JAVA

Linguagem de Programação Orientada à Objeto - POO

Prof. Celso Furtado

HISTÓRIA

- A linguagem Java teve seu lançamento oficial em **1995** pela **Sun Microsystems, Inc**.

HISTÓRIA

- **James Gosling** é considerado o criador da linguagem Java.

HISTÓRIA

- Mas, **James Gosling** não estava sozinho. Participaram com ele Patrick Naughton, Chris Warth, Ed Frank, Mike Sheridan e muitos outros. Eles eram conhecidos como o “**Green Team**”.

HISTÓRIA

- Em **2009** a **Oracle** adquiriu a **Sun Microsystems** pelo valor de **US$ 9 bi. Uau!**;

HISTÓRIA

- O mascote do Java é o **Duke**!

HISTÓRIA

- A linguagem Java começou a ser desenvolvida em 1991, e no início seu nome era “**OAK**”, que significa carvalho. Somente em 1995 ela passou a se chamar Java;
- A criação da linguagem Java não foi motivada pela Internet, e sim da necessidade de se ter uma **linguagem independente de plataforma**;
- O Java estava sendo criado para ser usado em vários dispositivos domésticos, tais como torradeiras, fornos de micro-ondas, controle remoto, etc.

HISTÓRIA

- Durante o desenvolvimento do Java, surgia a Internet e esta teve uma grande influência no desenvolvimento da linguagem;
- O Java possibilitou que as páginas web pudessem interagir de uma forma mais dinâmica com o usuário.

Applets Java

- A Internet ajudou a impulsionar o Java para a dianteira da programação e, por sua vez, Java teve um efeito profundo na Internet;
- O Java inovou com um tipo de programa de rede chamado ***applet***, que mudou a forma do mundo online pensar em conteúdo.

Applets Java

- O ***applet*** é um tipo especial de programa Java que é transmitido pela Internet e executado em um navegador web compatível;
- Desta forma é possível executar muitas tarefas no cliente ao invés de executá-las no servidor.

Java Virtual Machine (JVM)

- Uma das principais vantagens do Java em relação as outras linguagens é o que chamamos de “**Portabilidade**”;
- Ou seja, você pode desenvolver um aplicativo Java na plataforma **Windows** e executá-lo em **GNU/Linux**, por exemplo.

Java Virtual Machine (JVM)

- O segredo para esta portabilidade da linguagem Java é o **bytecode**;
- Quando um programa Java é compilado a saída não é um código executável. Em vez disso, o **bytecode**.

Java Virtual Machine (JVM)

- O bytecode é um conjunto de instruções altamente otimizado para ser executado pelo sistema de tempo de execução Java, que se chama **Java Virtual Machine (JVM).**

