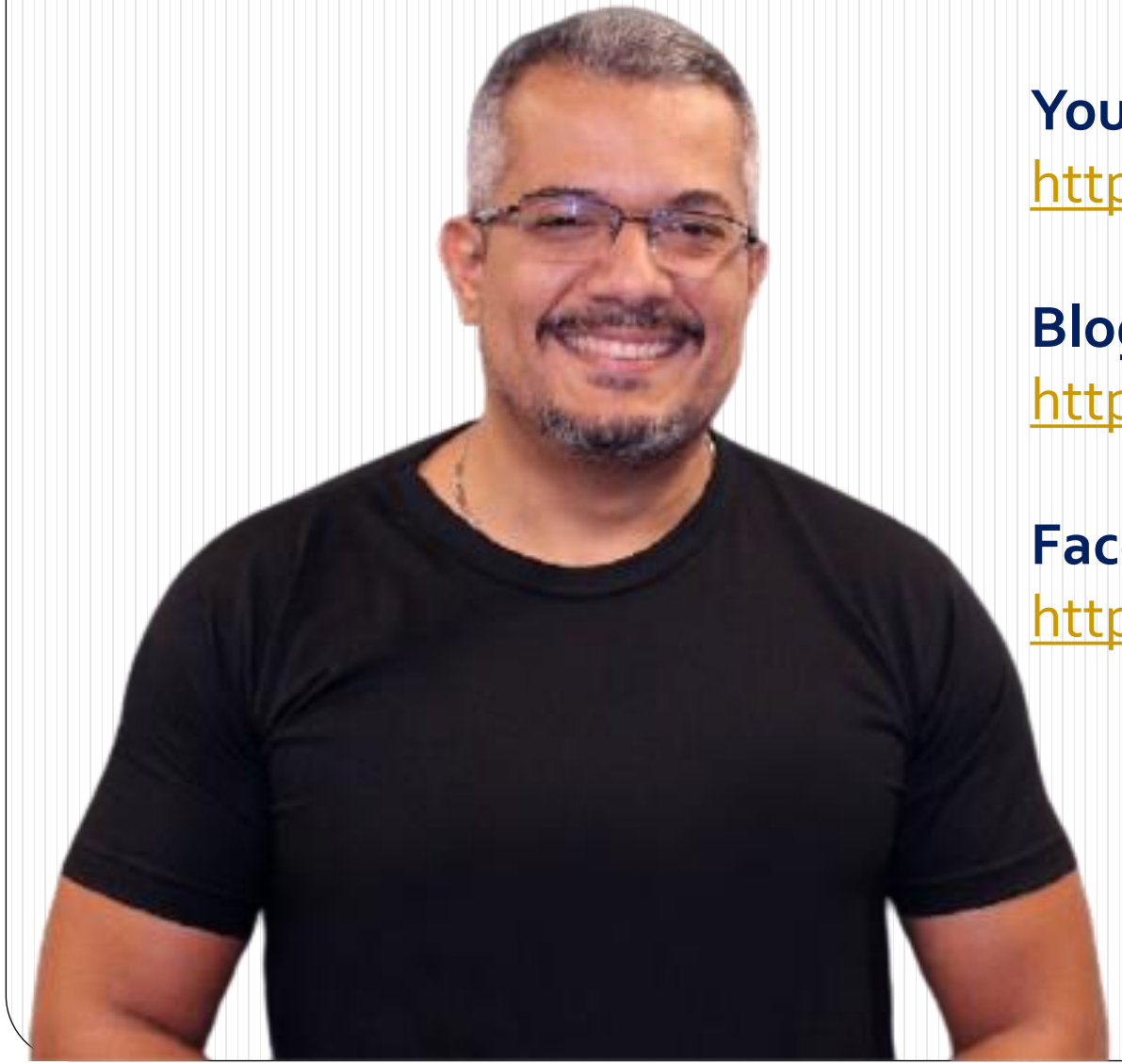
Linguagem Java SE 11 – Parte I

Prof. Rogerão Araújo

www.instagram.com/profRogeraoAraujo

PROVAS DE TI
TUDO PARA VOCÊ PASSAR

[Professor] – Rogerão Araújo



Instagram:
http://www.instagram.com/profrogeraoaraujo

Youtube:
http://www.youtube.com/rgildoaraujo

Blog:
http://www.rogeraoaraujo.com.br

Facebook:
http://www.facebook.com/professorRogerioAraujo

ROGERAOARAUJO
www.rogeraoaraujo.com.br

Módulos

Módulos


Básico da Linguagem Java
POO com Java
Mais da Linguagem Java

Básico da Linguagem Java

Introdução ao Java

Variáveis

Tipos primitivos e valores literais

Operadores

Expressões, declarações e blocos

Controle de fluxos

Arrays

Programação orientada a objetos com Java

Introdução à POO	Classes e objetos	Herança	Polimorfismo
Classes e métodos abstratos	Interfaces	Tipos enumerados	Annotations

Mais da Linguagem Java

Tratamento de exceções

Threads

Números e Strings

Generics

Reflection

Collections

Expressões lambda

I/O

Algoritmos em geral

Básico da Linguagem Java

Introdução ao Java

Introdução e conceituação

O que é Java?

Java

Linguagem de programação de alto nível

Plataforma

Plataformas da tecnologia Java

Java SE

Java Standard Edition

Java EE

Java Enterprise Edition

Java ME

Java Micro Edition

JavaFX

Plataforma de software multimídia

Características-chave da linguagem

Simples, familiar e orientada a objetos

Arquitetura neutra e portável

Robusta e segura

Interpretada, multi-thread e dinâmica

De alto desempenho

Questões de concursos

[IBFC 2021 IBGE – Supervisor de Pesquisa – Tecnologia de Informação e Comunicação] Quanto às linguagens de programação, assinale a alternativa que esteja tecnicamente incorreta. (Marque CERTO ou ERRADO o texto da questão)

- [A] Java é uma das principais representantes das linguagens orientadas a objetos.

Questões de concursos

[IBFC 2021 IBGE – Supervisor de Pesquisa – Tecnologia de Informação e Comunicação] Quanto às linguagens de programação, assinale a alternativa que esteja tecnicamente incorreta. (Marque CERTO ou ERRADO o texto da questão)

- [A] Java é uma das principais representantes das linguagens orientadas a objetos.
  - Gabarito: **CERTO**.

Visão geral do desenvolvimento com Java

Arquivos .java e .class

Arquivos .java

São arquivos textos

Contêm o código fonte

Arquivos .class

São arquivos gerados da compilação dos arquivos .java pelo compilador javac

Contêm os bytecodes

Arquivos .class

São arquivos gerados da compilação dos arquivos .java pelo compilador javac

Não contêm código nativo para um processador específico

São interpretados e executados

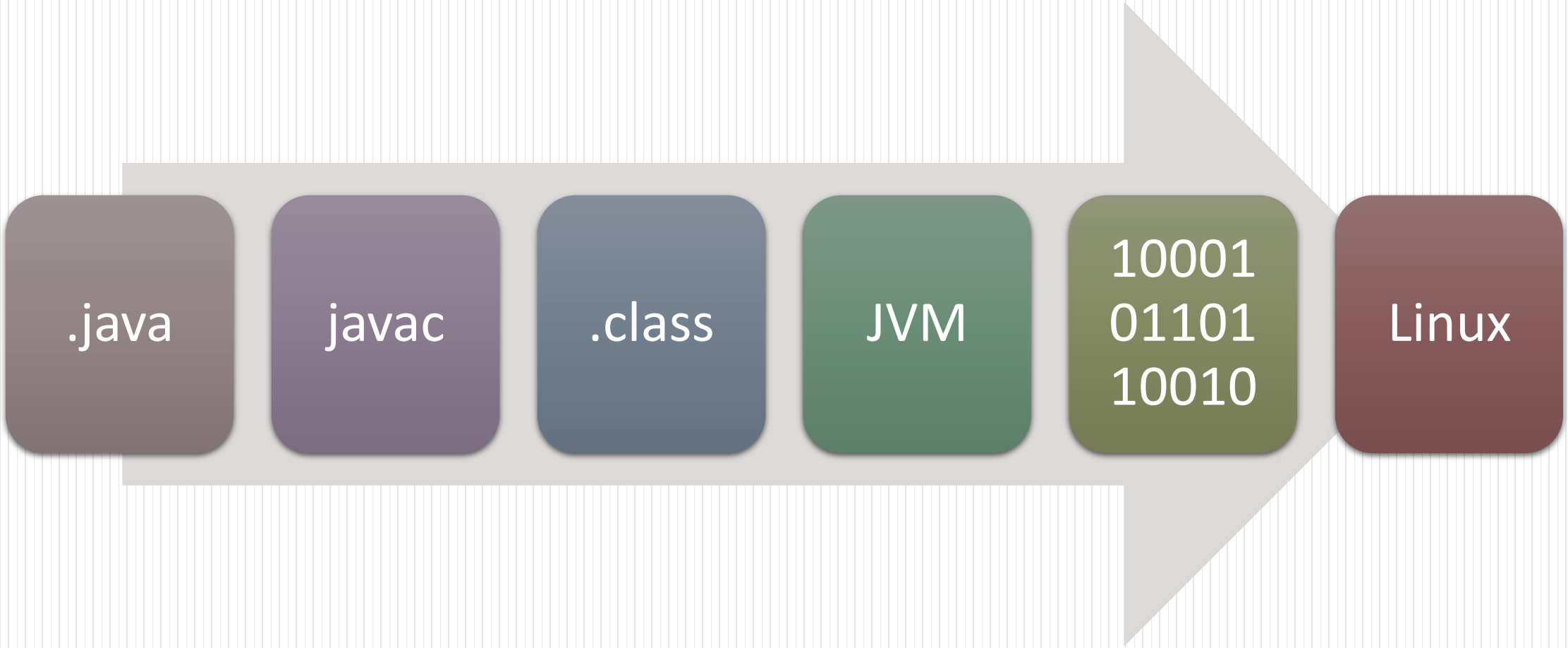
Contêm bytecodes

Pelo comando java

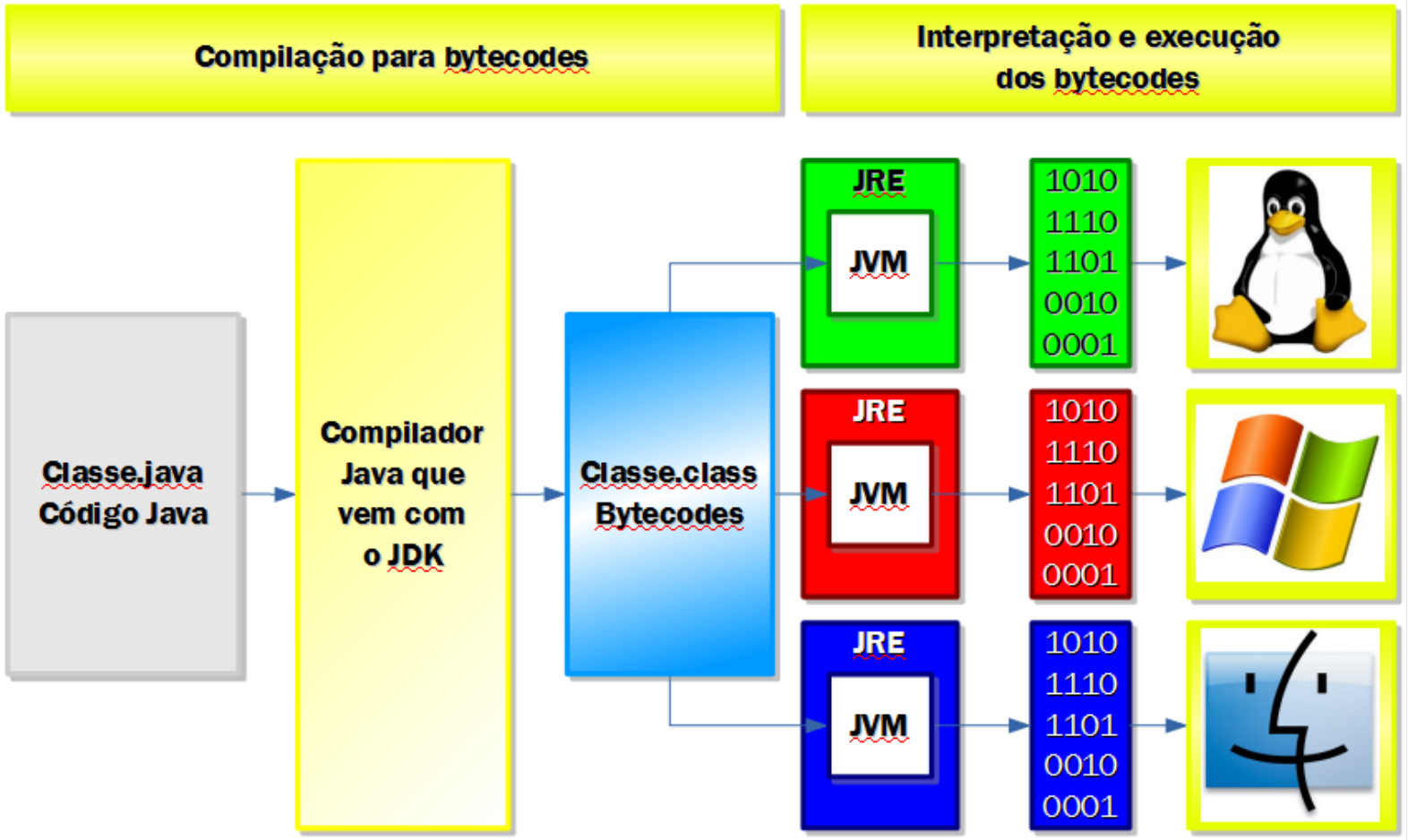
Para criar uma instância da JVM

É uma linguagem da Máquina do Virtual Java (JVM)

