Programação Orientada a Objetos em Java

Tipos de Dados e Instância

Prof. Henrique Louro
henrique.louro@cps.sp.gov.br

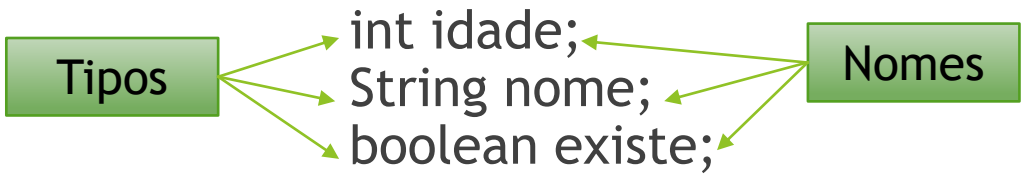
Cetec
Unidade de Ensino Médio e Técnico

Centro Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO

Declarando variáveis

- As declarações de variáveis consistem de um tipo e um nome de variável, como no exemplo a seguir:



- Os nomes de variáveis podem começar com uma letra, um sublinhado (_), ou um cifrão ($). Elas **não** podem começar com um número. Depois do primeiro caractere pode-se colocar qualquer letra ou número.

Atribuição

```
idade = 18;
nome = “Fulano”;
existe = true;
```

Tipos primitivos de dados

Tipo	Ocupa (Bytes)	Faixa de valores
byte	1	-128 a 127
short	2	-32.768 a 32.767
int	4	-2.147.483.648 a 2.147.483.647
long	8	-9x10e18 a 9x10e18
float	4	$-3x10^{38}$ a $3x10^{38}$ (com 6 dígitos significativos)
double	8	$-1,7x10^{308}$ a $1,7x10^{308}$ (com 15 dígitos significativos)
char	2	Utiliza padrão unicode, tem como subconjunto o ASCII
boolean	1	True ou False
String	-	Coleção de caracteres. Não é considerado um tipo primitivo e sim por referência

Conceitos de POO

- **Classe** – Um modelo para um objeto, que contém variáveis (atributos), para descrever o objeto e métodos para descrever como o objeto se comporta. As classes podem herdar atributos e métodos de outras classes.
- **Objeto** – Uma instância de uma classe. Vários objetos que são instância da mesma classe têm acesso aos mesmos métodos, mas normalmente possuem valores diferentes para seus atributos de instância.
- **Instância** – O mesmo que um objeto. Cada objeto é uma nova ocorrência ou instância de alguma classe.
- **Método** – Um grupo de instruções em uma classe, que define como seus objetos se comportarão. Os métodos precisam estar sempre localizados dentro de uma classe.

Conceitos de POO

- **Variável de classe** – Uma variável que descreve um atributo de uma classe, em vez das instâncias específicas da classe.
- **Variável de instância** – Uma variável que descreve um atributo de uma instância de uma classe, em vez da própria classe.
- **Interface** – Uma especificação de comportamento abstrato, que as classes individuais podem então implementar.
- **Pacote** – Uma coleção de classes e interfaces.
- **Subclasse** – Uma classe mais abaixo na hierarquia de classes do que do que outra classe, sua superclasse. Uma classe pode ter tantas subclasses quantas forem necessárias.
- **Superclasse** – Uma classe mais acima na hierarquia de classes do que outra classe, sua subclasse. Uma subclasse só pode ter uma superclasse imediatamente acima dela.

Instâncias

```
Conta conta1 = new Conta();
```

Classe

Nome do Objeto

Operador

Método Construtor

Tipos de Referência

- Os tipos de referência facilitam a manipulação dos objetos em memória, agilizando a construção do código, a gerência de memória e a navegação entre os objetos do sistema.

- Uma variável de referência permite o acesso a instâncias de objetos em memória, pois a ela é atribuído o endereço de memória de um objeto. Essa associação é que recebe o nome de referência.

Tipos de Referência


```
public class TesteConta {
      public static void main(String [] args) {
            Conta conta1 = new Conta();
            conta1.numero = 10;
            conta1.saldo = 500;
            System.out.println(conta1);
            System.out.println("Numero: "+conta1.numero);
            System.out.println("Saldo: "+conta1.saldo);
            Conta conta2 = new Conta();
            conta2.numero = 11;
            conta2.saldo = 5330;
            System.out.println(conta2);
            System.out.println("Saldo: "+conta2.numero);
            System.out.println("Saldo: "+conta2.numero);
      }
}
```

Nesse exemplo é possível perceber que os objetos **conta1** e **conta2** são do tipo referência, por duas razões:

1- Não são de tipo primitivo;

2- Estão associados ao operador *new*.