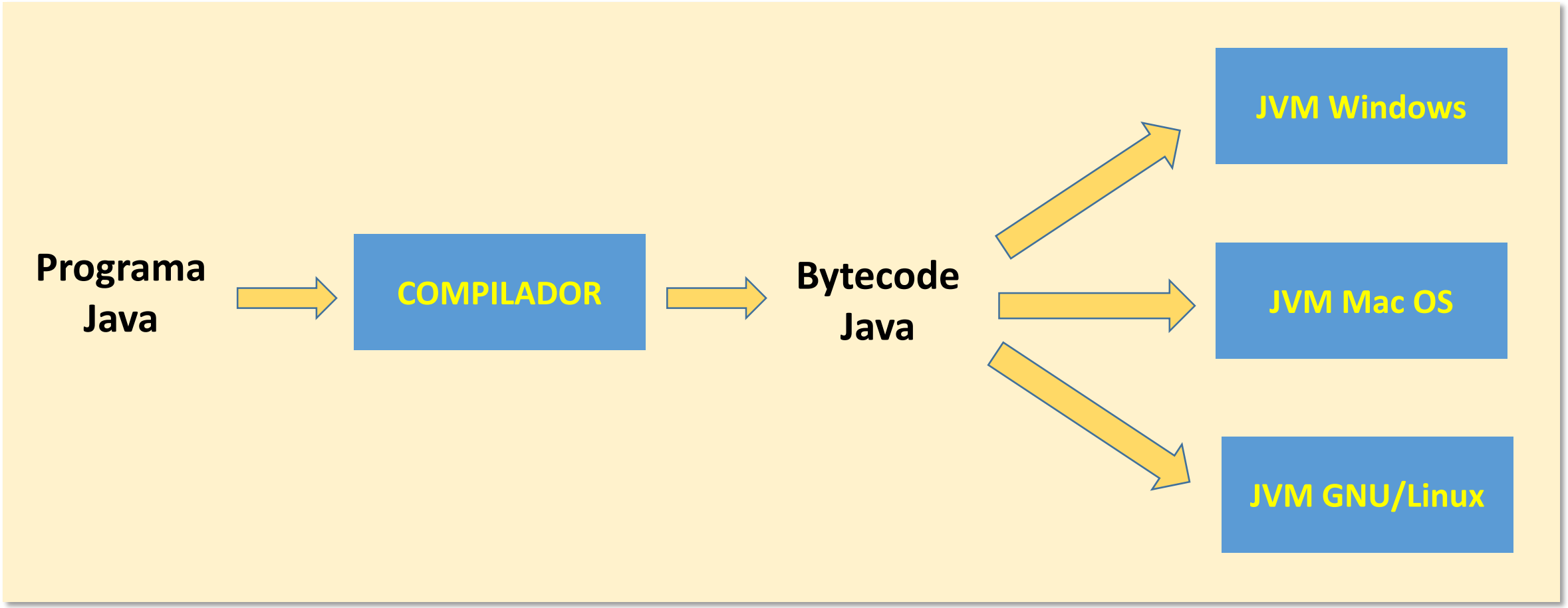
Java Virtual Machine (JVM)

- A maioria das linguagens de programação compila o código e gera um executável para um sistema operacional determinado;
- Este executável roda somente no sistema operacional para o qual foi compilado;
- Desta forma é necessário haver várias versões do programa, uma para cada plataforma que irá rodar.

Java Virtual Machine (JVM)

- No caso do Java, é necessário apenas uma versão da JVM para cada plataforma, que irá rodar o bytecode;
- Assim, você pode gerar o programa no Windows, gerar os bytecodes e rodar em qualquer JVM em qualquer plataforma.

Java Virtual Machine (JVM)



Programação Orientada a Objetos

- A **OOP** (Object-oriented Programming) ou **POO** (Programação Orientada a Objetos) é a essência de Java;
- A metodologia OOP é inseparável da linguagem, e todos os programas Java são, pelo menos até certo ponto, orientados a objetos.

Programação Orientada a Objetos

- Em POO temos três características como princípios básicos, que são:

  - **Encapsulamento;**
  - **Polimorfismo;**
  - **Herança.**

Programação Orientada a Objetos

- **Encapsulamento**:

  - É um mecanismo que vincula o código e os dados que ele trata. Dizemos que o conteúdo de uma classe fica oculto às outras classes.

Programação Orientada a Objetos

- **Polimorfismo**:

  - Do grego, “muitas formas”. É a capacidade de uma classe assumir diversas formas diferentes através dos seus métodos.

Programação Orientada a Objetos

- **Herança**:

  - É o processo pelo qual um objeto pode adquirir as propriedades de outro objeto.

Obtendo o Java Development Kit - JDK

- Agora que a teoria básica sobre Java já foi explicada é hora de começar a escrever programas em Java;
- Para isso é necessário que tenhamos o **Java Development Kit**, ou seja, o kit de desenvolvimento Java.

Obtendo o Java Development Kit - JDK

- O **JDK** é um conjunto de ferramentas que permitirão que você possa compilar e executar os seus programas em Java;
- Atualmente o Java se encontra em sua versão **8**, que é conhecida como **Java SE 8** (Java Standard Edition)

Obtendo o Java Development Kit - JDK

- O **JDK** pode ser baixado de:

http://www.oracle.com/technetwork/pt/java/javase/downloads/jdk8-downloads-2133151.html

- Simplesmente acesse a página e siga as instruções para o tipo de computador que você tem.