Compilação, interpretação e execução



Compilação, interpretação e execução

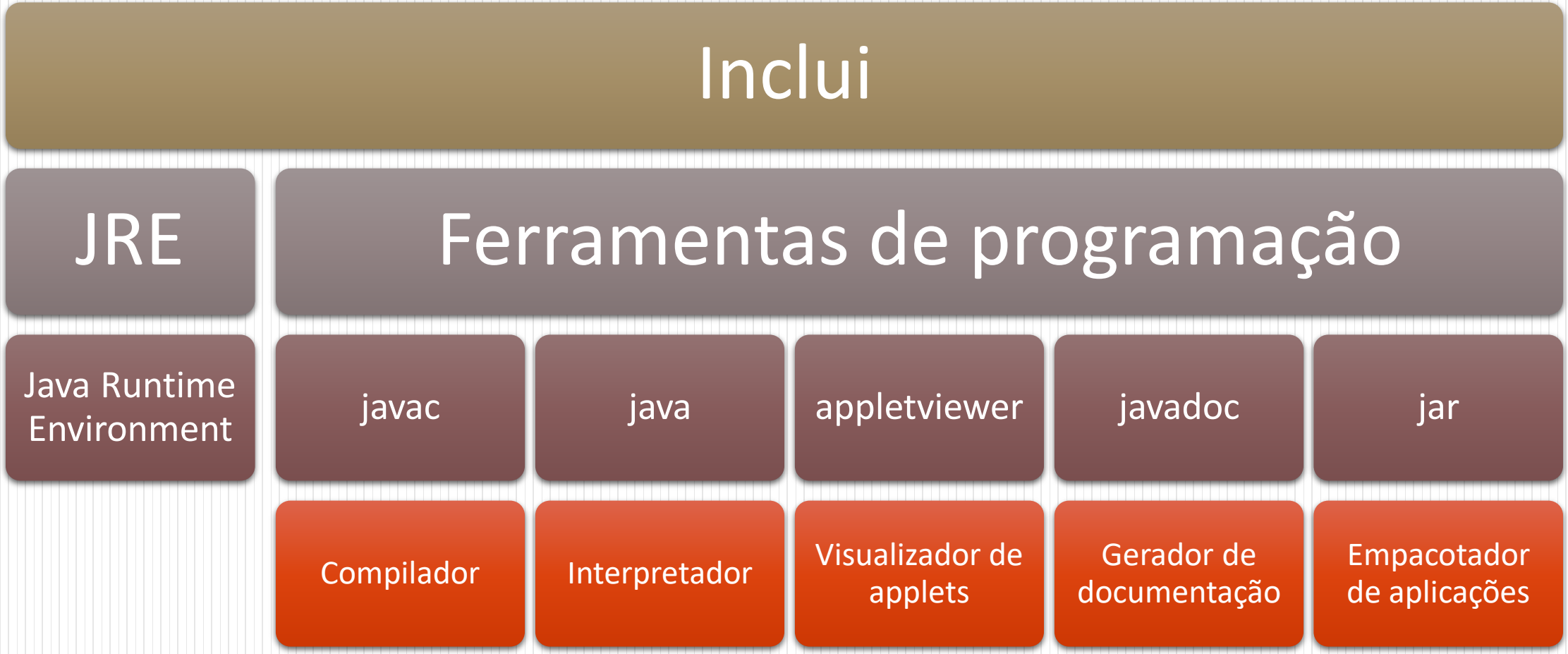


JDK

Java Development Kit

É o conjunto de ferramentas necessárias para realizar o desenvolvimento de aplicações Java

JDK


Inclui
JRE
Ferramentas de programação
Java Runtime Environment
javac
java
appletviewer
javadoc
jar
Compilador
Interpretador
Visualizador de applets
Gerador de documentação
Empacotador de aplicações

JRE

Java Runtime Environment

É a plataforma Java

JRE

É composto pelos componentes

JVM

Java Virtual Machine

Java API

Java Application Programming Interface

É uma biblioteca de componentes que possui vários recursos úteis

É utilizada para execução de aplicações Java

JRE

JRE específico de uma plataforma

É necessário instalá-lo um para a plataforma desejada

Pois junto com ele vem uma JVM que

Saberá lidar com essa plataforma

Conseguirá executar aplicações Java naquele ambiente

JVM

- **Java Virtual Machine**
- É a **peça-chave** para **fornecer** **capacidade de multiplataforma** para as **aplicações Java**
  - **“Write once, run everywhere”**
- Está **disponível** em **muitos sistemas operacionais diferentes**
  - Com isso, os mesmos arquivos .class são capazes de funcionar nesses sistemas operacionais
    - Microsoft Windows, Solaris OS, Linux, Mac OS, etc
- **Pode ser desenvolvida** por **qualquer organização**
  - Desde que siga as especificações para construção de uma JVM

JVM

- É **responsável** por **interpretar** e **executar** o **bytecode**
- É **provedora** de **formas** e **meios** de o **aplicativo conversar** com o **sistema operacional**
- No **tempo de execução**:
  - **Carrega** os **arquivos de classe** que **contém bytecodes**
  - **Determina** a **semântica** de **cada bytecode individual**
  - **Executa** a **computação apropriada**
- O uso adicional do processador e da memória durante a interpretação significa que um aplicativo Java executa mais lentamente do que um aplicativo nativo

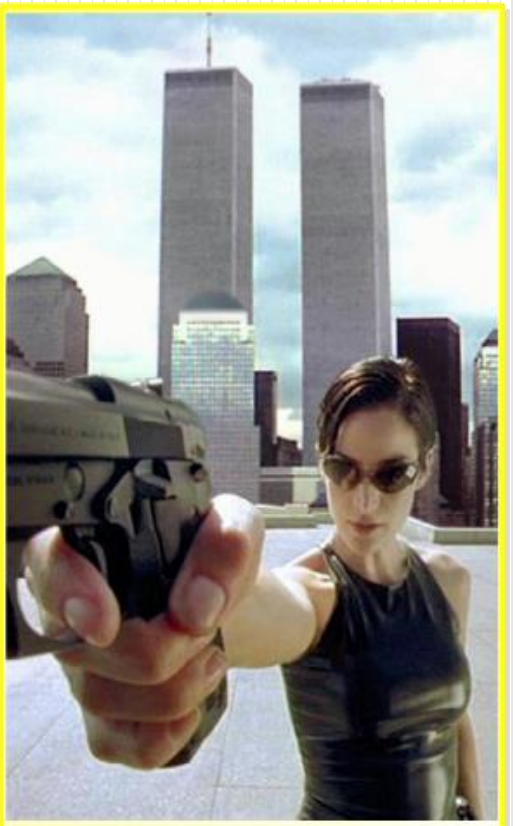
JIT

- **Compilador Just-in-time**
- É um **componente do JRE** que **melhora** o **desempenho** de **aplicativos Java** no **tempo de execução**
- **Ajuda** a **melhorar** o **desempenho** de **programas Java**
  - **Compilando bytecodes** no **código de máquina nativo** no tempo de execução
- É **ativado** quando um **método Java** é **chamado**
  - Os **bytecodes desse método** são **compilados** no **código de máquina nativo**
  - Quando um método tiver sido compilado:
    - A JVM chama o código compilado desse método diretamente
      - Em vez de interpretá-lo

Esquema 1

JDK

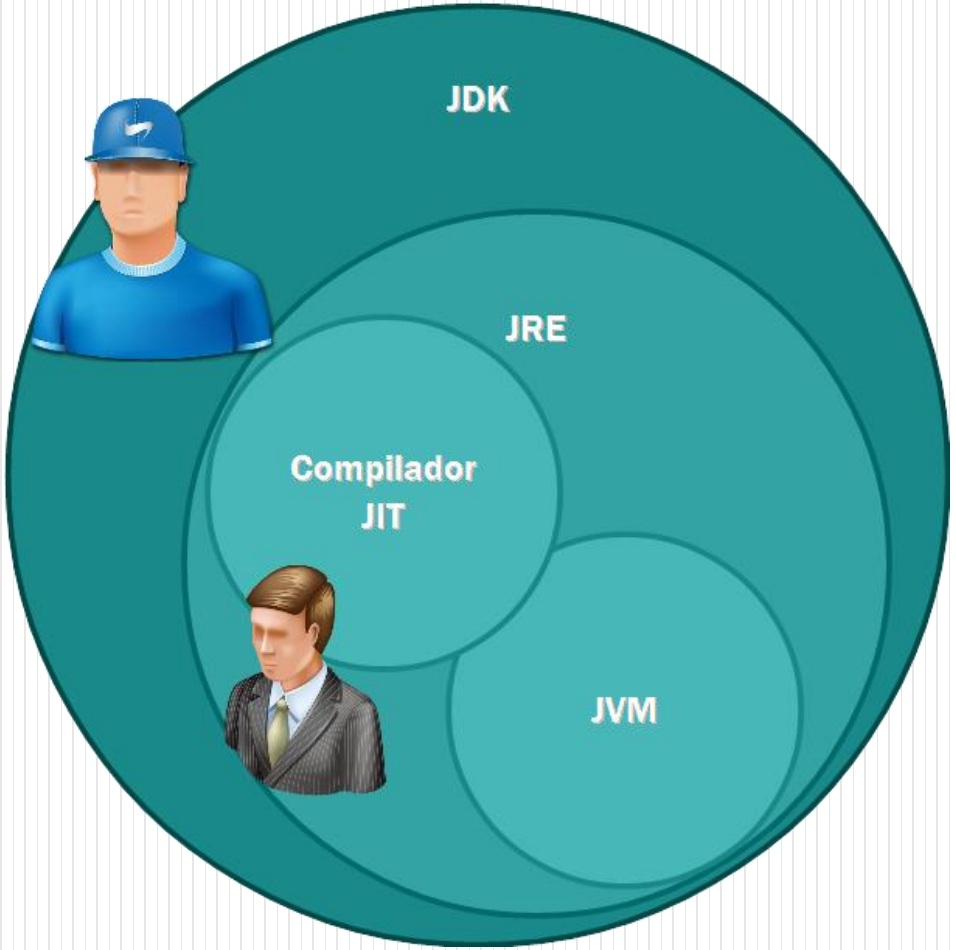
JRE



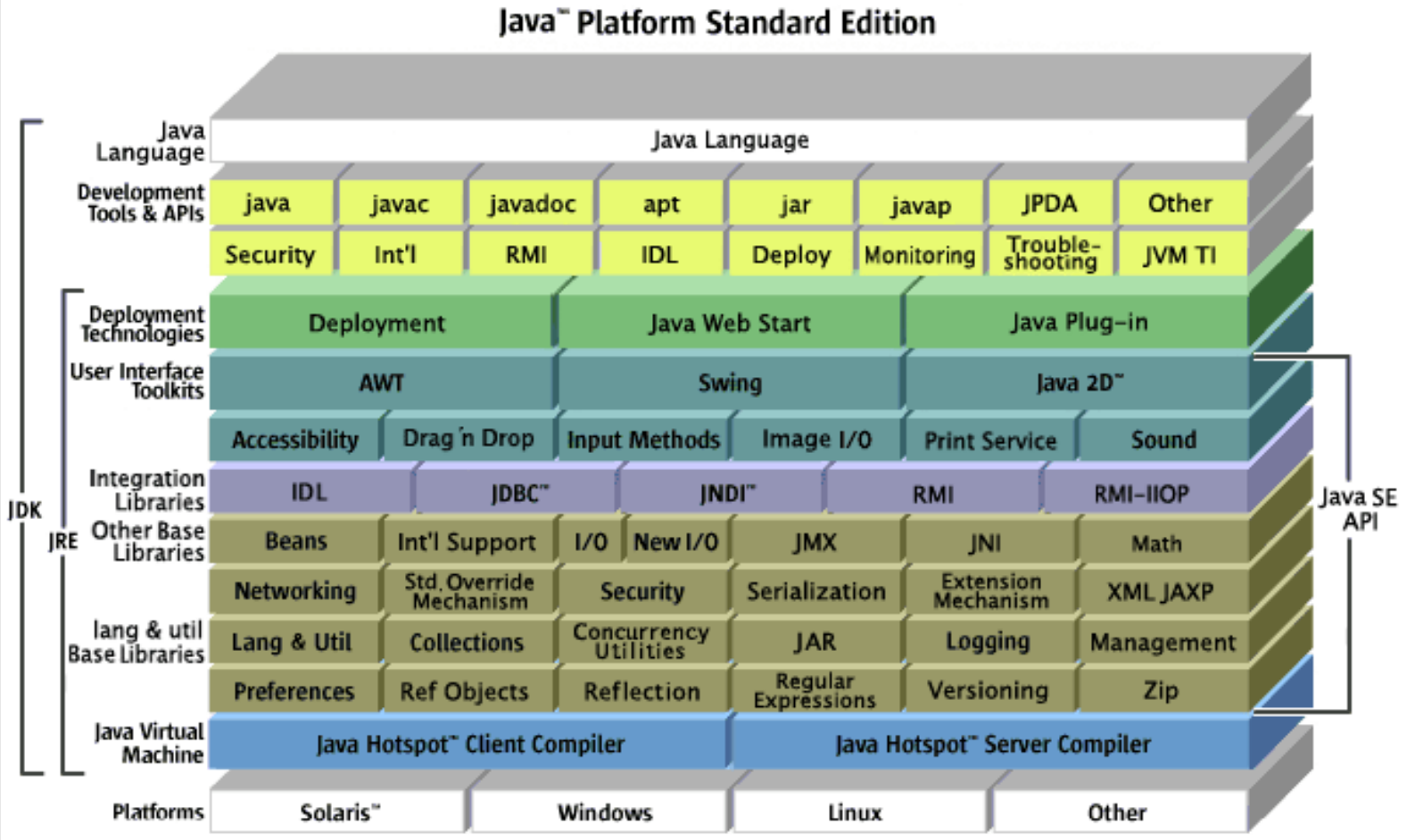
JVM

Esquema 2

Esquema 3



Esquema 4


Java™ Platform Standard Edition
Java Language
Java Language
Development Tools & APIs
java
javac
javadoc
apt
jar
javap
JPDA
Other
Security
Int'l
RMI
IDL
Deploy
Monitoring
Trouble-shooting
JVM TI
Deployment Technologies
Deployment
Java Web Start
Java Plug-in
User Interface Toolkits
AWT
Swing
Java 2D™
Accessibility
Drag 'n Drop
Input Methods
Image I/O
Print Service
Sound
Integration Libraries
IDL
JDBC™
JNDI™
RMI
RMI-IIOP
Other Base Libraries
Beans
Int'l Support
I/O
New I/O
JMX
JNI
Math
Networking
Std. Override Mechanism
Security
Serialization
Extension Mechanism
XML JAXP
lang & util Base Libraries
Lang & Util
Collections
Concurrency Utilities
JAR
Logging
Management
Preferences
Ref Objects
Reflection
Regular Expressions
Versioning
Zip
Java Virtual Machine
Java Hotspot™ Client Compiler
Java Hotspot™ Server Compiler
Platforms
Solaris™
Windows
Linux
Other
JDK
JRE
Java SE API

Questões de concursos

[Instituto AOCP 2021 ITEP/RN – Perito Criminal – Computação] Assinale a alternativa correta sobre as características da linguagem de programação Java.

- [A] A execução do código-objeto é feita diretamente pelo sistema operacional.
- [B] Não possui tratamento de exceções.
- [C] A linguagem é totalmente interpretada.
- [D] Não suporta o uso de ponteiros.
- [E] Possui 32 palavras reservadas.

Questões de concursos

[Instituto AOCP 2021 ITEP/RN – Perito Criminal – Computação] Assinale a alternativa correta sobre as características da linguagem de programação Java.

- [A] A execução do código-objeto é feita diretamente ~~pelo sistema operacional~~ **pela JVM**.
- [B] ~~Não~~ possui tratamento de exceções.
- [C] A linguagem é ~~totalmente interpretada~~ **híbrida**.
- **[D] Não suporta o uso de ponteiros.**
- [E] Possui ~~32~~ **52** palavras reservadas.

Questões de concursos

[IDIB 2021 CRF/MS – Analista de Informática] Considerando a arquitetura da linguagem de programação Java e o Java Development Kit (JDK) e o Java Runtime Environment (JRE), assinale a alternativa correta.

- [A] O JDK engloba o JRE e ferramentas como java, javac e javadoc.
- [B] O JDK e o JRE compartilham a JVM, bibliotecas e ferramentas.
- [C] O JRE é utilizado para desenvolver aplicações Java.
- [D] O JRE engloba todas as tecnologias do JDK.

Questões de concursos

[IDIB 2021 CRF/MS – Analista de Informática] Considerando a arquitetura da linguagem de programação Java e o Java Development Kit (JDK) e o Java Runtime Environment (JRE), assinale a alternativa correta.

