

Aula 2 – Comandos básicos do Java

Prof. Me. Mateus Henrique Dal Forno

Programação III

Instituto Federal Farroupilha – Campus Frederico Westphalen
Curso Técnico em Informática

27 de fevereiro de 2024



- ▶ Desenvolvida em 1995 pela extinta *Sun Microsystems*
 - Adquirida pela *Oracle* em 2010.
- ▶ Apesar de ser uma linguagem **relativamente nova**, teve uma enorme **aceitação** no mundo inteiro.

► **Orientação a Objetos**

- Paradigma mais utilizado no mundo!
- Permite reaproveitamento de código
- Facilita a manutenibilidade dos sistemas.

► **Simplicidade de robustez**

- Representa um “aperfeiçoamento” da linguagem C++.
- Apresenta características que permitem a criação de programas de forma mais rápida.
- Tratamento (obrigatório!) de exceções.

► **Gerenciamento automático de memória**

- Não trabalha com acesso direto à memória (ponteiros).
- *Garbage Collector* – Mecanismo de alocação/liberação automático de memória.

► Independência de plataforma

- Um programa escrito em uma plataforma pode ser utilizado em outra plataforma qualquer.
- JVM (Java Virtual Machine) – Interpretador de programa Java para código de máquina específico da plataforma em questão.

► Multi-threading

- Possibilidade de realizar várias tarefas em paralelo.
- Thread – é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente.

O Java opera sobre 3 plataformas diferentes. Cada uma tem seu objetivo específico.

▶ **Java Standard Edition (JSE)**

- Ferramentas e APIs essenciais para qualquer aplicação java.
- Utilizada para desenvolver aplicações desktop com ou sem interface gráfica.

▶ **Java Enterprise Edition (JEE)**

- Ferramentas e APIs para o desenvolvimento de aplicações distribuídas.

▶ **Java Micro Edition (JME)**

- Ferramentas e APIs para o desenvolvimento de aplicações para aparelhos eletrônicos diversos (celulares, eletrodomésticos, palms, etc. . .)

Componentes necessários para JSE

- JRE Java Runtime Environment** – Necessário para executar aplicações java;
- JDK Java Development Kit** – Compilador, bibliotecas, API e JVM (máquina virtual);
- IDE Integrated Development Environment** – Eclipse, NetBeans. . .

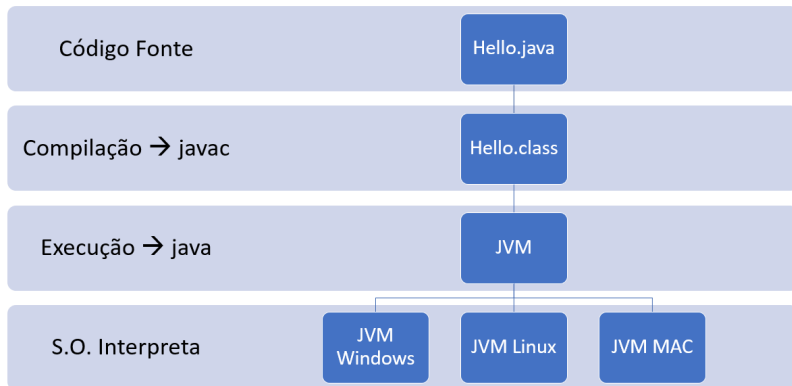
► Compilação

- Ao invés do programa ser compilado para código de máquina da plataforma que o programa será executado, o código será compilado para um bytecode (arquivo `.class`).
- O bytecode é genérico, isto é, não é específico para nenhum sistema operacional específico.

