Introdução ao Java

Dentre tantas alternativas, escolhemos Java (uma linguagem criada em 1991 pela Sun Microsystems) para ser nossa próxima linguagem. Temos alguns motivos para tal: total portabilidade, semelhança com outras linguagens oriundas do C, uso bastante difundido na Internet, facilidade de obtenção de ferramentas de software, possibilidade de se montar desde um *applet* simples até um sistema cliente/servidor completo em plataforma *web*, etc.

É bom deixar claro que existem diferenças conceituais entre o Pascal e o C vistos anteriormente e Java (bem como entre todas as linguagens – senão não existiriam tantas...). Traduzir um programa de uma linguagem para outra não é simplesmente traduzir comandos... Isto só funciona em rotinas bem simples. Devemos, pois, tentar compreender os prós e os contras (vide final deste texto) das linguagens para poder optar por esta ou aquela, dependendo do problema a ser resolvido.

Como em todas as linguagens derivadas de C, em Java existe diferenciação entre letras maiúsculas e minúsculas, ou seja: pode-se declarar uma variável ALUNO e outra variável aluno (e outra Aluno, etc., que podem representar coisas diferentes).

Comentários

```
// comentário de uma linha
/* comentário de uma
   ou mais linhas */
/** comentário para uso em documentação emitida posteriormente (com o utilitário Javadoc) */
```

Estrutura de um programa

```
public class Prog1
{
  public static void main (String args[])
  {
    // corpo do programa
  }
}
```

Importante: um programa Java é uma classe (class) e o seu ponto de partida para execução é um método (main). Daí ser fundamental compreender o modelo de Orientação a Objeto para programar em Java. Atente para o fato de que as classes em Java podem ou não conter o método main; se contiverem, serão executáveis, caso contrário serão apenas classes definindo atributos e métodos para uso em outras classes.

Variáveis

A declaração de variáveis é feita com a sintaxe:

```
Tipo nome_da_variável ;
```

Os tipos básicos são os seguintes:

byte, short, int, long (tipos inteiros);
float, double (tipos reais);
char (caracter) – entre apóstrofos
boolean (lógico) (true/false)

Existe também a String (cadeia de caracteres) – entre aspas – mas deve-se atentar para o fato de ser uma classe, não um tipo de variável.

A declaração é obrigatória, mas pode ser feita dentro do corpo do programa e não apenas no início, como visto em C. Exemplos de declaração:

```
int a;
char c1;
```

A atribuição é feita com o sinal de igual. Pode-se atribuir um valor já na declaração:

```
int x = 1;
```

Ou pode-se atribuir um valor depois da declaração:

```
a = 1;
```

Comandos de Entrada/Saída

Não há um paralelo direto com os comandos Leia e Escreva (ReadLn/WriteLn do Pascal), nem mesmo com as funções scanf/printf (da biblioteca stdio.h, do C), porque o Java, sendo independente de plataforma, não poderia lidar diretamente com os diversos *hardwares* envolvidos na entrada e saída de dados. Isto, obviamente, poderia ser encapsulado em uma classe e disponibilizado como uma extensão da linguagem em uma forma mais amigável, mas os projetistas de Java devem ter preferido deixá-la mais "pura"... (outro motivo plausível é argumentar que não é mais usual escrever e ler algo diretamente na tela, já que existem objetos como caixas de edição e *labels* para tal). De qualquer forma, existe a classe System, que lida - ainda que de uma forma mais complexa - com entrada e saída. Para escrever algo na tela, veja o exemplo abaixo:

```
System.out.println("o valor da variavel x é " + x);
```

Para ler diretamente do teclado é um pouco mais complexo. Veja o exemplo completo abaixo que lê um caracter e diz qual seu código ASCII (cada caracter tem um número pelo qual é reconhecido no computador; este número é definido no American Standard Code for Information Interchange – ASCII).

```
public class Tecla
{
  public static void main(String args[])
  throws java.io.IOException
  {
    System.out.print("Digite uma tecla... ");
    int s = System.in.read();
    System.out.print("O codigo ASCII da tecla e': " + s );
  }
}
```