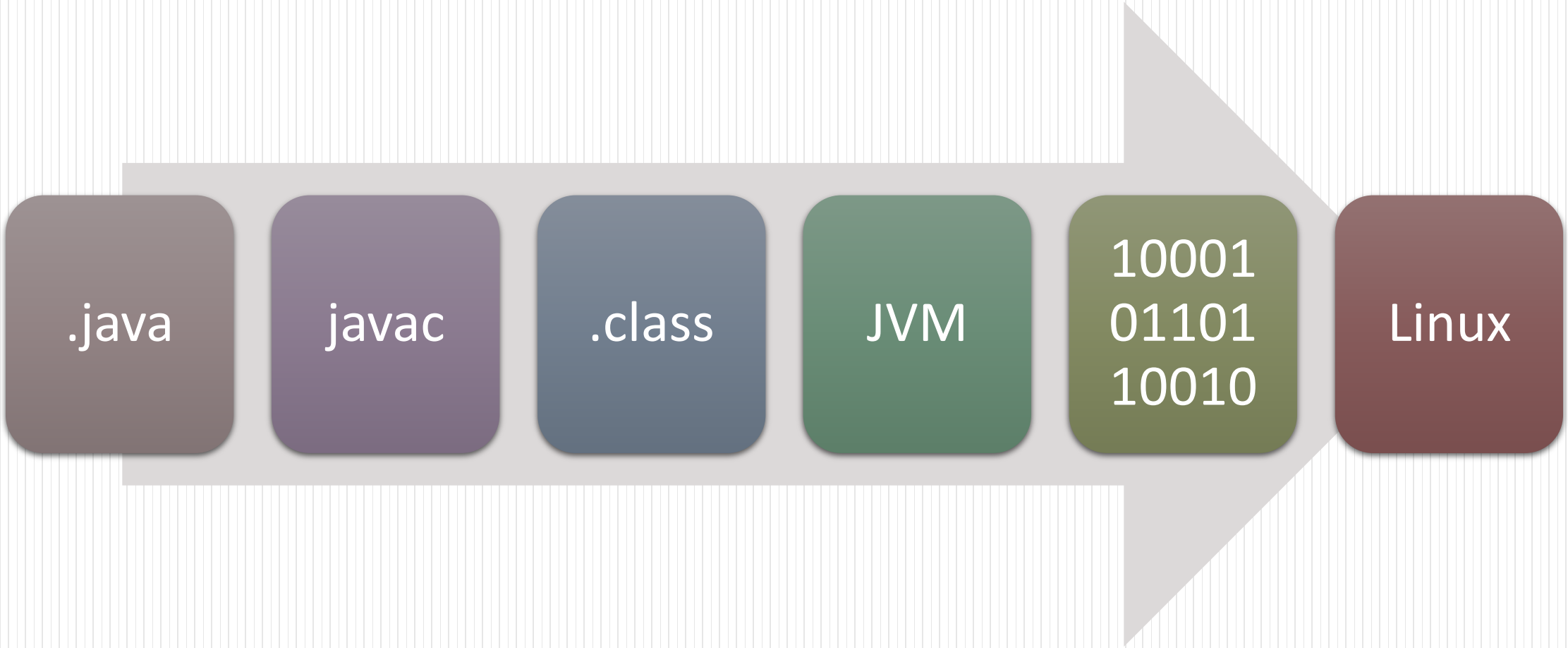
- **[A] O JDK engloba o JRE e ferramentas como java, javac e javadoc.**
- [B] O JDK e o JRE compartilham a JVM, bibliotecas e ferramentas.
- [C] O JRE é utilizado para desenvolver aplicações Java.
- [D] O JRE engloba todas as tecnologias do JDK.

Questões de concursos

[CESPE/CEBRASPE 2021 SERPRO – Analista – Especialização: Ciência de Dados] Sobre a linguagem de programação JAVA, julgue o próximo item.

- A compilação de um programa JAVA para ambiente Windows gera um programa com extensão EXE, o qual é executado pelo sistema operacional.

Comentários



Questões de concursos

[CESPE/CEBRASPE 2021 SERPRO – Analista – Especialização: Ciência de Dados] Sobre a linguagem de programação JAVA, julgue o próximo item.

- A compilação de um programa JAVA para ambiente Windows gera um ~~**programa**~~ **arquivo** com extensão ~~**EXE**~~ **.class**, o qual é ~~**executado pelo sistema operacional**~~ **interpretado pela JVM**.
  - Gabarito: **ERRADO**.

Questões de concursos

[CESPE/CEBRASPE 2021 SEED/PR – Professor – Educação Básica e Jornada] Java é uma linguagem construída a partir de um legado das linguagens C e C++. No entanto, ela apresenta características únicas que a diferem das demais, como: (Marque CERTO ou ERRADO o texto do item)

- [II] a saída de seu compilador não gera um código executável e, sim, um *bytecode*.

Questões de concursos

[CESPE/CEBRASPE 2021 SEED/PR – Professor – Educação Básica e Jornada] Java é uma linguagem construída a partir de um legado das linguagens C e C++. No entanto, ela apresenta características únicas que a diferem das demais, como: (Marque CERTO ou ERRADO o texto do item)

- [II] a saída de seu compilador não gera um código executável e, sim, um *bytecode*.
  - Gabarito: **CERTO**.

Questões de concursos

[CESPE/CEBRASPE 2021 SEED/PR – Professor – Educação Básica e Jornada] Java é uma linguagem construída a partir de um legado das linguagens C e C++. No entanto, ela apresenta características únicas que a diferem das demais, como: (Marque CERTO ou ERRADO o texto do item)

- [III] o fato de um programa Java ser executado somente pela Java *virtual machine* (JVM).

Questões de concursos

[CESPE/CEBRASPE 2021 SEED/PR – Professor – Educação Básica e Jornada] Java é uma linguagem construída a partir de um legado das linguagens C e C++. No entanto, ela apresenta características únicas que a diferem das demais, como: (Marque CERTO ou ERRADO o texto do item)

- [III] o fato de um programa Java ser executado somente pela Java *virtual machine* (JVM).
  - Gabarito: **CERTO**.

Garbage collection e garbage collector

Garbage collection

Coleta de lixo

É o processo em que o JRE exclui objetos

Quando aquele determina que estes não estão mais sendo usados

Não é necessário a destruição explícita dos objetos criados

Garbage collection

- Um **objeto é elegível para coleta de lixo** quando **não houver** mais **referências a esse objeto**
  - Todas as referências a um objeto devem ser descartadas antes que o objeto seja elegível para coleta de lixo
- Referências mantidas em uma variável são normalmente descartadas quando:
  - A variável sai do escopo em que ela se encontra
  - Um valor null é atribuído à variável-objeto
    - Explicitamente

Garbage collector

É um programa que está dentro do JRE

Não é controlado pelo programador

Libera periodicamente a memória usada por objetos que não são mais referenciados

A JVM decide quando executá-lo

Garbage collector

Faz o seu trabalho automaticamente

Quando determina que é a hora certa

Há a possibilidade do programador chamar o gabarge collector

Não há um momento pré-determinado

Mesmo assim não há garantia de que este irá realmente ser executado no momento que é chamado

Método finalize() da classe Object

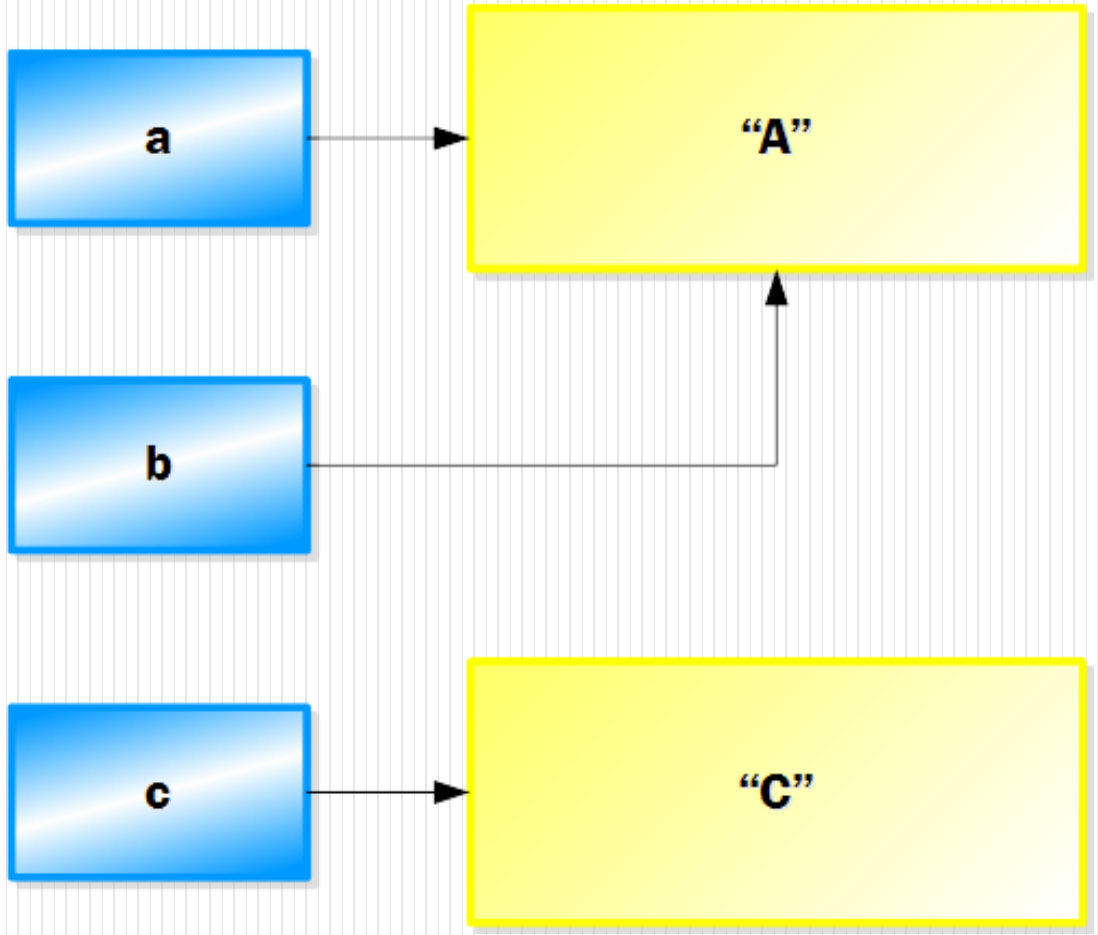
- **protected void finalize()**
- **Pertence** à **classe Object**
- É **chamado** pelo **Garbage Collector** para um **objeto**
  - Quando o processo **Garbage Collection determina** que **não há mais referências** para esse objeto

Método gc() da classe System

- **public static void gc()**
- **Pertence** à **classe System**
- **Força** chamar o **Garbage Collector**
- Chamar esse método sugere que a JVM envie esforços para a reciclagem de objetos não utilizados
  - A fim de tornar a memória que eles atualmente ocupam disponível para reutilização rápida
  - Nada garante que ele realmente vá coletar lixo naquele momento
    - Mesmo tentando “forçar” a coleta de lixo

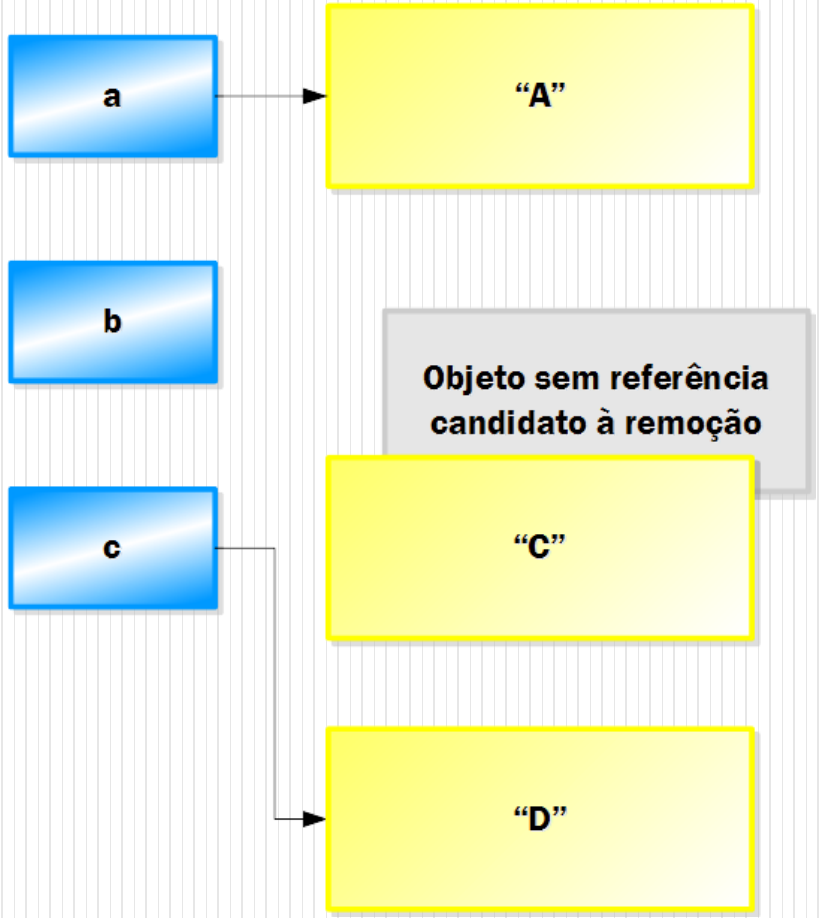
Exemplo

- Classe a;
- Classe b;
- Classe c;
- a = new Classe(“A”);
- b = a;
- c = new Classe(“C”);



Exemplo (continuação)

- b = null;
- c = new Classe(“D”);



Questões de concursos

[UERJ 2021 UERJ – Analista de Tecnologia da Informação] Tendo em vista o suporte para orientação a objetos na linguagem JAVA, caso um objeto do tipo enumeração fosse criado, para liberar seu armazenamento na memória seria necessário:

- [A] invocar o coletor de lixo
- [B] aguardar o coletor de lixo
- [C] invocar um método delete()
- [D] aguardar a chamada automática do método dispose()

Questões de concursos

[UERJ 2021 UERJ – Analista de Tecnologia da Informação] Tendo em vista o suporte para orientação a objetos na linguagem JAVA, caso um objeto do tipo enumeração fosse criado, para liberar seu armazenamento na memória seria necessário:

- [A] invocar o coletor de lixo
- **[B] aguardar o coletor de lixo**
- [C] invocar um método delete()
- [D] aguardar a chamada automática do método dispose()

Questões de concursos

[VUNESP 2021 TJM/SP – Técnico em Comunicação e Processamento de Dados Judiciário (Desenvolvedor)] Na plataforma Java SE 8, o coletor de lixo (garbage collector) somente libera o espaço ocupado pelo objeto A na memória quando

- [A] o método destrutor do objeto A é invocado.
- [B] um novo objeto, do mesmo tipo que o objeto A, é instanciado.
- [C] todos os demais objetos referenciados pelo objeto A são apagados.
- [D] não existem referências a objetos imutáveis (como String) dentro do objeto A.
- [E] não há mais referências para o objeto A no programa.

