Módulo 5

Desenvolvimento de Aplicações Móveis

Lição 10

Outros Tópicos

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

Timers e *TimeTasks* permitem programar tarefas para que sejam executadas em um horário determinado. A tarefa pode ainda ser programada para que se repita num intervalo de tempo designado pelo programador.

Ao final desta lição, o estudante será capaz de:

- Programar tarefas utilizando *Timers* (marcadores)
- Registrar o recebimento de conexões no Registro

2. Timers

É possível criar uma tarefa utilizando-se a herança, estendendo a classe *TimerTask* e implementando o método *run*(). Este método será executado baseado na programação do *Timer*.

```
class CounterTask extends TimerTask {
  int counter = 0;
  public void run() {
    System.out.println("Counter: " + counter++);
  }
}
```

Para programar uma tarefa, cria-se um objeto do tipo *Timer* e utiliza-se o método *schedule*() do *Timer* para que se possa configurar o andamento da tarefa. Cada *Timer* roda em uma *thread* independente. O método *schedule*() possui várias formas diferentes. É possível especificar o tempo de início para a tarefa utilizando um tempo em milissegundos ou especificando uma data absoluta (*java.util.Date*). O terceiro parâmetro para o método é o período de repetição da tarefa. Se o período de repetição for especificado, a tarefa será executada a cada "período" de milissegundos definido.

```
Timer timer = new Timer();
TimerTask task = new CounterTask();
// inicia a tarefa em 8 segundos, e repete a cada segundo
timer.schedule(task, 8000, 1000);
```

Para parar a execução do objeto *Timer*, utiliza-se o método *close*(). Isto fará com que a *Thread* contendo o *Timer* seja interrompida e a tarefa agendada, descartada. Lembre-se sempre de que uma vez que o *Timer* tenha sido parado, não poderá ser reiniciado.

`void schedule( TimerTask task, long delay)`	Agendar a tarefa para que seja executada a cada período (em milissegundos).
`void schedule( TimerTask task, long delay, long period)`	Agendar a tarefa para que ela seja repetida, começando após o período especificado (em milissegundos).
`void schedule( TimerTask task, Date time)`	Agenda a tarefa para que seja executada num tempo específico.
`void schedule( TimerTask task, Date time, long period)`	Agenda a tarefa para que ela seja repetida, começando pelo tempo especificado.
`void cancel()`	Interrompe o *timer* e descarta qualquer tarefa agendada.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.*;
import java.util.Timer;
import java.util.TimerTask;
import java.util.Date;

public class TimerMidlet extends MIDlet implements CommandListener{
    private Command exitCommand;
    private Form form;
    private StringItem textField;
    private Display display;
    private Timer timer;

    public TimerMidlet() {
        exitCommand = new Command("Exit", Command.EXIT, 1);
        textField = new StringItem("Counter", "");
        timer = new Timer();
```

```
        TimerTask task = new CounterTask(this);
        timer.schedule(task, 2000, 1000);
        form = new Form("Timer Test");
        form.addCommand(exitCommand);
        form.append(textField);
    }
    public void startApp() {
        display = Display.getDisplay(this);
        form.setCommandListener(this);
        display.setCurrent(form);
    }
    public void pauseApp() {}
    public void destroyApp(boolean unconditional) {
        timer.cancel();
    }
    public void commandAction(Command c, Displayable d) {
        if (c == exitCommand) {
            destroyApp(true);
            notifyDestroyed();
        }
    }
    public void setText(String text){
        textField.setText(text);
    }
}
class CounterTask extends TimerTask {
    int counter = 0;
    TimerMidlet midlet;

