DANIEL ABELLA
www.daniel-abella.com

Daniel Abella

Profissional PMP, PMI-ACP e CSM, Gerente de Projetos no Virtus & UniFacisa e Professor do Curso de Sistemas de Informação da UniFacisa. Autor dos livros Gestão A3 (Ágil) e Scrum Arretado.

Conteúdo:

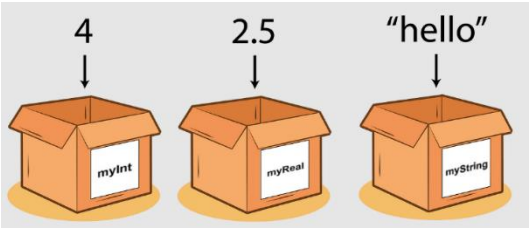
Capítulo 2

Variáveis e Controle de Fluxo

Livro Java do Zero (Uma Viagem ao Mundo Java)

01 Variáveis

- Variáveis são nomes que atribuímos para um "pedacinho" da memória do computador. Podemos pensar em uma caixa, que por sua vez, pode armazenar números, textos, arquivos, entre outros. Na imagem abaixo, temos a analogia de 3 variáveis.



- No código a seguir, temos exemplos dos principais tipos de variáveis em Java. Note que, diferente do Python, **precisamos definir o tipo da variável**.

```
1 class Teste {
2     public static void main(String[] args) {
3         int myInt = 4;
4         double myReal = 2.5;
5         String myString = "hello";
6         char myLetter = 'D';
7         boolean myBool = true;
8     }
9 }
```

02 Escopo de Variáveis

- **Existem basicamente 3 tipos de escopos de variáveis: instância, classe e local.**

Variáveis Locais

- As variáveis locais são declaradas dentro de métodos, construtores ou blocos e são apenas visíveis dentro dos locais aonde foram declaradas. Ou seja, se uma variável nomeCompleto for criada dentro de um método main, esta apenas estará "viva" durante a execução do método.
- Por fim, para uma variável local precisamos inicializar com um valor na hora de declarar ou inicialize antes de usar.

Sinal de Atribuição

Se você precisar iniciar a variável com um valor, após o nome da variável, precisamos colocar o sinal de atribuição =, que é o mesmo do Python.

Valor Inicial

Após o sinal de atribuição, temos o valor inicial da variável. Se a variável for do tipo int, valores possíveis podem ser 1 e não 1.0 (apenas se fosse double).

```
int var = 10;
```

Exemplos de Declarações de Variáveis Locais

```
int idade = 18;
double salario = 1000.50;
possuiNomeSerasa = true;
String nomeCompleto = "Daniel Abella";
```

Variáveis de Instância

- Enquanto as variáveis locais estão dentro de métodos/construtores/blocos, as **variáveis de instância estão fora dos métodos, mas dentro da classe**.

Modificadores de Acesso

São palavras-chave que determinam a visibilidade e acessibilidade de classes, métodos e variáveis em um programa, sendo elas: public (var1), protected (var2), private (var4) e o padrão, quando não especificamos nenhum dos 3, também conhecido por default ou friendly-package, que é o caso da var3.

```
public int var1 = 10;
protected int var2 = 10;
int var3 = 10;
private int var4 = 10;
```

- Nas variáveis locais, sempre tínhamos a necessidade de inicializar a variável antes de usá-la. Em variáveis de classe, caso não seja inicializada, recebe o valor padrão. Para inteiros, valor padrão é 0, números reais é 0.0, boolean é false e objetos (como String) é null.

Variáveis de Classe (Static)

- Variáveis de classe (também chamadas como estáticas ou *static*) pertencem à classe em vez de pertencerem a cada instância;
- Usando **static**, a variável agora não é mais da instância, mas da classe;
- Podemos pensar neste tipo de variável como variáveis globais da classe, que podem ser acessadas diretamente pelo nome da classe, sem a necessidade de criar uma instância/objeto.