Para ler strings é bem mais complicado e deve-se usar a classe BufferedReader. Veja exemplo abaixo:

```
...
System.out.print("Entre seu nome: ");
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String s = br.readLine();
...
```

Na versão 1.5 de Java foi criada uma nova classe (Scanner) para ler e fazer "parses" em qualquer String (maiores detalhes: http://java.sun.com/j2se/1.5.0/docs/api/java/util/Scanner.html). Na prática, isto possibilita uma forma mais fácil de fazer leitura de caracteres do teclado. Veja os exemplos abaixo:

```
import java.util.*;

public class leitura
```

```
{
  public static void main(String args[])
  {
    System.out.print("Digite um numero inteiro: ");
    try // o try-catch é só para garantir controle de erros...
    {
      // exemplo de leitura de um inteiro
      int i = new Scanner(System.in).nextInt(); // lê o primeiro inteiro
      System.out.println("voce digitou "+i);
      /* também poderíamos criar um objeto Scanner e depois usá-lo,
         como abaixo:
         Scanner sc=new Scanner(System.in);
         int i = sc..nextInt();
      */
    }
    catch (Exception e)
    {
      System.out.println("digitou algo errado!");
    }
  }
}
```

para ler um String:

```
...
System.out.print("Digite um nome qualquer: ");
String s = new Scanner(System.in).nextLine();
// acima lê a "frase" toda (usando “next” delimita por espaço)
System.out.println("voce digitou "+s);
...
```

para leitura de qualquer coisa (um float por exemplo) é só ler um string e fazer a devida conversão...

```
...
try // de novo, o try-catch é só para garantir...
{
  System.out.print(
    "Digite um numero real (use ponto decimal, nao virgula): ");
  String sf = new Scanner(System.in).next();
  // acima, lê a primeira String antes de um espaço em branco
  float f = (float) Float.parseFloat(sf); // converte
  System.out.println("voce digitou "+f);
}
catch (Exception e)
{
  System.out.println("digitou algo errado!");
}
...
```

Operadores

Matemáticos:
+ - * / (os clássicos)
++ (incremento, por exemplo X++ é equivalente a X=X+1)
-- (decremento, por exemplo X-- é equivalente a X=X-1)

Relacionais:
== (a comparação é com dois sinais de igual)
>, >=, <, <= (maior, maior ou igual, menor, menor ou igual)

!= (diferente – ou não igual...)

Lógicos:
&& (e)
|| (ou)
! (não)

Comandos de Decisão

```
if (expressão) // é o “se”
{
  ...comandos...
}
else
{
  ...comandos...
}

switch (expressão) // é o “caso”. A expressão deve ser int ou char
{
case valor1:
  ...comandos...
  break; // usado para sair do case.
case valor2:
  ...comandos...
  break;
case valor3:
  ...comandos...
  break;
}
```

Comandos de Repetição

```
for (expr_inicial; condição_boolean; incremento)
{
  ...comandos...
}

while(condição_boolean)
{
  ...comandos...
}

do // parece o “repita”, mas a condição é “enquanto”, não “até”
{
  ...comandos...
}
while(condição_boolean);
```

Vetores

Em Java, um Vetor é uma classe, que deve ser declarada e seus objetos criados. Para declarar, basta adicionar chaves à declaração. Veja exemplo:

```
char s[];
```

Criando um Vetor

Para criar um vetor, defina a classe e o objeto, usando a palavra new. Veja exemplo:

```
String nomes[];
nomes = new String[3];
nomes[0]= “João”;
nomes[1]=“José”;
nomes[2]=“Maria”;
```

Outra possibilidade é juntar os valores para criação:

```
String nomes[] = { “João”, “José”, “Maria”};
```

