FACAMP
FACULDADES DE CAMPINAS

Laboratório de Programação

Aula 1

Revisão de linguagem Java

2º semestre de 2019
Prof José Martins Junior

A sintaxe Java

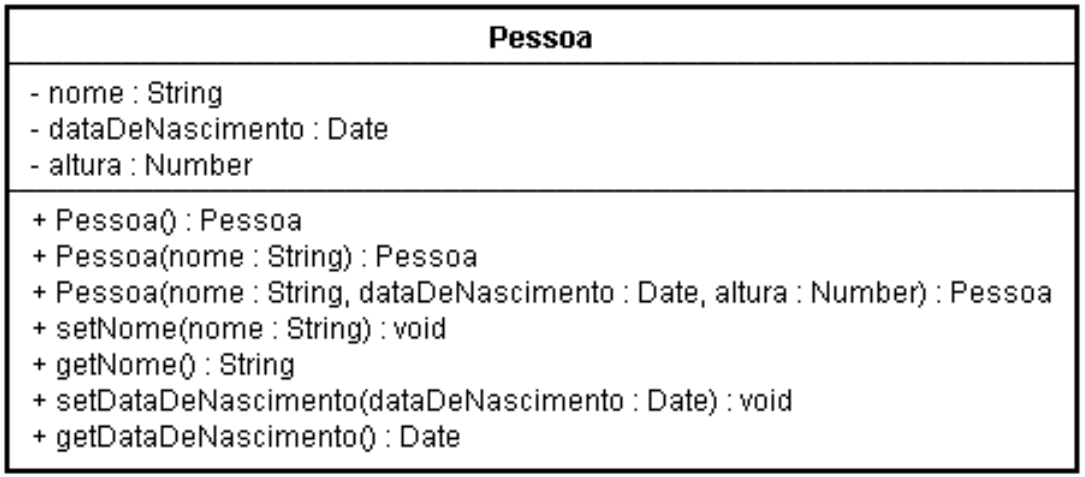
- Sintaxe parecida com a linguagem C
- *Case sensitive*
- Blocos são delimitados por chaves **{ }**
- Toda instrução deve terminar com ponto-e-vírgula **;**
- Bloco principal: nome da classe (= nome do arquivo)
- Blocos internos: métodos, construtores e destrutores
- Primeiras linhas: definição de pacote e **import** de classes
- Método **main**: obrigatório em aplicativos Java
- Variáveis: tipos básicos ou classes de objetos
- Java é uma linguagem fortemente tipada
- Comentários: // ou /* ... */ ou /** ... */ (JavaDoc)

Classes em Java

- As classes em Java
  - Definem modelos para instâncias de objetos
  - São identificadas por nomes simples ou compostos (sem espaços ou caracteres especiais)
  - O código-fonte de uma classe deve ser armazenado em um arquivo com o mesmo nome da classe + extensão **.java**
  - Uma classe deve ser compilada (javac) e produzida a sua versão *bytecode*, armazenada em um arquivo com o mesmo nome, mas com extensão **.class**

Exemplo de classe

```
//o código deve ser armazenado em um arquivo Pessoa.java
public class Pessoa {  //chaves delimitam o código da classe
    //atributos de classe - início
    private String nome;
    private java.util.Date dataDeNascimento;
    private Number altura;
    //atributos de classe - fim
    public Pessoa() {
    }
    public Pessoa(String n) {
        nome = n;
    }
    // ... outros métodos
}
```



Tipos primitivos de dados

- Inteiros
  - int (4 bytes)
  - long (8 bytes)
  - short (2 bytes)
  - byte (1 byte)
- Ponto flutuante: simples e dupla precisão
  - float (4 bytes)
  - double (8 bytes)
- Booleanos (boolean): admitem os valores true ou false

Tipos primitivos de dados

- Caractere (char): utiliza 2 bytes (Unicode)
  - Contém os caracteres ASCII/ANSI
  - Pode ser expresso em hexadecimal de '\u0000' a '\uFFFF', indicado por aspas simples
  - Sequências de escape e respectivos valores Unicode

\b (retrocesso)	\u0008
\r (retorno carro)	\u000d
\t (tab)	\u0009
\" (aspas)	\u0022
\n (nova linha)	\u000a
\' (apóstrofo)	\u0027
\\ (barra invertida)	\u005c