► Interpretação

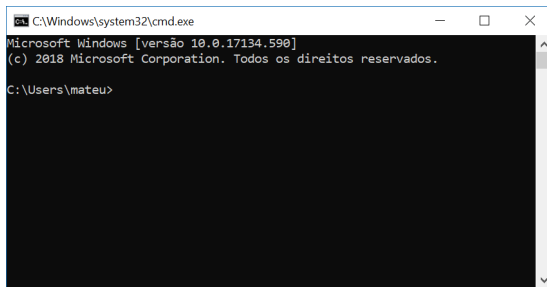
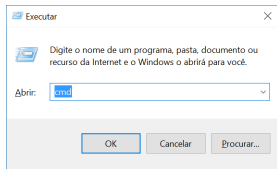
- Quando um programa java é executado, o arquivo bytecode é interpretado pela JVM.

Esquema de compilação e interpretação



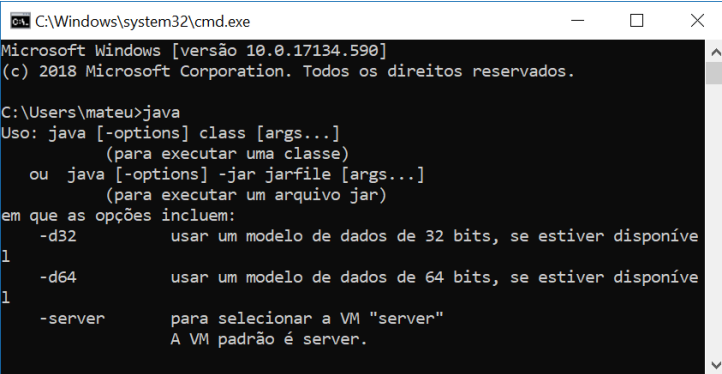
Verificando se o JRE e o JDK estão instalados

- ▶ Executar (*Windows Key + r*)
 - cmd



Verificando se o JRE está instalado

► java



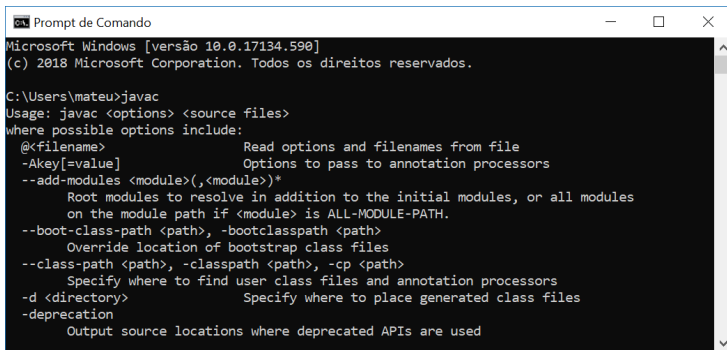
```
C:\Windows\system32\cmd.exe
Microsoft Windows [versão 10.0.17134.590]
(c) 2018 Microsoft Corporation. Todos os direitos reservados.

C:\Users\mateu>java
Uso: java [-options] class [args...]
        (para executar uma classe)
    ou  java [-options] -jar jarfile [args...]
        (para executar um arquivo jar)
em que as opções incluem:
    -d32          usar um modelo de dados de 32 bits, se estiver disponível
    -d64          usar um modelo de dados de 64 bits, se estiver disponível
    -server       para selecionar a VM "server"
                  A VM padrão é server.
```