Questões de concursos

[VUNESP 2021 TJM/SP – Técnico em Comunicação e Processamento de Dados Judiciário (Desenvolvedor)] Na plataforma Java SE 8, o coletor de lixo (garbage collector) somente libera o espaço ocupado pelo objeto A na memória quando

- [A] o método destrutor do objeto A é invocado.
- [B] um novo objeto, do mesmo tipo que o objeto A, é instanciado.
- [C] todos os demais objetos referenciados pelo objeto A são apagados.
- [D] não existem referências a objetos imutáveis (como String) dentro do objeto A.
- **[E] não há mais referências para o objeto A no programa.**

Questões de concursos

[CESGRANRIO 2011 Petrobrás – Analista de Sistemas Júnior - Engenharia de Software] Considere o seguinte código Java, contido no arquivo R.java

```
1.  class P {
2.  private int id;
3.  protected void finalize() {System.out.print(id);}
4.  public P(int i) {id = i;}
5.  }
6.  class R {
7.  public static void main(String[] args) {
8.  P p1 = null;
9.  for (int i = 0; i < 5; i++) {p1 = new P(i);}
10. System.gc();
11. }}
```

Questões de concursos

[CESGRANRIO 2011 Petrobrás – Analista de Sistemas Júnior - Engenharia de Software] No momento imediatamente anterior à execução da linha 10, quantos objetos do tipo P, que foram criados na linha 9, tornaram-se elegíveis para ser apanhados para a garbage collection?

- [A] 0
- [B] 1
- [C] 4
- [D] 5
- [E] 9

Comentário


p1
p1
p1
p1
p1
p1
• null
• new P(0)
• new P(1)
• new P(2)
• new P(3)
• new P(4)
Objetos elegíveis para coleta de lixo

Questões de concursos

[CESGRANRIO 2011 Petrobrás – Analista de Sistemas Júnior - Engenharia de Software] No momento imediatamente anterior à execução da linha 10, quantos objetos do tipo P, que foram criados na linha 9, tornaram-se elegíveis para ser apanhados para a garbage collection?

- [A] 0
- [B] 1
- **[C] 4**
- [D] 5
- [E] 9

Tipos primitivos e valores literais

Introdução

Estaticamente e fortemente tipada

Linguagem de programação Java é

Estaticamente tipada

Quando o tipo de cada variável deve ser conhecido em tempo de compilação

```
int x = 15;
```

Fortemente tipada

Quando há a exigência de que os valores armazenados em uma variável sejam compatíveis com o tipo dela

```
int x = 15;
```

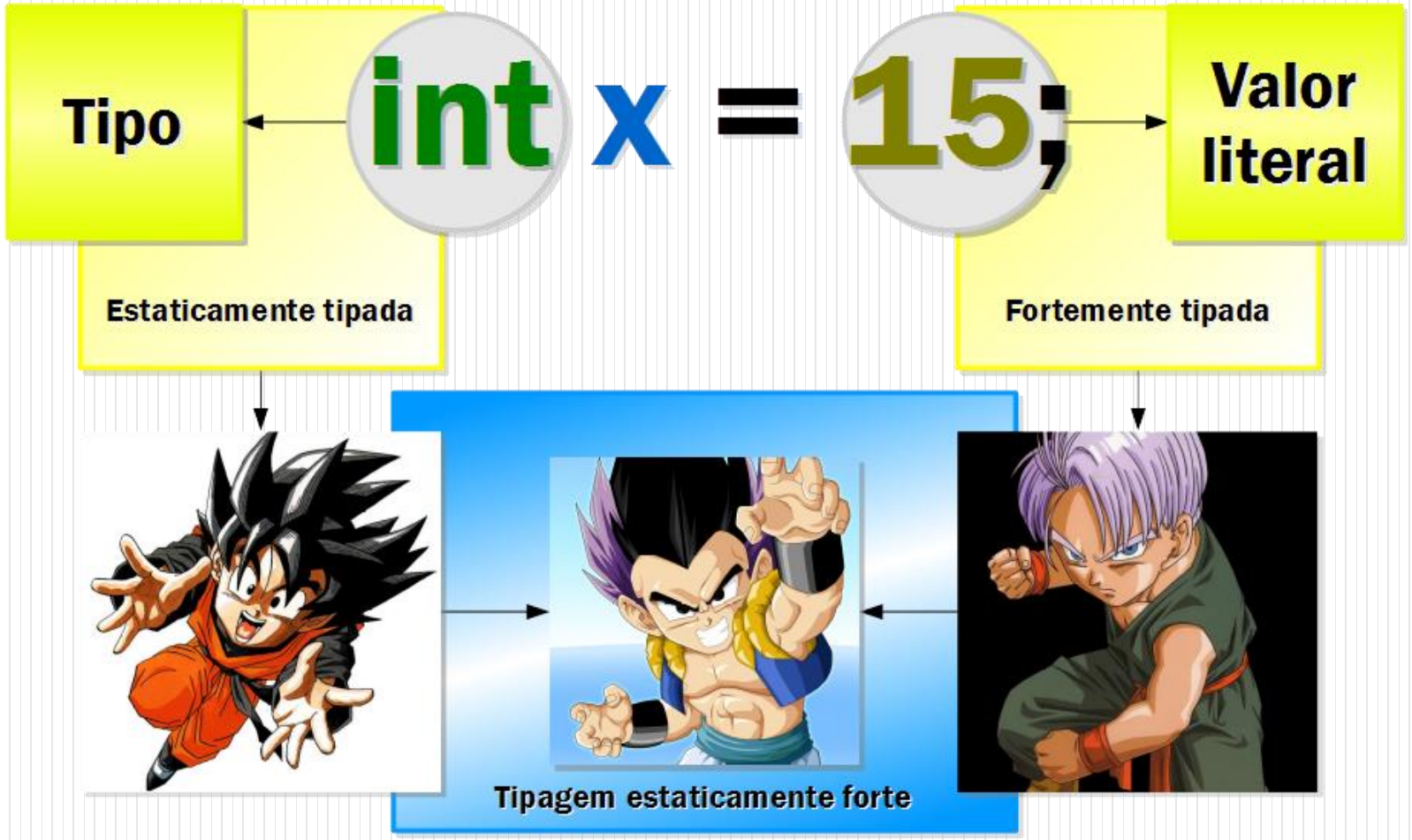
Tipagem estaticamente forte

Ajuda a detectar erros em tempo de compilação

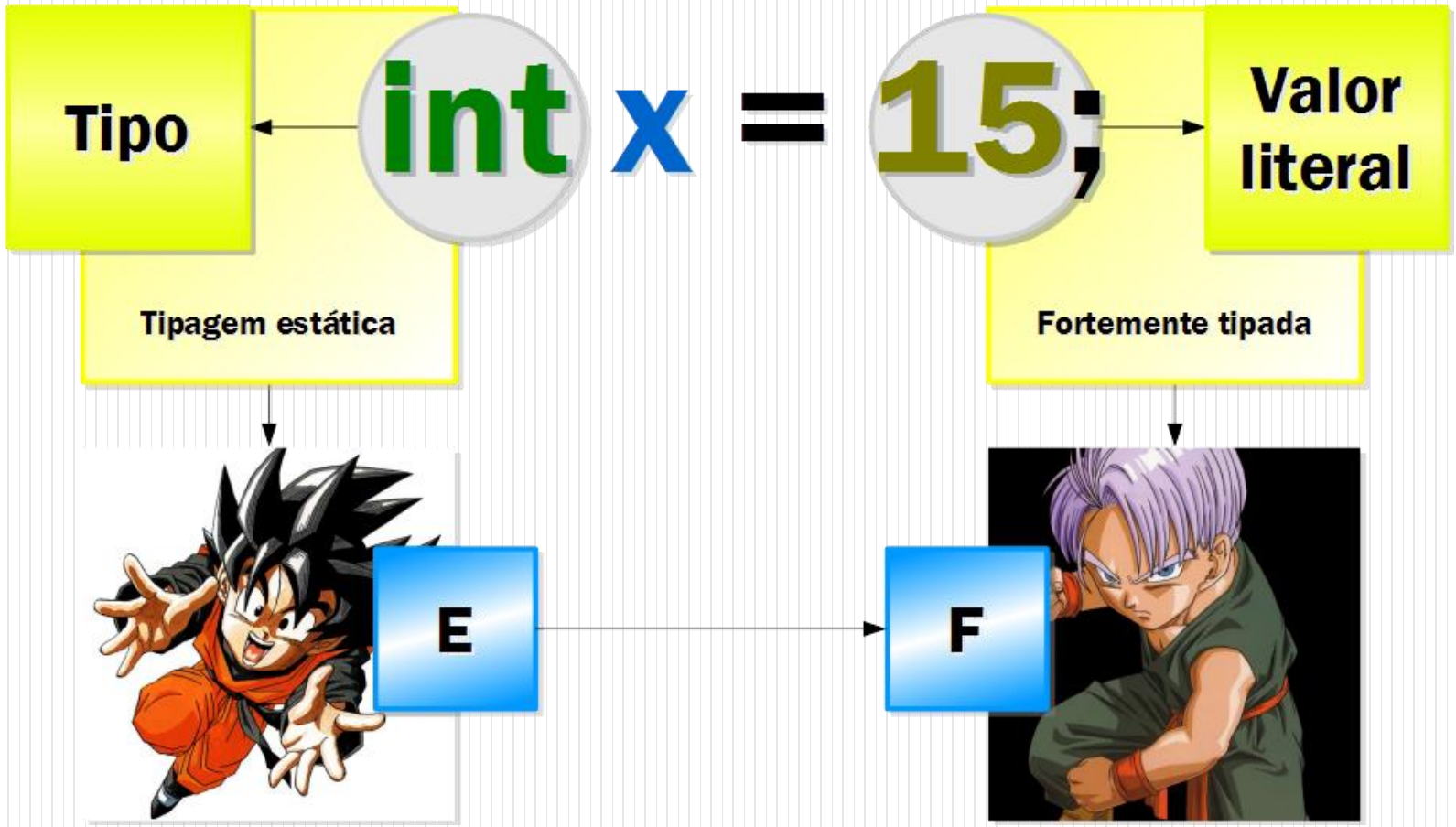
Exemplos

int x = 15.7;	int x = (int) 15.7;	int x = “15”;
Há erro, pois são o valor é incompatível com o tipo	Usa-se casting, mas há perda dos decimais	Há erro de compilação que nem o casting salva

Esquema


Tipo
int x = 15;
Valor literal
Estaticamente tipada
Fortemente tipada
Tipagem estaticamente forte

Esquema


Tipo
int x = 15;
Valor literal
Tipagem estática
Fortemente tipada
E
F

Questões de concursos

[Quadrix 2021 CRECI 14ª Região – Analista de TI] A respeito da linguagem de programação Java, julgue o item.

- A linguagem Java é amplamente conhecida como uma linguagem fracamente tipada, tendo em vista que ela não requer que todas as variáveis tenham um tipo.

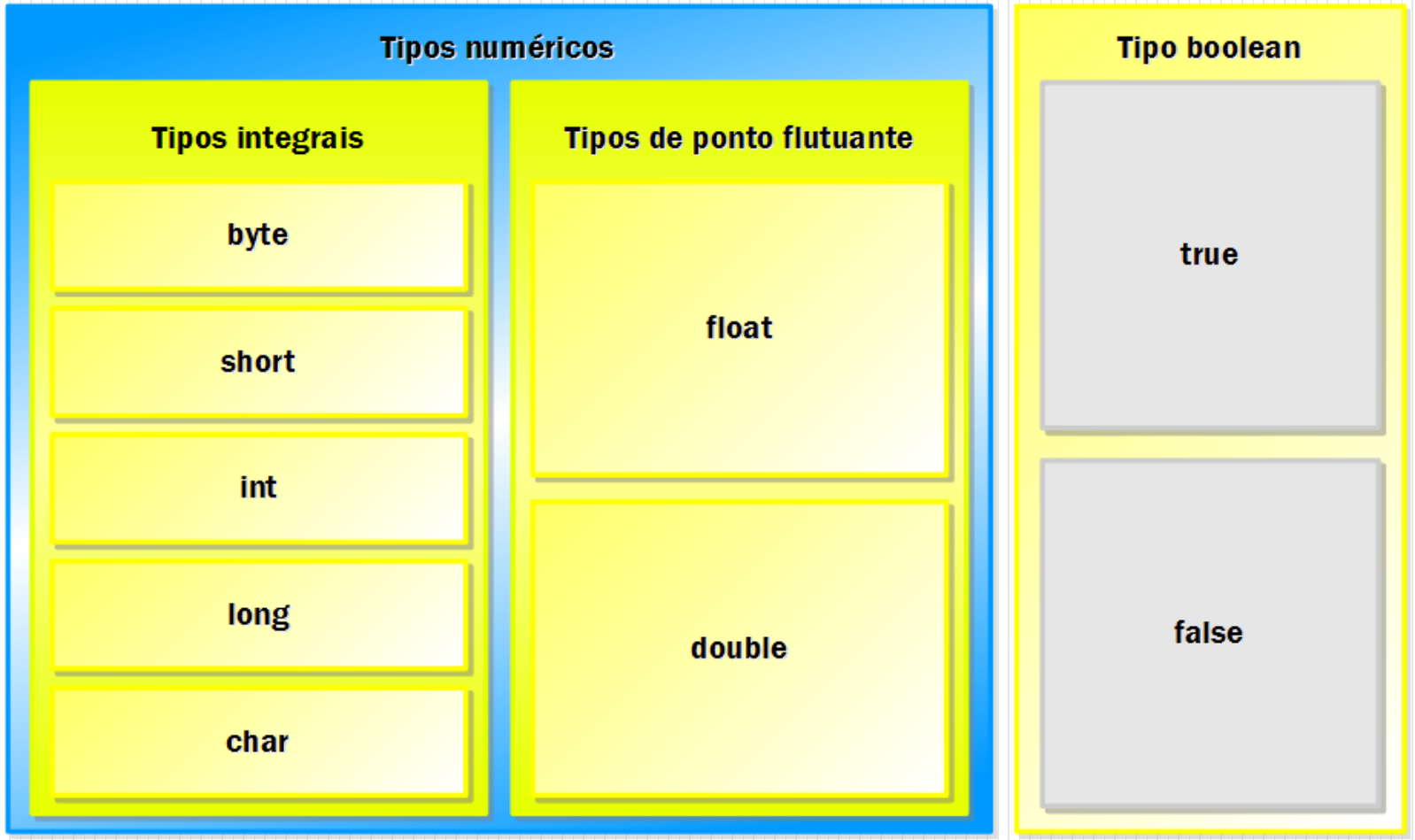
Questões de concursos

[Quadrix 2021 CRECI 14ª Região – Analista de TI] A respeito da linguagem de programação Java, julgue o item.

- A linguagem Java é amplamente conhecida como uma linguagem ~~**fracamente**~~ **estaticamente** tipada, tendo em vista que ela ~~**não**~~ requer que todas as variáveis tenham um tipo.
  - Gabarito: **ERRADO**.

