Inatel
Instituto Nacional de Telecomunicações

Programação Orientada a Objetos

Cap.2 - Variáveis Primitivas e Controle de Fluxo

Prof. Me. Renzo P. Mesquita

Capítulo 2
Variáveis Primitivas e Controle de Fluxo

2.1. Comentar é essencial!
2.2. Variáveis e tipos primitivos;
2.3. Casting e Promoção;
2.4. Saída de Dados;
2.5. Entrada de Dados;
2.6. Estruturas de Decisão;
2.7. Controlando Loops;
2.8. Escopo das Variáveis;

Objetivos

Aprender a trabalhar com recursos fundamentais da linguagem Java como: declaração de variáveis, casting de valores, controladores de fluxo (if-else, switch-case) e estruturas de repetição (for, while);

2.1. Comentar é essencial!

Como conversamos, programas em Java são desenvolvidos geralmente por uma equipe de programadores. Logo, comentar o código fonte é ESSENCIAL para manter a organização do projeto.

Os diferentes tipos de comentários em Java são:

1. *// Este é um comentário de uma linha*

2. *\/* Este é um comentário de várias linhas,*
   escreva o quanto quiser até que se indique
   o final do comentário\/*

3. *\/** Este é um comentário de documentação*
   @author <nomeDoAutor>
   @version <versaoDaAplicacao>
   @since <dataDeCriacaoDaAplicacao>
   **\/*

// *

/**

2.2. Variáveis e Tipos Primitivos

O Java é uma linguagem FORTEMENTE TIPADA, ou seja, o programador ao declarar uma variável obrigatoriamente deve colocar o seu tipo.

Uma variável em Java é declarada da seguinte forma:
tipoDaVariavel nomeDaVariavel = valorInicial;

Importante: o "=" significa atribuição (como na variável) e o "==" comparação (como no If-Else).

tipoDaVariavel pode-se usar por exemplo:

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^-46 a 3.40282347^38 com precisão simples (32 bits)
	double	Abrange de 4.94065645841246544^-324 a 1.7976931348623157^+308 com precisão dupla (64 bits)
Caracter	char	Pode armazenar um caracter unicode (16 bits)

Ué! Ta faltando outro tipo muito importante aí, não está?

2.2. Variáveis e Tipos Primitivos

Para representar textos o Java não possui um tipo primitivo, ele possui uma Classe chamada String. Essa classe pode ser usada de modo semelhante a um tipo primitivo e ainda conta com diversos métodos disponíveis para realizar manipulações sobre Strings.

Uma String em Java é declarada da seguinte forma:
String minhaString;

Exemplos de métodos úteis para manipulação de Strings:

```
minhaString.length(); // Mostra o tamanho da String
minhaString.toUpperCase(); // Coloca tudo em Maiúsculo
minhaString.equalsIgnoreCase(minhaString2); // Compara Strings
minhaString.startsWith("Java"); // Verifica se começa com uma palavra
minhaString.replace("C#", "Java"); // Troca um pedaço por outro
```

E muitos outros! Para uma lista completa, consulte no Javadoc a classe String.

2.3. Casting e Promoção

Alguns valores são imcompatíveis se tentarmos fazer uma atribuição direta.

Exemplos:

1.
```
double d = 3.1415;
int i = d; // não compila
```

2.
```
double d = 5; // ok, o double pode conter um número inteiro
int i = d; // não compila
```

3.
```
long x = 10000;
int i = x; // não compila, pois pode estar perdendo informação
```

O que podemos fazer para resolver estes tipos de problemas? Usarmos do Casting.

2.3. Casting e Promoção

O Casting evita que aconteça erros na compilação e permite moldar (cast) um valor de um tipo em uma variável de outro tipo.

Exemplos:

1.
```
double d3 = 3.14;
int i = (int) d3;
```
A valor de i agora é 3.

2.
```
long x = 10000;
int i = (int) x;
```
i recebe o valor sem problemas.

Castings possíveis no Java:

PARA: / DE:	byte	short	char	int	long	float	double
byte	----	*Impl.*	(char)	*Impl.*	*Impl.*	*Impl.*	*Impl.*
short	(byte)	----	(char)	*Impl.*	*Impl.*	*Impl.*	*Impl.*
char	(byte)	(short)	----	*Impl.*	*Impl.*	*Impl.*	*Impl.*
int	(byte)	(short)	(char)	----	*Impl.*	*Impl.*	*Impl.*
long	(byte)	(short)	(char)	(int)	----	*Impl.*	*Impl.*
float	(byte)	(short)	(char)	(int)	(long)	----	*Impl.*
double	(byte)	(short)	(char)	(int)	(long)	(float)	----

** Impl. significa que o Cast pode ser feito automaticamente pelo compilador.*

2.4. Saída de Dados

Acontece quando pegamos algo que esteja na memória do computador e mostramos ao usuário.

Suponha que tenhamos a seguinte variável:

float nota = 7.5;

Quais as diferentes formas de mostrarmos este valor na tela?

