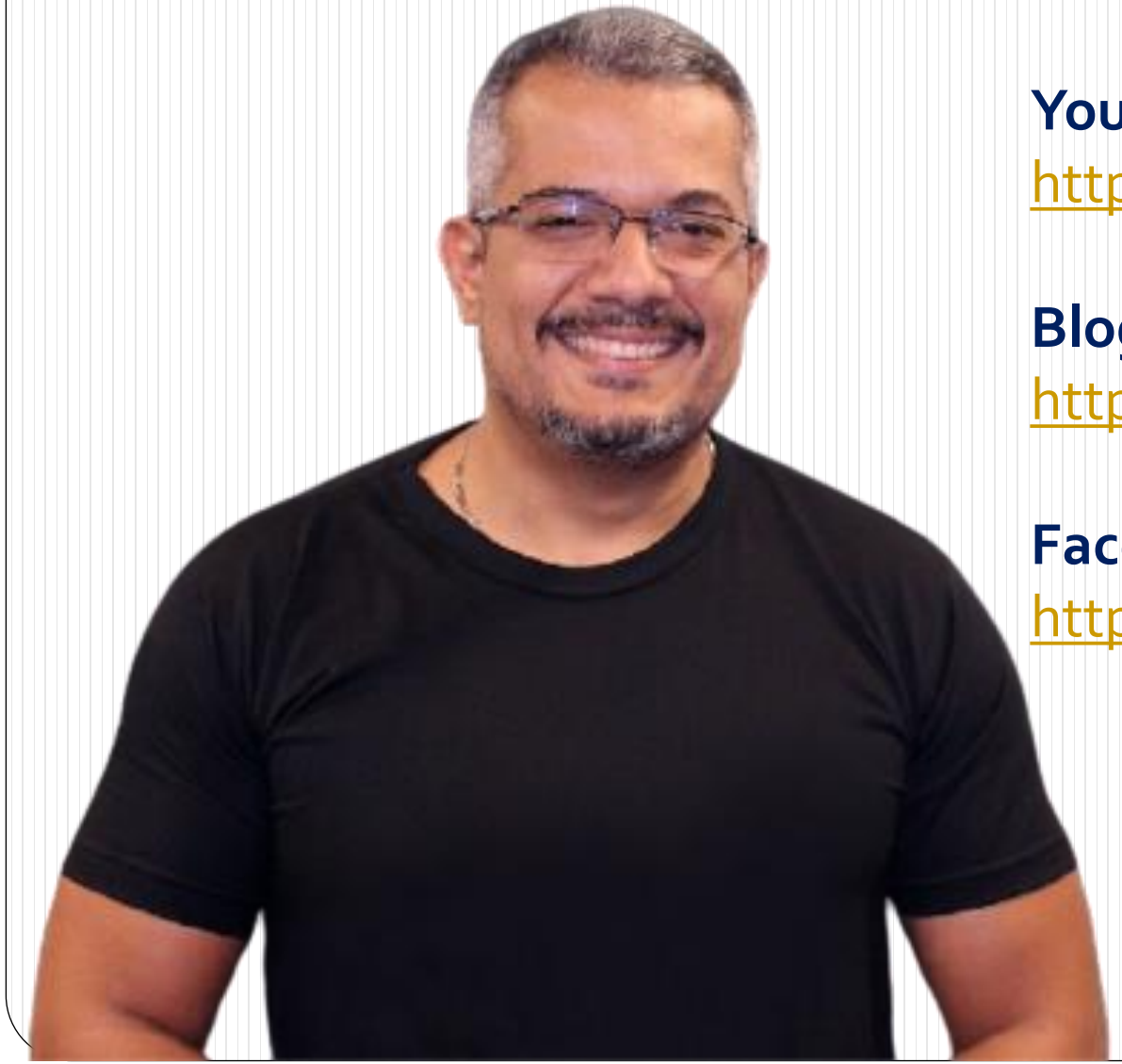
Linguagem Java SE 11 – Parte II

Prof. Rogerão Araújo

www.instagram.com/profRogeraoAraujo

[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

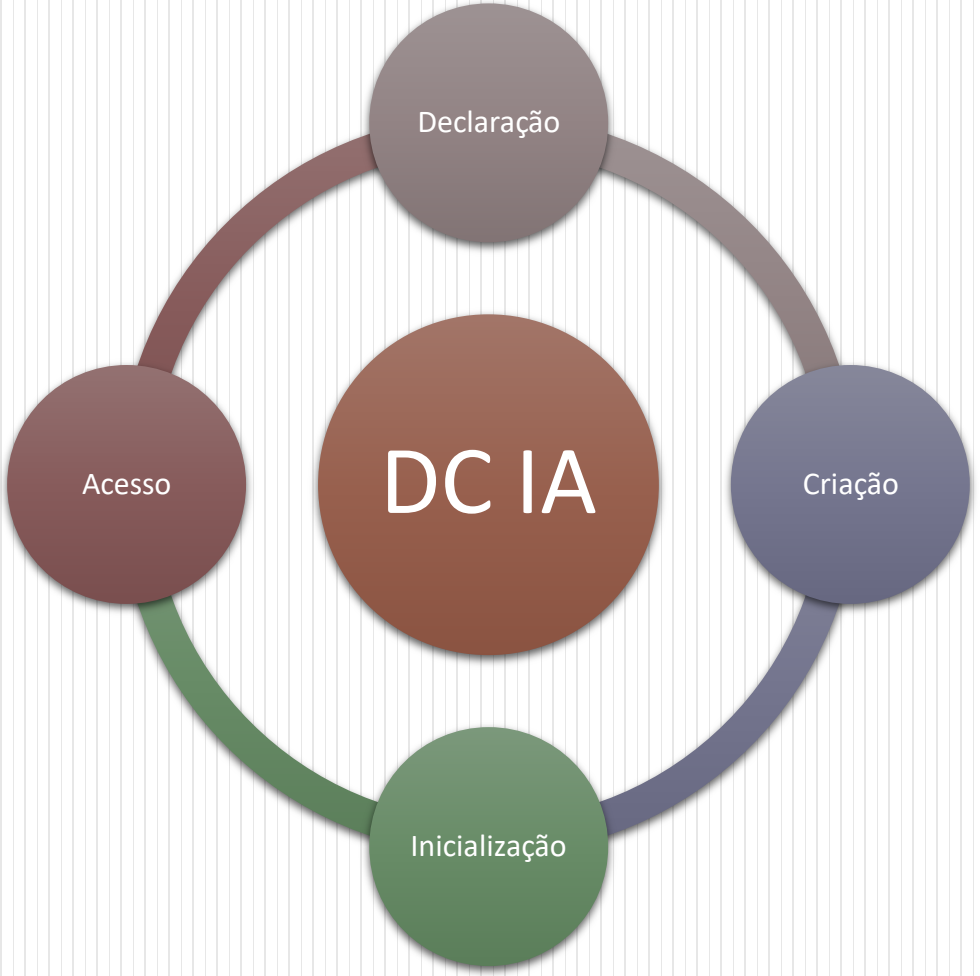
Básico da Linguagem Java

Arrays

Conceituação

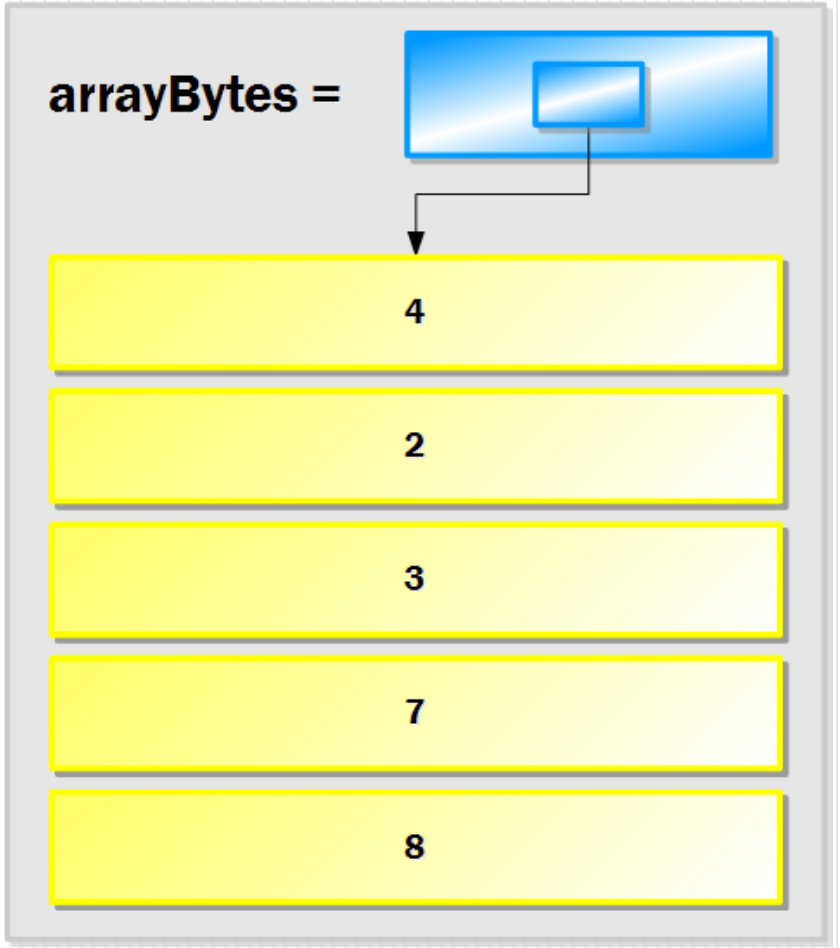
- São **objetos** que **contém** um **conjunto de valores** do **mesmo tipo**
- **Cada elemento de um array**
  - **Componente de array**
  - É uma **variável**
  - É **acessado** por um **índice**
    - De **0** a **n – 1**
      - Onde **n** é o **tamanho do array**
- **Tamanho de um array**
  - É **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};**
    - O **número de valores definem** 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};**
    - O **número de valores definem** 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

- Código:
  - public class ExemploForEach {
    - public static void main(String []args) {
      - 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