Modificadores de Acesso

São palavras-chave que determinam a visibilidade e acessibilidade de classes, métodos e variáveis em um programa, sendo elas: public (var1), protected (var2), private (var4) e o padrão, quando não especificamos nenhum dos 3, também conhecido por default ou friendly-package, que é o caso da var3.

```
public static int var1 = 10;
protected static int var2 = 10;
static int var3 = 10;
private static int var4 = 10;
```

Inferência de Tipos c/ Var

Java10

```
1  class Teste {
2      public static void main(String[] args) {
3          var nome = "Daniel Abella";
4          System.out.println(nome.getClass());
5      }
6  }
```

Output

```
java -cp /tmp/3ieNQKYFQe HelloWorld
class java.lang.String
```

03 Casting

- A conversão de tipo é quando você atribui um valor de um tipo de dados primitivo a outro tipo, excetuando-se o boolean.
- No código a seguir, a instrução destacada na seta 1, converte a variável pi (destacada na seta 2) para int. Ou seja, (int) é o cast que converte algo para inteiro, fazendo com que a variável meuInt receba 3 (parte inteira do 3.14).

```
1  class Teste {
2      public static void main(String[] args) {
3
4          double pi = 3.14;
5          int meuInt = (int) pi;
6          System.out.println(meuInt);
7      }
8  }
```

- No exemplo a seguir, note que, como a variável agora é short (e não int como antes), o cast será (short). Este tipo de conversão chamamos de Narrowing Casting, pois converte um tipo maior em um tipo menor.

```
1  class Teste {
2      public static void main(String[] args) {
3
4          double pi = 3.14;
5          short meuInt = (short) pi;
6          System.out.println(meuInt);
7      }
```

- Ainda existe um outro tipo de conversão, chamada de Widening Casting, que converte de forma automática, sem usar cast, um tipo menor em um tipo maior, conforme exemplo abaixo.

```
1  class Teste {
2      public static void main(String[] args) {
3
4          //exemplo 1
5          int myAgeInt = 37;
6          double myAgeDouble = myAgeInt;
7
8          //exemplo 2
9          short myShort = 100;
10         int myInt = myShort;
11     }
12 }
```

04 Lendo do Teclado c/ Scanner

- Em Python, para ler algo do teclado era imensamente fácil. Em Java, precisamos usar a classe Scanner. Para isso, precisamos importar esta classe, como fizemos na linha 1 do código abaixo.

```
1  import java.util.Scanner;
2
3  class Teste {
4      public static void main(String[] args) {
5
6          Scanner scanner = new Scanner(System.in);
7
8          System.out.print("Digite o seu nome: ");
9          String nome = scanner.next();
10         System.out.println("O nome lido foi " + nome);
11
12         System.out.println();
13         System.out.print("Digite a sua idade: ");
14         int minhaIdade = scanner.nextInt();
15         System.out.println("A idade lida foi " + minhaIdade);
16     }
17 }
```

MÉTODO	DESCRIÇÃO
nextInt()	Lê um inteiro do usuário
nextFloat()	Lê um float do usuário
nextBoolean()	Lê um boolean do usuário
nextLine()	Lê uma linha de texto do usuário
next()	Lê uma palavra do usuário
nextDouble()	Lê um double do usuário

05 Operadores Básicos

- No código a seguir, entre as linhas 7 e 11, apresentamos o uso dos operadores básicos da linguagem Java, sendo eles: soma (linha 7), subtração (linha 8), divisão (linha 9), multiplicação (linha 10) e resto da divisão (linha 11).

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

        int operando1 = 8;
        int operando2 = 12;

        var soma = operando1 + operando2;
        var subtracao = operando1 - operando2;
        var divisao = operando1 / operando2;
        var multiplicacao = operando1 * operando2;
        var modulo = operando1 % operando2;

        System.out.println("A soma deu " + soma + " reais");
    }
}
```

- Importante observar que, o operador +, além da soma, é usado para concatenar elementos. Ainda temos operadores importantes como o pré e pós incremento (++), bem como o pré e pós decremento (--).
- Se você não quer aumentar/diminuir em 1, mas em outro valor, temos o operadores +=, -=, *=, /= e %=.

06 Operador Condicional com IF

- O funcionamento do IF é bem parecido como nas outras linguagens. Em Java, usamos chaves para delimitar o alcance dos blocos IF e a condição está entre parênteses, conforme exemplo a seguir.

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

        int idade = 18;

        if (idade >= 18) {
            System.out.println("Maior de idade");
        } else {
            System.out.println("Menor de idade");
        }

        System.out.println("Codigo executado independente do IF/Else");
    }
}
```

Se a condição verdadeira

Se a condição falsa

- E, temos além do IF e Else, o Else If. Em resumo, <u>temos obrigatoriamente um IF, 0 ou mais Else If e, opcionalmente, ter o else ao fim.</u>

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

        int idade = 18;

        if (idade == 18) {
            System.out.println("Tem exatos 18 anos");
        } else if(idade < 18) {
            System.out.println("Menor de idade");
        } else {
            System.out.println("Mais de 18");
        }

        System.out.println("Codigo executado independente do IF/Else");
    }
}
```

- E, os <u>operadores And e Or no Java, são respectivamente && e ||.</u>

```java
if (idade >= 18 && salario >= 10000) {
```

```java
if (idade >= 18 || salario >= 10000) {
```

07 Repetição com While e Do-While

- While (cuja tradução é enquanto) é usado para criar loops, fazendo com que bloco de código seja repetido até que uma dada condição seja falsa.

LOOP WHILE

```java
while(condicao) {
    //linhas de código
}
```

- Exemplo apresentado a seguir.