    public CounterTask(TimerMidlet midlet){
        this.midlet = midlet;
    }
    public void run() {
        counter++;
        midlet.setText("" + counter);
        System.out.println("Counter: " + counter);
    }
}
```

3. Registro de Conexões

Permite que os *MIDlets* registrem as conexões com o software de gerenciamento da aplicação (AMS). Se o programa não estiver em execução, o AMS ficará esperando por conexões nos limites dos endereços registrados pelas aplicações. Quase todos os tipos de conexões são suportadas, incluindo *SMS, CBS* e *AMS*.

É possível fazer o registro de novas conexões de duas maneiras: da maneira estática, utilizando o arquivo *application descriptor* (JAD), ou dinamicamente durante o tempo de execução, utilizando a *API PushRegistry*.

Iremos demostrar como utilizar a *API PushRegistry*. Pressionar o botão direito do mouse sobre o *Project Name* e selecionar *properties* para abrir a página de propriedades do projeto. Selecionar a opção de *Push Registry*.

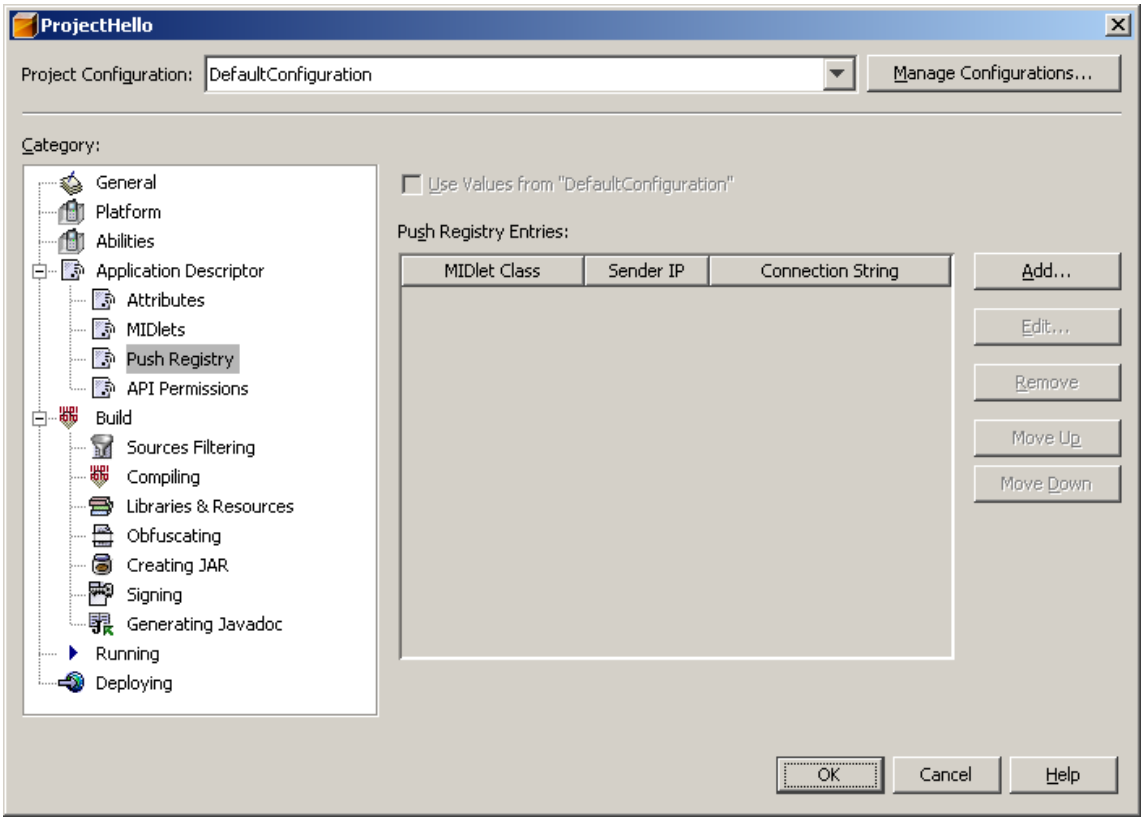

Figura 1: Janela de Propriedades do Projeto

Pressionar o botão *add...* para registrar uma nova conexão:

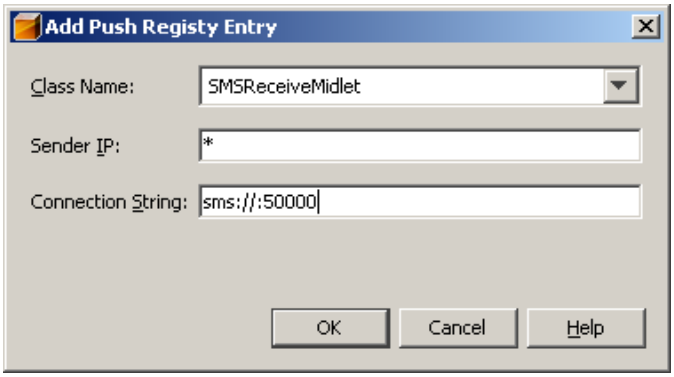

Figura 2: Adicionando um novo registro