Método Construtor

- É o método que comanda como os atributos de uma classe serão iniciados;
- No exemplo da classe Carro, no próximo slide, há apenas um método construtor. Ele pode ser reconhecido por utilizar o mesmo nome da classe e por não ter nenhum tipo de retorno;
- Pode-se definir quantos métodos construtores forem necessários. A isso dá-se o nome de **sobrecarga de métodos**;
- Cada construtor permite que sejam atribuídos valores específicos para os argumentos da classe, utilizando tipos primitivos e de referência;
- O compilador Java diferencia os construtores baseado na quantidade e nos tipos dos argumentos passados (**assinatura do método**).

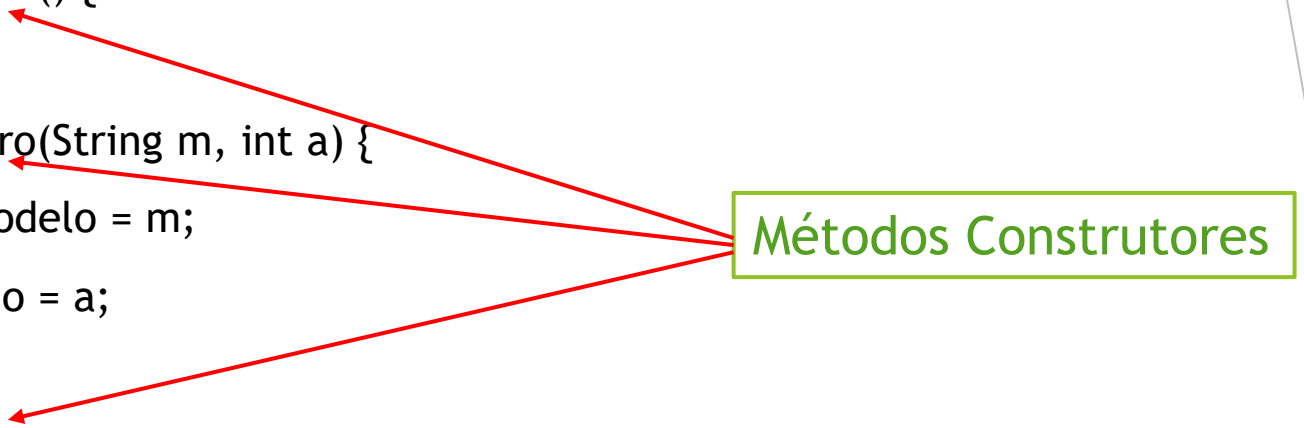
Método Construtor: exemplo

```
class Carro {
    private String modelo;
    private int ano;
    Carro(String m, int a) {
     modelo = m;
     ano = a;
    }
    String mostra()   {
      return  modelo + ano;
    }
}
```

Método Construtor

Métodos Construtores: exemplo


```
class Carro {
    private String modelo;
    private int ano;
    Carro() {
    }
    Carro(String m, int a) {
      modelo = m;
      ano = a;
    }
    Carro(String m) {
      modelo = m;
    }
    String mostra()  {
       return  modelo + ano;
    }
}
```



Métodos Construtores: exemplo


```
class TestaCarro {
    public static void main(String args[]){
      Carro carro1 = new Carro();
      System.out.println(carro1.mostra());
      Carro carro2 = new Carro("xyz",2009);
      System.out.println(carro2.mostra());
      Carro carro3 = new Carro("kwm");
      System.out.println(carro3.mostra());
    }
}
```

Nesse exemplo é possível perceber que os objetos **carro1**, **carro2** e **carro3** são instanciados utilizando-se cada um dos métodos construtores da classe **Carro**, vista no slide anterior.

Dúvidas?

Poste suas dúvidas preferencialmente no **Fórum de Dúvidas/Sugestões** da capacitação, na plataforma AVA.

Obrigado!

Prof. Henrique Louro
henrique.louro@cps.sp.gov.br