Em sala, veremos alguns exemplos de programas em Java. Para criá-los, basta um editor de textos simples – o Bloco de Notas, por exemplo. Gravamos o programa com a extensão “.java”. Para rodar o programa, devemos primeiro transformá-lo em um código “pseudo-compilado” (em *bytecodes*), com o aplicativo javac, gratuitamente disponível no *site* da Sun, criadora do Java. Este código gerado (que ganha a extensão “.class”) é que vem a ser portável para outras plataformas. De posse de um programa em *bytecodes*, pode-se levá-lo para qualquer computador (mesmo que use outro sistema operacional ou arquitetura de *hardware*), desde que tenha a Máquina Virtual Java (JVM – Java Virtual Machine) instalada.

Alguns Exemplos...

```
// um simples alô mundo

public class Alo
{
  public static void main(String args[])
  {
    System.out.println("Alo, mundo!");
  }
}
```

```
// uma saudação em tela com exemplo de entrada de string (refaça usando Scanner)

import java.io.*;

public class Saudacao
{
  public static void main(String args[])
  throws java.io.IOException
  {
    System.out.print("Entre seu nome: ");
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    String s = br.readLine();
    System.out.print("Ola, " + s );
  }
}
```

```
// o quadrado de um número fixo

public class Quad
{
  public static void main(String args[])
  {
    int b;
    b = 5;
    System.out.println("O quadrado de "+b+" e' "+b*b);
  }
}
```

```
// o quadrado de um número inteiro com entrada via teclado (refaça usando Scanner)

import java.io.*;

public class Quadrado
{
  public static void main(String args[])
  throws java.io.IOException
  {
    System.out.print("Entre um numero inteiro: ");
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    String s = br.readLine();
    int x =(new Integer(s)).intValue();
    System.out.print("O quadrado do numero e' " + (x*x) );
  }
}
```

```
// sorteio de dezenas de loto

public class Loto
{
  public static void main(String args[])
  {
    int d1,d2,d3,d4,d5;
    d1= (int) (Math.random()*100);
    do { d2= (int) (Math.random()*100); } while (d2==d1);
    do { d3= (int) (Math.random()*100); } while ((d3==d1)||(d3==d2));
    do { d4= (int) (Math.random()*100); } while ((d4==d1)||(d4==d2)||(d4==d3));
    do { d5= (int) (Math.random()*100); } while ((d5==d1)||(d5==d2)||(d4==d3)||(d5==d4));
    System.out.println("Dezenas: "+d1+"-"+d2+"-"+d3+"-"+d4+"-"+d5);
  }
}
```

```
// maior de 2 números fixos

public class Maior
{
  public static void main(String args[])
  {
    int x = 8;
    int y = 15;
    if (x>y)
    {
      System.out.println("O maior e' "+x);
    }
    else
    {
      System.out.println("O maior e' "+y);
    }
  }
}
```

```
// maior de 2 números digitados junto à chamada do programa (ex: java Maior2 15 23)

public class Maior2
{
  public static void main(String args[])
  {
    if (args.length != 2)
    {
      System.out.println("Erro! Sintaxe: java Maior2 Numero1 Numero2");
    }
    else
    {
      int N1 = Integer.parseInt(args[0]);
```

```
      int N2 = Integer.parseInt(args[1]);
      if (N1>N2)
      {
        System.out.println("O maior e' "+N1);
      }
      else
      {
        System.out.println("O maior e' "+N2);
      }
    }
  }
}
```

```
// maior de 2 números sorteados pelo programa

public class Maior3
{
  public static void main(String args[])
  {
    int N1 = (int) (Math.random()*100);
    int N2 = (int) (Math.random()*100);;
    System.out.println("Os numeros sorteados foram: "+N1+" e "+N2);
    if (N1>N2)
    {
      System.out.println("O maior e' "+N1);
    }
    else
    {
      System.out.println("O maior e' "+N2);
    }
  }
}
```

```
public class Objetos
{
  public static void main(String args[])
  {
    class FormasGeometricas
    {
      String cor;
      String nome;
    }
    class Triangulo extends FormasGeometricas
    {
      double base,altura;
    }
    class Circulo extends FormasGeometricas
    {
      double diametro;
    }
    class Retangulo extends FormasGeometricas
    {
      double altura,largura;
    }
    Circulo c1;
    c1 = new Circulo();
    c1.cor = "vermelho";
    c1.nome = "bola";
    c1.diametro = 20; // centimetros
    double d = c1.diametro;
    System.out.println("o circulo C1 , um(a) "+c1.nome);
    System.out.println("de cor "+c1.cor);
    System.out.println("cuja area e' de "+(3.1415*(d/2)*(d/2))+" centimetros quadrados");
  }
}
```

