Módulo 5

Desenvolvimento de Aplicações Móveis

Lição 9

Pacotes Opcionais

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, iremos aprofundar em como escrever, construir, utilizar o emulador e o empacotador de aplicações J2ME. O ambiente de programação que iremos utilizar será o *Netbeans*.

Nem todos os dispositivos são criados de maneira semelhante, pois cada um deles possui características diferentes. Por isso pode ser muito complicado para se criar uma especificação padrão que atenda a todos os dispositivos.

Para acomodar as diferentes capacidades de cada dispositivo, a tecnologia *MIDP* definiu vários pacotes opcionais. Esses pacotes são específicos para atender dispositivos específicos que tenham esses recursos. Iremos aprender como utilizar a *Mobile Media API* (*MMAPI*) e a *Wireless Messaging API* (*WMA*).

Ao final desta lição, o estudante será capaz de:

- Saber quais são as funcionalidade oferecidas pela Mobile Media API
- Reproduzir tons simples
- Reproduzir um arquivo de áudio de uma rede e de um JAR
- Enviar e receber mensagens SMS

2. Mobile Media API (MMAPI)

A *Mobile Media API* (*MMAPI*) permite-nos gerar tons, tocar e gravar áudio e vídeo nos dispositivos compatíveis. A reprodução e a gravação de mídia são tratadas por dois objetos: o *DataSource* e o *Player*.

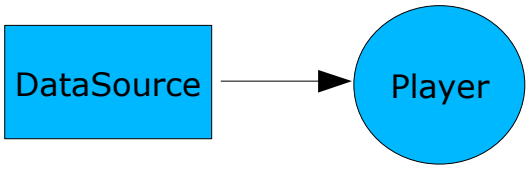

Figura 1: Relação entre DataSource e Player

O *DataSource* trata de detalhes de como obter o dado de uma fonte. A fonte pode ser um arquivo de um JAR ou de uma rede (via *HTTP*), um registro de um RMS, uma conexão de *streaming* de um servidor ou outra fonte proprietária. O *Player* não tem o que se preocupar sobre de onde o dado vem ou de que uma maneira ele pode ser obtido. Tudo o que o *Player* necessita fazer é ler um dado de um *DataSource*, processar e exibir ou reproduzir a mídia para o dispositivo de saída.

O terceiro ator na nossa cena é o *Manager*. O *Manager* cria *players* de *DataSources*. O *Manager* tem métodos para criar *Players* vindos dos locais de mídia (através de *URL*), *DataSources* e *InputStreams*.

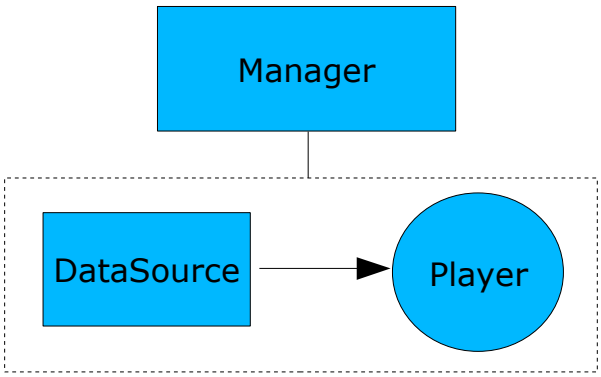

Figura 2: Relacionamento do Manager com o DataSource e Player

Pode-se consultar propriedades MMAPI via ***String System.getProperty(String key)***.

Chave	***Descrição***
microedition.media.version	A versão da especificação *MMAPI* implementada pelo dispositivo. Exemplo: "1.1"
supports.mixing	Retorna "true" se o dispositivo suporta mixagem de áudio. Pode reproduzir os dois últimos tons simultaneamente. Pode ter dois *Players* reproduzindo áudio simultaneamente e pode reproduzir um tom enquanto o outro *Player* está reproduzindo áudio ao mesmo tempo.
supports.audio.capture	Retorna "true" se a captura de áudio é suportada.
supports.video.capture	Retorna "true" se a captura de vídeo é suportada.
supports.recording	Retorna "true" se a gravação de áudio é suportada.

2.1. Geração de Tons

Para reproduzir tons é necessário chamar o método estático *Manager.playTone(int tom, int duration, int volume).* Os valores válidos para o parâmetro tom vão de 0 a 127. O parâmetro *duration* representa a duração da reprodução do tom e deve ser especificada em milissegundos. O parâmetro *volume* varia de 0 a 100.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.media.*;
import javax.microedition.media.control.*;
import java.io.*;

public class ToneMIDlet extends MIDlet implements CommandListener{
  private Command exitCommand, playCommand;
  private Form form;
  private Gauge volumeGauge;
  private Gauge durationGauge;
  private Gauge toneGauge;
  private Display display;
  private int duration = 2; // seconds
  private int volume = 100;
  private int tone = ToneControl.C4;
  private static int MAX_VOLUME = 100;
  private static int MAX_TONE = 127;
  private static int MAX_DURATION = 5;

  public ToneMIDlet() {
    playCommand = new Command("Play", Command.OK, 1);
    exitCommand = new Command("Exit", Command.EXIT, 1);
    volumeGauge = new Gauge("Volume", true, MAX_VOLUME, volume);
    toneGauge = new Gauge("Tone", true, MAX_TONE, tone);
    durationGauge = new Gauge("Duration",true,MAX_DURATION,duration);