```
Class Name: SMSReceiveMidlet
Sender IP: *
```

```
Connection String: sms://:50000
```

Verifique a conexão adicionada na figura abaixo.

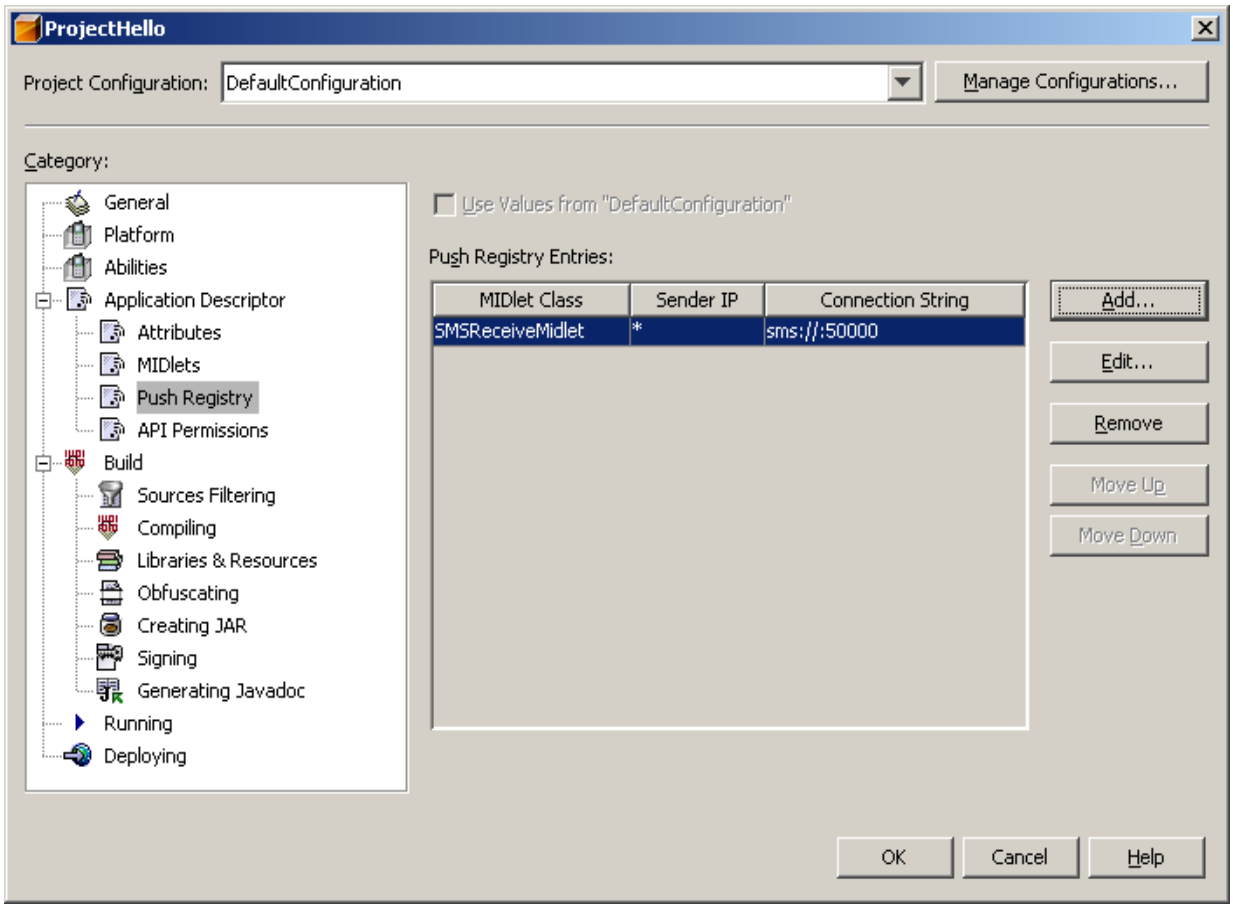

Figura 3: Registro da Conexão

Selecionar a opção *API Permissions* e pressionar o botão *Add* para inserir uma permissão do *MIDlet* para uma biblioteca em particular.

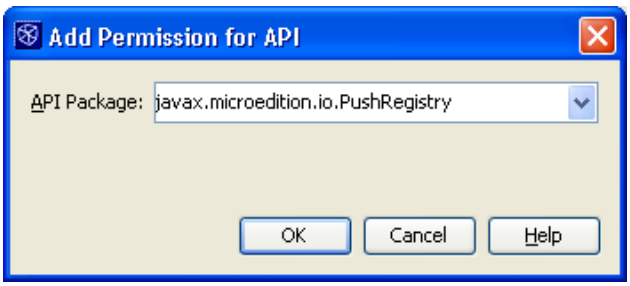

Figura 4: Adicionando permissões

Adicionar as seguintes bibliotecas*:*