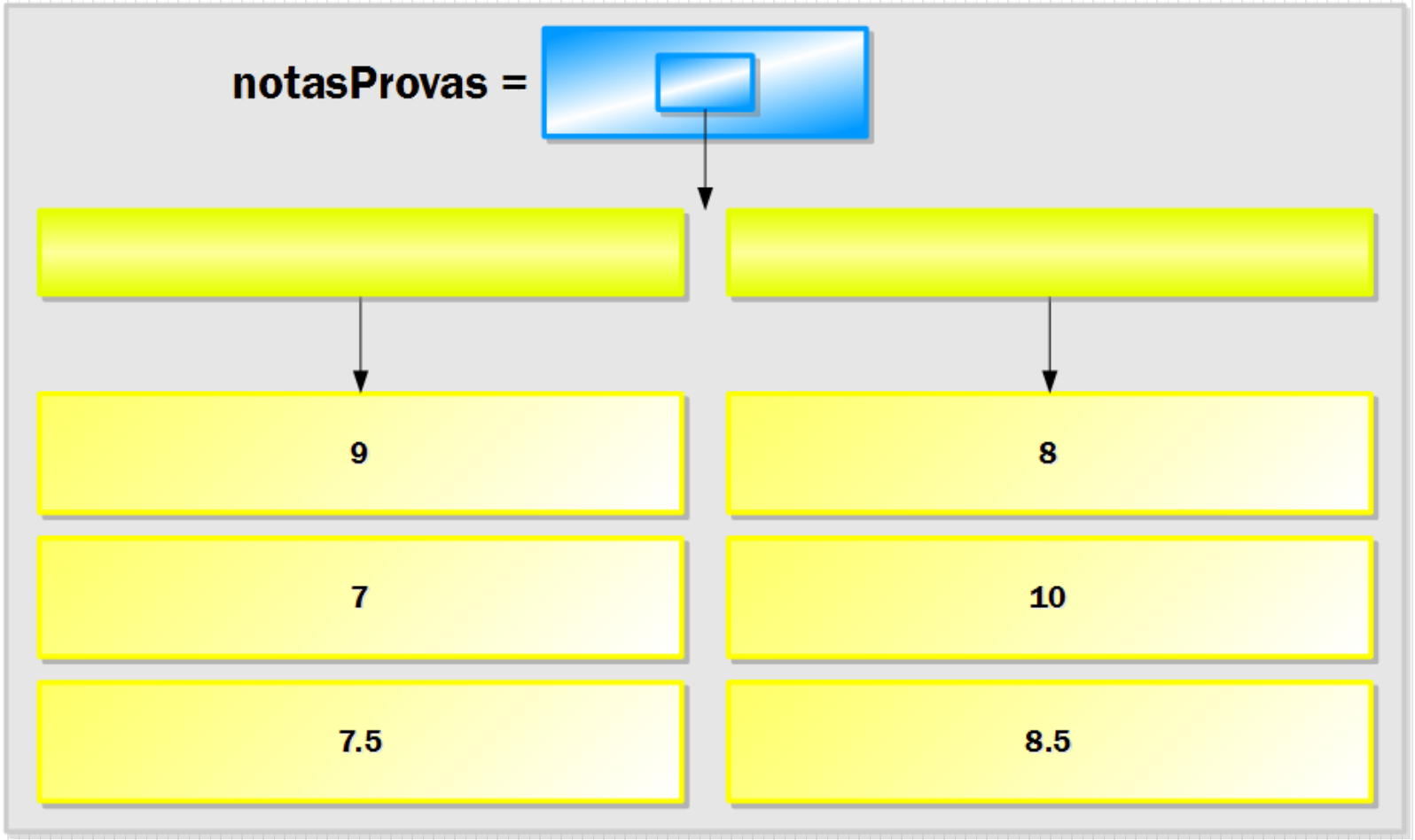
Arrays multidimensionais

- 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**.