    form = new Form("Tone Player");
    form.addCommand(playCommand);
    form.addCommand(exitCommand);
    form.append(volumeGauge);
    form.append(durationGauge);
    form.append(toneGauge);
  }
  public void startApp() {
    display = Display.getDisplay(this);
    form.setCommandListener(this);
    display.setCurrent(form);
  }
  public void pauseApp() {}
  public void destroyApp(boolean unconditional) {}
  public void commandAction(Command c, Displayable d) {
    if (c == exitCommand) {
      notifyDestroyed();
    }
    if (c == playCommand){
      try {
        volume = volumeGauge.getValue();
        tone = toneGauge.getValue();
        duration = durationGauge.getValue();
        Manager.playTone(tone, duration*1000, volume);
      } catch (MediaException mex){}
    }
  }
}
```

2.2. Tocando Áudio

Por conveniência, o método *Manager.createPlayer*(String URI) cria um objeto *player* a partir de uma URI.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.media.*;
import javax.microedition.media.control.*;
import java.io.*;
```

```
public class NetAudioMidlet extends MIDlet implements CommandListener{
  private Command exitCommand, playCommand;
  private Form form;
  private Gauge volumeGauge;
  private Display display;
  private int volume = 100;
  private static int MAX_VOLUME = 100;
  Player player;

  public NetAudioMidlet() {
    playCommand = new Command("Play", Command.OK, 1);
    exitCommand = new Command("Exit", Command.EXIT, 1);
    volumeGauge = new Gauge("Volume", true, MAX_VOLUME, volume);
    form = new Form("Audio Player");
    form.addCommand(playCommand);
    form.addCommand(exitCommand);
    form.append(volumeGauge);
  }
  public void startApp() {
    display = Display.getDisplay(this);
    form.setCommandListener(this);
    display.setCurrent(form);
    try {
      player = Manager.createPlayer("http://localhost:8084/Chapter07/bong.wav");
      player.realize();
      player.prefetch();
    } catch (IOException ioex) {
      display.setCurrent(new Alert("IO Exception",
        ioex.getMessage(),
        null, AlertType.ERROR));
    }  catch (MediaException mex) {
      display.setCurrent(new Alert("Media Exception",
        mex.getMessage(),
        null, AlertType.ERROR));
    }
  }
  public void pauseApp() {}
  public void destroyApp(boolean unconditional) {}
  public void commandAction(Command c, Displayable d) {
    if (c == exitCommand) {
      notifyDestroyed();
    }
    if (c == playCommand){
      try {
        VolumeControl control = (VolumeControl)
          player.getControl("VolumeControl");
        if (control != null){
          control.setLevel(volumeGauge.getValue());
        }

        player.start();
      }  catch (MediaException mex) {
        display.setCurrent(new Alert("Media Exception",
          mex.getMessage(),  null, AlertType.ERROR));
      } catch (Exception ex){
        display.setCurrent(new Alert("Exception",
          ex.getMessage(),  null, AlertType.ERROR));
      }
    }
  }
}
```

Também é possível tocar uma mídia a partir de um arquivo inserido no JAR do projeto. Entretanto, deve ser criado um objeto do tipo Stream que será repassado para o método Manager.createPlayer().

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.media.*;
import javax.microedition.media.control.*;
import java.io.*;

public class AudioMidlet extends MIDlet implements CommandListener{
  private Command exitCommand, playCommand;
  private Form form;
  private Gauge volumeGauge;
  private Display display;
  private int volume = 100;
  private static int MAX_VOLUME = 100;
  Player player;

  public AudioMidlet() {
    playCommand = new Command("Play", Command.OK, 1);
    exitCommand = new Command("Exit", Command.EXIT, 1);
    volumeGauge = new Gauge("Volume", true, MAX_VOLUME, volume);
    form = new Form("Audio Player");
    form.addCommand(playCommand);
    form.addCommand(exitCommand);
    form.append(volumeGauge);
  }
  public void startApp() {
    display = Display.getDisplay(this);
    form.setCommandListener(this);
    display.setCurrent(form);
  }
  public void pauseApp() {}
  public void destroyApp(boolean unconditional) {}
  public void commandAction(Command c, Displayable d) {
    if (c == exitCommand) {
      notifyDestroyed();
    }
    if (c == playCommand){
      try {
        InputStream stream = getClass().
          getResourceAsStream("bong.wav");
        player = Manager.createPlayer(stream, "audio/x-wav");
        player.realize();

        VolumeControl control = (VolumeControl)
          player.getControl("VolumeControl");
        if (control != null){
          control.setLevel(volumeGauge.getValue());
        }

        player.start();
      }  catch (MediaException mex) {
        display.setCurrent(new Alert("Media Exception",
          mex.getMessage(), null, AlertType.ERROR));

      } catch (Exception ex){
        display.setCurrent(new Alert("Exception",
          ex.getMessage(),  null, AlertType.ERROR));
      }
    }
  }
}
```

3. Wireless Messaging API (WMA)

Utilizar o *Wireless Messaging API* é quase semelhante à maneira que uma conexão é feita através de soquetes e *Datagramas*. Utiliza-se o mesmo aplicativo – *Generic Connection Framework* (GCF).

A URL de conexão possui o seguinte formato "sms://+639178888888", onde "+639178888888" é o número do telefone para qual se deseja enviar a mensagem.