```
javax.microedition.io.PushRegistry
javax.microedition.io.Connector.sms
javax.wireless.messaging.sms.receive
javax.wireless.messaging.sms.send
```

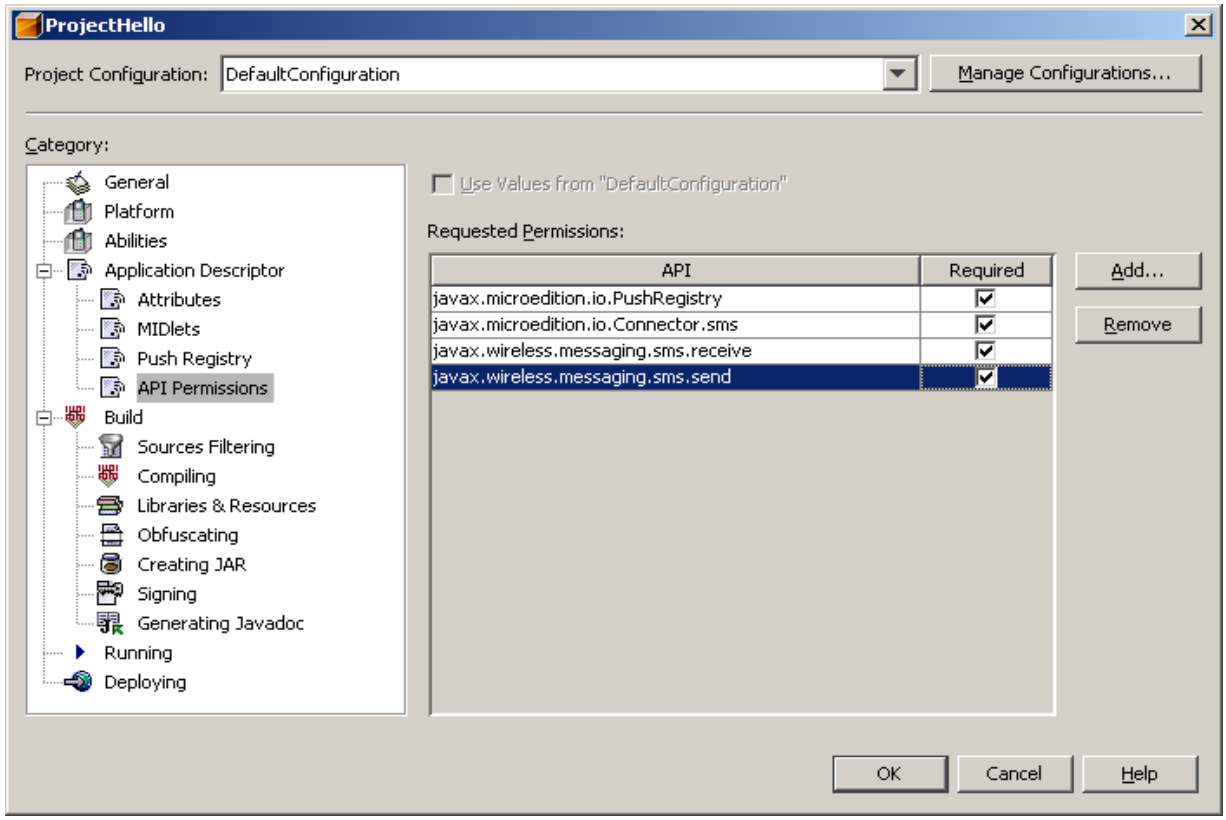

Figura 5: Permissões adicionadas

Para finalizar, criar um atributo para configurar automaticamente a porta, selecionar *Attributes* conforme a seguinte janela:

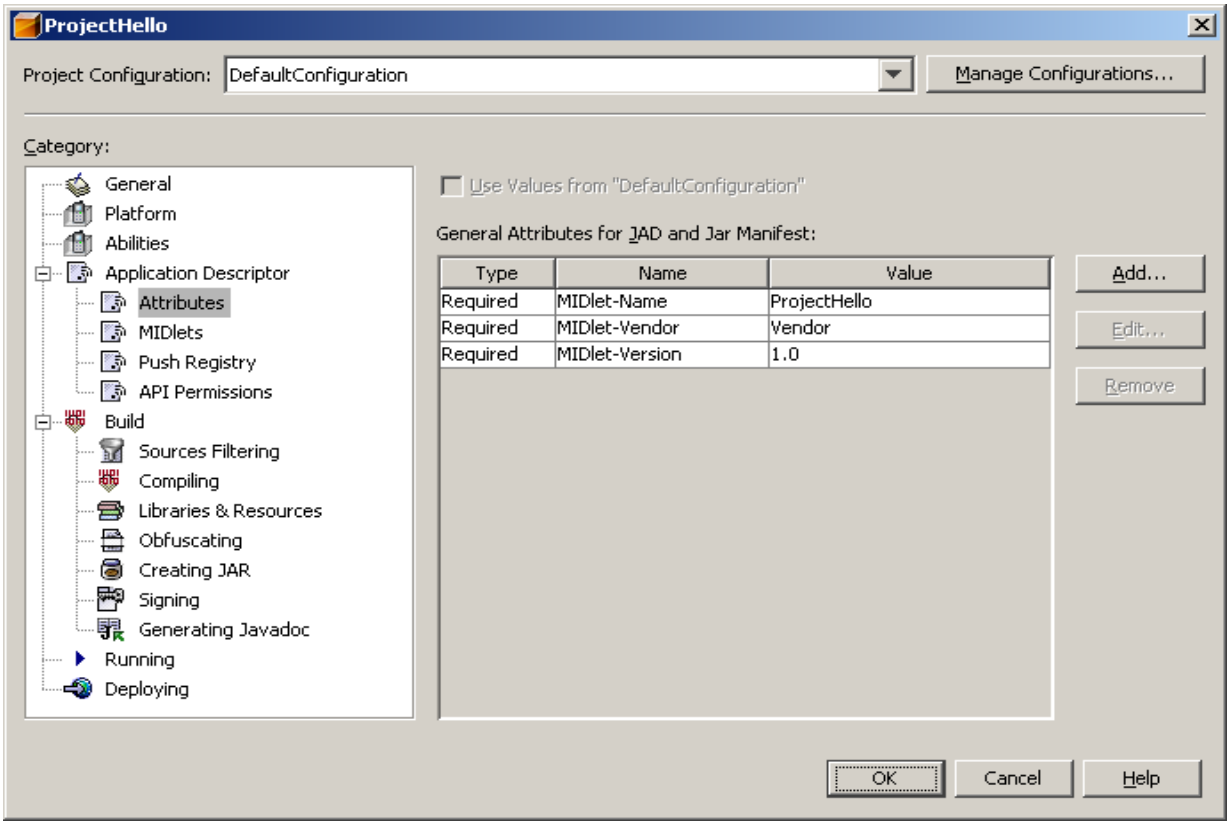

Figura 6: Adicionar novos atributos

Pressionar o botão *Add...* para adicionar um novo atributo do tipo *Custom*.

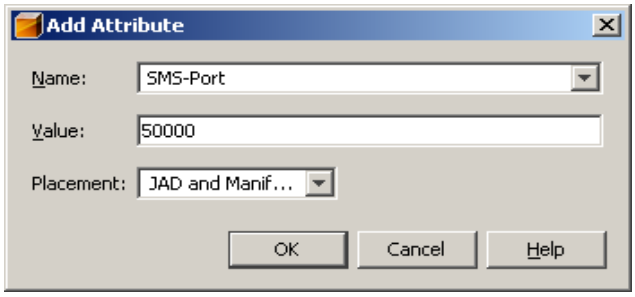

Figura 7: Adicionando o atributo