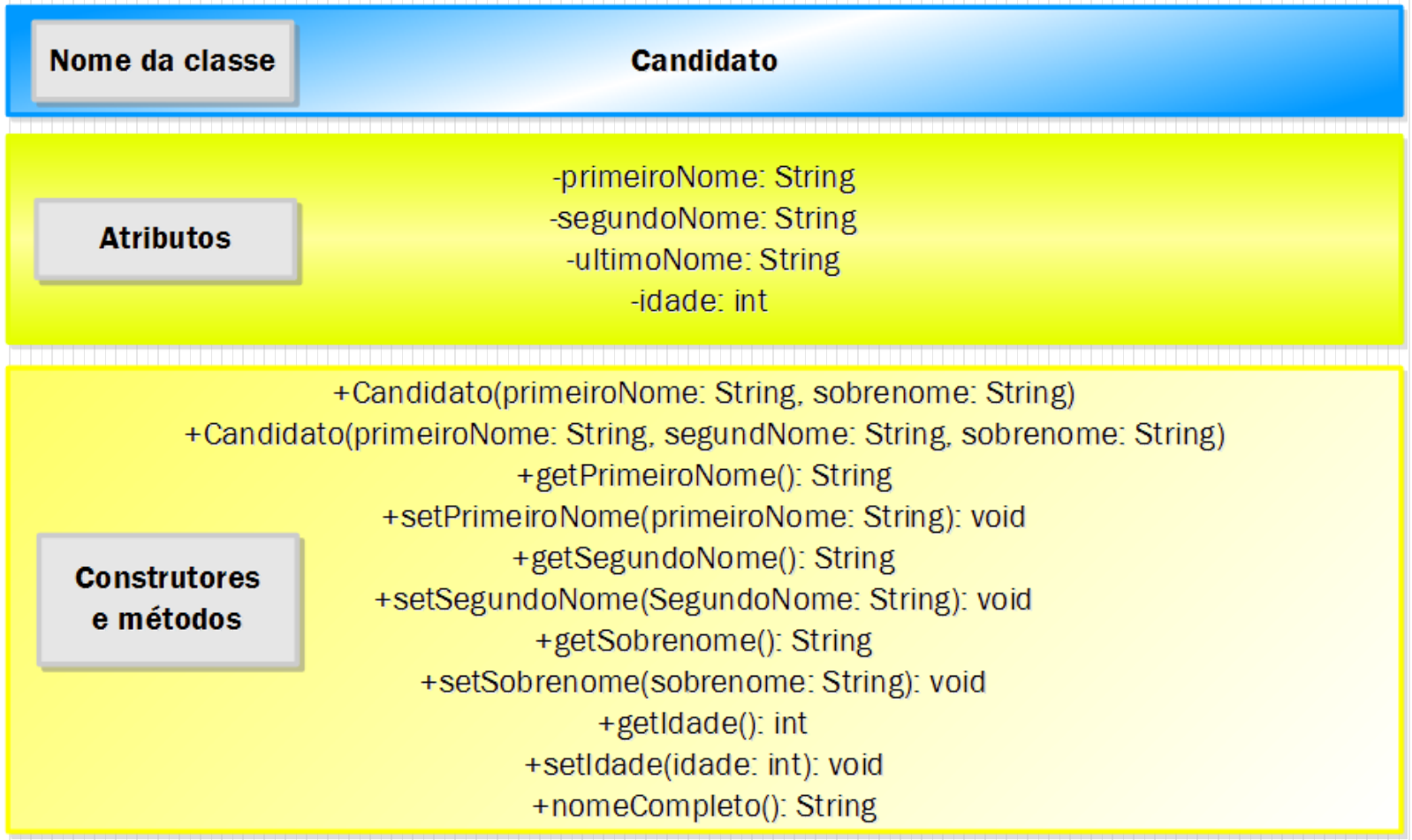
POO com Java

Classes e objetos

Estrutura de uma classe em Java

- // Nome da classe.
- public class Classe {
  - // Zero ou mais atributos.
  - public String nome;
  - // Zero ou mais construtores.
  - public Classe(String nome){
    - this.nome = nome;
  - }
  - // Zero ou mais métodos.
  - public String getNome() {
    - return this.nome;
  - }
- }

Estrutura de uma classe em Java



Modificadores acesso

São padrões de visibilidade de acessos a

- Classes
- Membros de uma classe
  - Atributos
  - Métodos

Determinam se uma classe pode usar uma outra

- Invocando
  - Um determinado atributo
  - Um determinado método

Modificadores de níveis de acesso em Java

Nível superior	Nível de membro	
Aplicados a	Aplicados a	
Classes	Atributos	Métodos

Modificadores de níveis de acesso em Java

Modificadores de nível superior

public

Torna uma classe visível

- Para qualquer outra classe
- Em qualquer pacote

default

Sem modificador explícito

Torna uma classe visível

- Apenas para classes do mesmo pacote

Modificadores de nível de membro

public

Torna um membro acessível

- Em qualquer lugar
- A qualquer outra classe que possa visualizar a classe que contém o membro

protected

Torna um membro acessível às classes

- Do mesmo pacote
- Através de herança

Os membros herdados não são acessíveis a outras classes fora do pacote em que foram declarados