► Para verificar a versão instalada utilize *java -version*

Verificando se o JDK está instalado

► javac

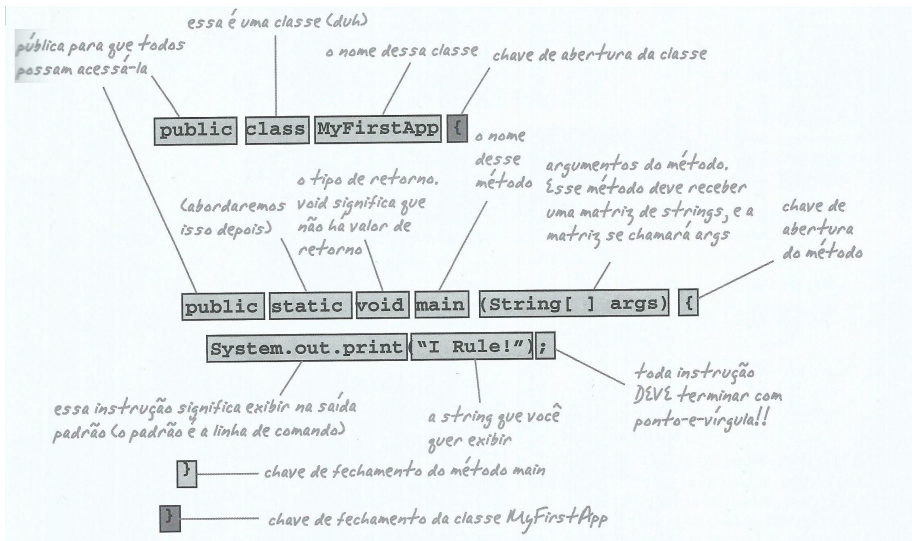


```
Prompt de Comando
Microsoft Windows [versão 10.0.17134.590]
(c) 2018 Microsoft Corporation. Todos os direitos reservados.

C:\Users\mateu>javac
Usage: javac <options> <source files>
where possible options include:
  @<filename>                Read options and filenames from file
  -Akey[=value]              Options to pass to annotation processors
  --add-modules <module>(,<module>)*
                             Root modules to resolve in addition to the initial modules, or all modules
                             on the module path if <module> is ALL-MODULE-PATH.
  --boot-class-path <path>, -bootclasspath <path>
                             Override location of bootstrap class files
  --class-path <path>, -classpath <path>, -cp <path>
                             Specify where to find user class files and annotation processors
  -d <directory>             Specify where to place generated class files
  -deprecation               Output source locations where deprecated APIs are used
```

► Para verificar a versão instalada utilize *javac -version*

Estrutura básica do código em Java



Princípios básicos da linguagem

- ▶ Java é *case-sensitive*.
- ▶ Todo nome de classe em Java Inicia com **letra maiúscula**.
- ▶ As classes, métodos ou blocos de código **sempre** estarão delimitados por chaves – “{” e “}”.
- ▶ Um comando deve **sempre** ser finalizado por um ponto e vírgula (“;”).
- ▶ Existem dois tipos de comentários
 - // comentário de linha
 - /* bloco de comentário */
- ▶ Nomes de variáveis devem sempre começar por **letras**, **\$** ou **_**

Componentes básicos da Classe

- ▶ **Variáveis de instância:** são os atributos de um objeto/da classe.
- ▶ **Métodos:** operações que a classe/objeto é capaz de executar.
- ▶ **Método main():** Uma aplicação java é caracterizada por possuir o método main().
- ▶ A declaração do método deve ser da seguinte maneira:

```
public static void main  
(String args[])
```

Gato
- raça : String - nome : String - cor : String - femea : boolean
+ miar() : void + lambear-se() : void + ronronar() : void + main(args : String[]) : void

Método main()

- ▶ É um método que representa o ponto de entrada para a execução de um programa java.
- ▶ Quando o programa é executado, o interpretador chamará primeiramente o método main da classe.
- ▶ É ele quem controla o fluxo de execução do programa e executa qualquer outro método necessário para a funcionalidade da aplicação.
- ▶ **Importante:** Nem toda classe terá um método main.
- ▶ Uma classe que não possui o método main não pode ser “executada”, pois não representa um programa.
- ▶ Uma classe sem o método main é utilizado como classe utilitária para a construção de outras classes ou mesmo de um programa.

Método main

```
00 public class MyFirstApp {  
02     public static void main (String [] args) {  
04         System.out.println("Mateus diz Ola Mundo ao Java!");  
06         ...  
    }  
}
```

MyFirstApp.java

Como imprimir no terminal

- ▶ Utilizando o comando `System.out.print!`
 - `System.out.print("texto");` mesma linha

Exemplo

```
00 public class MyFirstApp {  
02     public static void main (String [] args) {  
04         System.out.print("Mateus diz Ola Mundo ao Java!");  
06         System.out.print("Sejam bem vindos!");  
    }  
}
```

MyFirstApp1.java

Saída na tela

Mateus diz Ola Mundo ao Java! Sejam bem vindos!