Encerre a janela de configuração do projeto pressionando o botão OK e execute a suite do *MIDlet*. Executar a aplicação duas vezes para conseguir dois emuladores conforme as figuras:

Figura 8: Emulador do aplicativo

Observe que o primeiro telefone possui o número +5550000 e o segundo +5550001. Selecionar a opção *SMSReceiveMidlet* no primeiro telefone. Uma mensagem solicitando a autorização para comunicação via mensagem será mostrada, selecionar a opção *Yes* e deixar em modo de espera:

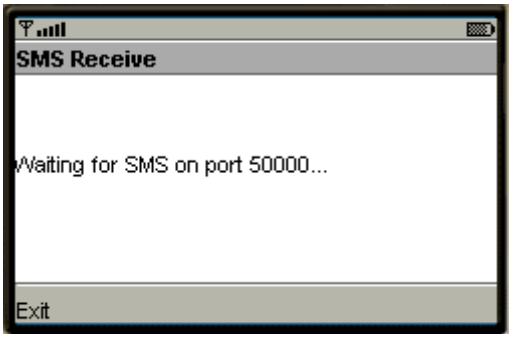

Figura 9: Modo de Espera da Mensagem

No segundo telefone, selecionar *SMSSendMidlet*, informar o endereço **5550000** e pressionar OK.

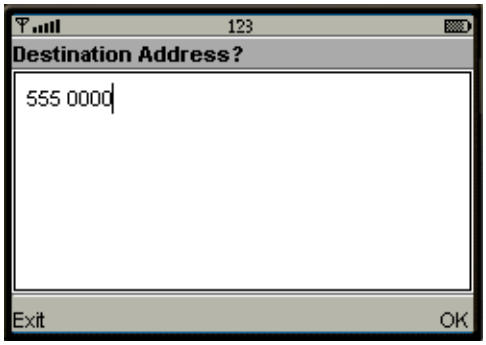

Figura 10: Emulador do aplicativo

Digitar uma mensagem e pressionar Send:

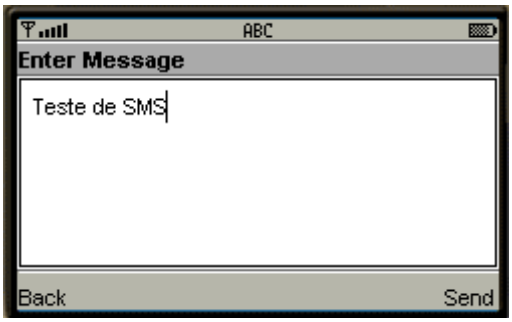

Figura 11: Janela de confirmação

Responder afirmativamente à mensagem de confirmação da comunicação:

Figura 12: Confirmação da Comunicação

Responder afirmativamente a próxima mensagem e aguardar o envio. Observar no primeiro telefone o recebimento da mensagem.

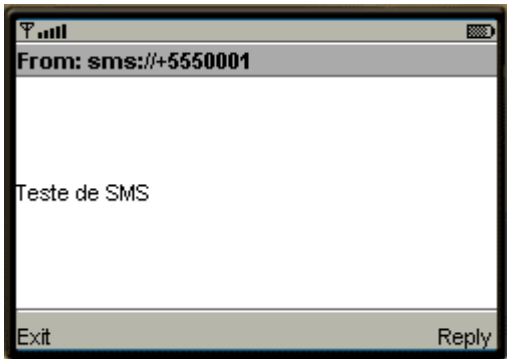

Figura 13: Aplicativo concluído

3.1. Programas

Classe Auxiliar