```
public void sendSMS(String number, String message) throws Exception{
  String url = "sms://" +  number;
  MessageConnection connection = (MessageConnection) Connector.open(url);
  TextMessage msg = (TextMessage) connection.newMessage(
    MessageConnection.TEXT_MESSAGE);
  msg.setPayloadText(message);
  connection.send(msg);
  connection.close();
}
```

3.1. Enviando uma mensagem SMS

O desenvolvimento de aplicações móveis com o *Netbeans* é muito simples. Não é necessário enviar mensagens reais de SMS apenas para testar a aplicação que estamos desenvolvendo. O Netbeans (com o pacote *Mobility*) possui a ferramenta *J2ME Wireless Toolkit*. Esta ferramenta vem com um emulador e inclui também aplicativos para testar o envio e recebimento de mensagens do tipo *SMS*. É possível configurar o número do telefone utilizando as preferências do *WMA* acessando a partir do menu principal:

- *Tools*
- *Java Platform Manager*
- *J2ME Wireless Toolkit* 2.2
- *Tools & Extensions*
  - *Preferences* -> WMA
  - *Utilities* -> WMA: Open Console

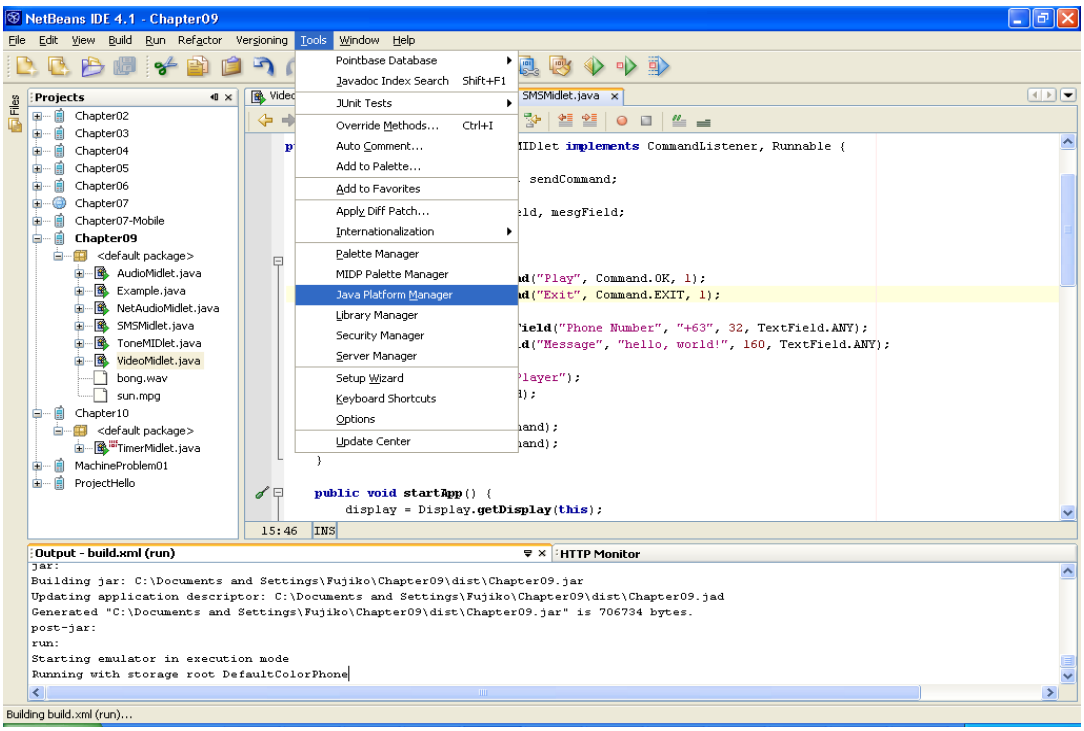

Figura 3: Java Platform Manager

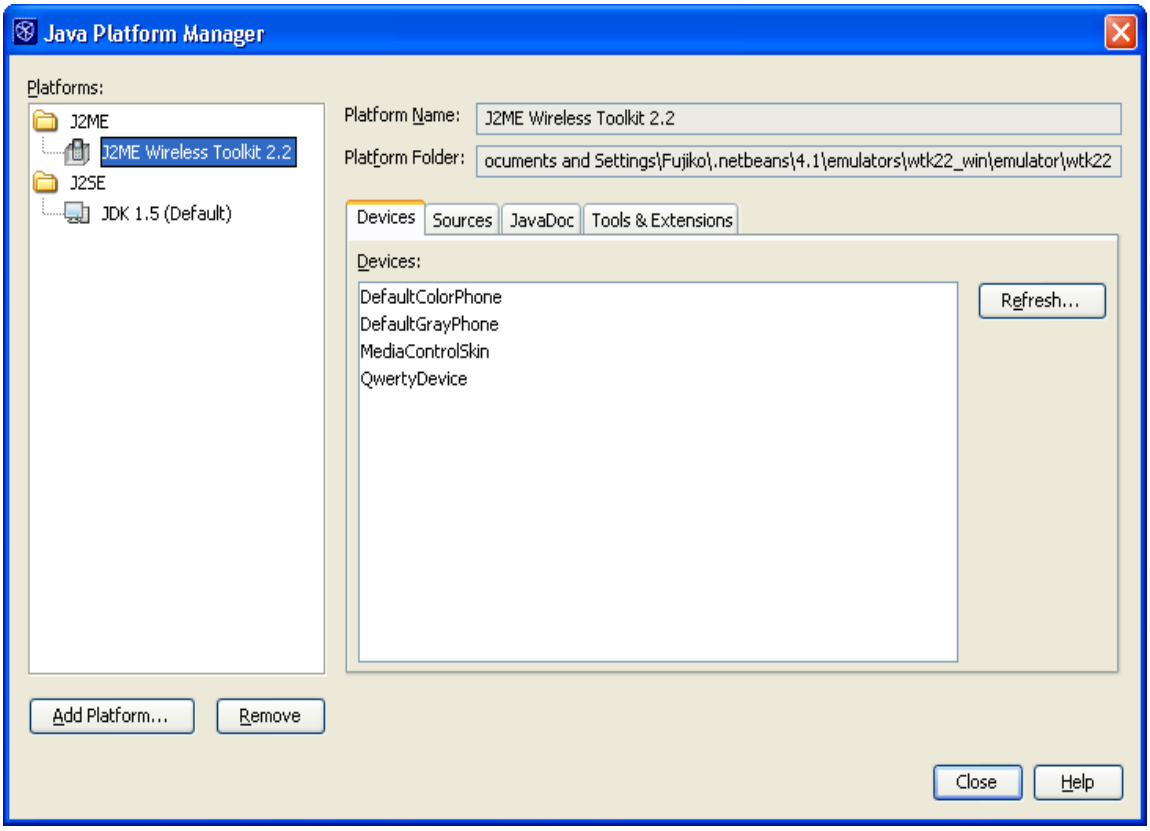

Figura 4: Java Platform Manager - Devices

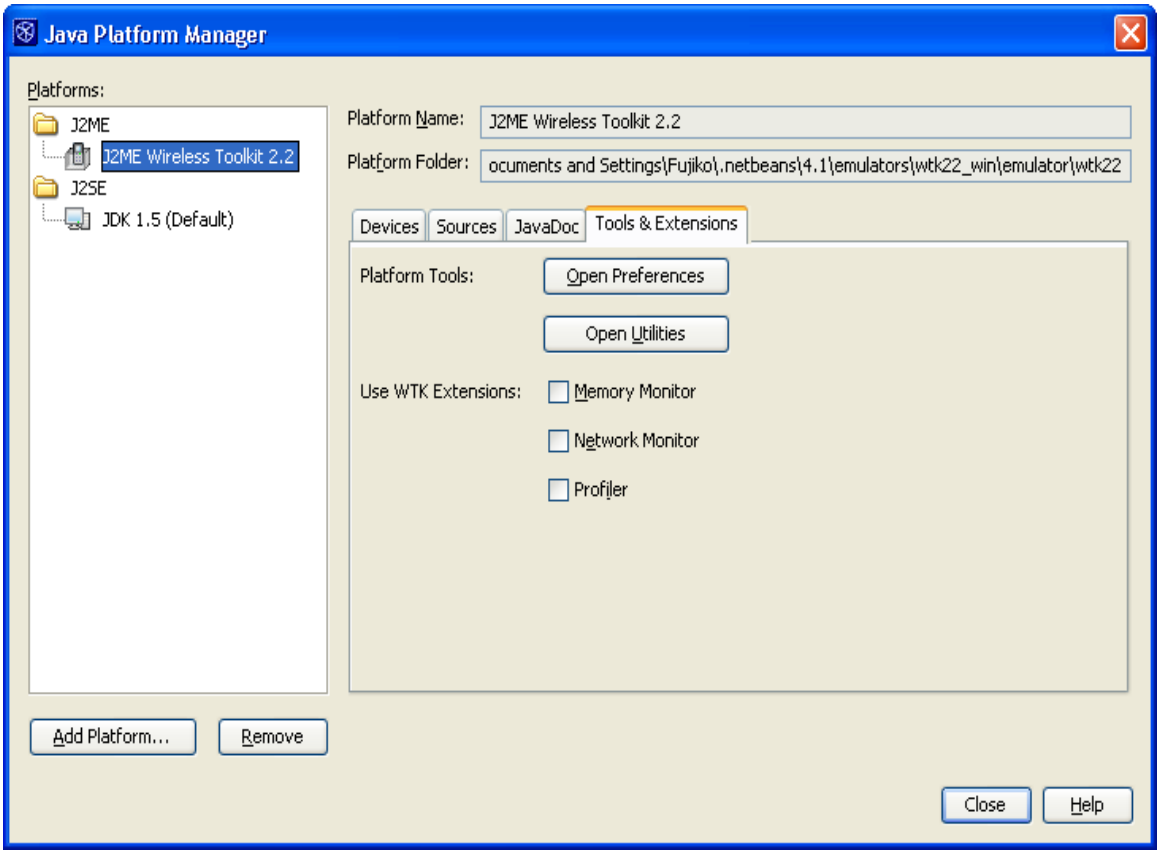

Figura 5: Java Platform Manager – Tools & Extensions

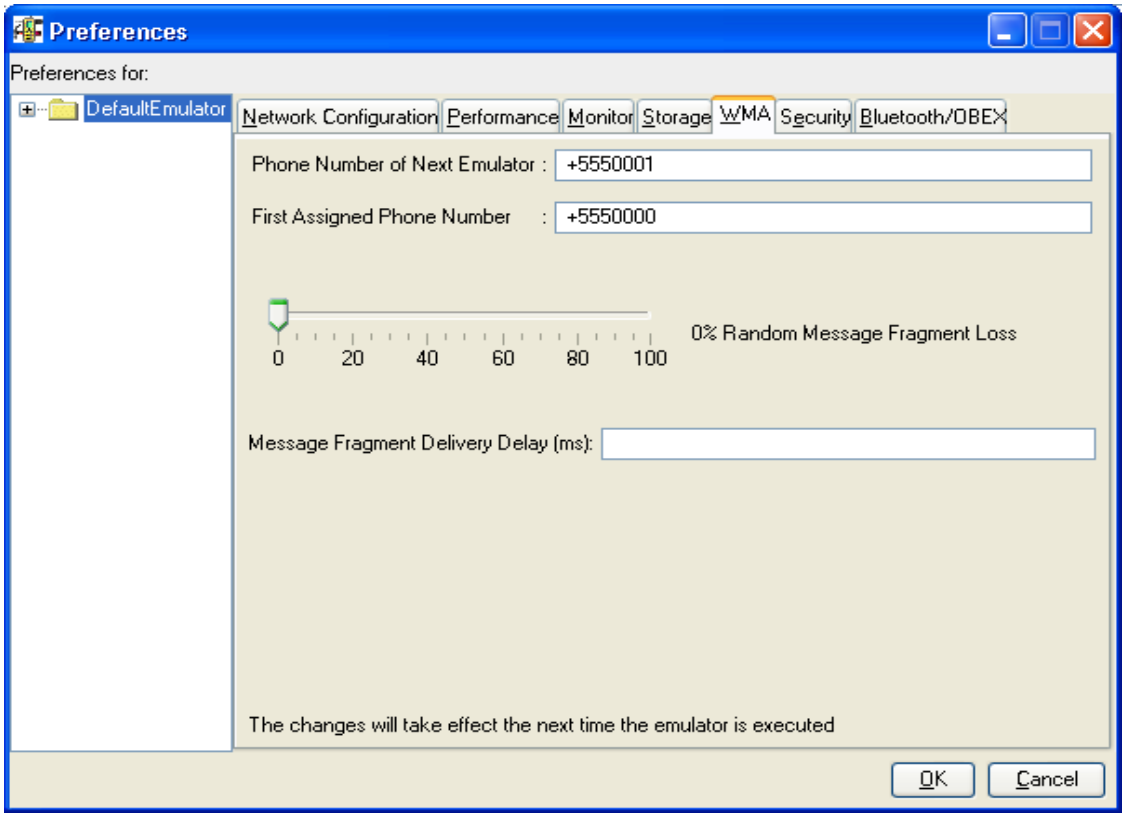

Figura 6: J2ME Wireless Toolkit – Preferences

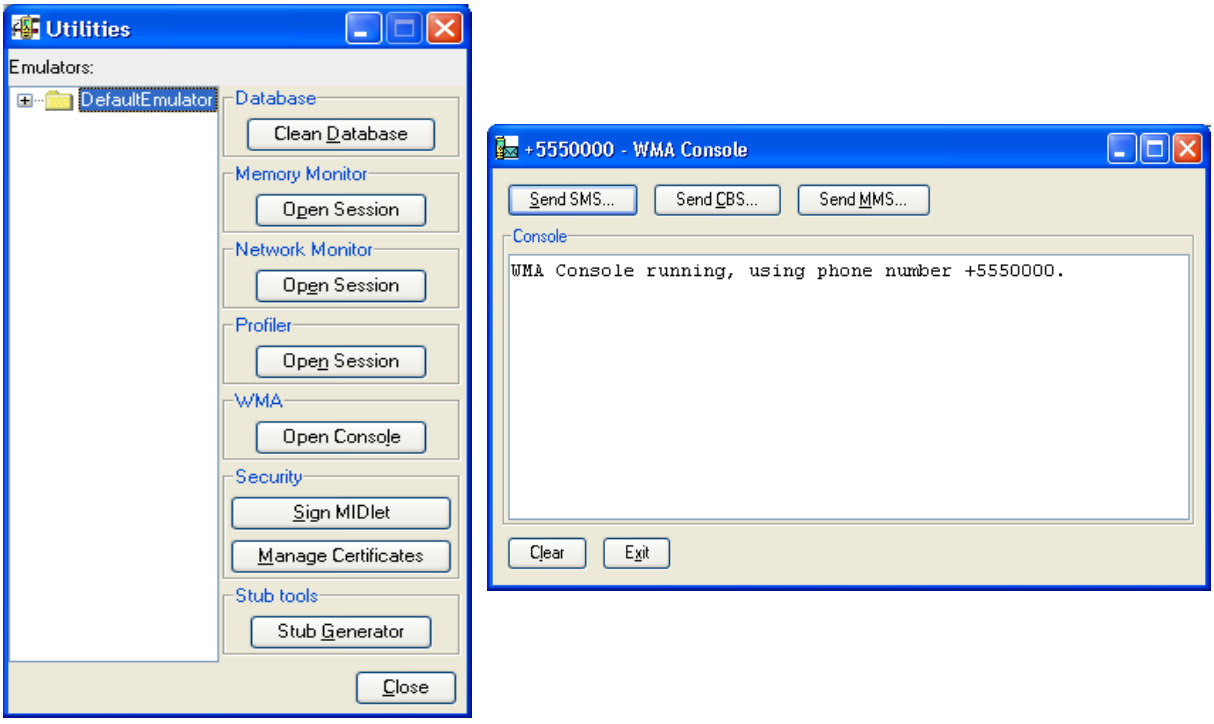

Figura 7: J2ME Wireless Toolkit – Utilities e Console

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

public class SMSMidlet extends MIDlet implements CommandListener, Runnable {

  private Command exitCommand, sendCommand;
```

```
    private Form form;
    private TextField addressField, mesgField;
    private Display display;

    public SMSMidlet() {