Obtendo o Java Development Kit - JDK

- Após instalar o JDK, você terá dois programas principais:
- **javac** – que é o compilador Java. Com ele você irá gerar o bytecode do seu programa;
- **java** – que é o interpretador padrão do Java. Ele será utilizado para iniciar o aplicativo que você criou.

Obtendo o Java Development Kit - JDK

- O JDK é executado no ambiente de prompt de comando e usa ferramentas de linha de comando. Ele não é um aplicativo de janelas;
- O JDK também não é um ambiente de desenvolvimento integrado (IDE).

Obtendo o Java Development Kit - JDK

- Mas existem várias IDEs que permitirão uma maior produtividade quando programamos em Java;
- As IDEs mais utilizadas para programação Java são o Eclipse e o NetBeans.
- Nós utilizaremos o Eclipse ☺!

Obtendo o Java Development Kit - JDK

www.eclipse.org

www.netbeans.org

Programando em Java

- A programação Java ocorrem em três etapas:

  - Inserir o programa;
  - Compilar o programa;
  - Executar o programa.

Programando em Java

- Abra o Bloco de Notas e digite o programa abaixo:

```
/*
    Primeiro programa Java simples
    Escrito por Celso Furtado
*/

class Exemplo {
    // Um programa Java começa com um método main ()
    public static void main(String args[]) {
        System.out.print(“Primeiro programa Java!”);
    }
}
```

Programando em Java

- O Java é case-sensitive, ou seja, letras maiúsculas e minúsculas são diferentes;
- Instruções digitadas com o tipo de caixa errado irão gerar erro no momento da compilação.

Programando em Java

- Salve o programa com o nome Exemplo.java. O nome do arquivo deve ser igual ao nome da classe;
- Abra o prompt de comando e acesse o diretório onde você gravou o seu arquivo.

Programando em Java

- Digite o comando abaixo:

```
> javac Exemplo.java
```

- Se tudo estiver correto o comando irá gerar um novo arquivo chamado **Exemplo.class.**

Programando em Java

- Para executar o programa digite:

```
> java Exemplo
```

- Note que não utilizamos a extensão .class para executarmos o programa.

Programando em Java

- Altere o seu código fonte acrescentando a linha em negrito:

```
/*
    Primeiro programa Java simples
    Escrito por Celso Furtado
*/

class Exemplo {
    // Um programa Java começa com um método main ()
    public static void main(String args[]) {
      System.out.print(“Primeiro programa Java!”);
      System.out.print(“SENAI Prof. Vicente Amato”);
    }
}
```

Programando em Java

- Compile o programa para atualizar o bytecode (.class) e execute o programa novamente;
- O Resultado esperado é o seguinte:

```
Primeiro programa Java!
SENAI Prof. Vicente Amato
```

- Seu programa apresentou a saída como esperado? O que aconteceu?
- Como você poderia resolver o problema de modo a conseguir o resultado esperado?

Programando em Java

Tipos de dados em Java

- Em Java, há dois tipos de dados, os **orientados a objetos** e os **não orientados a objetos**;
- Os orientados a objetos são definidos por classes, mas veremos isso depois ;)
- Porém, temos 8 tipos de dados na base Java, que são conhecidos como tipos primitivos.

Programando em Java

- ## Tipos de dados primitivos

Tipo	Significado
boolean	Representa valores verdadeiro/falso
byte	Inteiro de 8 bits
char	Caractere
double	Ponto flutuante de precisão dupla (Números com casas decimais. **Ex.:** 2.89)
float	Ponto flutuante de precisão simples (Números com casas decimais. **Ex.:** 2.89)
int	Números inteiros
long	Inteiro longo
short	Inteiro curto

Programando em Java

• Tipos de dados Inteiro

Tipo	Tamanho em bits	Intervalo
byte	8	-128 a 127
short	16	-32.768 a 32.767
int	32	-2.147.483.648 a 2.147.483.647
long	64	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807

Programando em Java

- **Tipos de dados ponto flutuante**

Tipo	Tamanho em bits	Intervalo
float	32	Precisão simples
double*	64	Precisão dupla

* Dos dois o double é o mais utilizado, pois todas as funções matemáticas da biblioteca Java utiliza este tipo de dados.