```
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.wireless.messaging.*;
import java.io.IOException;

public class SMSSender implements CommandListener, Runnable {
  Command sendCommand = new Command("Send", Command.OK, 1);
  Command backCommand = new Command("Back", Command.BACK, 2);
  Display display;
  String smsPort;
  String destinationAddress;
  TextBox messageBox;
  Displayable backScreen;
  Displayable sendingScreen;

  public SMSSender(String smsPort, Display display,
    Displayable backScreen, Displayable sendingScreen) {
    this.smsPort = smsPort;
    this.display = display;
    this.destinationAddress = null;
    this.backScreen = backScreen;
    this.sendingScreen = sendingScreen;
    messageBox = new TextBox("Enter Message", null, 65535, TextField.ANY);
    messageBox.addCommand(backCommand);
    messageBox.addCommand(sendCommand);
    messageBox.setCommandListener(this);
  }
  public void promptAndSend(String destinationAddress) {
    this.destinationAddress = destinationAddress;
    display.setCurrent(messageBox);
  }
  public void commandAction(Command c, Displayable s) {
    try {
      if (c == backCommand) {
        display.setCurrent(backScreen);
      } else if (c == sendCommand) {
        display.setCurrent(sendingScreen);
        new Thread(this).start();
      }
    } catch (Exception ex) {
      ex.printStackTrace();
    }
  }
  public void run() {
    String address = destinationAddress + ":" + smsPort;
    MessageConnection smsconn = null;
    try {
      smsconn = (MessageConnection)Connector.open(address);
      TextMessage txtmessage = (TextMessage)smsconn.newMessage(
        MessageConnection.TEXT_MESSAGE);
      txtmessage.setAddress(address);
      txtmessage.setPayloadText(messageBox.getString());
      smsconn.send(txtmessage);
    } catch (Throwable t) {
      System.out.println("Send caught: ");
      t.printStackTrace();
    }
    if (smsconn != null) {
      try {
        smsconn.close();
      } catch (IOException ioe) {
        System.out.println("Closing connection caught: ");
```

```
            ioe.printStackTrace();
          }
        }
      }
    }
```

Midlet para receber mensagem *SMS*

```
import javax.microedition.midlet.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.wireless.messaging.*;
import java.io.IOException;

public class SMSReceiveMidlet extends MIDlet
    implements CommandListener, Runnable, MessageListener {
    Command exitCommand  = new Command("Exit", Command.EXIT, 2);
    Command replyCommand = new Command("Reply", Command.OK, 1);
    Alert content;
    Display display;
    Thread thread;
    String[] connections;
    boolean done;
    String smsPort;
    MessageConnection smsconn;
    Message msg;
    String senderAddress;
    Alert sendingMessageAlert;
    SMSSender sender;
    Displayable resumeScreen;

    public SMSReceiveMidlet() {
        smsPort = getAppProperty("SMS-Port");
        display = Display.getDisplay(this);
        content = new Alert("SMS Receive");
        content.setTimeout(Alert.FOREVER);
        content.addCommand(exitCommand);
        content.setCommandListener(this);
        content.setString("Receiving...");
        sendingMessageAlert = new Alert("SMS", null, null, AlertType.INFO);
        sendingMessageAlert.setTimeout(5000);
        sendingMessageAlert.setCommandListener(this);
        sender = new SMSSender(smsPort, display, content, sendingMessageAlert);
        resumeScreen = content;
    }
    public void startApp() {
        String smsConnection = "sms://:" + smsPort;
        if (smsconn == null) {
            try {
                smsconn = (MessageConnection) Connector.open(smsConnection);
                smsconn.setMessageListener(this);
            } catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        connections = PushRegistry.listConnections(true);
        if (connections == null || connections.length == 0) {
            content.setString("Waiting for SMS on port " + smsPort + "...");
        }
        done = false;
        thread = new Thread(this);
        thread.start();

        display.setCurrent(resumeScreen);
    }
    public void notifyIncomingMessage(MessageConnection conn) {
        if (thread == null) {
```