Tipos primitivos

8 tipos primitivos



8 tipos primitivos

Tipo	Bits	Possui sinal	Tipo de valor	Faixa	Valor padrão
byte	8	Sim	Inteiro	-2^7 a $2^7 - 1$ (-128 a 127)	0
short	16	Sim	Inteiro	-2^{15} a $2^{15} - 1$ (-32.768 a 32.767)	0
int	32	Sim	Inteiro	-2^{31} a $2^{31} - 1$ (-2.147.483.648 a 2.147.483.647)	0
long	64	Sim	Inteiro	-2^{63} a $2^{63} - 1$ (-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807)	0L
boolean	1?	Não	Boolean	true ou false	False
char	16	Não	Caracteres UNICODE	'\u0000' a '\uffff'	'u\0000'
float	32	Sim	Ponto flutuante	+/-3.4E-38 a +/-3.4E+38	0.0f
double	64	Sim	Ponto flutuante	+/-1.7E-308 a +/-1.7E+308	0.0

Intervalos dos tipos numéricos integrais

PSIU

BSIL

1329

byte

-128 a 127

short

-32.768 a 32.767

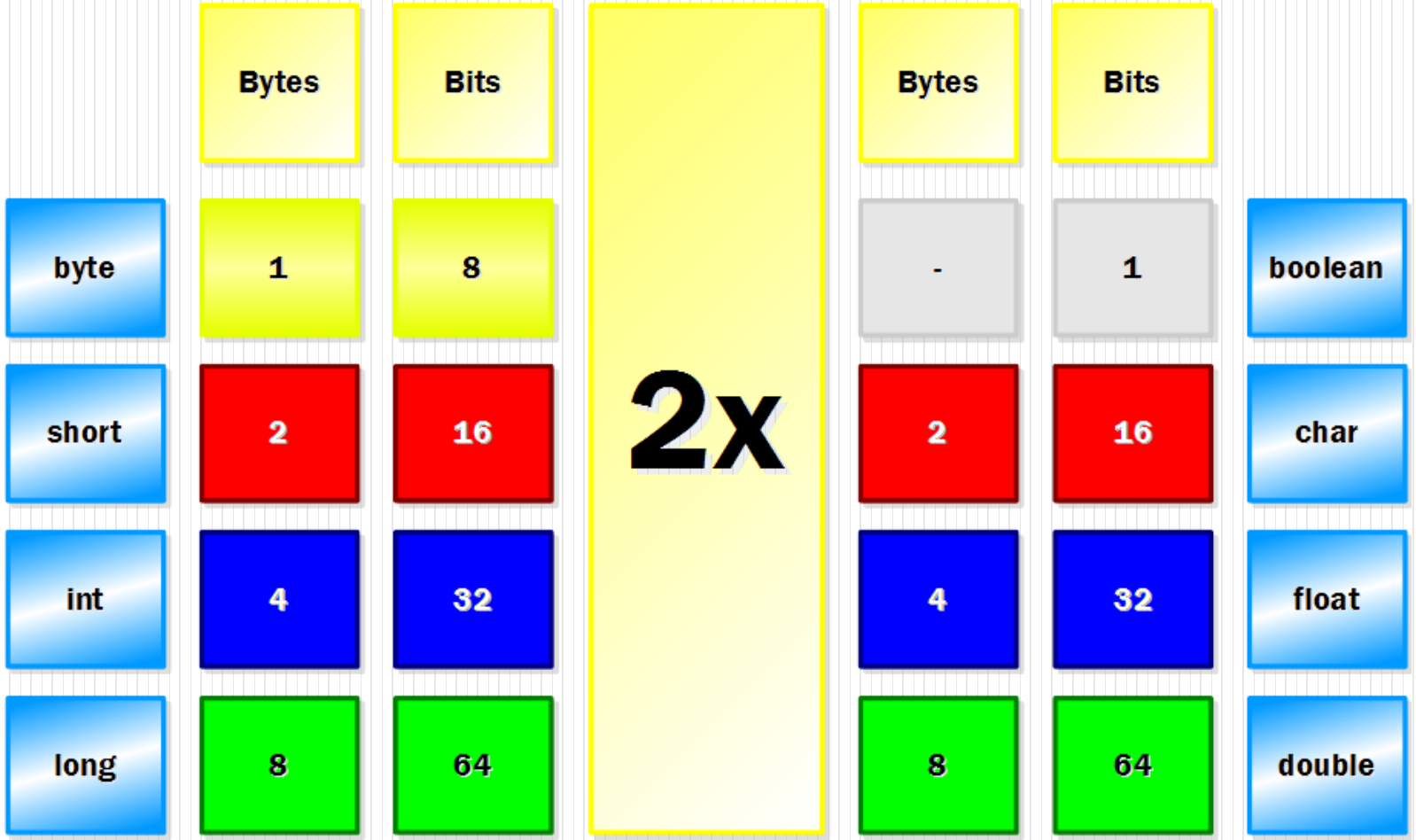
int

-2.147.483.648 a 2.147.483.647

long

-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807

Intervalos dos tipos numéricos integrais



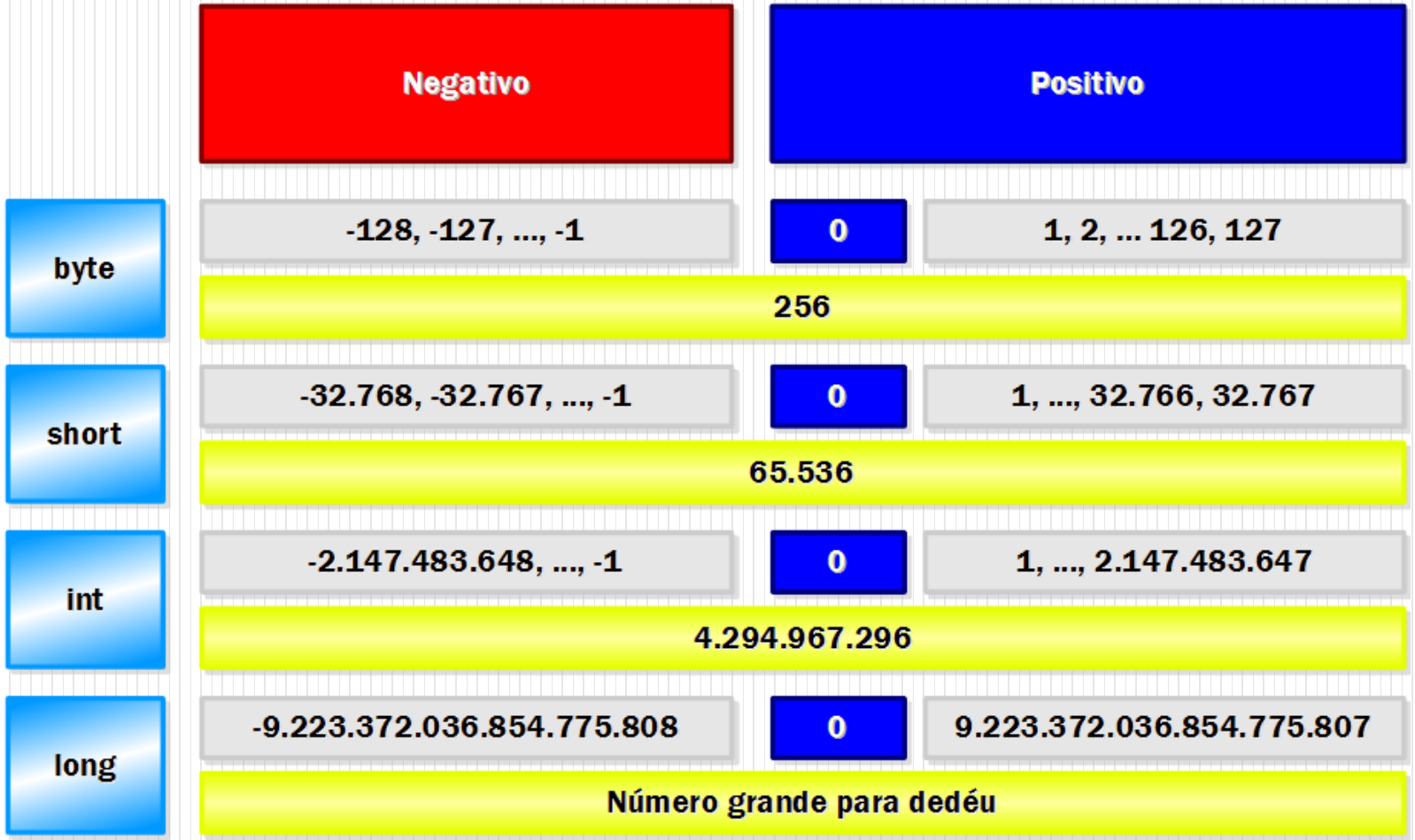
Intervalos dos tipos numéricos integrais

	Bits	Números negativos		Números positivos	
		Bit sinal	Bits do valor	Bit sinal	Bits do valor
byte	8	1	7	0	7
short	16	1	15	0	15
int	32	1	31	0	31
long	64	1	63	0	63

Intervalos dos tipos numéricos integrais

	Bits	Valor limite negativo	Valor limite positivo
		$-2^{\text{bits do valor}}$	$2^{\text{bits do valor}} - 1$
byte	8	$-2^7 = -128$	$2^7 - 1 = 127$
short	16	$-2^{15} = -32.768$	$2^{15} - 1 = 32.767$
int	32	$-2^{31} = -2.147.483.648$	$2^{31} - 1 = 2.147.483.647$
long	64	$-2^{63} =$ -número grande	$2^{63} - 1 =$ número grande

Intervalos dos tipos numéricos integrais



Operadores

Operadores unários

Operadores

Mais +	• Indica valor positivo • Não é necessário usá-lo em números positivos
Menos -	• Nega uma expressão ou um número
Incremento ++	• Incrementa um valor em um
Decremento --	• Diminui um valor em um
Complemento lógico !	• Inverte o valor de um boolean

Operadores de incremento e decremento

Pré-fixados

É avaliado o valor incrementado ou decrementado de uma variável

O valor trabalhado já é o alterado

Incrementado ou decrementado

Pós-fixados

É avaliado o valor original

O valor trabalhado é o valor original de uma variável

Depois o valor da variável é alterado

Incrementado ou decrementado

Operadores de incremento e decremento

Pré-fixados

```
public class PreFixados {
    public static void main(String []args) {
        int x = 15, y = 15;
        System.out.println("x: " + ++x);
        // x: 16
        System.out.println("y: " + --y);
        // y: 14
    }
}
```

Pós-fixados

```
public class PosFixados {
    public static void main(String []args) {
        int x = 15;
        System.out.println("x: " + x++);
        // x: 15
        System.out.println("x: " + x);
        // x: 16
    }
}
```

Questões de concursos

[IDIB 2020 CRM/MT – Técnico em Informática] A linguagem de programação Java foi desenvolvida na década de 90, permanecendo até hoje como umas das linguagens de programação orientadas a objeto mais utilizadas em todo o mundo. A respeito das operações matemáticas que podemos fazer com Java, analise o código fonte abaixo e assinale a alternativa que indica corretamente o valor que será exibido em tela após sua execução.

Questões de concursos

[IDIB 2020 CRM/MT – Técnico em Informática]

```
public class Main
{
        public static void main(String[] args) {
                int a = 0, b = 0, c = 0;
                a += 3;
                b = a++ + ++b;
                c = b % a;
                System.out.println(a+b+c);
        }
}
```

Comentários

- Código da questão:
  - public class Main {
    - public static void main(String []args) {
      - int a = 0, b = 0, c = 0;
      - a += 3;
      - b = a++ + ++b;
      - c = b % a;
      - System.out.println(a + b + c);
    - }
  - }

- Resultado da execução:
  - 8

Questões de concursos

[IDIB 2020 CRM/MT – Técnico em Informática] A respeito das operações matemáticas que podemos fazer com Java, analise o código fonte abaixo e assinale a alternativa que indica corretamente o valor que será exibido em tela após sua execução.

- [A] 1
- [B] 2
- [C] 8
- [D] 4

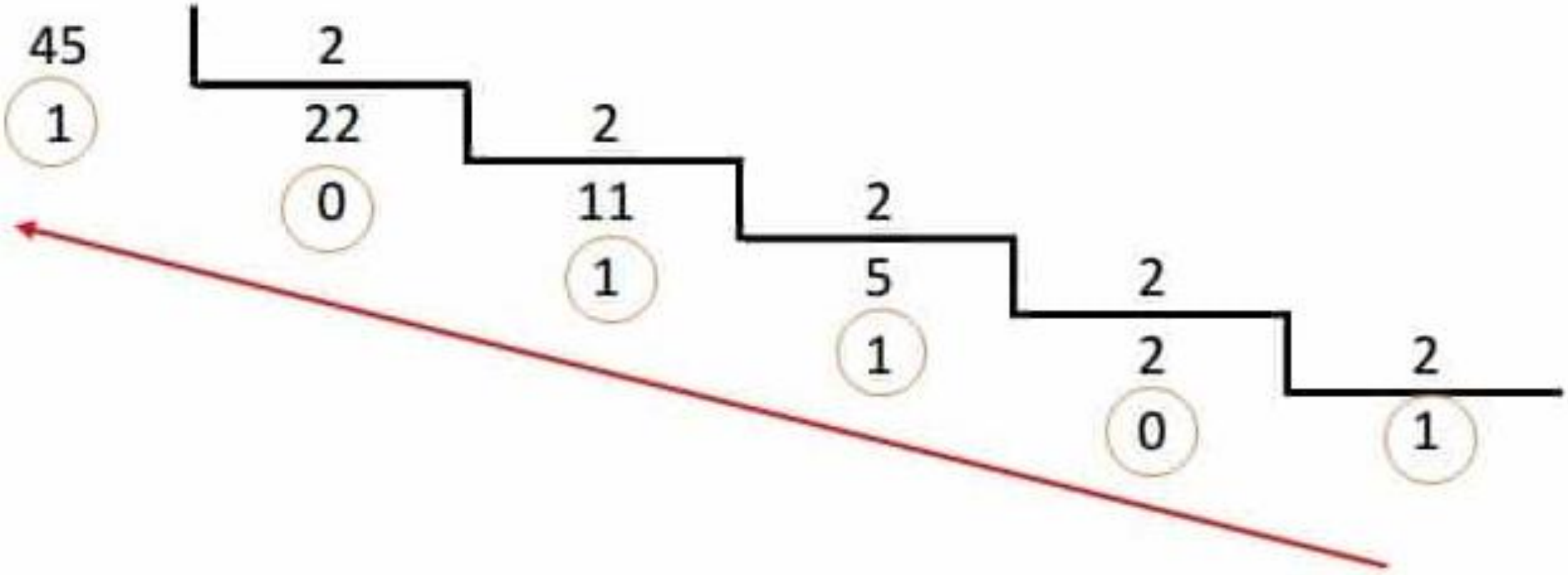
Questões de concursos

[IDIB 2020 CRM/MT – Técnico em Informática] A respeito das operações matemáticas que podemos fazer com Java, analise o código fonte abaixo e assinale a alternativa que indica corretamente o valor que será exibido em tela após sua execução.