Variáveis

- Devem ser **declaradas** quanto ao tipo (básico ou objeto), que precede o nome da variável, como na linguagem C
- Nomes de variáveis
  - Evitar palavras reservadas, caracteres especiais..
  - Iniciar com letra minúscula (em compostas, maiúscula na primeira letra de cada termo subsequente)
- Todas as variáveis devem ser **inicializadas**
- Conversões de tipo
  - Operações entre tipos diferentes de números - Java tratará os elementos como o maior tipo (Ex.: multiplicação int por double: ambos serão tratados como double e o resultado será um double)
  - *Casting*: permite a conversão para um tipo numérico menor (com perda de precisão). Ex.: float f = 10/3; int i = (int) f;
  - Conversão direta na atribuição: de um tipo menor para um maior
- Constantes: declaradas fora do método com **static final**

Objetos em “Variáveis”

- Além dos tipos básicos, uma “variável” em Java pode ser um objeto
  - Devem ser definidas (declaradas)
    - Ex.: String s;
  - O objeto a que se refere deve ser instanciado através do operador **new**, atribuição direta ou pelo retorno de um método
    - Exs.: s = new String("exemplo");
      s = "exemplo";
      String sub = s.substring(0,3);

- IMPORTANTE
  - Se tentarmos acessar um membro (dado) ou método de um objeto não instanciado através do nome da variável definida como tal objeto, obteremos um erro em tempo de execução (*NullPointerException*)

Operadores aritméticos

- Soma, subtração, multiplicação, divisão, resto: + - * / %

  / indica divisão inteira se ambos operadores forem inteiros, e ponto flutuante, caso contrário

  Exs.: 15/4 = 3 e 15/4.0 = 3.75

  % indica o resto (mod) de uma divisão inteira

  Ex.: 15%4 = 3

- Exponenciação (Math.pow())

  Ex.: y = Math.pow(x,a); (x^a ; x,a double)

- Incremento e decremento: incrementam/decrementam em 1
  - Sufixados: i++; i--; (são realizados depois da expressão)

    Ex.: int i = 2; int j = 10 * i++; (j será igual a 20)
  - Prefixados: ++i; --i; (são realizados antes da expressão)

    Ex.: int i = 2; int j = 10 * ++i; (j será igual a 30)

Operadores relacionais e booleanos

- Igualdade e desigualdade

  == (igual) != (diferente) < (menor) > (maior)

  <= (menor ou igual) >= (maior ou igual)
- Curto-circuito

  && (and) || (or)

  Se 1° termo for false em && ou true em ||, 2° não é calculado
- de Bit

  & (and) | (or) ^ (xor) ~ (not)

  >> (deslocamento a direita; estende o sinal para bits do alto)

  >>> (deslocamento a direita; preenche bits do alto com zeros)

  << (deslocamento a esquerda)
- Atalhos para atribuição

  += -= *= /= %= &= |= ^= <<= >>= >>>=

  Ex.: x += 4; (somará 4 ao valor anterior de x)

Hierarquia de operadores

[] () chamada de método	esquerda para direita
! ~ ++ -- +(unário) -(unário) Casting new	direita para esquerda
* / %	esquerda para direita
+ -	esquerda para direita
<< >> >>>	esquerda para direita
< <= > >= instanceof	esquerda para direita
== !=	esquerda para direita
&	esquerda para direita
^	esquerda para direita
|	esquerda para direita
&&	esquerda para direita
||	esquerda para direita
?:	esquerda para direita
= += -= *= /= %= &= |= ^= <<= >>= >>>=	direita para esquerda

Operações com Strings (1)

- Diferentemente de C, **String** é uma **classe** em Java
- Facilidade: pode receber atribuição direta

  Exs.:
  ```
  String s = "";
  String oi = "oi";
  ```
- Concatenação: utilizando sinal +

  Exs.:
  ```
  String saudacao = oi + " " + "amigo";
  System.out.println(oi + " amigo");
  System.out.println(saudacao);
  ```
- Conversão direta: de tipos numéricos

  Exs.:
  ```
  int soma = 1 + 1;
  String res = "Soma= " + soma;
  System.out.println("Soma = " + soma);
  System.out.println(res);
  System.out.println(”Soma = " + (1+1));
  ```

Operações com Strings (2)

- Métodos úteis da classe String
  - substring(i,j) - obtém String que é parte de uma String: i = índice do 1° caracter; j = i + tamanho da substring

