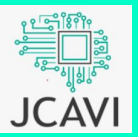
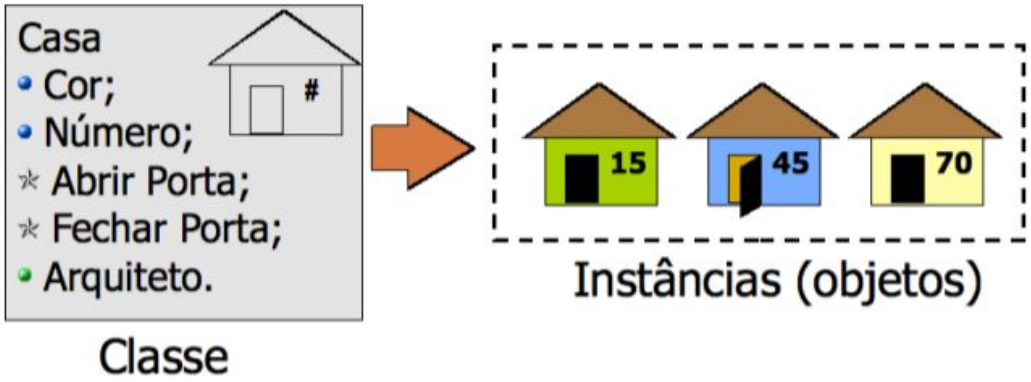
Estruturas Java

prof.: Marcelo da Silva



Classes

- Uma classe descreve um conjunto de objetos com as mesmas propriedades, o mesmo comportamento, os mesmos relacionamentos com outros objetos e a mesma semântica;
- Parecido com o conceito de tipo.



Objetos

- Objeto = Instância de classe;
- Paradigma OO norteia o desenvolvimento por meio de classificação de objetos:
  - Modelamos classes, e não objetos;
  - Objetos são entidades reais – executam algum papel no sistema;
  - Classes são abstrações – capturam a estrutura e comportamento comum a um conjunto de objetos.

Palavras Reservadas

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Criando Variáveis

O nome de um item no programa escrito em Java Variáveis, métodos, classes e labels. Não podem ser palavras reservadas Devem iniciar com letra, cifrão($) ou sublinhado(_). Por convenção: Nomes de variveis e métodos iniciam com minúsculas e as palavras subsequentes devem iniciar com maiúsculas. • Ex: nome, nomeComposto, etc... Nomes de classes iniciam com maiúsculas e as palaváras subsequentes também. • Ex: Classe, ClasseComposta, etc... Nomes de constantes são escritos em caixa alta e os nomes compostos separados pelo caracter '_'. • Ex: CONSTANTE, OUTRA_CONSTANTE

Caracteres Especiais

Código	Significado
\n	Quebra de linha (newline ou linefeed)
\r	Retorno de carro (carriage return)
\b	Retrocesso (backspace)
\t	Tabulação (horizontal tabulation)
\f	Nova página (form feed)
\'	Aspas simples (apóstrofo)
\"	Aspas
\\	Barra invertida (“\”)
\u233d	Caractere unicode de código 0x233d (hexadecimal)

Tipos Primitivos

Classificação	Tipo	Descrição
Lógico	boolean	Pode possuir os valores true (verdadeiro) ou false (falso)
Inteiro	byte	Abrange de -128 a 127 (8 bits)
	short	Abrange de -32768 a 32767 (16 bits)
	int	Abrange de -2147483648 a 2147483647 (32 bits)
	long	Abrange de -2^{63} a $(2^{63})-1$ (64 bits)
Ponto Flutuante	float	Abrange de 1.40239846^{-45} a 3.40282347^{+38} com precisão simples (32 bits)
	double	Abrange de $4.9406564584124654^{-324}$ a $1.7976931348623157^{+308}$ com precisão dupla (64 bits)
Caracter	char	Pode armazenar um caracteres unicode (16 bits) ou um inteiro entre 0 e 65535

Tipos de Variáveis

- Variáveis que pertencem à classe como um todo;
- Variáveis que pertencem a objetos (instâncias) da classe (atributos definidos para a classe);
- Variáveis locais.

```
public class Variavel {
  public static int c = 10; // De classe.
  public int i;             // De instância.
  public int func() {
     int n = 5;             // Local.
     i = 2 * n;
     return (i + c);
  }
}
```

Conversões entre tipos numéricos

- Tipos numéricos podem se misturar em operações, seguindo as seguintes regras:
  - Se um dos operandos for `double` e o outro não, será convertido para `double`;
  - Senão, se um dos operandos for `float` e o outro não, será convertido para `float`;
  - Senão, se um dos operandos for `long` e o outro não, será convertido para `long`;
  - Nos demais casos, os operandos serão sempre convertidos para `int`, caso já não o sejam.