1) System.out.print ("Sua nota é:" + nota);

Mostra o valor sem quebra de linha;

2) System.out.println ("Sua nota é:" + nota);

Mostra o valor com quebra de linha;

3) System.out.printf ("Sua nota é: %.2f", nota);

Mostra o valor sem quebra de linha e formatado de acordo com o/os flag(s) utilizado(s). Alguns exempos de flags: *%f - float;* *%s - String;* *%d - inteiro;* *%c - caracter*

2.5. Entrada de Dados

O pacote java.lang contém as classes que constituem recursos básicos da linguagem e que não precisam ser importados. Porém, este pacote não possui um recurso para entrada de dados.

Para realizar a entrada de dados via terminal no Java usa-se o pacote *java.util.Scanner*

Exemplo:

```
// Classe que permitirá a entrada de dados
Scanner valorTeclado = new Scanner(System.in);

// Recebe um valor inteiro digitado pelo usuário
int meuInteiro = valorTeclado.nextInt();
// Recebe um valor em float digitado pelo usuário
float meuFloat = valorTeclado.nextFloat();
// Recebe uma String digitada pelo usuário
String minhaString = valorTeclado.nextLine();
```

2.5. Entrada de Dados

Exercício - GoRangers

Zordon (chefe dos Power Rangers) precisa computar o número de vilões apreendidos/derrotados pelos Rangers.
Utilizando do pacote java.util.Scanner, faça com que o usuário entre com os valores de vilões apreendidos/derrotados pelos Rangers nos últimos 3 anos. Utilize também do método printf para mostrar o resultado na tela.

Dicas:

1. Inicialize suas variáveis com 0;
2. Scanner valorTeclado = new Scanner (System.in);
3. int primeiroMes = valorTeclado.nextInt();
4. System.out.printf("Primeiro mês: %d inimigos derrotados.",primeiroMes);

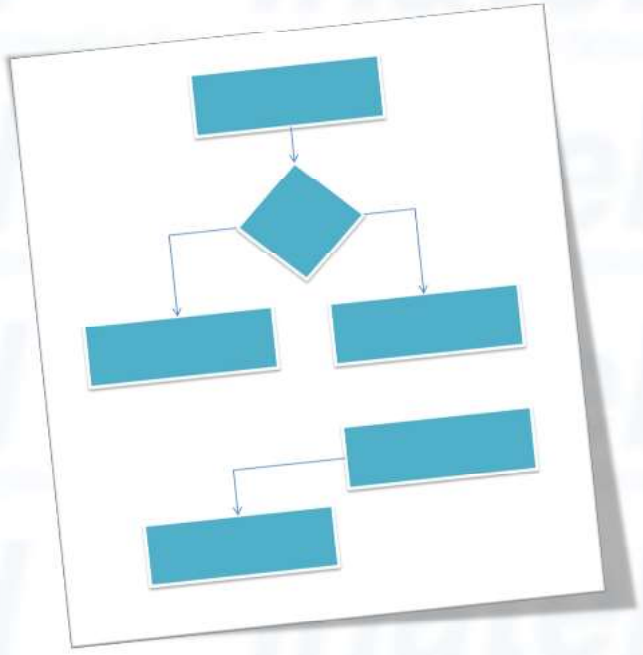
2.6. Estruturas de Decisão

As estruturas de decisão são utilizadas para controlar o fluxo de execução dos aplicativos, possibilitando que a leitura das instruções siga caminhos alternativos em função da análise de determinadas condições.

As estruturas de decisão mais populares são:

1) If-Else

2) Switch-Case



Vamos ver como elas funcionam em Java?

2.6. Estruturas de Decisão

2.6.1. If-Else

Quando usamos uma instrução if, apenas expressões com resultados booleanos (true ou false) podem ser avaliadas.

A sintaxe do if/else:

```
if (expressao booleana)
{
  //Comandos
}
else
{
  //Comandos
}
```

Uma condição booleana é qualquer expressão que retorne true ou false. Para isso, pode-se usar os operadores <, >, <=, >=, && (e), || (ou) e ! (negação).

Exemplo:

```
int idade = 15;
boolean amigoDoDono = true;
if (idade < 18 && amigoDoDono == false) {
    System.out.println("Não pode entrar");
}
else {
    System.out.println("Pode entrar");
}
```

2.6. Estruturas de Decisão

2.6.2. Switch-Case

O Switch-Case é uma forma para se definir diversos desvios no código a partir de uma única variável ou expressão.

A sintaxe do Switch-Case:

```
switch(variável ou expressão)
{
    case valor1:
        <instruções>
        break;
    case valorN:
        <instruções>
        break;
    default:
        <instruções>
}
```

Exemplo:

```
switch(valor)
{
    case 1:
        System.out.println("Valor 1");
        break;
    case 2:
        System.out.println("Valor 2");
        break;
    case 3:
        System.out.println("Valor 3");
        break;
    default:
        System.out.println("Nenhum dos 3!");
}
```

2.6. Estruturas de Decisão

Exercício - PokemonElementals

No mundo do anime Pokemon, cada criatura possui um elemental que o representa que pode ser Fogo, Água, Eletricidade, Pedra, Gelo, Planta entre outros. Faça um software que a partir da entrada de um elemental, o software responda qual é a sua fraqueza. Caso um elemental desconhecido seja inserido, o software deve mostrar um aviso ao usuário.