Programando em Java

Declarando uma variável

- A declaração de uma variável deve ser da seguinte forma

<tipo> <nome da variável>;

- Por exemplo para declararmos uma variável do tipo inteiro devemos fazer o seguinte:

```
int valor1;
```

Onde, int é o tipo e valor1 é o nome da variável.

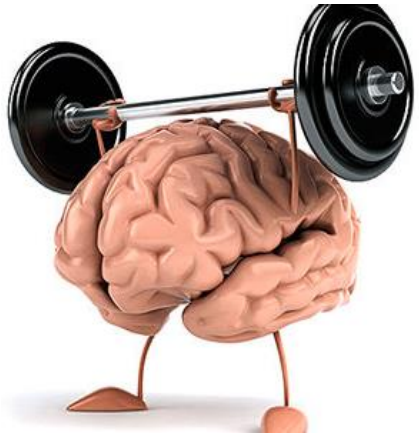
Programando em Java

- Digite o seguinte programa:

```
// Declarando variáveis no Java

class Idade {
    // Um programa Java começa com um método main ()
    public static void main(String args[]) {
      int idade = 18;
      System.out.print("A sua idade é: " + idade);
    }
}
```

- Compile e rode o programa observando o resultado.

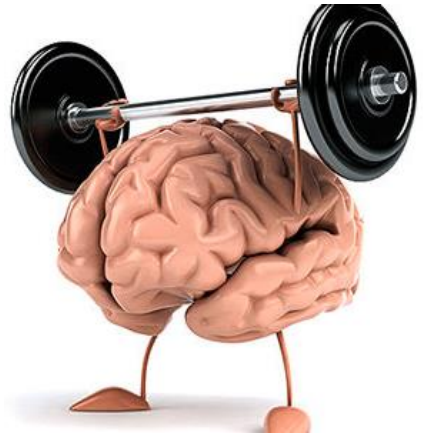


Programando em Java

DESAFIO 1

- Faça um programa, que dada uma idade qualquer, calcule a quantidade de dias que a pessoa já viveu. A saída do programa deve ser a seguinte:

Você tem 18 anos e já viveu 6570 dias!



Programando em Java

DESAFIO 2

- Faça um programa que, dados os gastos do primeiro trimestre do ano, calcule o total gasto no trimestre e a média de gastos mensais. Os dados são os seguintes:

  Janeiro: 13500

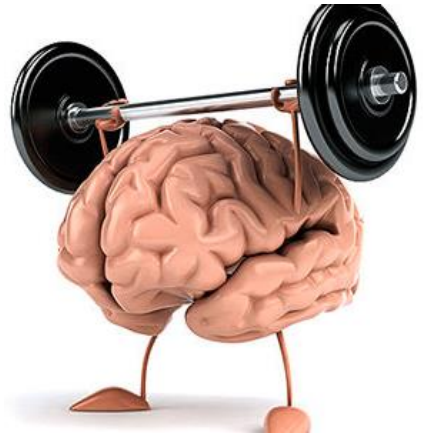
  Fevereiro: 12000

  Março: 14700

- Ao executar o programa a saída deve ser a seguinte:

O gasto total do trimestre foi de ----> 40200.

A média de gastos mensal foram de ----> 13400.



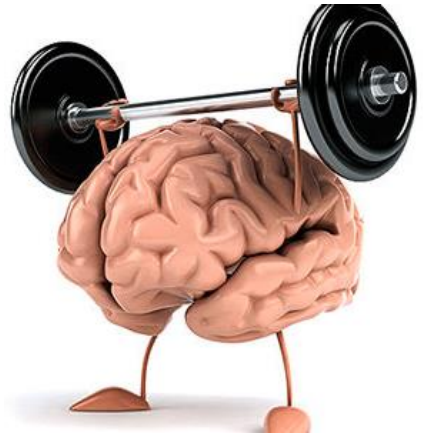
Programando em Java

DESAFIO 3

- Sabendo-se que um automóvel consome 1 litro de combustível a cada 7 Km percorridos, calcule a quantidade de combustível necessário para percorrer uma distância qualquer informada pelo programa.