      sendCommand = new Command("Send", Command.OK, 1);
      exitCommand = new Command("Exit", Command.EXIT, 1);

      addressField = new TextField(
        "Phone Number", "+5550000", 32, TextField.ANY);
      mesgField = new TextField(
        "Message", "hello, world!", 160, TextField.ANY);

      form = new Form("SMS Message");
      form.append(addressField);
      form.append(mesgField);
      form.addCommand(sendCommand);
      form.addCommand(exitCommand);
    }

    public void startApp() {
      display = Display.getDisplay(this);
      form.setCommandListener(this);
      display.setCurrent(form);
    }
    public void pauseApp() {}
    public void destroyApp(boolean unconditional) {}
    public void commandAction(Command c, Displayable d) {
      if (c == exitCommand) {
        notifyDestroyed();
      }
      if (c == sendCommand) {
        Thread thread = new Thread( this );
        thread.start();
      }
    }
    public void sendSMS(String number, String message) throws Exception{
      String url = "sms://" +  number;
      MessageConnection connection =
          (MessageConnection) Connector.open(url);
      TextMessage msg = (TextMessage) connection.newMessage(
          MessageConnection.TEXT_MESSAGE);
      msg.setPayloadText(message);
      connection.send(msg);
      connection.close();
    }
    public void run() {
      try {
        String address = addressField.getString();
        String message = mesgField.getString();
        sendSMS(address, message);
        display.setCurrent(new Alert("SMS Message", "Message Sent\n"
          + "To: " + address + "\n" + "Message: " + message, null,
          AlertType.INFO));
      } catch (Exception ex) {
        display.setCurrent(new Alert("SMS Error", ex.getMessage(),
            null, AlertType.ERROR));
      }
    }
}
```