Modificadores de nível de membro

default

Sem modificador explícito

Torna um membro acessível

Apenas para classes do mesmo pacote

private

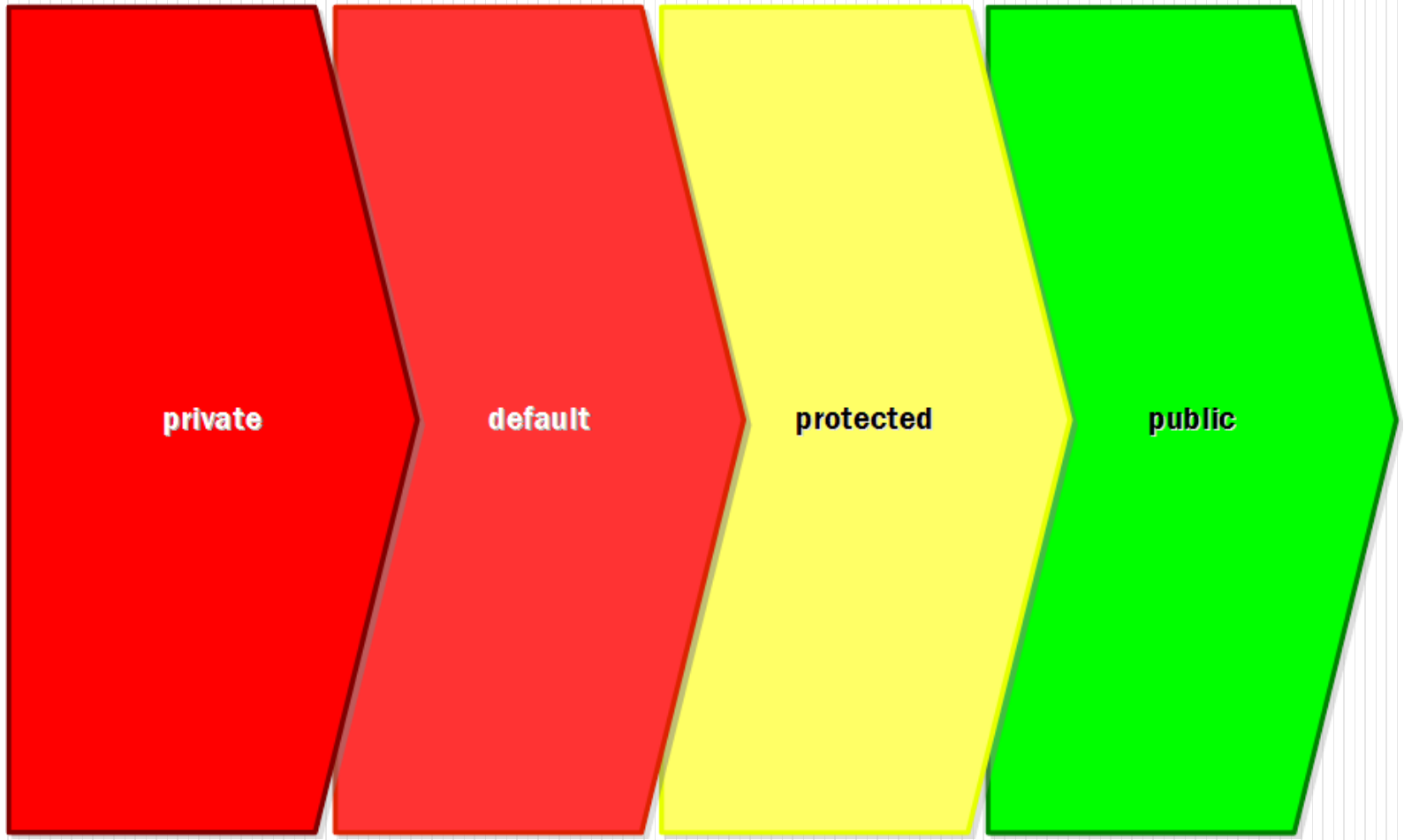
Torna um membro acessível

Apenas para a classe que o contém

Modificadores de nível de membro

	Classe	Pacote	Subclasse	Todos
public	☑	☑	☑	☑
protected	☑	☑	☑	
default	☑	☑		
private	☑			

Modificadores de nível de membro



Modificadores de nível de membro

Declaração de classes em Java

- **Modificadores de acesso de nível superior**:
  - **public**
  - **default**
- **Outros modificadores**:
  - **final**
    - **Torna** uma **classe** **impossível** de **ser estendida**
  - **abstract**
    - **Indica** uma **classe** **abstrata** que:
      - Pode ter ou não métodos abstratos
      - Não pode ser instanciada, mas pode ter subclasses

Declaração de classes em Java

- **Palavra-chave class**
- **Nome da classe**
  - **Começa** com **letra maiúscula** por **convenção**
  - A **sua nomeação segue** as **regras de nomeação de variáveis**
- **Nome da superclasse a ser estendida**
  - Se houver
  - É **precedida** da **palavra-chave extends**
  - Uma **classe deve estender** apenas **uma superclasse**

Declaração de classes em Java

- **Lista de nomes das interfaces a serem implementadas**
  - Se houver
  - É **precedida** da **palavra-chave implements**
  - Os **nomes** são **separados** por **vírgulas**
  - Uma **classe pode implemenar mais de uma interface**
- **Corpo da classe**
  - É **envolvido** por **chaves {}**

Construtores

- São **chamados** para **criar objetos** a partir de **uma classe**
  - Na **linguagem Java**, usa-se o **operador new** para **criar objetos**
    - Exemplo:
      - Classe objeto = **new** Classe();
- São **blocos declarados** com o **mesmo nome da classe**
  - A **sua declaração** parece com as **declarações de métodos**
    - Porém, os **construtores**:
      - Usam o **mesmo nome da classe**
      - **Não possuem** nenhum **tipo de retorno**
        - Nem mesmo void é declarado

Construtores

- **Construtor padrão (default)**
  - **Não recebe** nenhum **parâmetro de construtor**
  - **Possui** o **corpo vazio**
  - Na **linguagem Java**:
    - É **automaticamente fornecido** pelo **compilador**
      - Não é necessário declarar nenhum construtor para uma classe
    - Se houver a declaração de pelo menos um construtor para uma classe, o construtor padrão não é mais fornecido

Construtores

- **Construtores sobrecarregados**
  - São **construtores** com **número** e **tipos de parâmetros diferentes**
  - Não se pode declarar dois construtores com a mesma assinatura
- **Construtores chamando construtores**
  - Construtores podem chamar outros

Declaração de atributos

- **Modificadores de acesso de nível de membro**:
  - **private**
  - **public**
  - **protected**
  - **default**
- **Outros modificadores**:
  - **final**
    - **Torna** um **atributo** **imutável** (“constante” em Java)
  - **static**
    - **Torna** um **atributo** um **atributo de classe**
- **Tipo do atributo**
- **Nome do atributo**

Atributos estáticos

- **Variáveis de classe**
- **Possuem** o **modificador static**
  - Esse modificador informa ao compilador que haverá apenas uma cópia da variável
    - Não importando quantas instâncias a classe tenha
- São **manipulados a partir** do **nome da classe**
  - Sem a necessidade de criar uma instância dela
  - Exemplo:
    - Classe.atributo

Atributos estáticos

- **Mantêm informações** da **classe**
  - Não de suas instâncias
- **Possuem escopo** de **classe**
- São **compartilhados** por **cada instância da classe**
  - Qualquer objeto pode alterar o valor de um atributo estático

Declaração de métodos

- **Modificadores de acesso de nível de membro**:
  - **private**
  - **public**
  - **protected**
  - **default**
- **Outros modificadores**:
  - **final**
    - **Torna** um **método impossível** de **ser sobrescrito**
  - **abstract**
    - **Indica** um **método** que é **declarado sem uma implementação**
  - **static**
    - **Torna** um **método** um **metódo de classe**
      - Não opera em objetos