- Ao executar o programa a saída deve ser a seguinte:

Para percorrer 70 Km serão necessários 10 litros de combustível.



Programando em Java

DESAFIO 4

- É possível fazer com que os programas desenvolvidos anteriormente solicitem as entradas de dados para o usuários durante a execução, tornando o seu aplicativo mais dinâmico? Se sim, como você poderia implementar esta funcionalidade? .

Escopo de variável

- Escopo de variável é a **região** onde a variável é **visível**;
- O escopo também define quando a variável será **criada** e **destruída** na memória;
- Assim, escopo se refere a **visibilidade** e **tempo de vida** de uma variável.

Escopo de variável

- Existem quatro categorias de escopo:

  - **Membro da classe;**
  - **Variável em nível de método;**
  - **Variável local, visível apenas dentro de um bloco;**
  - **Variável local passada como parâmetro.**

Escopo de variável

- O nível mais restrito é a variável local, que é declarada dentro de um bloco. Por exemplo:

```
if(i > 10){
    //Variável local
    int x = 100;
    System.out.println("O valor de x é = " + x);
}
```

Escopo de variável

- Se a variável for declarada em um nível maior, como nível de método, por exemplo, ela será visível para todos os blocos daquele método. Exemplo:

```
public static void main(String[] args) {
    //Variável de método
    int i = 20;

    if(i > 10){
        //Variável local
        int x = 100;

        System.out.println("O valor de x é = " + x);

        System.out.println("O valor de i é = " + i);
    }

}
```

Escopo de variável

- Finalmente, temos a variável de classe, que é visível por toda a classe. Exemplo:

```
public class Escopo {
    //Variável de classe
    static String exemplo = "Escopo de variáveis!";

    public static void main(String[] args) {

        //Variável de método
        int i = 20;
        if(i > 10){
            //Variável local
            int x = 100;

            System.out.println(exemplo);

            System.out.println("O valor de x é = " + x);

            System.out.println("O valor de i é = " + i);
        }
    }

}
```

Programando em Java

Variáveis do tipo “final”

- Quando declaramos uma variável e utilizamos a palavra chave “final” estamos dizendo que esta variável é na verdade uma constante, ou seja, seu valor não poderá ser alterado depois da primeira atribuição. Exemplo:

```
final double PI = 3.1415;
```

OBS.: De acordo com as boas práticas Java, as variáveis do tipo final devem ser grafadas em letras maiúsculas, e separadas por “ _ “ quando houverem mais

Interpolação de Strings

- A interpolação é um recurso que evita o uso de concatenação de Strings com o uso do sinal de “+”. Exemplo:

```
int valor1 = 10;
int valor2 = 30;
int soma;

soma = valor1 + valor2;

// Sem interpolação
System.out.println("A soma de " + valor1 + " + " + valor2 + " é: " + soma);

// Com interpolação
System.out.printf("A soma de %d + %d é: %d", valor1, valor2, soma);
```

Inte

%d	int com sinal	Decimal com sinal
%u	int sem sinal	Decimal sem sinal
%o	int sem sinal	Octal sem sinal
%x ou %X	int sem sinal	Hexadecimal sem sinal
%z[n] ou %Z[n]	int sem sinal	Inteiro na base **n** (n deve ser um valor decimal)
%f	float	Número real padrão
%.[p]f	float	Número real na precisão **p** (p deve ser um valor decimal)
%e ou %E	float	Número real em notação cientifica
%g ou %G	float	Número real em notação cientifica (utilizado para números com grandes precisões)
%s	String	Texto
%c	char	Caractere
%p	Object	Exibe o conteúdo do método toString do objeto (veremos isso em detalhes a frente).

Pós e Pré-incremento e decremento

- Os operadores reduzidos **++** (incrementa o operando de 1) e **--** (decrementa o operando de 1) podem aparecer antes (pré-incremento) ou depois (pós-incremento);
- Quando o operador é colocado antes do operando, **++op/--op** o valor é alterado antes da utilização do mesmo;
- Quando o operador é colocado depois do operando **op++/op--** o valor é alterado depois da utilização do mesmo.

