Java – Variáveis e Controle Fluxo

Variáveis primitivas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Todo **bloco** em java é delimitado por **chaves** ({ });
- **Dentro de um bloco,** podemos declarar variáveis e usá-las;
- **Em Java, toda variável tem um tipo que não pode ser mudado, uma vez que declarado;**

```
tipoDaVariavel nomeDaVariavel;
```

- Por exemplo, é possível ter uma idade que guarda um **número inteiro**:

```
int idade;
```

- *Você pode **atribuir** um **valor** a **variável**:*

```
idade = 15;
```

Variáveis primitivas

- Além de **atribuir**, você pode **utilizar** esse valor:

```
// declara a idade
int idade;
idade = 15;

// imprime a idade
System.out.println(idade);
```

- Podemos usar uma **variável** para **alterar** ou **definir** uma segunda variável:

```
// calcula a idade no ano seguinte
int idadeNoAnoQueVem;
idadeNoAnoQueVem = idade + 1;
```

Variáveis primitivas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- No mesmo momento que você **declara** uma **variável**, também é possível **inicializá-la** por **praticidade**:

```
int idade = 15;
```

- Você pode usar os operadores **+, -, / e *** para **operar** com **números**, sendo eles responsáveis pela **adição**, **subtração**, **divisão** e **multiplicação**, respectivamente;

- Além desses **operadores básicos**, há o operador **%** (módulo) que nada mais é que o **resto de uma divisão inteira .**

Operadores Aritméticos

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- **Exemplos:**

```
int quatro = 2 + 2;
int tres = 5 - 2;

int oito = 4 * 2;
int dezesseis = 64 / 4;

int um = 5 % 2; // 5 dividido por 2 dá 2 e tem resto 1;
                // o operador % pega o resto da divisão inteira
```

Programa Simples

```
class TestaIdade {


    public static void main(String[] args) {
        // imprime a idade
        int idade = 20;
        System.out.println(idade);

        // gera uma idade no ano seguinte
        int idadeNoAnoQueVem;
        idadeNoAnoQueVem = idade + 1;

        // imprime a idade
        System.out.println(idadeNoAnoQueVem);
    }
}
```

Tipos de Dados

- Para armazenar um número com **ponto flutuante** (e que também pode armazenar um número inteiro) usamos o **double**:

```
double pi = 3.14;
double x = 5 * 10;
```

- O tipo **boolean** armazena um valor **verdadeiro** ou **falso**, e só: nada de **números**, **palavras** ou **endereços**, como em algumas outras linguagens.

```
boolean verdade = true;
```

Tipos de Dados

- **true** e **false** são palavras reservadas do Java.
- É comum que um boolean seja determinado através de uma **expressão booleana**, isto é, um trecho de código que retorna um booleano, como o exemplo:

```
int idade = 30;
boolean menorDeIdade = idade < 18;
```

- O tipo **char** guarda um, e apenas um, caractere. Esse caractere deve estar entre **aspas simples**;

```
char letra = 'a';
System.out.println(letra);
```

Tipos de Dados

- É comum que um **boolean** seja determinado através de uma **expressão booleana**, isto é, um trecho de código que retorna um booleano, como o exemplo:

```
int idade = 30;
boolean menorDeIdade = idade < 18;
```

- O tipo **char** guarda um, e apenas um, **caractere**. Esse caractere deve estar entre **aspas simples**;

```
char letra = 'a';
System.out.println(letra);
```

- Variáveis do tipo **char** são **pouco usadas** no dia a dia. Veremos mais a frente o uso das **Strings**, que usamos **constantemente**, porém estas não são **definidas por um tipo primitivo**.