    Ex.: String hello = "Hello";

    System.out.println(hello.substring(0,4)); //resultará “Hell”
  - equals(s) - compara Strings

    Ex.: hello.equals(hello.substring(0,4)+"o"); //retornará true
  - charAt(n) - retorna o caractere Unicode na posição n
  - length() - retorna o tamanho (int) de uma String
  - indexOf(s) - procura ocorrência de s na String e retorna a posição (int) do primeiro caractere de s na String
- Importante
  - O índice dos caracteres em uma String vai de 0 a length() -1
  - CUIDADO: (string1 == string2) só é verdadeira se as variáveis apontam para a mesma área na memória

Desvio condicional

- Formato geral

if (*condição***) {**

instruções; ...

}

[else {

instruções; ...

}]

  - Se a condição for verdadeira, o bloco de instruções da cláusula **if** é executado
  - Caso contrário, executa-se o bloco da cláusula **else** (se esta estiver presente)

Loop indeterminado

- Formato geral (duas formas: while e do-while)

while (*condição***) {**

instruções; ...

}

do {

instruções; ...

} while (*condição***);**

  - Enquanto a condição for verdadeira, o bloco de instruções é executado (não há número predefinido de execuções)
  - A diferença é que no do-while o bloco é executado a primeira vez, mesmo que a condição seja falsa (pois é testada no final)

Loop determinado

- Formato geral

  for (*inicialização* **;** *condição* **;** *passo***) {**

  instruções; ...

  }

  - O bloco de instruções é repetido por um número predefinido de iterações
  - Geralmente, uma variável (contador) recebe um valor inicial, é comparada a cada loop com um valor, após ser incrementada ou decrementada pelo passo definido

Seleções múltiplas

- Formato geral

switch (*escolha***) {**

case *valor***:** *instruções***;** ... **break;**

case *valor***:** *instruções***;** ... **break;**

default: *instruções***;**

}

  - Executa o bloco de instruções cujo valor da case corresponder com a variável de escolha (até a próxima instrução break)
  - Podem ser comparados valores inteiros (int) ou char; a partir da JDK7, também String

Arrays

- Em Java são objetos e devem ser instanciados

  Ex.: int[] arrayDeInts = new int[100];

  ```
  for (int i = 0; i < 100; i++) {
          arrayDeInts[i] = i;  //0 a 99
  }
  ```

- Atalho na atribuição

  Ex.: int[] numeros = {1,2,3,4,5,6,7,8,9,0};

- campo length - indica o tamanho do array

  Ex.: numeros.length vale 10

- método System.arraycopy(ArrayOrigem, posicaoInicial, arrayDestino, posicaoDestino, numDeElementosParaCopiar)
  - Realiza cópia de arrays em bloco na memória

Recebendo argumentos na chamada do aplicativo

- O método main
  - Recebe um Array de Strings como argumentos, provenientes da chamada do aplicativo (linha de comando)

```
public static void main (String[] args)
```

- Quantidade de argumentos fornecida
  - Pode ser obtida de args.length
- Os argumentos são acessados pelo índice do Array: de 0 a (args.length - 1)

```
if (args.length > 0) {
        String arg1 = args[0];
}
```

Exemplo: Exemplo1.java

```
public class Exemplo1 {
   public static void main(String[] args) {
     String primeiro = "";
     if (args.length < 1)
       System.out.println("Nenhum argumento foi fornecido");
     else {
       primeiro = args[0];
       System.out.println("Primeiro argumento: " + primeiro);
       System.out.print("Todos os argumentos: ");
       for (int i = 0; i < args.length; i++)
         System.out.print(args[i] + " ");
       System.out.print("\n");
     }
   }
}
```

Obtendo entrada em stdin

- A classe Scanner permite leitura de dados em um stream
  - Pode também ler dados da entrada padrão (System.in)
- O próximo exemplo ilustra o uso de tal classe

Exemplo: Exemplo2.java

```
import java.util.Scanner;
public class Exemplo2 {
  public static void main(String[] args) {
    String s;
    int i;
    Scanner scanner = new Scanner(System.in);
    System.out.print("Digite uma string: ");
    s = scanner.next();
    System.out.println("String digitada: " + s);
    System.out.print("Digite um numero inteiro: ");
    i = scanner.nextInt();
    System.out.println("Numero digitado: " + i);
  }
}
```