Pós e Pré-incremento e decremento

- Exemplo:

```
int i = 0;
System.out.println("Valor do i = " + i);
System.out.println("Valor do i++ = " + i++);
System.out.println("Valor do i = " + i);
System.out.println("Valor do ++i = " + ++i);
System.out.println("Valor do i = " + i);
```

Trabalhando com Arrays

- Arrays são um conjunto de variáveis do mesmo tipo, referenciadas por um nome comum;
- Em Java um array pode ter uma ou mais dimensões. O array unidimensional é o mais popular;
- Os arrays em Java são tratados com objetos.

Trabalhando com Arrays

- **Arrays unidimensionais:**
  - O array unidimensional é uma lista de variáveis relacionadas. Essas listas são muito comuns em programação;
  - Você poderia, por exemplo, usar um array para armazenar uma lista de usuários ativos em uma rede de computadores.

Trabalhando com Arrays

- **Arrays unidimensionais:**
  - A declaração de um array unidimensional é feito da seguinte forma:

```
tipo nome-array[] = new tipo[tamanho];
```

OBS.: O tipo do elemento array também pode ser chamado de "**tipo base**".

Trabalhando com Arrays

- **Arrays unidimensionais:**
  - Exemplo prático:

```
double temperatura[] = new double[4];
```

Com o comando acima criamos um array do tipo double com **4** posições.

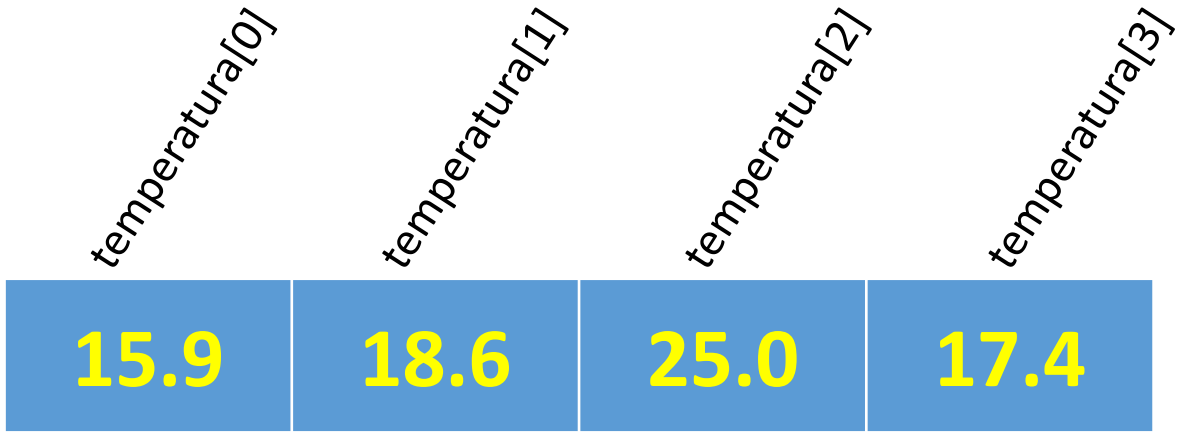
Trabalhando com Arrays

- **Preenchimento de arrays unidimensionais:**
  - O preenchimento de arrays é feito da seguinte forma:

```
temperatura[0] = 15.9;
temperatura[1] = 18.6;
temperatura[2] = 25.0;
temperatura[3] = 17.4;
```

Trabalhando com Arrays

- **Preenchimento de arrays unidimensionais:**
  - O array anterior pode ser representado da seguinte forma:



Trabalhando com Arrays

- **Arrays multidimensionais:**
  - O array multidimensional é um array composto por arrays;
  - A forma mais simples de array multidimensional é o array bidimensional, que é um array de arrays unidimensionais.

Trabalhando com Arrays

- **Arrays multidimensionais:**
  - Exemplo prático:

```
int tabela[][] = new int[3][4];
```

Com o comando acima criamos um array do tipo int com **4** posições.