```java
class TesteWhile {
    public static void main(String[] args) {
        int i = 1;

        while(i <= 4) {
            System.out.println(i);
            i++;
        }
    }
}
```

- Ao contrário do while convencional, onde a condição é verificada antes da primeira execução do bloco de código, o do-while executa o bloco de código pelo menos uma vez e, em seguida, verifica a condição. Se a condição for verdadeira, o bloco de código será executado novamente. Se a condição for falsa, a execução do loop será interrompida e o programa continuará a partir do próximo trecho de código após o do-while.

LOOP DO-WHILE

```java
do {
    //linhas de código
} while(condicao);
```

08 Repetição com For e ForEach

- O comando For tradicional permite criar loops para repetir um bloco de código um determinado número de vezes.

LOOP FOR

```java
for(inicialização;condição;iteração) {
    //linhas de código
}
```

LOOP ENHANCED FOR

```
for(tipo var; elemento) {
    //linhas de código
}
```

- O Enhanced For, também conhecido como For Each, possui um funcionamento mais fácil, porém não indexado.

Atenção ao Break e Continue

Em Java, temos o break e continue como palavras-chave úteis para controle o fluxo do programa, mais especificamente em estruturas de repetição. O break, com exemplo abaixo, é utilizado para interromper completamente a execução do loop.

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break; // O loop é interrompido quando i é igual a 5
    }
    System.out.println(i);
}
```

Por outro lado, continue é usado para pular para a próxima iteração sem concluir o restante do código dentro da iteração atual, conforme exemplo a seguir.

```
for (int i = 0; i < 10; i++) {
    if (i % 2 == 0) {
        continue; // Pula para a próxima iteração se i for par
    }
    System.out.println(i);
}
```

09 Comentários em Java

- Comentários são textos que são ignorados pelo compilador, sendo usado geralmente para documentar o código. Em Java, temos 3 tipos de comentários: de linha, bloco (múltiplas linhas) e JavaDOC.

```
class Teste {
    /*
     * Este é um
     * comentário de bloco
     */
    public static void main(String[] args) {
        //Comentário de linha 1
        //System.out.println("Abella")
        System.out.println("Olá, mundo!")
    }
}
```

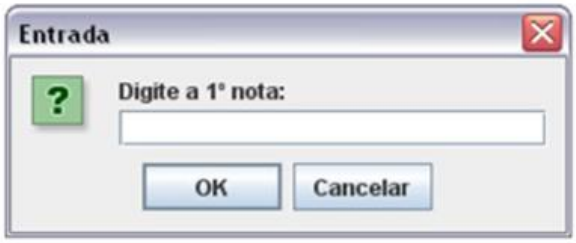
- O JavaDOC segue a mesma sintaxe dos comentários de várias linhas, mas começam com dois asteriscos (/**) ao invés de um (/*). A seguir, temos um exemplo.

```
/**
 * Esta classe representa uma Classe de exemplo.
 * Ela possui métodos para executar ações específicas.
 */
class Teste {

    /**
     * Nosso método main
     * @param args Argumentos de linha de comando
     */
    public static void main(String[] args) {
        //Comentário de linha 1
        //System.out.println("Abella")
        System.out.println("Olá, mundo!")
    }
}
```

10 JOptionPane

- De maneira simples, podemos criar algumas telinhas até bonitinhas para imprimir algo na tela, bem como ler do teclado. Vejam o exemplo abaixo, onde a primeira telinha é o "nosso print", enquanto que a segunda é o "nosso scanner".



- Vamos ao nosso primeiro exemplo? Abaixo, na primeira linha, tivemos que importar a classe JOptionPane. Para ler do teclado, usamos o método showInputDialog da classe JOptionPane. É importante observar que, o método showInputDialog retorna uma String.

```
import javax.swing.JOptionPane;

class HelloWorld {
    public static void main(String[] args) {
        String nome = JOptionPane.showInputDialog("Digite o seu nome");
        JOptionPane.showMessageDialog(null,"O seu nome eh " + nome);
    }
}
```