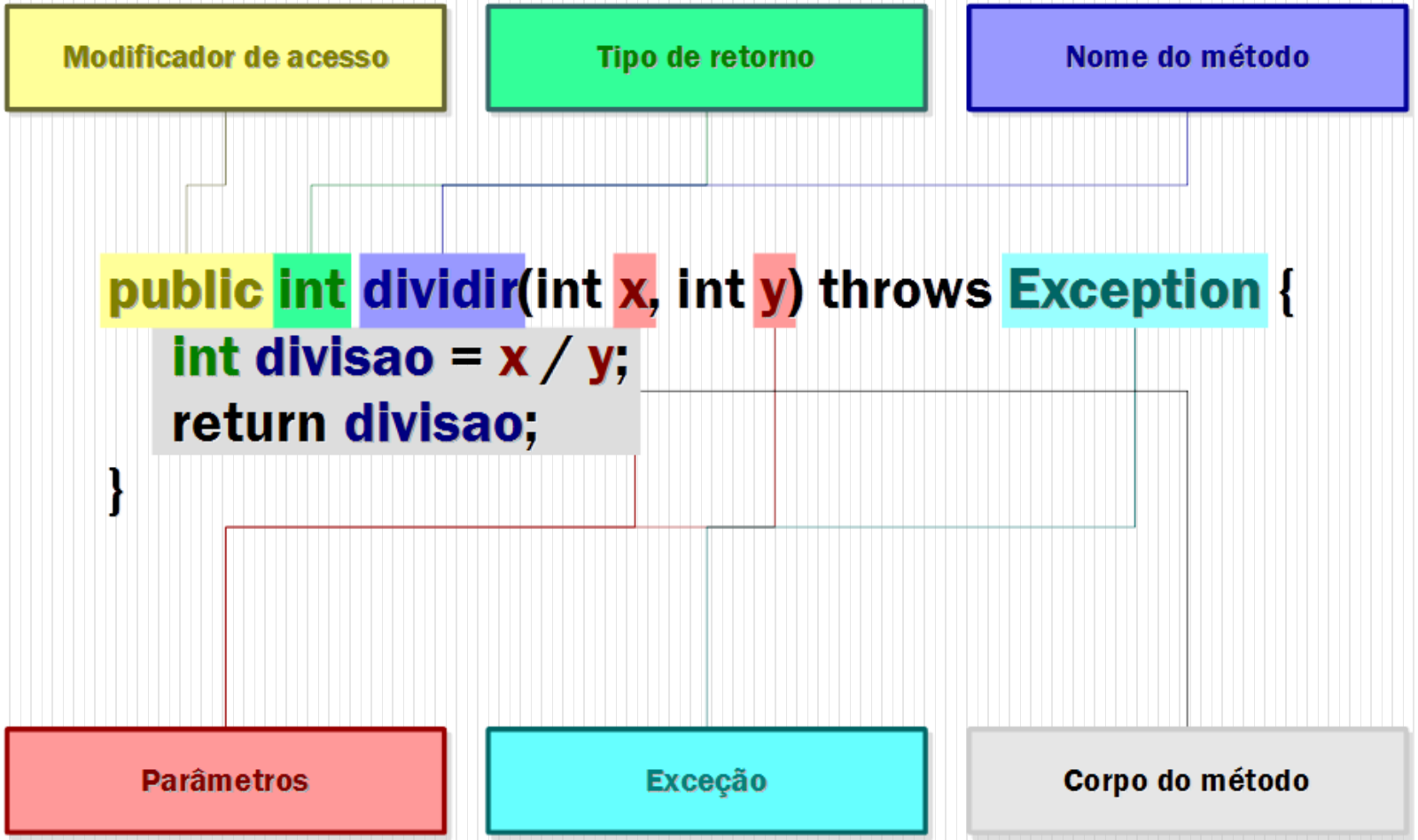
Declaração de métodos

- **Tipo de retorno do método**
  - É o **tipo do valor retornado** pelo **método** pela a declaração **return**
  - **Usa-se void**
    - Se o método não retornar nada
- **Nome do método**
  - A **sua nomeação segue** as **regras de nomeação de variáveis**
- **Lista de parâmetros**
  - É **envolvida** por **parênteses ()**

Declaração de métodos

- **Lista de exceções**
  - Se houver
- **Variáveis locais**
  - São **declaradas dentro** do **corpo do método**
- **Corpo do método**
  - É **envolvido** por **chaves {}**

Declaração de métodos



Métodos estáticos

- **Possuem** o **modificador static**
- São **invocados a partir** do **nome da classe**
  - Sem a necessidade de criar uma instância dela
  - Exemplo:
    - Classe.metodo(argumentos)
- **Podem ser referidos a partir** das **instâncias da classe**
  - Mas **não é recomendado** porque questão de clareza
  - Exemplo:
    - instancia.metodo(argumentos)
- Seu **uso comum** é **acessar atributos estáticos**

Métodos de instância x métodos estáticos

Podem acessar diretamente	Métodos de instância	Métodos estáticos
Atributos de instância	Sim	Não
Métodos de instância	Sim	Não
Atributos estáticos	Sim	Sim
Métodos estáticos	Sim	Sim

Métodos sobrecarregados

- São **métodos** com:
  - **Nomes iguais**
  - **Número** e **tipos de parâmetros diferentes**
- O **tipo de retorno dos métodos não é considerado** pelo **compilador** para **diferenciá-los**
  - Não se pode declarar dois métodos com a mesma assinatura
    - Mesmo que tenham um tipo de retorno diferente
- Devem ser usados com moderação
  - Já que podem tornar o código muito menos legível

Métodos sobrecarregados

- Exemplo:
  - public class Escrita {
    - public void escrever(int x) { // Método sobrecarregado.
      - System.out.println("int: " + x);
    - }
    - public void escrever(double x) { // Método sobrecarregado.
      - System.out.println("double: " + x);
    - }
    - public int escrever(double x) { // Método não sobrecarregado. Erro na compilação.
      - return x;
    - }
  - }

Retorno de métodos

Um método retorna para o código que o chamou quando:

- O método completa todas as declarações no método
- Se atinge uma instrução return
- O método lança uma exceção

