PROGRAMAÇÃO
ORIENTADA A OBJETO E
QUALIDADE DE CÓDIGO

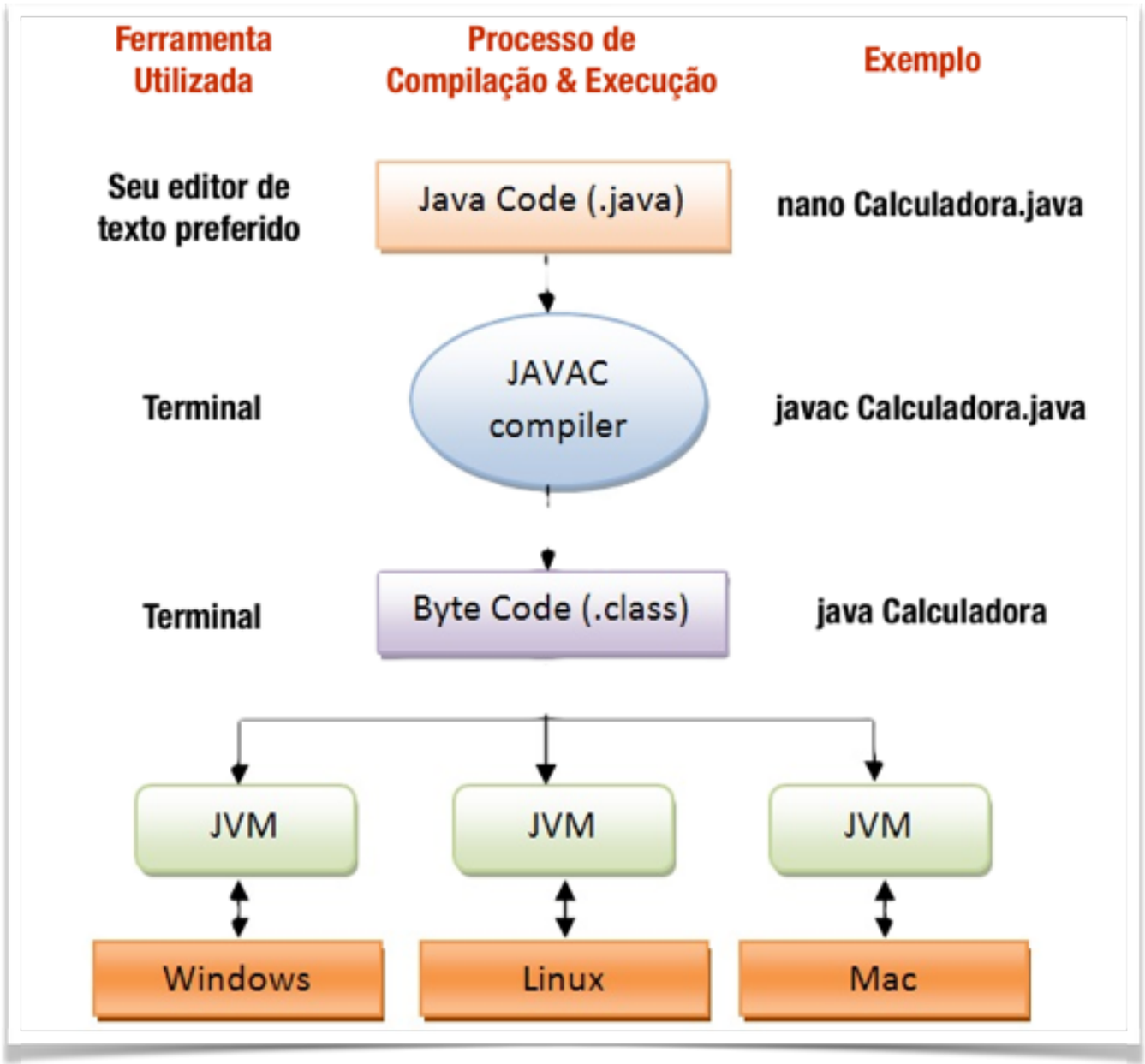
Conceitos de Java
[Parte I]

Prof. MsC. Reinaldo de O. Castro
[a.k.a. Reinaldo Luckman]
reinaldo@ufscar.br
reinaldo@doitlabs.com

VANTAGENS DA LINGUAGEM JAVA

- **Independente** de sistema operacional
  - Compile seu programa uma única vez e execute em qualquer sistema operacional (Linux, Mac, Windows, Solaris, etc)
- Não possui ponteiros explícitos (como C e C++), somente **ponteiros implícitos**
- A plataforma Java é **abrangente** e possui API's para desenvolvimento de soluções desktop, web e mobile, tanto stand-alone quanto cliente-servidor.