Java: Prós

Portabilidade

Java é realmente multiplataforma, ou seja: roda em Windows, Mac, Solaris, HP-UX, Linux, IBM-Unix e qualquer dispositivo que disponha de capacidade de processamento e que tenha uma JVM (Máquina Virtual Java) embutida (quase tudo tem uma JVM embutida…).

Dimensionabilidade

De acordo com o tamanho do problema e do equipamento a ser utilizado, Java vem em três edições: Micro Edition (J2ME, para dispositivos menores em termos de capacidade de processamento, como celulares), Standard Edition (J2SE, a edição padrão, para a maioria das aplicações comuns, gráficas ou não) e Enterprise Edition (J2EE, para aplicações empresarias multicamadas, distribuídas, mais complexas).

Além disso, temos uma ampla gama de possibilidades usando Java: programas em modo texto ou gráfico, mono ou multisuário, cliente-servidor tradicional desktop ou *web*, *applets*, *servlets*, JSP e até uma linguagem de *script* (Javascript) que, embora seja meio "bastarda" (não é cria da Sun, é da Netscape) vem completar a família.

Comandos da linguagem C

A Sun não reinventou a roda onde não era necessário. Os comandos básicos de Java são os mesmos de C (if, switch, for, while, do...) facilitando a adaptação de quem já conhece a sintaxe.

OO "de nascença"

Java já nasceu orientada a objeto, ao contrário de outras linguagens que tiveram de sofrer adaptações para passarem a ser OO, o que possivelmente a torna mais robusta neste sentido.

Pronta para a internet

Java nasceu praticamente junto com a popularização da internet e incorporou nativamente várias formas de uso junto à *web*.

Várias ferramentas *free*

Java conta com uma série de aplicativos (seja da própria Sun, seja de terceiros), muitas vezes gratuitos e/ou open-source para facilitar e difundir o uso da linguagem.

Java: Contras

Complexidade

Algumas tarefas são, pela natureza da linguagem (e um pouco pelo purismo de alguns programadores da Sun), mais complicadas. Por exemplo: para escrever algo na tela em puro Pascal usamos apenas Writeln(...), um comando simples e direto. Em Java devemos chamar um método de um atributo de uma classe para tal: System.out.print(...);

O equivalente a um simples Readln(x), pior ainda...

Lentidão

Por ser "pseudo-compilado" e ter que carregar a JVM antes de ser executado, um programa escrito em Java pode acabar rodando um pouco mais lento que um similar escrito em linguagem compilada nativa de um determinado sistema (embora exista a possibilidade de compilar nativamente um programa Java em alguns ambientes, mas aí talvez seja melhor usar uma ferramenta específica para o ambiente).

Requer software adicional:

Você deve ter a JVM antes de rodar programas em Java (felizmente, isto é gratuito e fácil de achar).

Concorrência

Obviamente, não é um problema da linguagem. Mas nunca devemos esquecer que, embora a Sun seja uma grande empresa, a plataforma briga diretamente contra a gigante Microsoft e também contra algumas alternativas, digamos, mais *free* (PHP, por exemplo). Estas brigas podem enfraquecer, em longo prazo, a penetração do Java no mercado. Como não tenho bola de cristal, não sei quem acabará levando a melhor. É bom conhecer várias alternativas para se preparar para o futuro.