Elemental	Fraqueza
Fogo	Água
Água	Eletricidade
Eletricidade	Pedra
Pedra	Gelo
Gelo	Fogo
Planta	Fogo

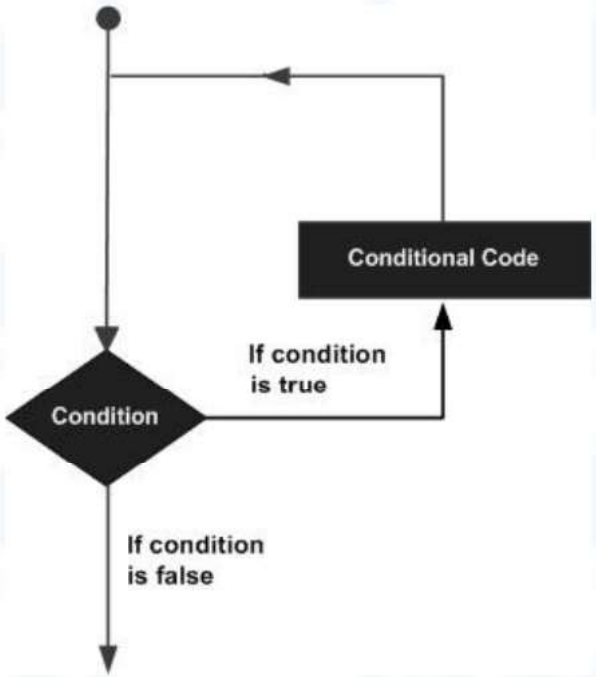
2.7. Controlando Loops

Um loop nos permite executar uma declaração ou um grupo de declarações várias vezes.

O Java oferece os seguintes tipos de loops:

1) while

Repete uma declaração ou um grupo delas enquanto a condição seja verdadeira. Ela testa a condição antes de executar o corpo do loop;



2) do-while

Mesmo caso do while, porém, testa a condição no final do corpo do loop;

3) for

Mesmo caso do while, porém, possui um espaço para inicialização de variáveis e seus modificadores;

2.7. Controlando Loops

Exemplos:

1) while

2) do-while

3) for

```
// Repetindo 10 vezes o quanto o Java é legal!
int i = 0;
while (i < 10)
{
    System.out.println("Java é legal!");
    i++;
}
// Reforçando mais 20 vezes o quanto o Java é legal!
int i = 0;
do
{
    System.out.println("Java é legal!");
    i++;
}
while (i < 20);
// Sem mais! O Java é demais mesmo! :)
for(int i=0;i<100;i++)
{
    System.out.println("Java é legal!");
}
```

2.7. Controlando Loops

Exercício - CanYouGuessTheNumber

Escreva um aplicativo Java que gere um número aleatório inteiro entre 1 e 10, e por meio de testes condicionais, você tem que adivinhar que número é esse. O aplicativo não pode finalizar até que o usuário descubra o número sorteado.

Dicas:

1. Random randomGenerator = new Random();
2. numAleatorio = randomGenerator.nextInt(10) + 1;

2.8. Escopo das Variáveis

No Java, é possível declarar variáveis a qualquer momento. Porém, dependendo de onde as declaramos, ela vai valer apenas dentro de um determinado ponto.

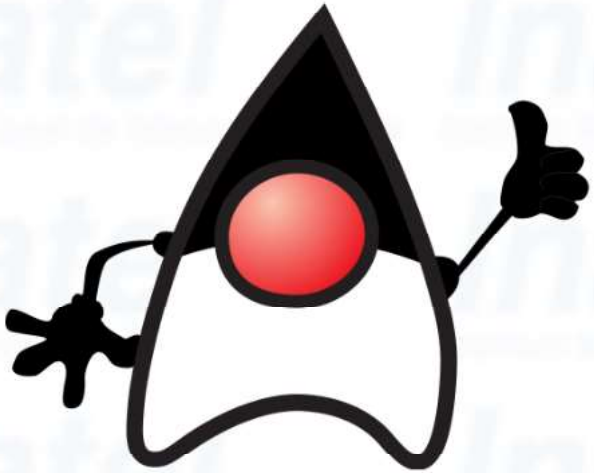
O ESCOPO DE UMA VARIÁVEL é o nome dado ao trecho de código que aquela variável existe e onde é possível acessá-la. Quando abrimos um novo bloco com as chaves, as variáveis declaradas ali dentro só valem até o fim daquele bloco.

Exemplos:

```
// Aqui o i é válido em outros blocos
// Já o j é valido apenas dentro do if
int i = 10;
if(algumBoolean)
{
    int j = 20;
}
```

```
// O y é válido apenas
// dentro do while
while(algumBoolean)
{
    float y = 20.5f;
}
```

FIM
DO
CAPÍTULO 2



Próximo Capítulo

Introdução a Orientação a Objetos