Módulo 1

Introdução à Programação I

Lição 5

Capturando entrada de dados através do teclado

Versão 1.0 - Jan/2007

Autor
Florence Tiu Balagtas

Equipe
Joyce Avestro
Florence Balagtas
Rommel Feria
Reginald Hutcherson
Rebecca Ong
John Paul Petines
Sang Shin
Raghavan Srinivas
Matthew Thompson

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

Alexandre Mori
Alexis da Rocha Silva
Aline Sabbatini da Silva Alves
Allan Wojcik da Silva
André Luiz Moreira
Andro Márcio Correa Louredo
Antoniele de Assis Lima
Antonio Jose R. Alves Ramos
Aurélio Soares Neto
Bruno da Silva Bonfim
Bruno dos Santos Miranda
Bruno Ferreira Rodrigues
Carlos Alberto Vitorino de Almeida
Carlos Alexandre de Sene
Carlos André Noronha de Sousa
Carlos Eduardo Veras Neves
Cleber Ferreira de Sousa
Cleyton Artur Soares Urani
Cristiano Borges Ferreira
Cristiano de Siqueira Pires
Derlon Vandri Aliendres
Fabiano Eduardo de Oliveira
Fábio Bombonato
Fernando Antonio Mota Trinta
Flávio Alves Gomes
Francisco das Chagas
Francisco Marcio da Silva
Gilson Moreno Costa
Givailson de Souza Neves
Gustavo Henrique Castellano
Hebert Julio Gonçalves de Paula
Heraldo Conceição Domingues
Hugo Leonardo Malheiros Ferreira
Ivan Nascimento Fonseca
Jacqueline Susann Barbosa
Jader de Carvalho Belarmino
João Aurélio Telles da Rocha
João Paulo Cirino Silva de Novais
João Vianney Barrozo Costa
José Augusto Martins Nieviadonski
José Leonardo Borges de Melo
José Ricardo Carneiro
Kleberth Bezerra G. dos Santos
Lafaiete de Sá Guimarães
Leandro Silva de Morais
Leonardo Leopoldo do Nascimento
Leonardo Pereira dos Santos
Leonardo Rangel de Melo Filardi
Lucas Mauricio Castro e Martins
Luciana Rocha de Oliveira
Luís Carlos André
Luís Octávio Jorge V. Lima
Luiz Fernandes de Oliveira Junior
Luiz Victor de Andrade Lima
Manoel Cotts de Queiroz
Marcello Sandi Pinheiro
Marcelo Ortolan Pazzetto
Marco Aurélio Martins Bessa
Marcos Vinicius de Toledo
Maria Carolina Ferreira da Silva
Massimiliano Giroldi
Mauricio Azevedo Gamarra
Mauricio da Silva Marinho
Mauro Cardoso Mortoni
Mauro Regis de Sousa Lima
Namor de Sá e Silva
Néres Chaves Rebouças
Nolyanne Peixoto Brasil Vieira
Paulo Afonso Corrêa
Paulo José Lemos Costa
Paulo Oliveira Sampaio Reis
Pedro Antonio Pereira Miranda
Pedro Henrique Pereira de Andrade
Renato Alves Félix
Renato Barbosa da Silva
Reyderson Magela dos Reis
Ricardo Ferreira Rodrigues
Ricardo Ulrich Bomfim
Robson de Oliveira Cunha
Rodrigo Pereira Machado
Rodrigo Rosa Miranda Corrêa
Rodrigo Vaez
Ronie Dotzlaw
Rosely Moreira de Jesus
Seire Pareja
Sergio Pomerancblum
Silvio Sznifer
Suzana da Costa Oliveira
Tásio Vasconcelos da Silveira
Thiago Magela Rodrigues Dias
Tiago Gimenez Ribeiro
Vanderlei Carvalho Rodrigues Pinto
Vanessa dos Santos Almeida
Vastí Mendes da Silva Rocha
Wagner Eliezer Roncoletta

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
- **Regina Mariani -** Coordenadora responsável pela parte jurídica
- **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

Agora que já estudamos alguns conceitos básicos e escrevemos alguns códigos simples, vamos fazer as aplicações ficarem mais interativas começando com a captura de dados digitados pelo usuário. Nesta lição, discutiremos três modos de obter dados de entrada (input). O primeiro é através do uso da classe BufferedReader do pacote java.util; o segundo, através do uso da nova classe Scanner no mesmo pacote; e, por fim, envolveremos a utilização da interface gráfica utilizando JOptionPane.

Ao final desta lição, o estudante será capaz de:

- Criar códigos para a captura de dados pelo teclado.
- Usar a classe BufferedReader para captura, através de uma janela de console, dos dados digitados no teclado.
- Utilizar a classe Scanner para captura, através de uma janela de console, dos dados digitados no teclado.
- Utilizar a classe JOptionPane para captura, através da uma interface gráfica, dos dados digitados no teclado.

