelloMidlet*) e pressionar o botão ***Finish***

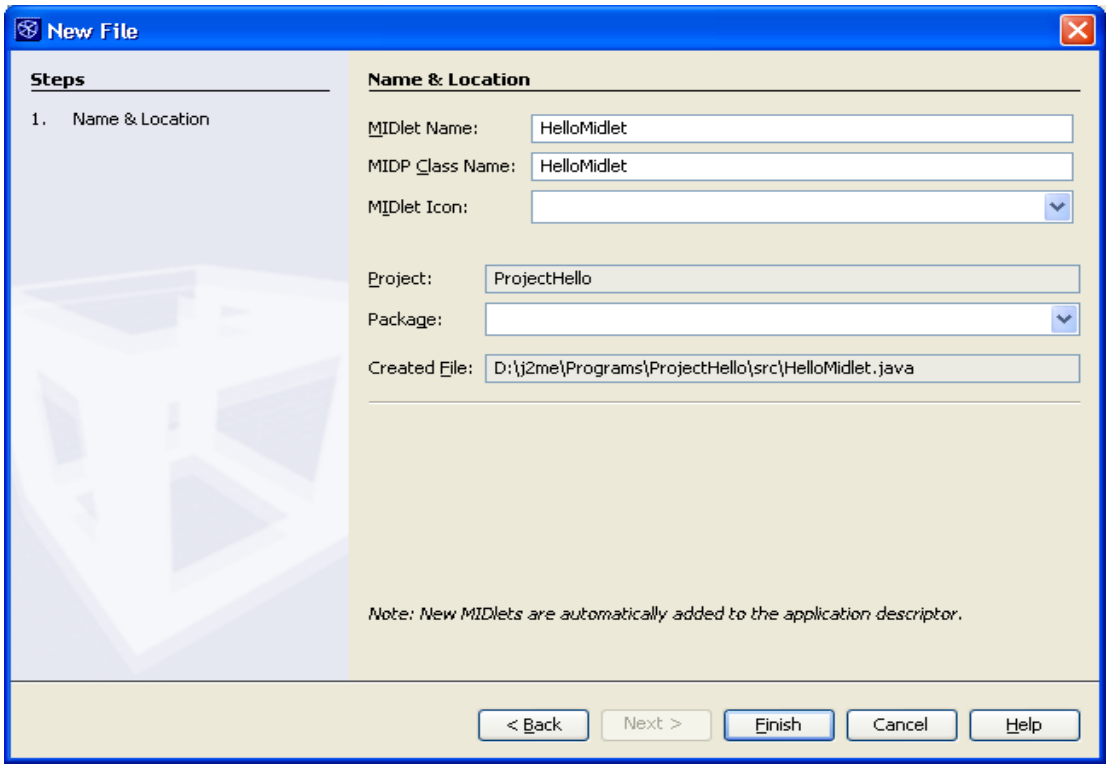

Figura 9: Janela New Project – Selecionando o nome e localização do projeto

Passo 9: Modificar o código do programa conforme descrito anteriormente

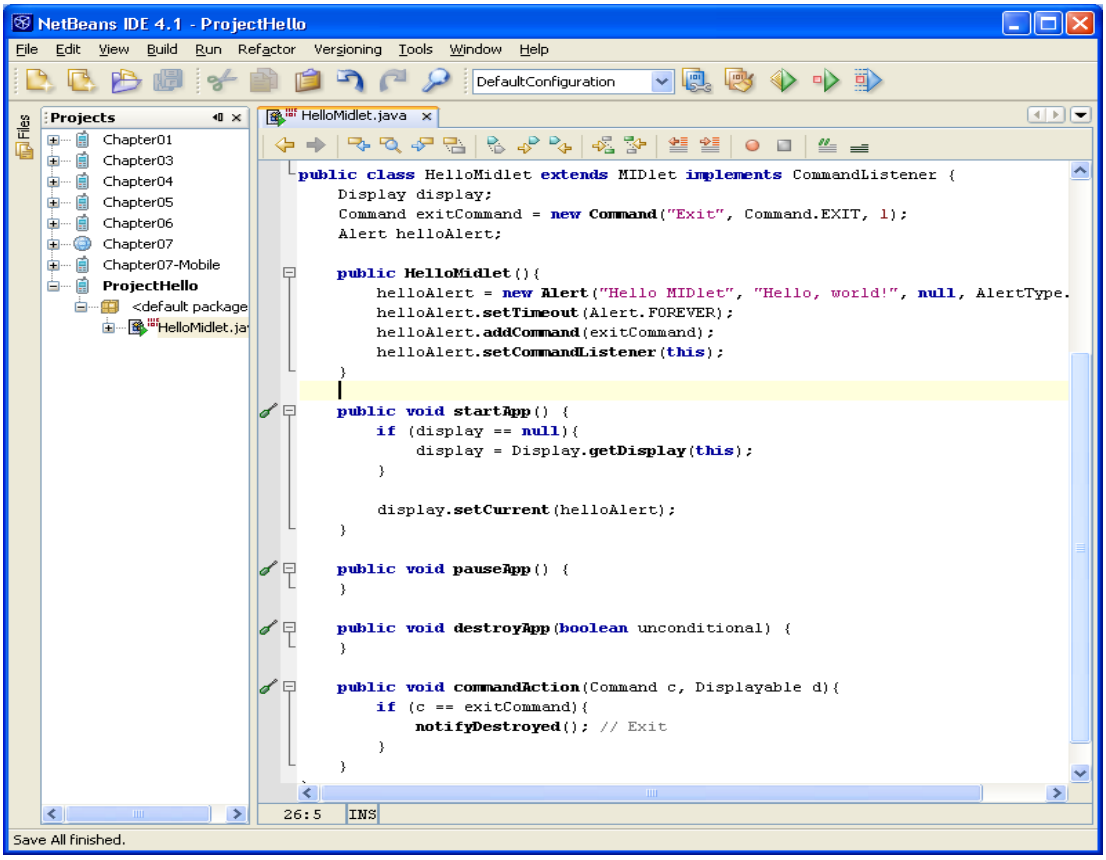

Figura 10: Janela New Project – Selecionando o nome e localização do projeto

Passo 10: Compilar e Executar o *MIDlet* de maneira idêntica à vista nos projetos anteriores