Retorno de métodos

Tipo de retorno do método

É o tipo do valor retornado pelo método

É usada a declaração return

Usa-se void

Se o método não retornar nada

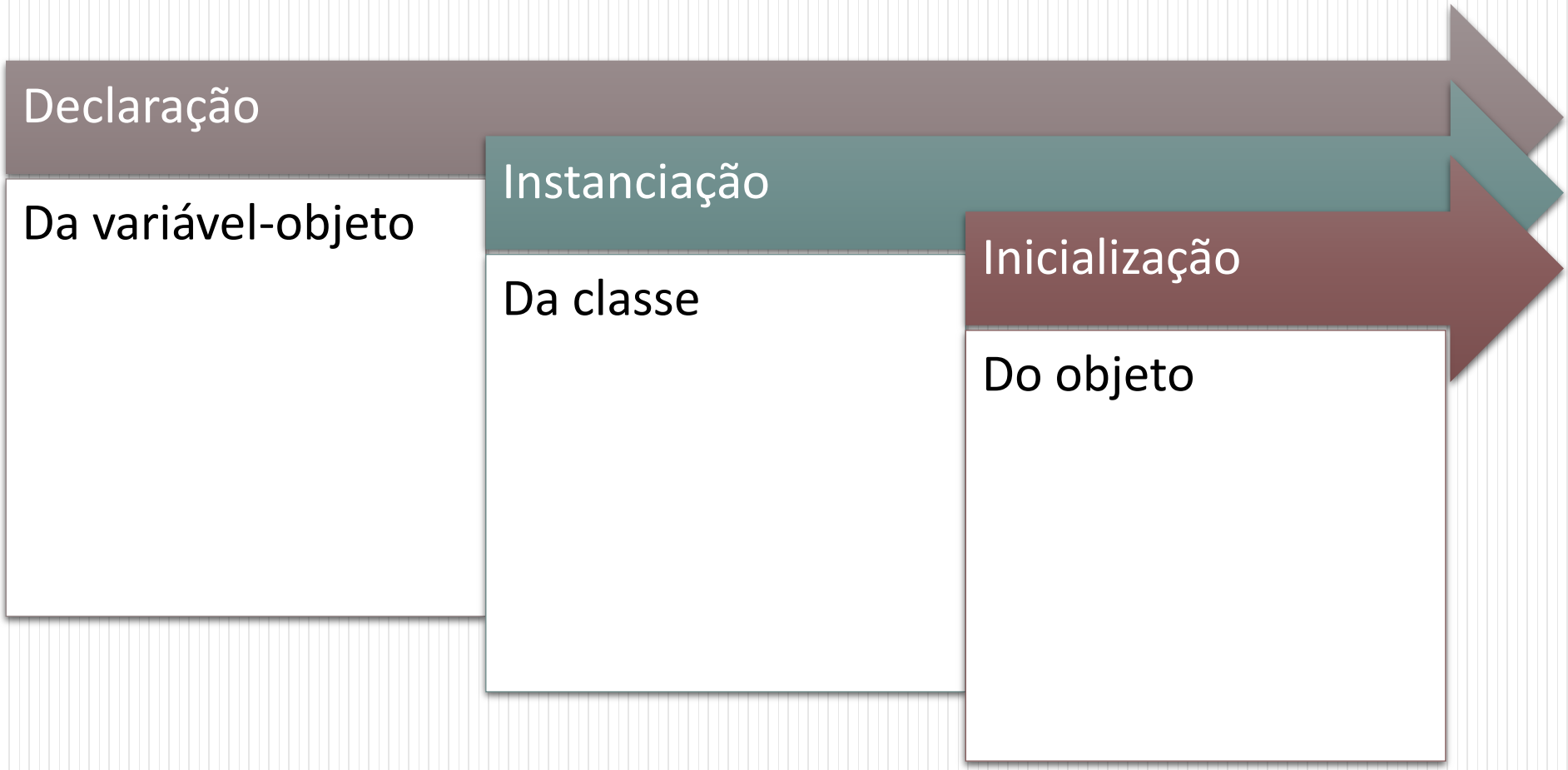
Não é necessário incluir return nesse caso

Se for usada, a instrução return irá forçar a saída do método

Retorno de métodos

- **Erros de compilação**
  - Informar um valor de retorno em métodos que não retornam nada (void)
    - public **void** metodo(String x) {
      - **return 15; // Errado.**
      - **return**; // Certo. Também poderia omiti-lo.
    - }
  - Informar um tipo de valor de retorno diferente do tipo informado na declaração do método
    - public **int** metodo() {
      - **return “15”; // Errado.**
      - **return 15**; // Certo.
    - }