2. BufferedReader para capturar dados

Primeiramente, utilizaremos a classe **BufferedReader** do pacote java.io para capturar dados de entrada através do teclado.

Passos para capturar os dados digitados, tomemos por base o programa visto na lição anterior:

1. Digite a seguinte instrução no início do programa:

```
import java.io.*;
```

2. Adicione as seguintes instruções no corpo do método **main**:

```
BufferedReader dataIn = new BufferedReader(
    new InputStreamReader(System.in));
```

3. Declare uma variável temporária do tipo String para gravar os dados digitados pelo usuário e chame o método readLine() que vai capturar linha por linha do que o usuário digitar. Isso deverá ser escrito dentro de um bloco **try-catch** para tratar possíveis exceções.

```
try {
    String temp = dataIn.readLine();
} catch (IOException e) {
    System.out.println("Error in getting input");
}
```

Abaixo, segue o programa completo:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class GetInputFromKeyboard {
    public static void main(String[] args) {
      BufferedReader dataIn = new BufferedReader(new
     InputStreamReader(System.in));
      String name = "";
      System.out.print("Please Enter Your Name:");
      try {
          name = dataIn.readLine();
            } catch (IOException e) {
          System.out.println("Error!");
      }
      System.out.println("Hello " + name +"!");
    }
}
```

Faremos uma análise deste programa linha por linha:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
```

Estas linhas acima mostram que estamos utilizando as classes **BufferedReader**, **InputStreamReader** e **IOException** cada qual dentro do pacote **java.io**. Essas APIs ou

Interfaces de Programação de Aplicações (Application Programming Interface) contêm centenas de classes pré-definidas que se pode usar nos programas. Essas classes são organizadas dentro do que chamamos de **pacotes**.

Pacotes contêm classes que se relacionam com um determinado propósito. No exemplo, o pacote **java.io** contém as classes que permitem capturar dados de entrada e saída. Estas linhas poderiam ser reescritas da seguinte forma:

```
import java.io.*;
```

que importará todas as classes encontradas no pacote **java.io**, deste modo é possível utilizar todas classes desse pacote no programa.

As próximas linhas:

```
public class GetInputFromKeyboard {
    public static void main( String[] args ) {
```

já foram discutidas na lição anterior. Isso significa que declaramos uma classe nomeada **GetInputFromKeyboard** e, em seguida, iniciamos o método principal (main).

Na instrução:

```
BufferedReader dataIn = new BufferedReader(new
    InputStreamReader(System.in));
```

declaramos a variável **dataIn** do tipo **BufferedReader**. Não se preocupe com o significado da sintaxe, pois será abordado mais à frente.

A seguir, declaramos a variável **name** do tipo String:

```
String name = "";
```

na qual armazenaremos a entrada de dados digitada pelo usuário. Note que foi inicializada como uma String vazia "". É uma boa prática de programação inicializar as variáveis quando declaradas.

Na próxima instrução, solicitamos que o usuário escreva um nome:

```
System.out.print("Please Enter Your Name:");
```

As seguinte linhas definem um bloco **try-catch:**

```
try {
    name = dataIn.readLine();
} catch (IOException e) {
    System.out.println("Error!");
}
```

que asseguram, caso ocorram exceções serão tratadas.

Falaremos sobre o tratamento de exceções na última parte deste curso. Por hora, é necessário adicionar essas linhas para utilizar o método **readLine()** e receber a entrada de dados do usuário.

Em seguida:

```
name = dataIn.readLine();
```

capturamos a entrada dos dados digitados pelo usuário e as enviamos para a variável String criada anteriormente. A informação é guardada na variável **name**.

Como última instrução:

```
System.out.println("Hello " + name + "!");
```

montamos a mensagem final para cumprimentar o usuário.

3. Classe Scanner para capturar dados

Vimos uma maneira para obter dados de entrada através do teclado. O JDK 5.0 lançou uma nova classe chamada **Scanner** que engloba diversos métodos para facilitar este serviço.

Abaixo, segue o programa completo utilizando esta classe:

```
import java.util.Scanner;

public class GetInputFromScanner
{
    public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Please Enter Your Name:");
      String name = sc.next();
      System.out.println("Hello " + name +"!");
    }
}
```

Compare-o com o programa visto anteriormente. Percebe-se que fica mais simples conseguir a mesma funcionalidade.