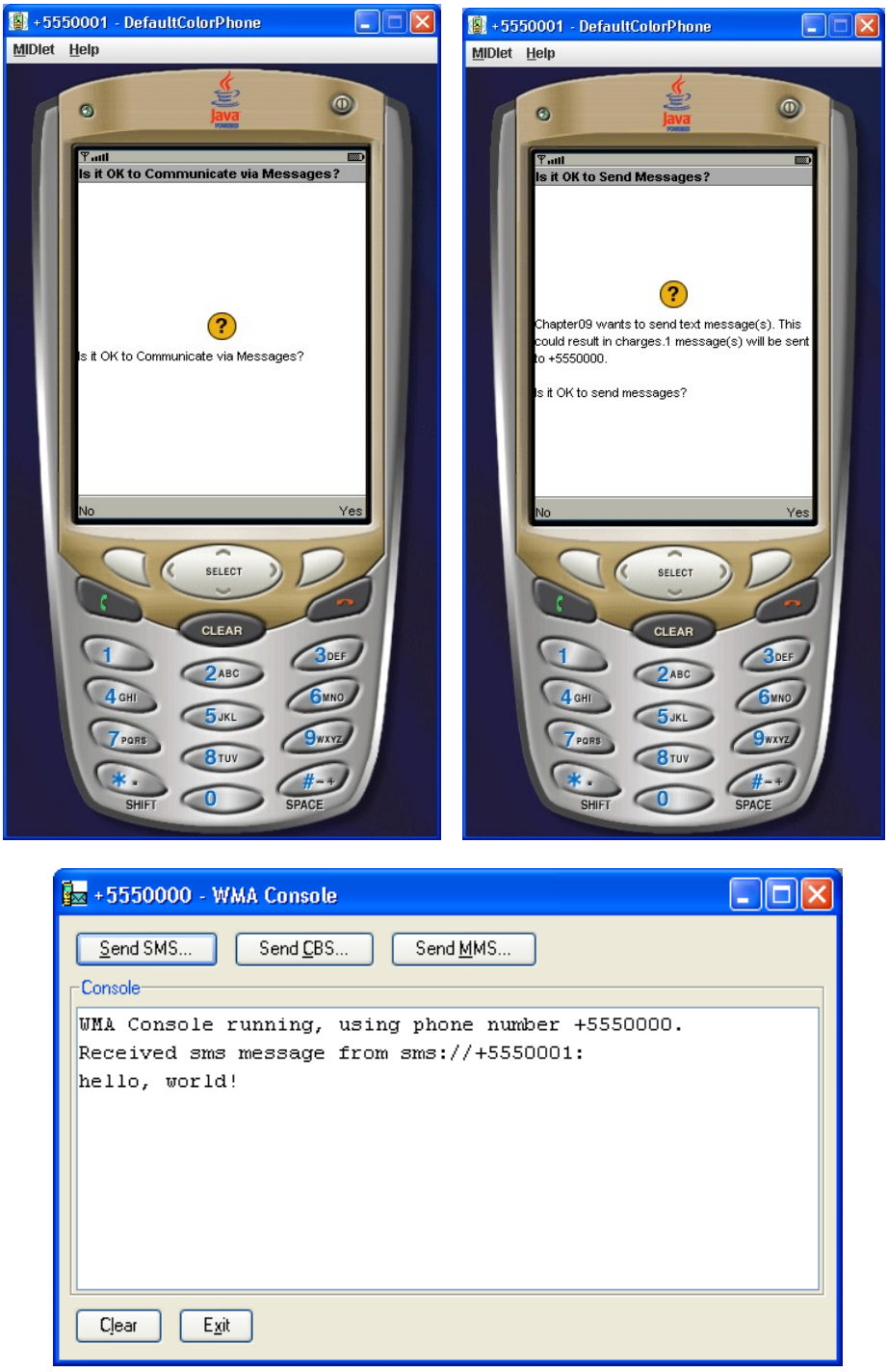

Figura 8: Execução da aplicação

3.2. Recebendo mensagens SMS

Para receber um mensagem de texto, abra uma *MessageConnection* especificando uma porta. O Protocolo para mensagem SMS é "sms". Este comando ficará esperando até receber uma mensagem de SMS pela porta 8888:

```
conn = (MessageConnection) Connector.open("sms://:8888");
```

Devemos registrar nossa aplicação para ser um receptor da mensagem de forma que o *AMS* notifique o *MIDlet* da chegada da mensagem.

```
conn.setMessageListener(this);
```

O método *notifyIncomingMessage* será chamado pelo *AMS* quando uma mensagem for recebida

pelo dispositivo. Precisamos criar uma Thread separada para as mensagens de leitura de forma que o método que for chamado novamente possa ter uma saída imediata.

```
public void notifyIncomingMessage(MessageConnection messageConnection) {
  if (thread == null){
    thread = new Thread(this);
    thread.start();
  }
}
```

E, deste modo, será utilizado o método run(), do qual obteremos a mensagem:

```
public void run(){
    try {
        Message mesg = conn.receive();
        if (mesg != null && mesg instanceof TextMessage) {
            TextMessage text = (TextMessage) mesg;
            addressField.setText(text.getAddress());
            mesgField.setText(text.getPayloadText());
            dateField.setText("" + text.getTimestamp());
            statusField.setText("Message received.");
        }
    } catch (Exception e) {
        statusField.setText("Error: " + e.getMessage());
    }
    thread = null;
}
```

Este é o código completo para receber e listar a mensagem *SMS*:

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

public class SMSReceiverMidlet extends MIDlet
    implements CommandListener, MessageListener, Runnable  {
  private Command exitCommand, sendCommand;
  private Form form;
  private StringItem statusField, addressField, mesgField, dateField;
  private Display display;
  private MessageConnection conn;
  private Thread thread;
  private String port = "8888";

  public SMSReceiverMidlet() {
    exitCommand = new Command("Exit", Command.EXIT, 1);
    statusField = new StringItem("Status:", "");
    addressField = new StringItem("From:", "");
    mesgField = new StringItem("Message:", "");
    dateField = new StringItem("Timestamp:", "");
    form = new Form("SMS Receiver");
    form.append(statusField);
    form.append(addressField);
    form.append(mesgField);
    form.append(dateField);
    form.addCommand(exitCommand);
  }
  public void startApp() {
    display = Display.getDisplay(this);
    form.setCommandListener(this);

    startReceiver();
    display.setCurrent(form);
  }
  public void pauseApp() {
    thread = null;
```