- [A] 1
- [B] 2
- **[C] 8**
- [D] 4

Operadores de bits

Transformação da base 10 para base 2



Transformação da base 10 para base 2

Valor incial	128	64	32	16	8	4	2	1	Valor final
93	**0**	1	0	1	1	1	0	1	**93**
20	**0**	0	0	1	0	1	0	0	**20**
13	**0**	0	0	0	1	1	0	1	**13**
-21	**1**	1	1	0	1	0	1	1	**-21**

Números binários negativos

- A **linguagem Java utiliza** a **notação de complemento de dois**
- -21 (10) representa 11101011 (2) em byte:
  - Transformar-se o número negativo para positivo
    - 21
  - Transformar-se o número positivo da base 10 para a base 2
    - 00010101
  - Inverte-se os bits
    - 11101010
  - Soma-se 1 aos bits
    - 11101011

Números binários negativos

- A **linguagem Java utiliza** a **notação de complemento de dois**
- 11101011 (2) em byte representa -21 (10):
  - O bit de sinal é 1
    - Então é um número negativo
  - Diminui-se 1 dos bits
    - 11101010
  - Inverte-se os bits
    - 00010101
  - Transformar-se o número de base 2 para base 10
    - 21
  - Conclui-se que 11101011 (2) é -21 (10)

Operadores de bits

Complemento ~

AND
&

OR
|

XOR
^

Deslocamento à esquerda
<<

Deslocamento à direita
>>

Deslocamento à direita sem sinal
>>>

Operador complemento ~

- **Inverte** o **bits** de um **número**
- Exemplo:
  - byte x = **~**20;
    - x recebe -21
- **Operador unário complemento lógico !**
  - **Inverte** o **valor** de um **boolean**

Operador complemento ~

Valor incial	128	64	32	16	8	4	2	1	Valor final
20	**0**	0	0	1	0	1	0	0	**20**
~20	**1**	1	1	0	1	0	1	1	**-21**

Operadores AND &, OR | e XOR ^

- **Operam** sobre **literais inteiros**
- Se os **operandos** forem **booleanos**:
  - O **resultado** será:
    - **Igual** ao obtido com **operadores**:
      - **AND condicional &&**
      - **OR condicional ||**
    - Mas **sem curto-circuito**
  - Todos os **operandos** serão **avaliados**
    - Mesmo sem necessidade
- **Possuem apenas** um **símbolo cada operador**
  - Os **operadores condicionais AND && e OR || possuem dois símbolos cada**

Operadores AND &, OR | e XOR ^

Valor incial	128	64	32	16	8	4	2	1	Valor final
93	**0**	1	0	1	1	1	0	1	**93**
-21	**1**	1	1	0	1	0	1	1	**-21**
AND &	**0**	1	0	0	1	0	0	1	**73**
OR I	**1**	1	1	1	1	1	1	1	**-1**
XOR ^	**1**	0	1	1	0	1	1	0	**-74**

Operador de descolamento à esquerda

- **Operador de descolamento à esquerda <<**
- **Desloca**:
  - Os **bits** do **primeiro operando**
  - Para **esquerda**
  - Pelo **número especificado** pelo **segundo operando**
- **Preenche** na **direita** com **zero**

Operador de descolamento à esquerda

- Exemplo:
  - byte x = 13 << 2; // x recebe 52.
    - 13: **0**00000000000000000000000**0**0001101
    - 52: **0**00000000000000000000000**0**0110100
  - byte y = -21 << 2; // y recebe -84.
    - -21: **1**11111111111111111111111**1**1101011
    - -84: **1**11111111111111111111111**1**0101100

Operador de descolamento à esquerda

Valor incial	128	64	32	16	8	4	2	1	Valor final
13	**0**	0	0	0	1	1	0	1	**13**
13 << 2	**0**	0	1	1	0	1	0	0	**52**
-21	**1**	1	1	0	1	0	1	1	**-21**
-21 << 2	**1**	0	1	0	1	1	0	0	**-84**

Operador de descolamento à direita

- **Operador de descolamento à direita >>**
- **Desloca**:
  - Os **bits** do **primeiro operando**
  - Para **direita**
  - Pelo **número especificado** pelo **segundo operando**
- **Preenche** na **esquerda** com **zero** ou **um**
  - **Dependendo** do **bit de sinal**

Operador de deslocamento à direita

- Exemplo:
  - byte x = 13 >> 2; // x recebe 3.
    - 13: 00000000000000000000000000001101
    - 03: 00000000000000000000000000000011
  - byte y = -21 >> 2; // y recebe -6.
    - -21: 11111111111111111111111111101011
    - -06: 11111111111111111111111111111010

Operador de deslocamento à direita

Valor incial	128	64	32	16	8	4	2	1	Valor final
13	**0**	0	0	0	1	1	0	1	**13**
13 >> 2	**0**	0	0	0	0	0	1	1	**3**
-21	**1**	1	1	0	1	0	1	1	**-21**
-21 >> 2	**1**	1	1	1	1	0	1	0	**-6**

Operador de deslocamento à direita sem sinal

- **Operador de deslocamento à direita sem sinal >>>**
- **Desloca**:
  - Os **bits** do **primeiro operando**
  - Para **direita**
  - Pelo **número especificado** pelo **segundo operando**
- **Preenche** na **esquerda** com **zero**
  - **Independentemente** do **bit de sinal**

Operador de deslocamento à direita sem sinal

- Exemplo:
  - byte x = 13 >>> 2; // x recebe 3.
    - 13: **0**00000000000000000000000**0**000**1101**
    - 03: **00**00000000000000000000000**0**0000**11**
  - byte y = (byte) (-21 >>> 2); // y recebe -6.
    - -21: **1**11111111111111111111111**1**11**01011**
    - 1073741818 (byte, -6): : **00**11111111111111111111111**1**1111**010**

Operador de deslocamento à direita sem sinal

Valor incial	128	64	32	16	8	4	2	1	Valor final
13	**0**	0	0	0	1	1	0	1	**13**
13 >>> 2	**0**	0	0	0	0	0	1	1	**3**
-21	**1**	1	1	0	1	0	1	1	**-21**
-21 >>> 2	**1**	1	1	1	1	0	1	0	**-6**

Questões de concursos

[FGV 2021 IMBEL – Supervisor – Tecnologia de Informação] Com relação aos operadores bitwise do Java, considere os valores binários

- a = 00111100
- b = 00001101

Questões de concursos

[FGV 2021 IMBEL – Supervisor – Tecnologia de Informação] Os valores resultantes das operações a&b e a|b são, respectivamente,

- [A] 00011100 e 11111101
- [B] 00001100 e 00111101
- [C] 00001111 e 00111111
- [D] 11001110 e 00001100
- [E] 01101100 e 00100101

Comentários

Valor incial	128	64	32	16	8	4	2	1	Valor final
60	**0**	0	1	1	1	1	0	0	**60**
13	**0**	0	0	0	1	1	0	1	**13**
AND &	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**12**
OR I	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**61**
XOR ^	**0**	0	1	1	0	0	0	1	**49**

Questões de concursos

[FGV 2021 IMBEL – Supervisor – Tecnologia de Informação] Os valores resultantes das operações a&b e a|b são, respectivamente,

- [A] 00011100 e 11111101
- **[B] 00001100 e 00111101**
- [C] 00001111 e 00111111
- [D] 11001110 e 00001100
- [E] 01101100 e 00100101

Controle de fluxo

Declarações de controle de fluxo

Conceituação

Conceituação

- **Controlam** a **ordem** que **outras declarações** serão **executadas**
- Exemplo:
  - public class ControleFluxo {
    - public static void main(String []args) {
      - int i = 0;
      - while (i < 10) {
        - System.out.println(i);
        - i++;
      - }
    - }
  - }

Estruturas de condição e de repetição

Estruturas de condição

- if-else
- switch

Estruturas de repetição

- while
- do-while
- for

Estruturas de quebra de controle de fluxo

break

continue

return

Estruturas de condição

if-else

if

- É a **declaração mais comum**
  - Dentre todas as declarações de controles de fluxos
- **Instruções dentro dela**
  - Serão **executadas** se a **condição da estrutura** for **true**
- **Condição da estrutura**
  - **Deve**:
    - **Vir** entre **parênteses**
    - **Retornar** um **valor booleano**

if

- Código:
  - public class ExemploIf {
    - public static void main(String []args) {
      - int x = 15;
      - if (x <= 20)
        - System.out.println("x <= 20");
    - }
  - }

- Resultado da execução:
  - x <= 20

else

- É **opcional**
- É **correspondente** com o **if mais próximo**
  - Exemplo:
    - int x = 15;
    - **if** (x <= 15) **if** (x == 15) y = 0; **else** y = 1;
- **Utilização de chaves**
  - É uma **boa prática** para **organizar o código**
  - Exemplo:
    - **if** (x <= 15) { **if** (x == 15) y = 0; **else** y = 1; }

else

- **Identação do código**
  - **Melhora** ainda mais a **visualização do código**
  - Exemplo:
    - if (x <= 15) {
      - if (x == 15)
        - y = 0;
      - else
        - y = 1;
    - }

else

- Código:
  - public class ExemploElse {
    - public static void main(String []args) {
      - int x = 21;
      - if (x <= 20)
        - System.out.println("x <= 20");
      - else
        - System.out.println("x > 20");
    - }
  - }

- Resultado da execução:
  - x > 20

switch

Conceituação

- **Múltiplas seleções ou escolhas**
- É uma **declaração mais fácil** de **entender** e **manter** do que uma **declaração if/else** com **várias condições** com **muitas alternativas**
- **Corpo da declaração**
  - **Bloco switch**
  - É **composto** por:
    - **Um** ou **mais rótulos cases**
      - **Trabalham** os **casos específicos**
    - Apenas **um rótulo default**
      - **Trabalham** os **casos não específicos**

Exemplo

- Código:
  - public class ExemploSwitch {
    - public static void main(String []args) {
      - int x = 15;
      - switch(x) {
        - case 14:
          - System.out.println("x = 14");
          - break;
        - case 15:
          - System.out.println("x = 15");
          - break;
        - default:
          - System.out.println(x);
      - }
    - }
  - }

- Resultado da execução:
  - x = 15

Tipos trabalhados

Tipos primitivos

- byte
- short
- char
- int

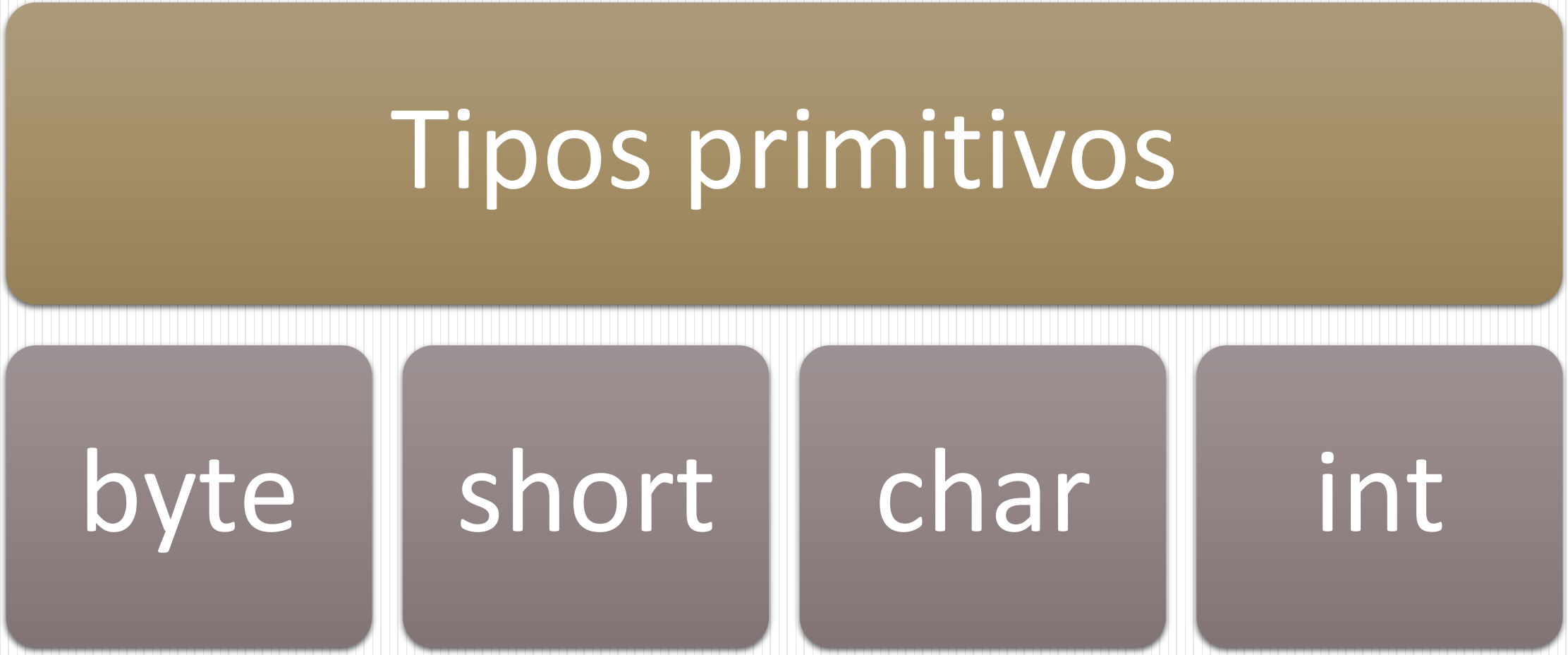
Classes wrappers

- Byte
- Short
- Character
- Integer

Outros tipos

- Strings
- Tipos enumerados

Tipos trabalhados até ao Java 6



if/else x switch

- **Declaração if/else**
  - **Pode testar expressões** com base em **faixas de valores ou condições**
- **Declaração switch**
  - **Testa expressões** com base apenas em **um único valor dos tipos**:
    - Tipos primitivos:
      - byte, short, char e int
    - Classes wrappers:
      - Byte, Short, Character e Integer
    - String
    - Tipos enumerados

Questões de concursos

[IDECAN 2021 PEFOCE – Análise de Sistemas Ciências da Computação] No que diz respeito à linguagem de programação Java, a estrutura switch-case equivale a um conjunto de instruções if encadeadas, fornecendo maior inteligibilidade e eficiência durante a execução.

Questões de concursos

[IDECAN 2021 PEFOCE – Análise de Sistemas Ciências da Computação]
Assinale a alternativa que apresente corretamente sua sintaxe.

- [A]

```
case (<instruções>)
{
  switch 1 : instruções; break;
  switch 2 : instruções; break;
  switch 3 : instruções; break;
  else: instruções;
}
```

- [B]

```
case (<instruções>)  switch
{
  1 : instruções; break;
  2 : instruções; break;
  3 : instruções; break;
  default: instruções;
}
```

Questões de concursos

[IDECAN 2021 PEFOCE – Análise de Sistemas Ciências da Computação]
Assinale a alternativa que apresente corretamente sua sintaxe.

- [C]

```
switch case (<instruções>)
{
  1 : instruções; break;
  2 : instruções; break;
  3 : instruções; break;
  otherwise: instruções;
}
```

- [D]

```
switch (<instruções>)  case
{
  1 : instruções; break;
  2 : instruções; break;
  3 : instruções; break;
  else: instruções;
}
```

Questões de concursos

[IDECAN 2021 PEFOCE – Análise de Sistemas Ciências da Computação]
Assinale a alternativa que apresente corretamente sua sintaxe.

- [E]

```
switch (<instruções>)
{
  case 1 : instruções; break;
  case 2 : instruções; break;
  case 3 : instruções; break;
  default: instruções;
}
```

Questões de concursos

[IDECAN 2021 PEFOCE – Análise de Sistemas Ciências da Computação]
Assinale a alternativa que apresente corretamente sua sintaxe.

- **[E]**

```
switch (<instruções>)
{
  case 1 : instruções; break;
  case 2 : instruções; break;
  case 3 : instruções; break;
  default: instruções;
}
```

Questões de concursos

[Quadrix 2021 CRECI 14ª Região – Analista de TI] A respeito da linguagem de programação Java, julgue o item.

- As instruções if, if...else e *switch* são exemplos de instruções de seleção contidas na linguagem Java.

Questões de concursos

[Quadrix 2021 CRECI 14ª Região – Analista de TI] A respeito da linguagem de programação Java, julgue o item.

- As instruções if, if...else e *switch* são exemplos de instruções de seleção contidas na linguagem Java.
  - Gabarito: **CERTO**.