Tipos Primitivos e Valores

- **Tipo Primitivos do Java e seus tamanhos;**

TIPO	*TAMANHO*
boolean	1 bit
byte	1 byte
short	2 bytes
char	2 bytes
int	4 bytes
float	4 bytes
long	8 bytes
double	8 bytes

Tipos Primitivos e Valores

- Nos tipos primitivos o valor que elas guardam são o **real conteúdo** da **variável**;

- Quando você utilizar o **operador de atribuição =** o valor será **copiado.**

```
int i = 5; // i recebe uma cópia do valor 5
int j = i; // j recebe uma cópia do valor de i
i = i + 1; // i vira 6, j continua 5
```

- Apesar da linha 2 fazer j = i, a partir desse momento essas **variáveis não tem relação** nenhuma: o que **acontece** com **uma**, não **reflete** em **nada** com a **outra**.

Casting e Promoções

- Alguns valores são **incompatíveis** se você **tentar** fazer uma **atribuição direta**;

- Enquanto um **número real** costuma ser representado em uma **variável** do tipo **double**, tentar atribuir ele a uma variável **int não funciona**;

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

O mesmo ocorre no seguinte trecho:

```
int i = 3.14;
```

Apesar de 5 ser um bom valor para um int, o compilador não tem como saber que valor estará dentro desse double no momento da execução

O mais interessante, é que nem mesmo o seguinte código compila:

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

Casting e Promoções

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- **O que aconteçe neste caso:**

```
int i = 5;
double d2 = i;
```

- O código **acima compila** sem **problemas**, já que um **double** pode **guardar** um **número** com ou sem **ponto flutuante**;

- Todos os **inteiros representados** por uma **variável** do tipo **int** podem ser guardados em uma **variável double**;

Casting e Promoções

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Às vezes, precisamos que um **número quebrado** seja **arredondado** e **armazenado** num número **inteiro**.

- Para fazer isso sem que **haja** o **erro** de **compilação**, é preciso ordenar que o número quebrado seja **moldado (casted)** como um **número inteiro**

```
double d3 = 3.14;
int i = (int) d3;
```

- O **casting** foi feito para **moldar** a variável d3 como um int. O valor de i agora é 3

Casting e Promoções

O mesmo ocorre entre valores `int` e `long`.

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

E, se quisermos realmente fazer isso, fazemos o casting:

```
long x = 10000;
int i = (int) x;
```

Castings Possíveis

- Tabela com os casts possíveis na linguagem 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)	----

- A indicação **Impl.** quer dizer que aquele **cast** é **implícito** e **automático**, ou seja, você não **precisa indicar** o **cast** explicitamente.
- Lembrando que o tipo **boolean** não **pode** ser **convertido** para **nenhum** outro tipo;

Exercício

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

1) Na empresa onde trabalhamos, há tabelas com o quanto foi gasto em cada mês. Para fechar o balanço do primeiro trimestre, precisamos somar o gasto total. Sabendo que, em Janeiro, foram gastos 15000 reais, em Fevereiro, 23000 reais e em Março, 17000 reais, faça um programa que calcule e imprima o gasto total no trimestre. Siga esses passos:

   a) Crie uma classe chamada `BalancoTrimestral` com um bloco main, como nos exemplos anteriores;

   b) Dentro do `main` (o miolo do programa), declare uma variável inteira chamada `gastosJaneiro` e inicialize-a com 15000;

   c) Crie também as variáveis `gastosFevereiro` e `gastosMarco`, inicializando-as com 23000 e 17000, respectivamente, utilize uma linha para cada declaração;

   d) Crie uma variável chamada `gastosTrimestre` e inicialize-a com a soma das outras 3 variáveis:

```
int gastosTrimestre = gastosJaneiro + gastosFevereiro + gastosMarco;
```

   e) Imprima a variável `gastosTrimestre`.

2) Adicione código (sem alterar as linhas que já existem) na classe anterior para imprimir a média mensal de gasto, criando uma variável `mediaMensal` junto com uma mensagem. Para isso, concatene a `String` com o valor, usando `"Valor da média mensal = "+ mediaMensal`.