Como imprimir no terminal

- ▶ Utilizando o comando `System.out.print!`
 - `System.out.println("texto");` nova linha

Exemplo

```
00 public class MyFirstApp {  
02     public static void main (String [] args) {  
04         System.out.println("Mateus diz Ola Mundo ao Java!");  
06         System.out.println("Sejam bem vindos!");  
    }  
}
```

MyFirstApp2.java

Saída na tela

Mateus diz Ola Mundo ao Java!
Sejam bem vindos!

Capturando dados do teclado

- Utiliza-se o Scanner;

Exemplo

```
00 import java.util.Scanner;
02
04 public class MyFirstApp {
06     public static void main (String[] args) {
08         Scanner teclado = new Scanner(System.in);
09         int teste = teclado.nextInt();
10         System.out.print(teste);
11     }
12 }
```

MyFirstApp3.java

Saída na tela (exemplo informando valor 10)

10

10

Capturando dados do teclado

Algumas observações do código:

```
00 import java.util.Scanner; // biblioteca importada
02 public class MyFirstApp {
04     public static void main (String[] args) {
06         Scanner teclado = new Scanner(System.in); // objeto scanner
           (esta linha vai ser sempre igual nos projetos)
           int teste = teclado.nextInt(); // lê o valor digitado e
           armazena na variável teste
           System.out.print(teste); // imprime o valor da variável
           teste na tela
08     }
}
```

MyFirstApp4.java

Variáveis em java

- Variáveis são tipadas e podem ser inicializadas;

```
00 public class MyFirstApp {  
02     public static void main (String [] args) {  
04         int valor;  
04         String nome = "Maria";  
06         boolean aposentado = false;  
06     }  
    }
```

MyFirstApp5.java

Concatenando uma variável para imprimir

```
00 import java.util.Scanner;
02 public class MyFirstApp {
04     public static void main (String [] args) {
06         Scanner teclado = new Scanner(System.in);
08         int valor = teclado.nextInt();
        System.out.print("Voce digitou " + valor);
    }
}
```

MyFirstApp6.java

Saída na tela (exemplo informando valor 10)

10

Voce digitou 10

Tipos primitivos de variáveis

Tipo	Categoria	Tamanho
byte	inteiro	8 bits
short	inteiro	16 bits
int	inteiro	32 bits
long	inteiro	64 bits
float	com casas decimais	32 bits
double	com casas decimais	64 bits
boolean	lógico	true/false
char	caracterer	16 bits

- ▶ Para armazenar palavras ou textos, utiliza-se **String**;
 - **String não é tipo primitivo**;

Operações com variáveis

```
00 import java.util.Scanner;
02
03 public class MyFirstApp {
04     public static void main (String [] args) {
05         Scanner teclado = new Scanner(System.in);
06         int valor = teclado.nextInt();
07         valor = valor * 2;
08         System.out.print("Voce digitou a metade de " + valor);
09     }
10 }
```

MyFirstApp7.java

Saída na tela (exemplo informando valor 10)

10


Voce digitou a metade de 20


Operações com variáveis

Variáveis ponto flutuante (com casas decimais)

```
01 import java.util.Scanner;
02
03 public class MyFirstApp {
04
05     public static void main(String[] args) {
06
07         Scanner teclado = new Scanner(System.in);
08         int litros = teclado.nextInt();
09         double valor = teclado.nextDouble();
10         System.out.println("Valor = R$ " + String.format("%.2f", (litros*valor))); //String.format serve para delimitar
        //quantas casas decimais queremos exibir
    }
}
```

MyFirstApp8.java

 DEITEL, P. *Java: como programar*. 10. ed. São Paulo: Pearson Education do Brasil, 2017.

 SIERRA, K.; BATES, B. *Use a cabeça! Java*. 2. ed. Rio de Janeiro: Alta Books, 2009.