Estruturas de repetição

while

Conceituação

- **Executa** uma **instrução** ou um **conjunto de instruções enquanto** uma **condição for verdadeira**
- **Condição da estrutura**
  - É **feita** no **início** da **declaração**
  - **Deve**:
    - **Vir** entre **parênteses**
    - **Retornar** um **valor booleano**

Exemplo

- Código:
  - public class ExemploWhile {
    - public static void main(String []args) {
      - int x = 1;
      - while (x < 11) {
        - System.out.print(x + " ");
        - x++;
      - }
    - }
  - }

- Resultado da execução:
  - 1 2 3 4 5 6 7 8 9 10

do-while

Conceituação

- **Executa** uma **instrução** ou um **conjunto de instruções enquanto** uma **condição for verdadeira**
- **Condição da estrutura**
  - É **feita** no **final** da declaração
  - **Deve**:
    - **Vir** entre **parênteses**
    - **Retornar** um **valor booleano**

Exemplo

- Código:
  - public class ExemploDoWhile {
    - public static void main(String []args) {
      - int x = 1;
      - do {
        - System.out.print(x + " ");
        - x++;
      - } while (x < 11);
    - }
  - }

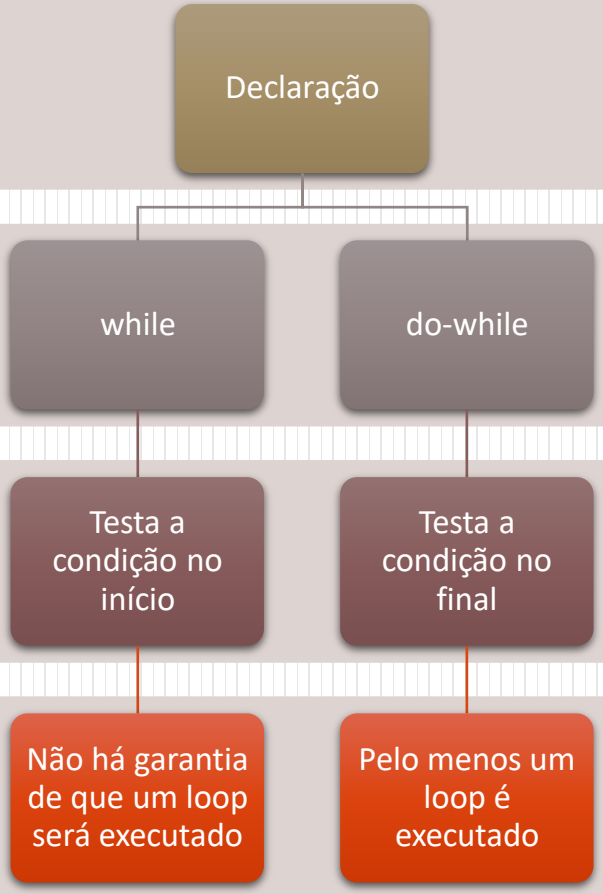
- Resultado da execução:
  - 1 2 3 4 5 6 7 8 9 10

while x do-while

Declaração

Diferença

Consequência



Questões de concursos

[CESPE/CEBRASPE 2021 SEED/PR – Professor – Educação Básica e Jornada] Em Java, a estrutura de repetição que permite que um conjunto de instruções não seja executada nenhuma vez é representada por

- [A] while.
- [B] switch.
- [C] do...while.
- [D] case.
- [E] continue.

Questões de concursos

[CESPE/CEBRASPE 2021 SEED/PR – Professor – Educação Básica e Jornada] Em Java, a estrutura de repetição que permite que um conjunto de instruções não seja executada nenhuma vez é representada por

- **[A] while.**
- [B] switch.
- [C] do...while.
- [D] case.
- [E] continue.

Questões de concursos

[Quadrix 2021 CRECI 14ª Região – Analista de TI] A respeito da linguagem de programação Java, julgue o item.

- O controle de *loops* com variáveis de ponto flutuantes pode resultar em valores de contador imprecisos. Para evitar tais situações, devem ser usados números inteiros para se controlar os loops de contagem.

Comentários

- Erro comum de programação
  - Como os valores em ponto flutuante podem ser valores aproximados
    - Controlar a contagem de loops com variáveis de ponto flutuante pode resultar em:
      - Valores imprecisos de contadores e exames incorretos da condição de terminação
- Boa prática de programação
  - Controle a contagem das repetições (loops) com valores inteiros

Questões de concursos

[Quadrix 2021 CRECI 14ª Região – Analista de TI] A respeito da linguagem de programação Java, julgue o item.

- O controle de *loops* com variáveis de ponto flutuantes pode resultar em valores de contador imprecisos. Para evitar tais situações, devem ser usados números inteiros para se controlar os loops de contagem.
  - Gabarito: **CERTO**.

for

Conceituação

- É uma **declaração** que **fornece** uma **forma compacta** para **repetir** uma **faixa de valores**
- Sintaxe:
  - **for** (**inicialização**; **condição**; **incremento**) {
    - **Instruções**;
  - }

Sintaxe

Expressão de inicialização	• Armazena a inicialização de um contador do loop • É executada apenas uma vez
Expressão de condição	• Contém a condição a ser testada antes de cada nova passagem no loop
Expressão de incremento	• Atualiza o contador
Pode suportar mais de um contador	• Porém, fica uma estrutura difícil de se manter

Exemplo

- Código:
  - public class ExemploFor {
    - public static void main(String []args) {
      - for (int x = 1; x < 11; x++) {
        - System.out.print(x + " ");
      - }
    - }
  - }

- Resultado da execução:
  - 1 2 3 4 5 6 7 8 9 10

for each

- É **mais compacto** e **melhorado**
- É **Utilizado** para **trabalhar** com:
  - **Collections**
  - **Arrays**
- O **contador** vai **assumir** o **valor de cada componente** de:
  - Uma **collection**
  - Um **array**
- Recomenda-se a sua utilização sempre que possível

for each

- Código:
  - public class ExemploForEach {
    - public static void main(String []args) {
      - String[] meses = {"J", "F", "M"};
      - for (String valor : meses)
        - System.out.print(valor + " ");
      - System.out.println();
      - for (int i = 0; i < meses.length; i++)
        - System.out.print(meses[i] + " ");
    - }
  - }

- Resultado da execução:
  - J F M
  - J F M

for each

- Código:
  - public class ExemploForEach {
    - public static void main(String []args) {
      - List<String> meses = new ArrayList<>();
      - meses2.add("J");
      - meses2.add("F");
      - meses2.add("M");
      - for (String valor : meses)
        - System.out.print(valor + " ");
      - System.out.println();
      - for (int i = 0; i < meses2.size(); i++)
        - System.out.print(meses.get(i) + " ");
    - }
  - }

- Resultado da execução:
  - J F M
  - J F M

Estruturas de quebra de controle de fluxo

break

Conceituação

Termina as declarações mais internas das quais o break faz parte

- swicth
- while
- do-while
- for

Exemplo

- Código:
  - public class ExemploBreak {
    - public static void main(String []args) {
      - for (int x = 1; x < 11; x++) {
        - if (x == 3)
          - break;
        - System.out.print(x + " ");
      - }
    - }
  - }

- Resultado da execução:
  - 1 2 3

Questões de concursos

[CESPE 2020 TJ/PA – Analista Judiciário – Programador]

```
class GeraNumeros {
   public static void main (String args []) {
      int num;
      num = 36;
      for(int i=0; i < num; i++) {
        if(i*i >= num) break;
        System.out.print(i + " ");
      }
    }
}
```

Comentário

- Código:
  - public class GeraNumeros {
    - public static void main(String []args) {
      - int num = 36;
      - for (int i = 0; i < num; i++) {
        - if (i * i >= num)
          - break;
        - System.out.print(i + " ");
      - }
    - }
  - }

- Resultado da execução:
  - 0 1 2 3 4 5

Questões de concursos

[CESPE 2020 TJ/PA – Analista Judiciário – Programador] Assinale a opção que apresenta corretamente a saída gerada pelo código Java precedente.

- [A] 36
- [B] 1 2 3 4 5 6
- [C] 1 2 3 4 5
- [D] 0 1 2 3 4 5
- [E] 0 1 2 3 4 5 6

Questões de concursos

[CESPE 2020 TJ/PA – Analista Judiciário – Programador] Assinale a opção que apresenta corretamente a saída gerada pelo código Java precedente.

- [A] 36
- [B] 1 2 3 4 5 6
- [C] 1 2 3 4 5
- **[D] 0 1 2 3 4 5**
- [E] 0 1 2 3 4 5 6

continue

Conceituação

Salta uma iteração das declarações mais internas das quais o continue faz parte

while

do-while

for

Exemplo

- Código:
  - public class ExemploContinue {
    - public static void main(String []args) {
      - for (int x = 1; x < 11; x++) {
        - if (x == 3)
          - continue;
        - System.out.print(x + ", ");
      - }
    - }
  - }

- Resultado da execução:
  - 1, 2, 4, 5, 6, 7, 8, 9, 10,

Questões de concursos

[SELECON 2021 EMGEPRON – Analista de Sistemas (Desenvolvimento de Sistemas)] Em Java, é uma tecnologia de desenvolvimento, sendo simultaneamente, uma linguagem e uma plataforma. Nesse contexto, apresenta-se o código a seguir.

```
public class Emgepron {
        public static void main (String args[ ]){
          for (int i = 0; i < 4; i++) {
              if (i==2)
                 continue;
              System.out.print(i + "");
          }
        }
}
```

Comentário

- Código:
  - public class Emgepron {
    - public static void main(String []args) {
      - for (int i = 0; i < 4; i++) {
        - if (i == 2)
          - continue;
        - System.out.print(i + " ");
      - }
    - }
  - }

- Resultado da execução:
  - 0 1 3

Questões de concursos

[SELECON 2021 EMGEPRON – Analista de Sistemas (Desenvolvimento de Sistemas)] Após a execução, a saída gerada será:

- [A] 0 1 3
- [B] 0 1 3 4
- [C] 0 1 2 3
- [D] 0 1 2 3 4

Questões de concursos

[SELECON 2021 EMGEPRON – Analista de Sistemas (Desenvolvimento de Sistemas)] Após a execução, a saída gerada será:

- **[A] 0 1 3**
- [B] 0 1 3 4
- [C] 0 1 2 3
- [D] 0 1 2 3 4

return

Conceituação

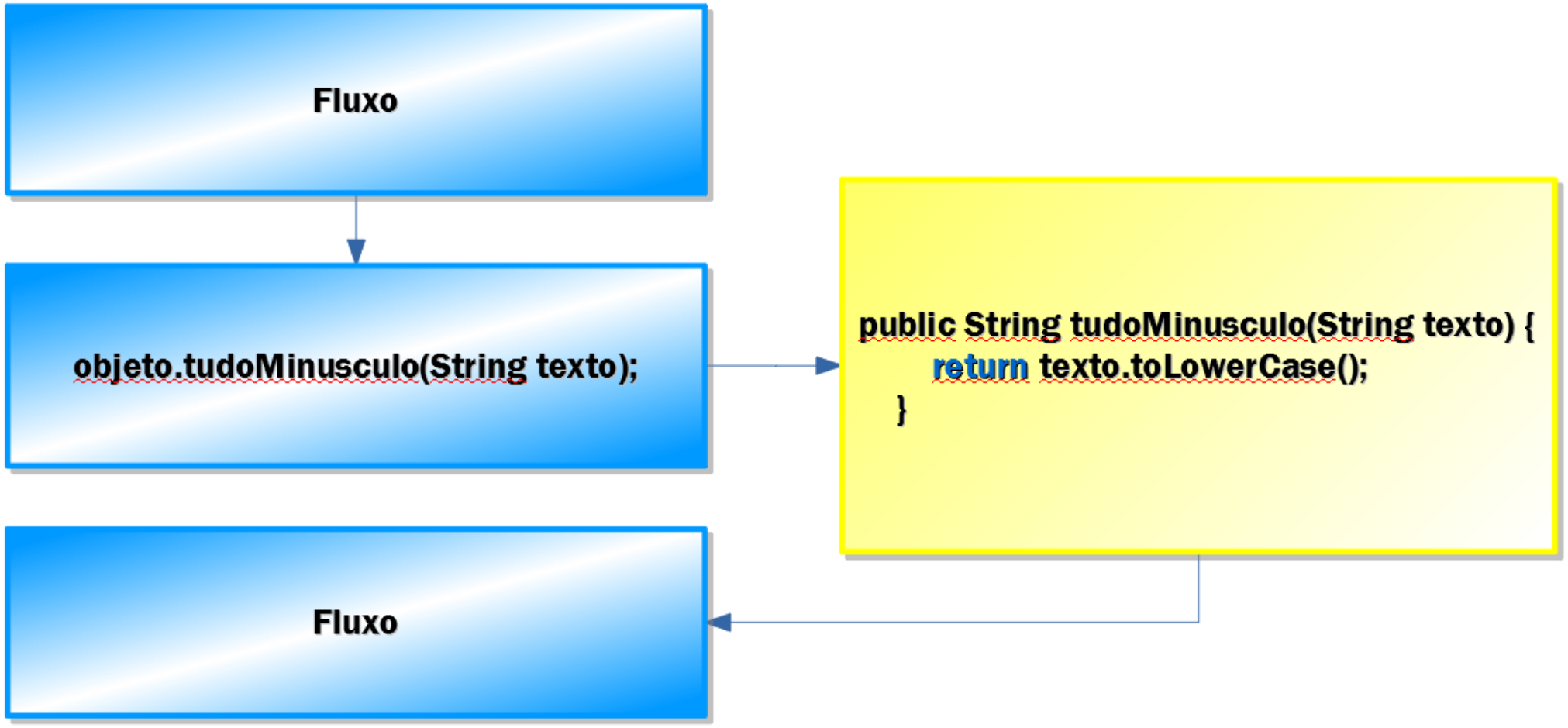
Sai do método atual

O controle de fluxo retorna onde o método foi invocado

O que pode retornar

- **Algum valor**
  - O **tipo do valor retornado deve ser igual** ao **tipo declarado no método**
  - public **String** tudoMinusculo(String texto) {
    - **return texto.toLowerCase()**;
  - }
- **Nada**
  - É **usado** em **métodos declarados** com **void**
    - São métodos que não retornam nenhum valor
  - **Não** é **obrigatório** o **uso** do **return**
  - public void naoFazNada() {
    - **return**;
  - }

Esquema


Fluxo
objeto.tudoMinusculo(String texto);
public String tudoMinusculo(String texto) {
return texto.toLowerCase();
}
Fluxo

Exemplo

- Código:

```
public class Exemplo08_Return {
    // Return que retorna algo.
    public static String tudoMinusculo(String texto) {
        return texto.toLowerCase();
    }

    // Return que retorna nada.
    public static void imprime(String texto) {
        System.out.println(texto);
        return;
    }

    public static void main(String[] args) {
        System.out.println("return ***************************************************");
        System.out.println(tudoMinusculo("ROGÉRIO GILDO ARAÚJO"));
        imprime("ROGÉRIO GILDO ARAÚJO");
    }
}
```