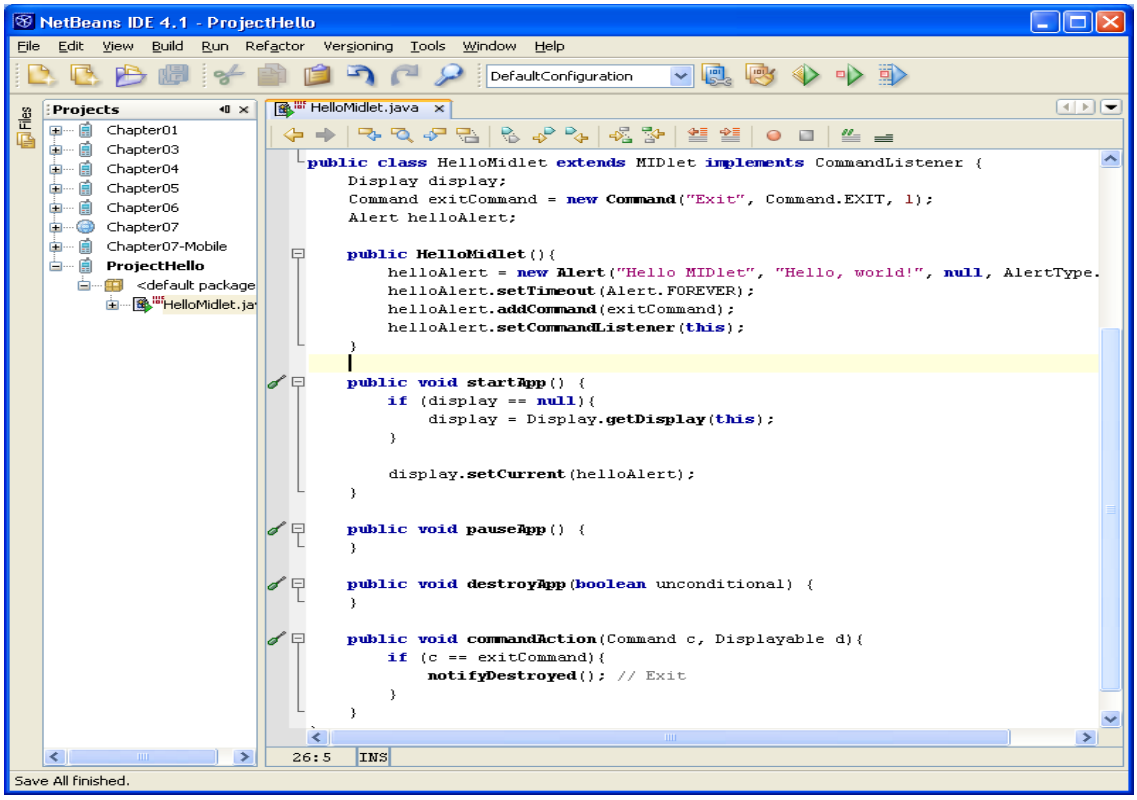

Figura 11: Janela New Project – Selecionando o nome e localização do projeto

Passo 11: Testar o *MIDlet* no Emulador

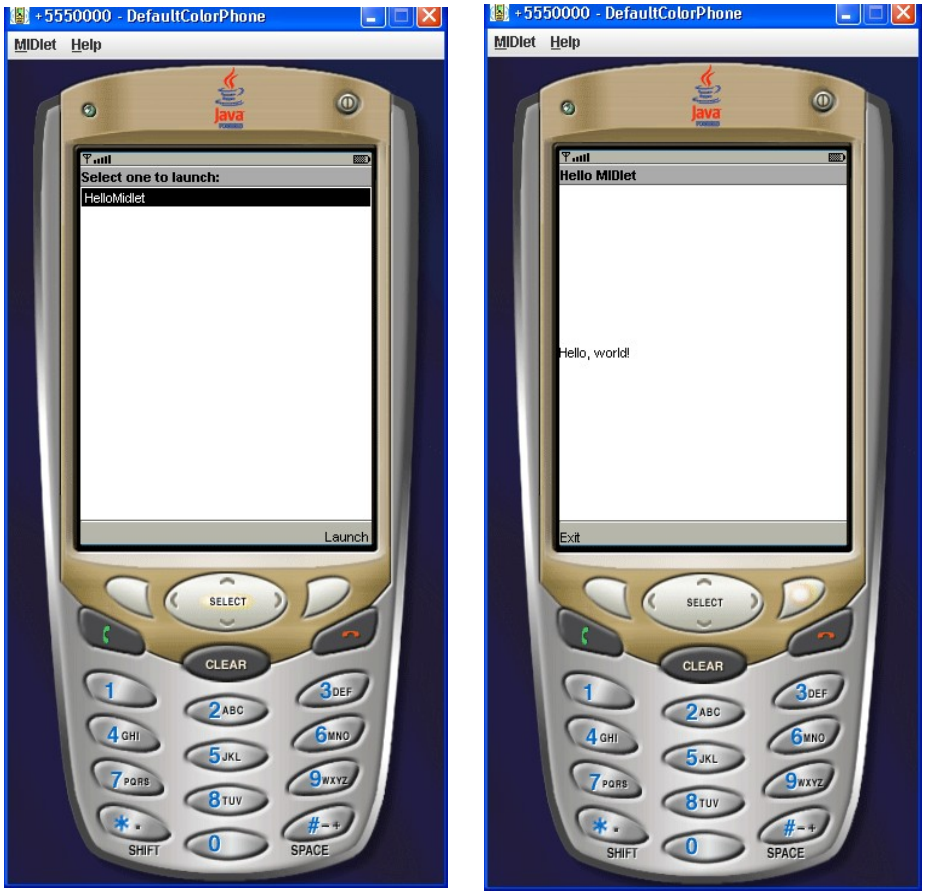

Figura 12: Janelas do Emulador da Aplicação

4. Exercícios

4.1. Múltiplos MIDlets em uma suite MIDlet

Adicionar um novo *MIDlet* ao projeto *ProjectHello*. Observar que o Netbeans adiciona automaticamente um novo *MIDlet* no arquivo de aplicações JAD quando é utilizado o auxílio "New File..."

Parceiros que tornaram JEDI™ possível

Instituto CTS
Patrocinador do DFJUG.

Sun Microsystems
Fornecimento de servidor de dados para o armazenamento dos vídeo-aulas.

Java Research and Development Center da Universidade das Filipinas
Criador da Iniciativa JEDI™.

DFJUG
Detentor dos direitos do JEDI™ nos países de língua portuguesa.

Banco do Brasil
Disponibilização de seus *telecentros* para abrigar e difundir a Iniciativa JEDI™.

Politec
Suporte e apoio financeiro e logístico a todo o processo.

Borland
Apoio internacional para que possamos alcançar os outros países de língua portuguesa.

Instituto Gaudium/CNBB
Fornecimento da sua infra-estrutura de hardware de seus servidores para que os milhares de alunos possam acessar o material do curso simultaneamente.