PROCESSO DE COMPILAÇÃO E DE EXECUÇÃO DE UM APLICATIVO JAVA



HELLO JAVA

- Em seu editor preferido, digite o código a seguir e salve-o como HelloJava.java:

```
1  public class HelloJava {
2      public static void main(String[] args) {
3          System.out.println("Hello Java!");
4      }
5  }
```

- Para compilar, abra um terminal e digite:
  javac HelloJava.java
- Para executar, digite (sem a extensão .class mesmo):
  java HelloJava

HELLO JAVA

Sempre crie uma classe (**class**) dentro de um arquivo .java

O nome da classe é exatamente igual ao nome do arquivo .java

```
1  public class HelloJava {
2      public static void main(String[] args) {
3          System.out.println("Hello Java!");
4      }
5  }
```

Todo aplicativo em Java deve ter uma classe que contenha o método **public static void main(String[] args)**

Imprime a mensagem na tela do computador

HELLO JAVA

- É muito importante você lembrar que:
  - O nome de um arquivo com a extensão **.java** deve ser exatamente ao nome da classe declarada nesse arquivo, incluindo letras maiúsculas e minúsculas (ou seja, sensível ao caso)
  - Enquanto na compilação de uma classe usa-se a extensão **.java**, quando executamos essa mesma classe, a extensão **.class** é omitida no comando **java**.

HELLO JAVA

- Exercício (sem olhar na transparências anteriores):
  - Crie uma classe chamada CursoDeEspecializacao e, dentro do método *main*, imprima "Vou me formar com louvor nesta especialização! :)"
  - Compile-a usando o programa **javac** e execute o bytecode usando o programa **java**

VARIÁVEIS EM JAVA

- Uma variável em Java, como em qualquer outra linguagem, é um **apelido** para um endereço de memória que contém um valor
- A **sintaxe genérica** para se declarar (e opcionalmente inicializar) uma variável em Java é: **<tipo> <nome_variavel> [= <valor_inicial>];**
- Em java temos dois tipos de variáveis: **primitivas e objetos**

VARIÁVEIS EM JAVA

- Exemplos de declaração de variáveis em Java:
  - Variável primitiva, somente declaração: **int idade;**
  - Variável primitiva, declaração e inicialização: **int idade = 10;**
  - Variável objeto, somente declaração: **Pessoa p;**
  - Variável objeto, declaração e inicialização: **Pessoa p = new Pessoa();**
  - A única variável objeto que pode ser inicializada sem o operador **new** é a do tipo **String**, que permite uma atribuição direta: **String nome = "Márcia";**

QUALIDADE DE CÓDIGO

NOMEAÇÃO DE VARIÁVEIS

- Dicas para nomear bem variáveis
  - Sempre descreva o que a variável realmente significa, ou seja, não use ‘x' e sim ‘somaSalarios'
  - Utilize, em média, de 8 a 20 caracteres no nome de uma variável; mais que isso o código se torna ilegível

QUALIDADE DE CÓDIGO

NOMEAÇÃO DE VARIÁVEIS

- Dicas para nomear bem variáveis (cont)
  - Nomes extremamente curtos geralmente estão dentro de um escopo pequeno e estão relacionados com alguma questão sintática; por exemplo, a variável 'i' para controlar laços
  - Evite chamar variáveis de ‘aux' ou ‘temp’; ela no geral tem um significado melhor que isso
  - Evitar nomes invertidos: 'formatoArquivo' e 'arquivoFormato'

VARIÁVEIS EM JAVA

- Tipos das variáveis primitivas em Java:

Type	Contains	Default	Size	Range
`boolean`	`true` or `false`	`false`	1 bit	NA
`char`	Unicode character	`\u0000`	16 bits	`\u0000` to `\uFFFF`
`byte`	Signed integer	0	8 bits	-128 to 127
`short`	Signed integer	0	16 bits	-32768 to 32767
`int`	Signed integer	0	32 bits	-2147483648 to 2147483647
`long`	Signed integer	0	64 bits	-9223372036854775808 to 9223372036854775807
`float`	IEEE 754 floating point	0.0	32 bits	±1.4E-45 to ±3.4028235E+38
`double`	IEEE 754 floating point	0.0	64 bits	±4.9E-324 to ±1.7976931348623157E+308

- Obs: os valores padrão são válidos somente para propriedades de uma classe e não variáveis locais

VARIÁVEIS EM JAVA

- Exercícios:
  - Crie uma classe Java chamada *Soma* que declare uma variável do tipo *short* chamada *resultado* e atribua o valor da soma 100 + 100 para esta variável, imprimindo seu resultado logo em seguida
  - Altere os valores 100 para 30000, compile. O que acontece?
  - Rescreva a linha da soma novamente dessa forma: ***short resultado = (short) (30000 + 30000);*** Compile e execute. Qual é o resultado da soma?

VETORES (ARRAYS) EM JAVA

- Um array é uma lista de **itens similares**, acessíveis via um índice que representa a posição do item que queremos obter
- Um array deve sempre ter:
  - Um **nome**
  - Um **tipo de dado**
  - Um **tamanho**

VETORES (ARRAYS) EM JAVA

- O tamanho de um array não pode ser alterado depois de sua criação, ou seja, sempre tem o **tamanho fixo**
- Em Java, a primeira posição do vetor é sempre no **índice de valor 0**. Assim, se criarmos um array de 5 posições, os respectivos índices de acesso são 0, 1, 2, 3 e 4

VETORES (ARRAYS) EM JAVA

- Como declarar um array:
  int myArray[];
- Como criar um array:
  myArray = new int[3];
- Como declarar e criar um array
  int myArray[] = new int[3];

VETORES (ARRAYS) EM JAVA

- Como declarar, criar e inicializar um array
  int myArray = new int[]{10, 20, 30};
- Como acessar uma posição de um array
  int myValue = myArray[2];
- Como obter o tamanho de um array
  myArray.length

VETORES (ARRAYS) EM JAVA

- Exemplo simples que declara um array de ***String*** e imprime logo em seguida

```java
public class CarArray {
    public static void main(String[] args) {
        String[] carros = new String[]{"Porshe", "Ferrari", "Maserati"};
        for (int i = 0; i < carros.length ; i++) {
            System.out.println(carros[i]);
        }
    }
}
```

VETORES (ARRAYS) EM JAVA

- Exercícios:
  - Para que serve o vetor de ***String*** chamado ***args*** passado ao método ***public static void main*** de toda classe principal em Java?
  - Altere a classe Soma para que agora ela some todos os números inteiros passados como parâmetro na linha de comando; deixe o tipo da variável *resultado* como ***short*** mesmo e execute primeiro com valores que caibam dentro do limite da variável e depois com valores que estourem esse limite
  - Dica: para converter uma ***String*** para um ***short***, use **Short.parseShort("10")**

APÊNDICE

ENTRADA DE DADOS VIA TECLADO

- Para deixar nossos programas mais interessantes, vamos verificar como permitir que o usuário entre com dados via teclado por meio da classe **Console**

```
1  import java.io.Console;
2  public class KeyboardReader {
3      public static void main(String[] args) {
4          Console c    = null;
5          String  nome = null;
6          try {
7              c = System.console();
8              if (c != null) {
9                  nome = c.readLine("Digite seu nome: ");
10                 System.out.println("O nome digitado foi: " + nome);
11             }
12         } catch (Exception e) {
13             e.printStackTrace();
14         }
15     }
16 }
```

APÊNDICE

ENTRADA DE DADOS VIA TECLADO

Importação da classe Console
(equivale ao #include de C)

Lê o que foi digitado
pelo usuário

```java
import java.io.Console;
public class KeyboardReader {
    public static void main(String[] args) {
        Console c    = null;
        String  nome = null;
        try {
            c = System.console();
            if (c != null) {
                nome = c.readLine("Digite seu nome: ");
                System.out.println("O nome digitado foi: " + nome);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Tratamento de exceção