Exemplo

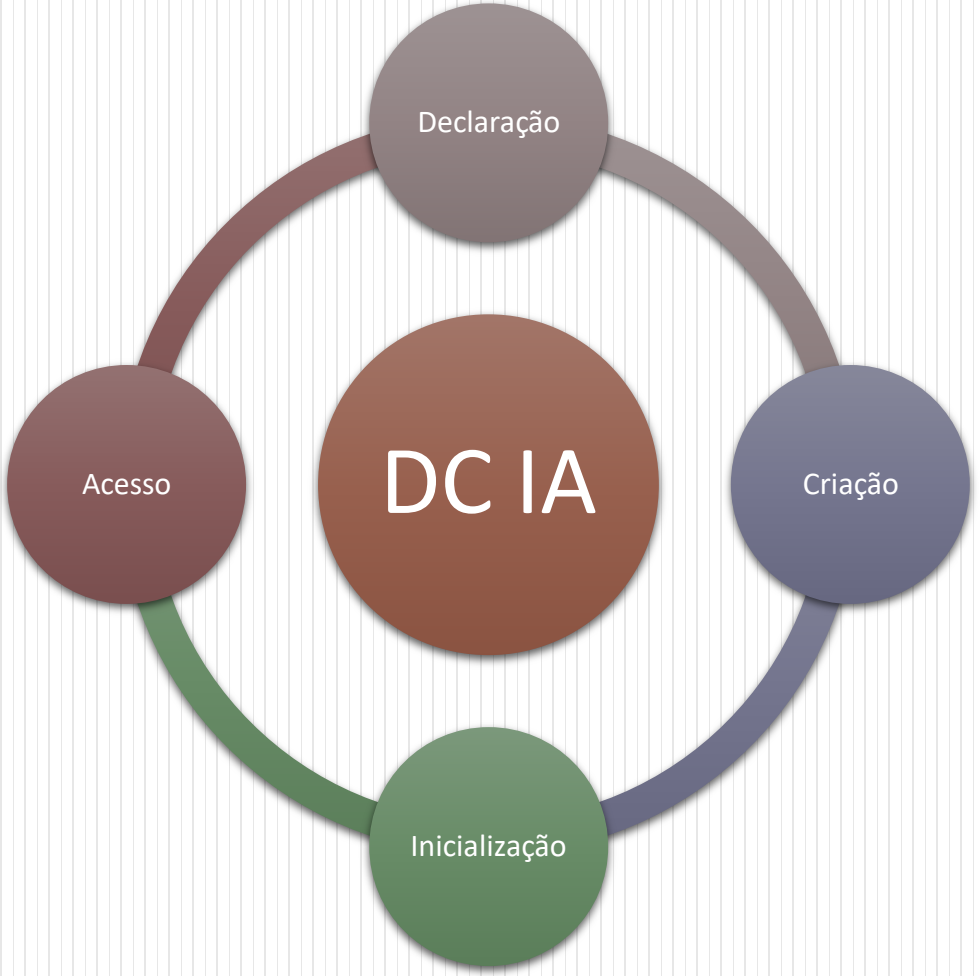
- Resultado da execução:
  - return **********
  - rogério gildo araújo
  - ROGÉRIO GILDO ARAÚJO

Arrays

Conceituação

- São **objetos** que **contém** um **conjunto de valores** do **mesmo tipo**
- **Cada elemento de um array**
  - **Componente de array**
  - É uma **variável**
  - É **acessado** por um **índice**
    - De **0** a **n – 1**
      - Onde **n** é o **tamanho do array**
- **Tamanho de um array**
  - É **estabelecido** quando o **array** é **criado**
  - É **fixo**

Etapas de um array



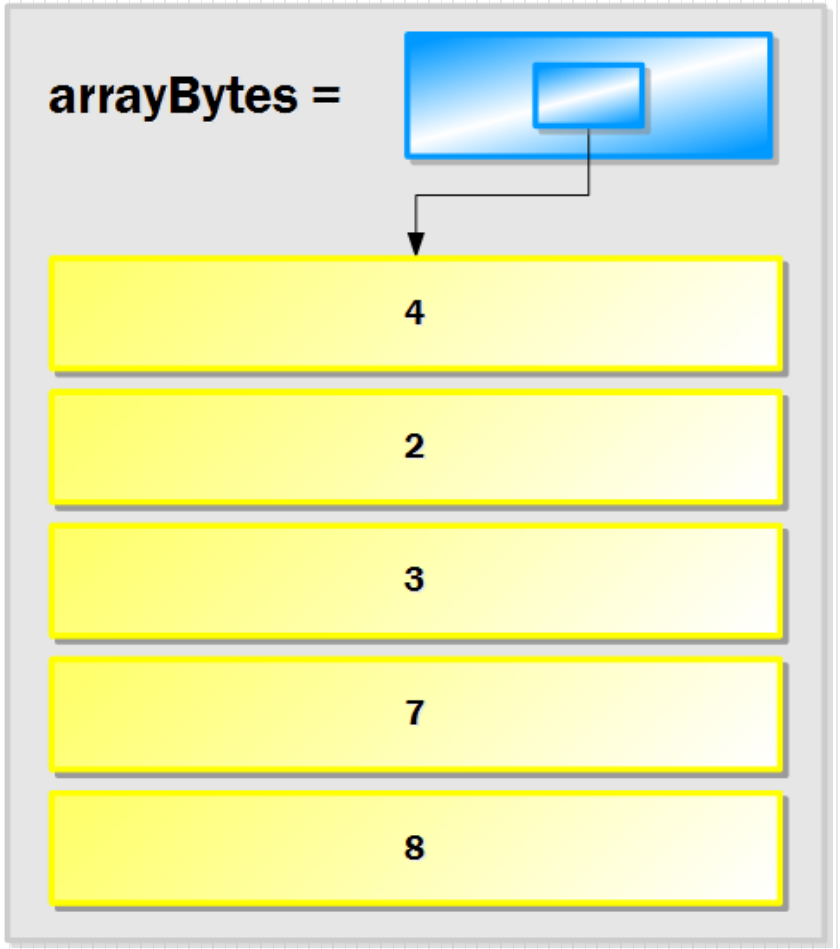
Etapas de um array

```
// Declarando um array.
byte[] arrayBytes;

// Criando o array.
arrayBytes = new byte[5];

// Inicializando o array.
arrayBytes[0] = 4;
arrayBytes[1] = 2;
arrayBytes[2] = 3;
arrayBytes[3] = 7;
arrayBytes[4] = 8;

// Acessando um elemento.
int x = arrayBytes[3];
```



Declaração de um array

- **Colchetes**
  - São **utilizados** na **declaração** de um **array**
- **Duas formas**:
  - **tipo[ ] nomeArray;** // É a forma mais recomendada.
  - **tipo nomeArray[ ];**
- Exemplos:
  - byte[] arrayBytes;
  - Candidato arrayCandidatos[];

Criação de um array

- É **onde**:
  - O **tamanho do array** é:
    - **Definido**
    - **Fixado**
  - Os **componentes do array** são **inicializados automaticamente** com **valores padrões dos tipos**

Criação de um array

Duas formas

Utilizando o operador new

Seguido

Do tipo

Do tamanho do array

Fazendo junção com a fase de inicialização

Criação de um array

- **Utilizando o operador new seguido do tipo e o tamanho do array**
  - **tipo[ ] array = new tipo[n];**
    - Sendo **n** o **tamanho do array**
  - Exemplos:
    - // Array declarado e criado.
    - byte[] arrayBytes = new byte[10];

    - // Array declarado e criado.
    - Candidato[] arrayCandidatos = new Candidato[5];

Criação de um array

- **Fazendo junção com a fase de inicialização**
  - **tipo[ ] array = {valo1, valo2, valo3, …, valorN};**
    - Sendo o **número de valores** o **tamanho do array**
    - Os **valores devem estar entre chaves { }**
  - Exemplo:
    - // Array declarado, criado com tamanho 6 e inicializado.
    - byte[] arrayBytes = {4, 5, 7, 8, 23, 45};

Inicialização de um array

Duas formas

Fazendo junção com a fase de criação

Fazendo junção com a fase de acesso

Inicialização de um array

- **Fazendo junção com a fase de criação**
  - **tipo[ ] array = {valo1, valo2, valo3, …, valorN};**
    - Sendo o **número de valores** o **tamanho do array**
    - Os **valores devem estar entre chaves { }**
  - Exemplo:
    - // Array declarado, criado com tamanho 6 e inicializado.
    - byte[] arrayBytes = {4, 5, 7, 8, 23, 45};

Inicialização de um array

- **Fazendo junção com a fase de acesso**
  - **array**[**0**] = **valo1**;
  - **array**[**1**] = **valor2**;
  - **array**[**n – 1**] = **valorN**; // Sendo n o tamanho do array.
  - Exemplo:
    - byte[] arrayBytes = new byte[2];
    - arrayBytes[0] = 4;
    - arrayBytes[1] = 5;

Acesso aos componentes de um array

- **Cada elemento de um array**
  - **Componente de array**
  - É:
    - Uma **variável**
    - **Acessado** por um **índice**
      - De **0** a **n – 1**
        - Onde **n** é o **tamanho do array**
- **Sintaxe**:
  - **array**[**0**] = **valo1**;
  - **array**[**1**] = **valor2**;
  - **array**[**n – 1**] = **valorN**;

Tamanho de um array

- Trecho de código:
  - byte[] array = {4, 5, 7, 8, 23};
  - System.out.println(array.length);
- Resultado da execução:
  - 5

for each

- Código:
  - public class ExemploForEach {
    - public static void main(String []args) {
      - String[] meses = {"J", "F", "M"};
      - for (String valor : meses)
        - System.out.print(valor + " ");
      - System.out.println();
      - for (int i = 0; i < meses.length; i++)
        - System.out.print(meses[i] + " ");
    - }
  - }

- Resultado da execução:
  - J F M
  - J F M

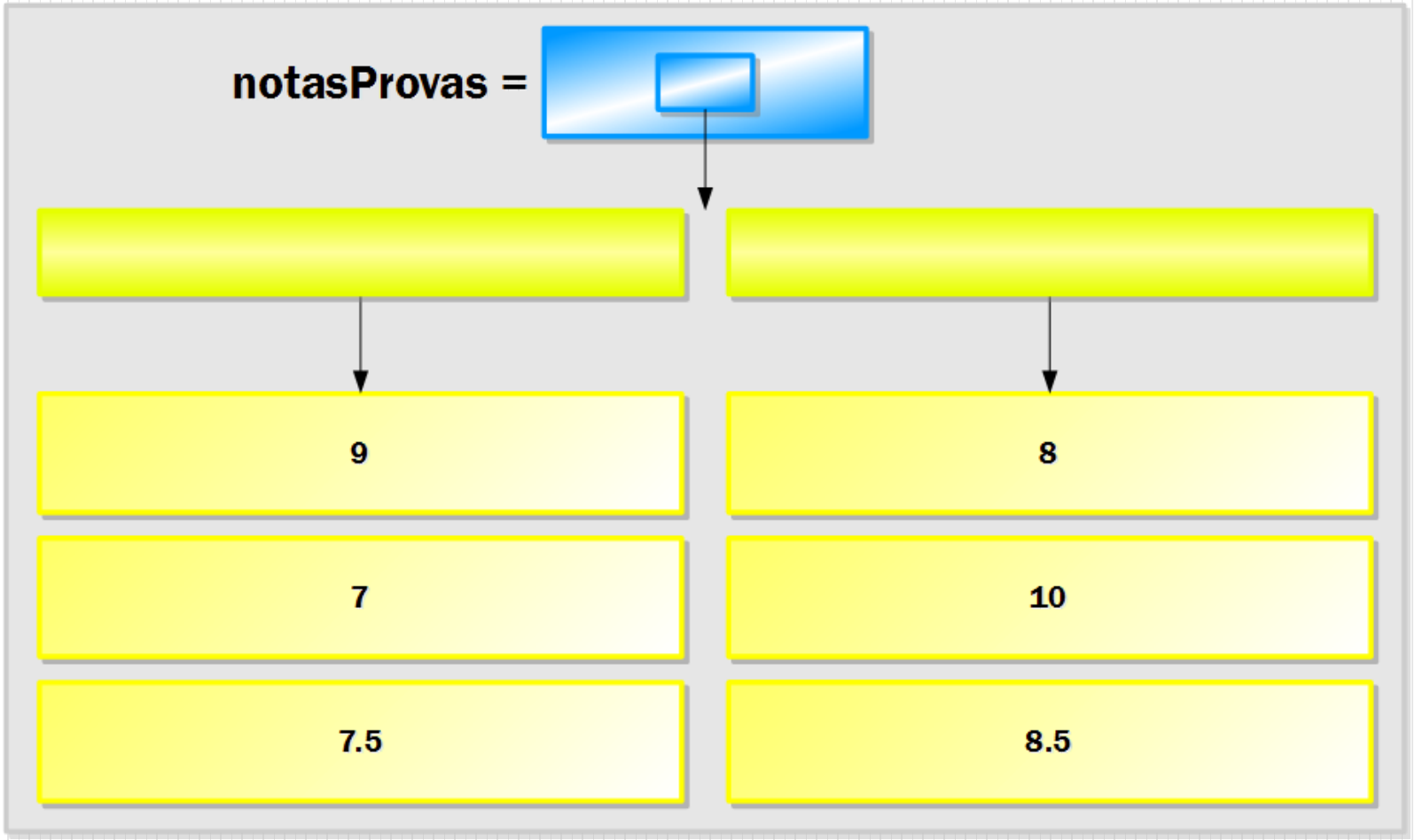
Arrays multidimensionais

- São **arrays** de **arrays**
- **Sintaxe**:
  - **tipo[ ][ ] array = new tipo[n][m];**
    - Onde o **par [ ]** **define** uma **dimensão**

Arrays multidimensionais

- Exemplo:
  - // Dois candidatos e três notas para cada um.
  - double[][] notasProvas = new double[2][3];
  - notasProvas[0][0] = 9;
  - notasProvas[0][1] = 7;
  - notasProvas[0][2] = 7.5;
  - notasProvas[1][0] = 8;
  - notasProvas[1][1] = 10;
  - notasProvas[1][2] = 8.5;

Arrays multidimensionais



Questões de concursos

[Quadrix 2021 CRECI 14ª Região – Analista de TI] A respeito da linguagem de programação Java, julgue o item.

- Na linguagem Java, a criação de um array é realizada por meio da palavra-chave create.

Questões de concursos

[Quadrix 2021 CRECI 14ª Região – Analista de TI] A respeito da linguagem de programação Java, julgue o item.

- Na linguagem Java, a criação de um array é realizada por meio da palavra-chave ~~**create**~~ **new**.
  - Gabarito: **ERRADO**.
  - Ou fazendo junção com a fase de inicialização

Questões de concursos

[UERJ 2021 UERJ – Analista de Tecnologia da Informação] Um tipo de dados define uma coleção de valores de dados e um conjunto de operações pré-definidas sobre ele. O sistema de tipos de uma linguagem de programação define como um tipo é associado com cada expressão na linguagem e inclui suas regras para equivalência e compatibilidade de tipos. Entender seu sistema de tipos é uma das partes mais importantes para entender a semântica de uma linguagem de programação.

Questões de concursos

[UERJ 2021 UERJ – Analista de Tecnologia da Informação] De acordo com essa afirmação e com os conceitos da linguagem de programação Java, é correto afirmar que: (Marque CERTO ou ERRADO o texto da letra)

- [A] o tipo de dado matriz é um agregado homogêneo de elementos de dados no qual um elemento individual é identificado por sua posição na agregação. Na linguagem Java, os elementos de uma matriz não precisam ser do mesmo tipo.

Questões de concursos

[UERJ 2021 UERJ – Analista de Tecnologia da Informação] De acordo com essa afirmação e com os conceitos da linguagem de programação Java, é correto afirmar que: (Marque CERTO ou ERRADO o texto da letra)

- [A] o tipo de dado matriz é um agregado homogêneo de elementos de dados no qual um elemento individual é identificado por sua posição na agregação. Na linguagem Java, os elementos de uma matriz ~~não~~ precisam ser do mesmo tipo.
  - Gabarito: **ERRADO**.

Dúvidas?

Prof. Rogerão Araújo

www.instagram.com/profRogeraoAraujo

PROVAS DE TI
TUDO PARA VOCÊ PASSAR