```
            done = false;
            thread = new Thread(this);
            thread.start();
        }
    }
    public void run() {
        try {
            msg = smsconn.receive();
            if (msg != null) {
                senderAddress = msg.getAddress();
                content.setTitle("From: " + senderAddress);
                if (msg instanceof TextMessage) {
                    content.setString(((TextMessage)msg).getPayloadText());
                } else {
                    StringBuffer buf = new StringBuffer();
                    byte[] data = ((BinaryMessage)msg).getPayloadData();
                    for (int i = 0; i < data.length; i++) {
                        int intData = (int)data[i] & 0xFF;
                        if (intData < 0x10) {
                            buf.append("0");
                        }
                        buf.append(Integer.toHexString(intData));
                        buf.append(' ');
                    }
                    content.setString(buf.toString());
                }
                content.addCommand(replyCommand);
                display.setCurrent(content);
            }
        } catch (IOException e) {
            // e.printStackTrace();
        }
    }
    public void pauseApp() {
        done = true;
        thread = null;
        resumeScreen = display.getCurrent();
    }
    public void destroyApp(boolean unconditional) {
        done = true;
        thread = null;
        if (smsconn != null) {
            try {
                smsconn.close();
            } catch (IOException e) {
                // Ignora erros no caso de finalização
            }
        }
    }
    public void commandAction(Command c, Displayable s) {
        try {
            if (c == exitCommand || c == Alert.DISMISS_COMMAND) {
                destroyApp(false);
                notifyDestroyed();
            } else if (c == replyCommand) {
                reply();
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    private void reply() {
        String address = senderAddress.substring(6);
        String statusMessage = "Sending message to " + address + "...";
        sendingMessageAlert.setString(statusMessage);
        sender.promptAndSend(senderAddress);
    }
```

```
}
```

Midlet para enviar mensagem *SMS*

```
import javax.microedition.midlet.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.wireless.messaging.*;
import java.io.IOException;

public class SMSSendMidlet extends MIDlet implements CommandListener {
    Command exitCommand  = new Command("Exit", Command.EXIT, 2);
    Command okCommand = new Command("OK", Command.OK, 1);
    Display display;
    String smsPort;
    TextBox destinationAddressBox;
    Alert errorMessageAlert;
    Alert sendingMessageAlert;
    SMSSender sender;
    Displayable resumeScreen = null;

    public SMSSendMidlet() {
        smsPort = getAppProperty("SMS-Port");
        display = Display.getDisplay(this);
        destinationAddressBox = new TextBox("Destination Address?",
            null, 256, TextField.PHONENUMBER);
        destinationAddressBox.addCommand(exitCommand);
        destinationAddressBox.addCommand(okCommand);
        destinationAddressBox.setCommandListener(this);
        errorMessageAlert = new Alert("SMS", null, null, AlertType.ERROR);
        errorMessageAlert.setTimeout(5000);
        sendingMessageAlert = new Alert("SMS", null, null, AlertType.INFO);
        sendingMessageAlert.setTimeout(5000);
        sendingMessageAlert.setCommandListener(this);
        sender = new SMSSender(smsPort, display, destinationAddressBox,
            sendingMessageAlert);
        resumeScreen = destinationAddressBox;
    }
    public void startApp() {
        display.setCurrent(resumeScreen);
    }
    public void pauseApp() {
        resumeScreen = display.getCurrent();
    }
    public void destroyApp(boolean unconditional) {
    }
    public void commandAction(Command c, Displayable s) {
        try {
            if (c == exitCommand || c == Alert.DISMISS_COMMAND) {
                destroyApp(false);
                notifyDestroyed();
            } else if (c == okCommand) {
                promptAndSend();
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    private void promptAndSend() {
        String address = destinationAddressBox.getString();
        if (!SMSSendMidlet.isValidPhoneNumber(address)) {
            errorMessageAlert.setString("Invalid phone number");
            display.setCurrent(errorMessageAlert, destinationAddressBox);
            return;
        }
        String statusMessage = "Sending message to " + address + "...";
        sendingMessageAlert.setString(statusMessage);
```

```
            sender.promptAndSend("sms://" + address);
        }
        private static boolean isValidPhoneNumber(String number) {
            char[] chars = number.toCharArray();
            if (chars.length == 0) {
                return false;
            }
            int startPos = 0;
            if (chars[0] == '+') {
                startPos = 1;
            }
            for (int i = startPos; i < chars.length; ++i) {
                if (!Character.isDigit(chars[i])) {
                    return false;
                }
            }
            return true;
        }
    }
```

4. Exercícios

4.1. Relógio no Celular

Criar um MIDlet que mostre a data e hora atual e que seja atualizada a cada segundo. Utilize um *Timer* para atualizar a data e hora e um *StringItem* para exibi-las.

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.