Inicialmente, definimos a chamada ao pacote que contém a classe **Scanner**:

```
import java.util.Scanner;
```

Em seguida, as instruções que define a classe e o método main:

```
public class GetInputFromScanner
{
    public static void main(String[] args) {
```

Definimos uma variável, denominada **sc**, que será criada a partir da classe **Scanner** e direcionada para a entrada padrão:

```
Scanner sc = new Scanner(System.in);
```

De forma semelhante, mostramos uma mensagem solicitando informação do usuário:

```
System.out.println("Please Enter Your Name:");
```

Utilizamos a variável **sc** para chamarmos o método que fará o recebimento dos dados digitados:

```
String name = sc.nextLine();
```

A classe **Scanner** possui diversos métodos que podem ser utilizados para realizar este serviço. Os principais métodos que podemos utilizar, neste caso, são:

Método	Finalidade
next()	Aguarda uma entrada em formato String
nextInt()	Aguarda uma entrada em formato Inteiro
nextByte()	Aguarda uma entrada em formato Inteiro

nextLong()	Aguarda uma entrada em formato Inteiro Longo
nextFloat()	Aguarda uma entrada em formato Número Fracionário
nextDouble()	Aguarda uma entrada em formato Número Fracionário

*Tabela 1: Métodos da Classe **Scanner** para obter dados*

Por fim, mostramos o resultado e encerramos o método main e a classe:

```
        System.out.println("Hello " + name +"!");
    }
}
```

4. Utilizando a JOptionPane para receber dados

Um outro modo de receber os dados de entrada é utilizar a classe **JOptionPane**, que pertence ao pacote **javax.swing**. A **JOptionPane** possui métodos que conseguem criar caixas de diálogo na qual o usuário pode informar ou visualizar algum dado.

Dado o seguinte código:

```
import javax.swing.JOptionPane;

public class GetInputFromKeyboard {
    public static void main( String[] args ){
        String name = "";
        name = JOptionPane.showInputDialog("Please enter your name");
        String msg = "Hello " + name + "!";
        JOptionPane.showMessageDialog(null, msg);
     }
}
```

esta classe apresentará o seguinte resultado:

Figura 1: Aguardando dados no JOptionPane

Figura 2: Digitando florence no JOptionPane

Figura 3: Respondendo com JOptionPane

A primeira instrução:

```
import javax.swing.JOptionPane;
```

mostra que estamos importando a classe **JOptionPane** do pacote **javax.swing**.

Poderíamos, de forma semelhante, escrever estas instruções do seguinte modo:

```
import javax.swing.*;
```

A instrução seguinte:

```
name = JOptionPane.showInputDialog("Please enter your name");
```

cria uma caixa de entrada que exibirá um diálogo com uma mensagem, um campo de texto para receber os dados do usuário e um botão OK, conforme mostrado na **figura 1**. O resultado será armazenado na variável do tipo String **name**.

Na próxima instrução, criamos uma mensagem de cumprimento, que ficará armazenada na variável **msg**:

```
String msg = "Hello " + name + "!";
```

Finalizando a classe, exibiremos uma janela de diálogo que conterá a mensagem e o botão de OK, conforme mostrado na **figura 3**.

```
JOptionPane.showMessageDialog(null, msg);
```

5. Exercícios

5.1. As 3 palavras (versão Console)

Utilizando a classe **BufferedReader** ou **Scanner**, capture três palavras digitadas pelo usuário e mostre-as como uma única frase na mesma linha. Por exemplo:

```
Palavra 1: Goodbye
Palavra 2: and
Palavra 3: Hello

Goodbye and Hello
```

5.2. As 3 palavras (versão Interface Gráfica)

Utilizando a classe **JOptionPane**, capture palavras em três caixas de diálogos distintas e mostre-as como uma única frase. Por exemplo:

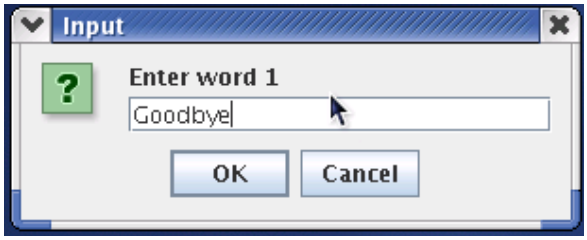

Figura 4: Primeira Palavra

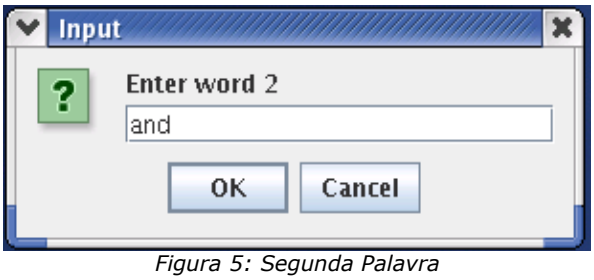

Figura 5: Segunda Palavra

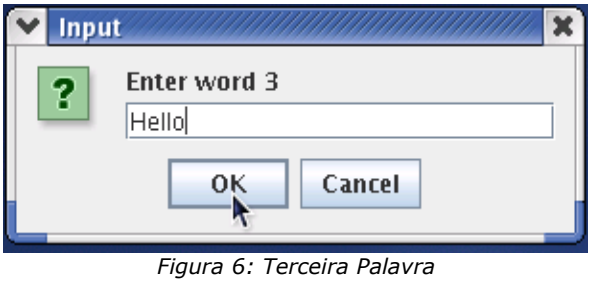

Figura 6: Terceira Palavra

Figura 7: Mostrando a Mensagem

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.