Conversões

```
byte b = 2; short s = 4; int i = 8;
long l = 16; float f = 1.1f; double d = 9.9;

d = s + l + d;
    long
        double

// i + b resulta em int, convertido pra long;
l = i + b;

// Erro: b + s resulta em int!
s = b + s;
```

Float é o único que precisa especificar o tipo do valor literal.

Conversões entre tipos

- Conversões de um tipo menor para um tipo maior ocorrem automaticamente;
- Podemos forçar conversões no sentido contrário, sabendo das possíveis perdas;
- Utilizamos o operador de coerção (*cast*):

```
double x = 9.997;
int nx = (int)x;
System.out.println(nx); // 9

nx = (int)Math.round(x);
System.out.println(nx); // 10
```

Constantes

- Para declarar constantes, basta usar a palavra-chave final:

```
public class Constantes {
  public static void main(String[] args) {
    final double CM_POR_POLEGADA = 2.54;
    CM_POR_POLEGADA = 2.55; // Erro!

    double larguraPol = 13.94;
    double larguraCm = larguraPol * CM_POR_POLEGADA;

    System.out.println(larguraCm);
  }
}
```

Constantes

- A convenção é que seu nome seja escrito com letras maiúsculas;
- É mais comum encontrar constantes como membros de classes ao invés de propriedades de instância ou variáveis locais.

```
public class MinhaClasse {
  public static final int CONSTANTE = 100;
  private static final int CONST_INTERNA = 1;

  /* ... */
}
```

Comentários

- Ignorados pelo compilador;
- Usados pelo programador para melhorar a legibilidade do código;
- Existem três tipos:
  - Comentários de uma linha: // ...;
  - Comentários de múltiplas linhas: /* ... */;
  - Comentários JavaDoc: /** ... */ – utilizados pela ferramenta javadoc para criar uma documentação HTML das classes, atributos e métodos.
  - A ferramenta javadoc vem com o JDK;
  - Mais informações na apostila.

Operadores

- Símbolos especiais que recebem um ou mais argumentos e produzem um resultado;
- Os operadores Java trabalham somente com tipos primitivos (e wrappers), exceto:
  - ✓ =, == e != podem ser aplicados a objetos;
  - ✓ + e += podem ser aplicados a *strings*.

Operadores Aritméticos

Símbolo	Significado	Exemplo
+	Adição	a + b
-	Subtração	a - b
*	Multiplicação	a * b
/	Divisão	a / b
%	Resto da divisão inteira	a % b
-	(Unário) inversão de sinal	-a
+	(Unário) manutenção de sinal	+a
++	(Unário) Incremento	++a ou a++
--	(Unário) Decremento	--a ou a--

Operadores Relacionais

Símbolo	Significado	Exemplo
==	Igual	a == b
!=	Diferente	a != b
>	Maior que	a > b
>=	Maior que ou igual a	a >= b
<	Menor que	a < b
<=	Menor que ou igual a	a <= b

- Observações:
  - Os parâmetros devem ser de tipos compatíveis;
  - Não confunda = (atribuição) com == (igualdade).

Operadores Lógicos

Símbolo	Significado	Exemplo
&&	AND (E)	a && b
&	AND sem curto-circuito	a & b
||	OR (OU)	a || b
|	OR sem curto-circuito	a | b
^	XOR (OU exclusivo)	a ^ b
!	NOT (NÃO, inversão de valor, unário)	! a

- Observações:
  - Só operam sobre valores lógicos;
  - Podem ser combinados em expressões grandes.

Atribuição Composta

- Une-se um operador binário com o sinal de atribuição:

Expresão	Equivale a
x += y	x = x + y
x -= y	x = x - y
x *= y	x = x * y
x /= y	x = x / y
x %= y	x = x % y

Expresão	Equivale a
x &= y	x = x & y
x |= y	x = x | y
x ^= y	x = x ^ y
x >>= y	x = x >> y
x <<= y	x = x << y

Incremento e Decremento

- Somar / subtrair 1 de uma variável inteira é tão comum que ganhou um operador só para isso:
  - ✓ ++x e x++ equivalem a x = x + 1;
  - ✓ --x e x-- equivalem a x = x - 1.
- Quando parte de uma expressão maior, a forma prefixada é diferente da pós-fixada:

```
int x = 7;
int y = x++; // y = 7, x = 8.

x = 7;
y = ++x;     // y = 8, x = 8.
```

Operador Ternário

- Forma simplificada de uma estrutura `if – else` (que veremos no próximo capítulo);
- Produz um valor de acordo com uma expressão:
  - ✓ `<expressão> ? <valor 1> : <valor 2>`
  - ✓ Se `<expressão>` for `true`, o resultado é `<valor 1>`, do contrário o resultado é `<valor 2>`.

```
int x = 7;

int y = (x < 10) ? x * 2 : x / 2;
System.out.println("y = " + y); // y = 14
```