```
    }
    public void destroyApp(boolean unconditional) {
      thread = null;
      if (conn != null){
        try {
          conn.close();
        } catch (Exception ex){}
      }
    }
    public void commandAction(Command c, Displayable d) {
      if (c == exitCommand) {
        notifyDestroyed();
      }
    }
    private void startReceiver(){
      try {
        String addr = "sms://:" + port;
        if (conn == null){
          conn = (MessageConnection) Connector.open(addr);
          conn.setMessageListener(this);
          statusField.setText(
              "waiting for message at port " + port);
        }
      } catch (Exception ex){
        statusField.setText("Cannot open connection on port "
            + port + ":" + ex.getMessage());
      }
      thread = new Thread(this);
      thread.start();
    }
    public void notifyIncomingMessage(MessageConnection messageConn) {
      if (thread == null){
        thread = new Thread(this);
        thread.start();
      }
    }
    public void run(){
      try {
        Message mesg = conn.receive();
        if (mesg != null && mesg instanceof TextMessage) {
          TextMessage text = (TextMessage) mesg;
          addressField.setText(text.getAddress());
          mesgField.setText(text.getPayloadText());
          dateField.setText("" + text.getTimestamp());
          statusField.setText("Message received.");
        } else {
          statusField.setText(
            "Non-text message received: "
            + mesg.getClass().toString());
        }
      } catch (Exception e) {
        statusField.setText("Error: " + e.getMessage());
      }
      thread = null;
    }
  }
```

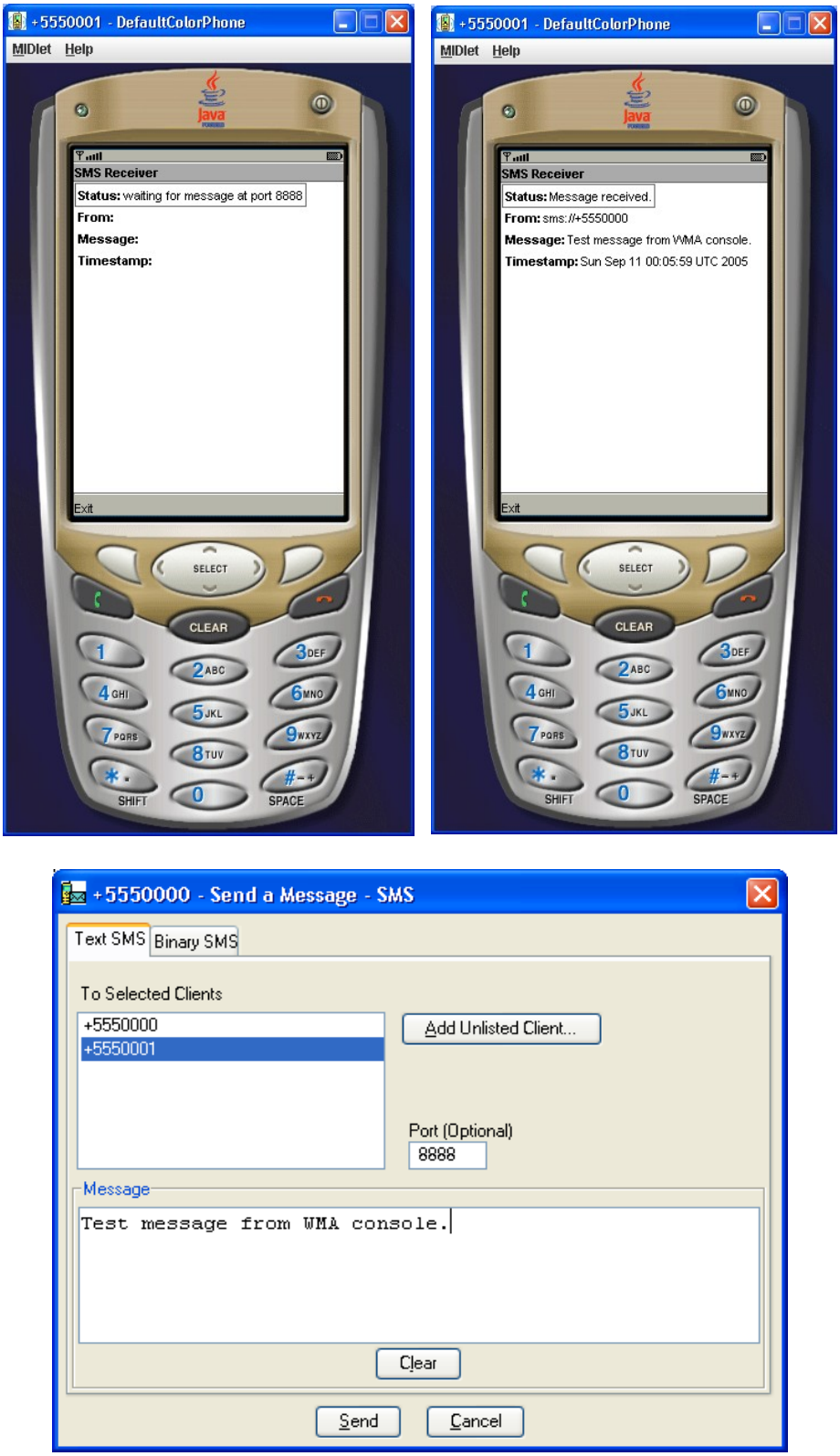

Figura 9: Recebimento da mensagem

4. Exercícios

4.1. Tocador de áudio

Crie uma *MIDlet* que toque um arquivo de áudio por um número indefinido de vezes. O arquivo de áudio deve ser lido a partir do JAR da aplicação. Dica: envie uma propriedade para o objeto Player controlar o laço.

4.2. Auto-respondedor de SMS

Crie uma *MIDlet* que responda automaticamente quando receber uma mensagem de texto. Dica: modifique a classe *SMSReceiverMidlet* e utilize a mesma conexão para enviar a mensagem de resposta.

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.