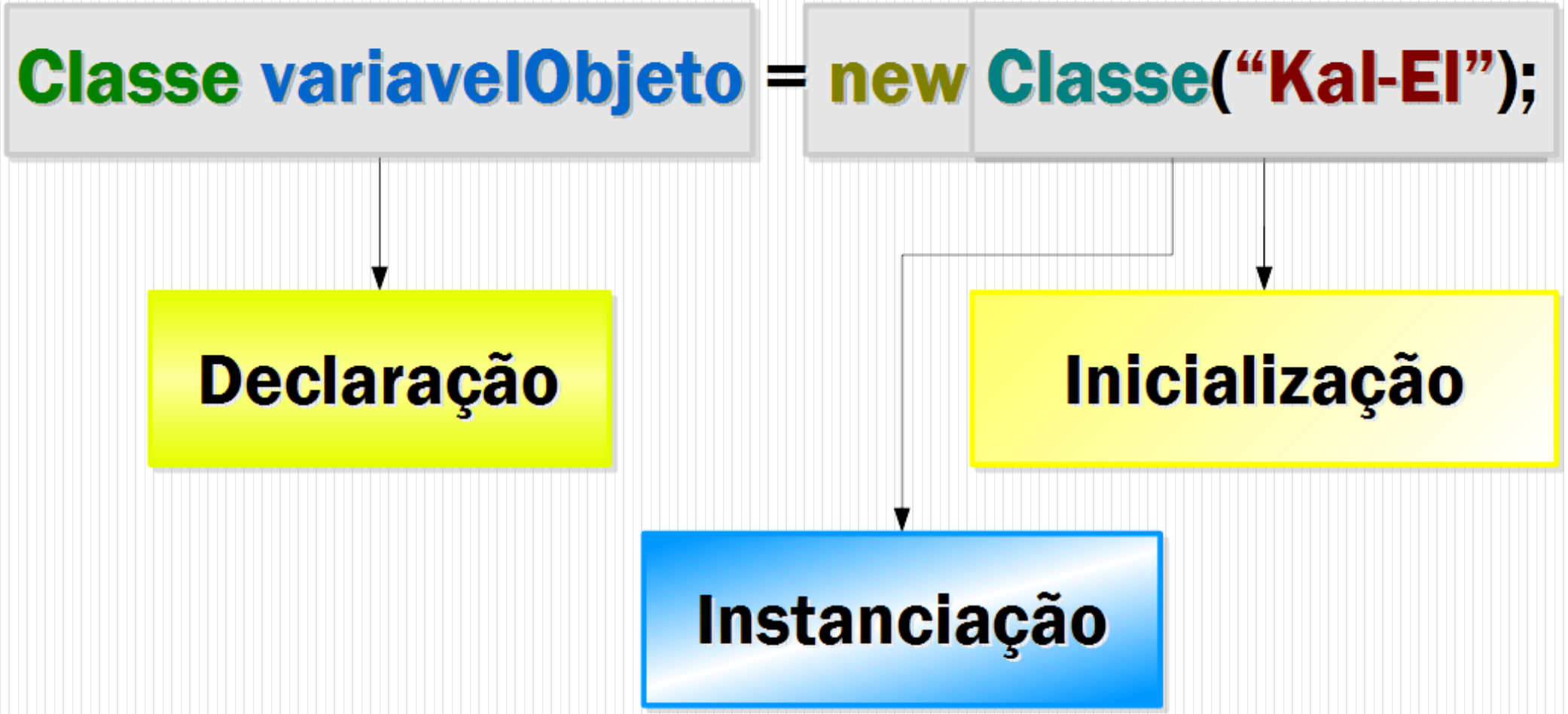
Fases da criação de um objeto



Fases da criação de um objeto

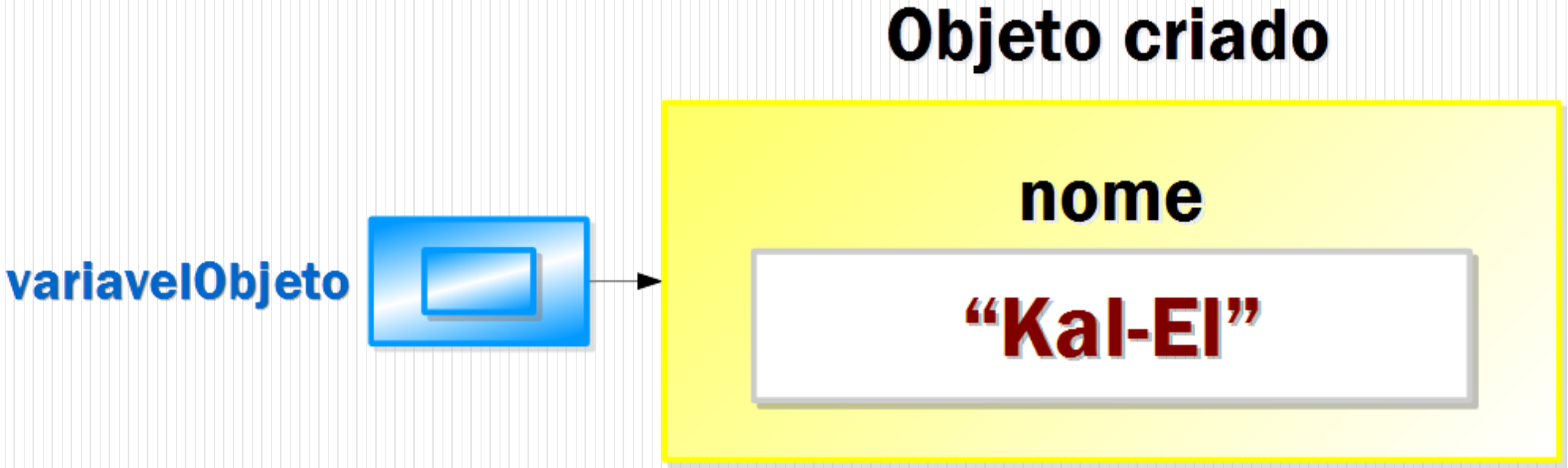
- **Declaração**
  - É a **etapa** de **definição** da **variável-objeto** para **referenciar** um **novo objeto**
  - Neste momento, não se pode utilizar o objeto, pois ele ainda não foi criado
- **Instanciação**
  - É a **etapa** de **utilização** do **operador new** para **criar** um **novo objeto**
  - O **operador new instancia** uma **classe**
    - Usando um dos seus construtores
    - Alocando memória para um novo objeto
    - Retornando uma referência para a memória
      - Esta referência é geralmente atribuída a uma variável do tipo apropriado
- **Inicialização**
  - É a **etapa** de **inicialização** do **novo objeto** através de um **construtor**

Fases da criação de um objeto



Fases da criação de um objeto

```
Classe variavelObjeto = new Classe("Kal-El");
```

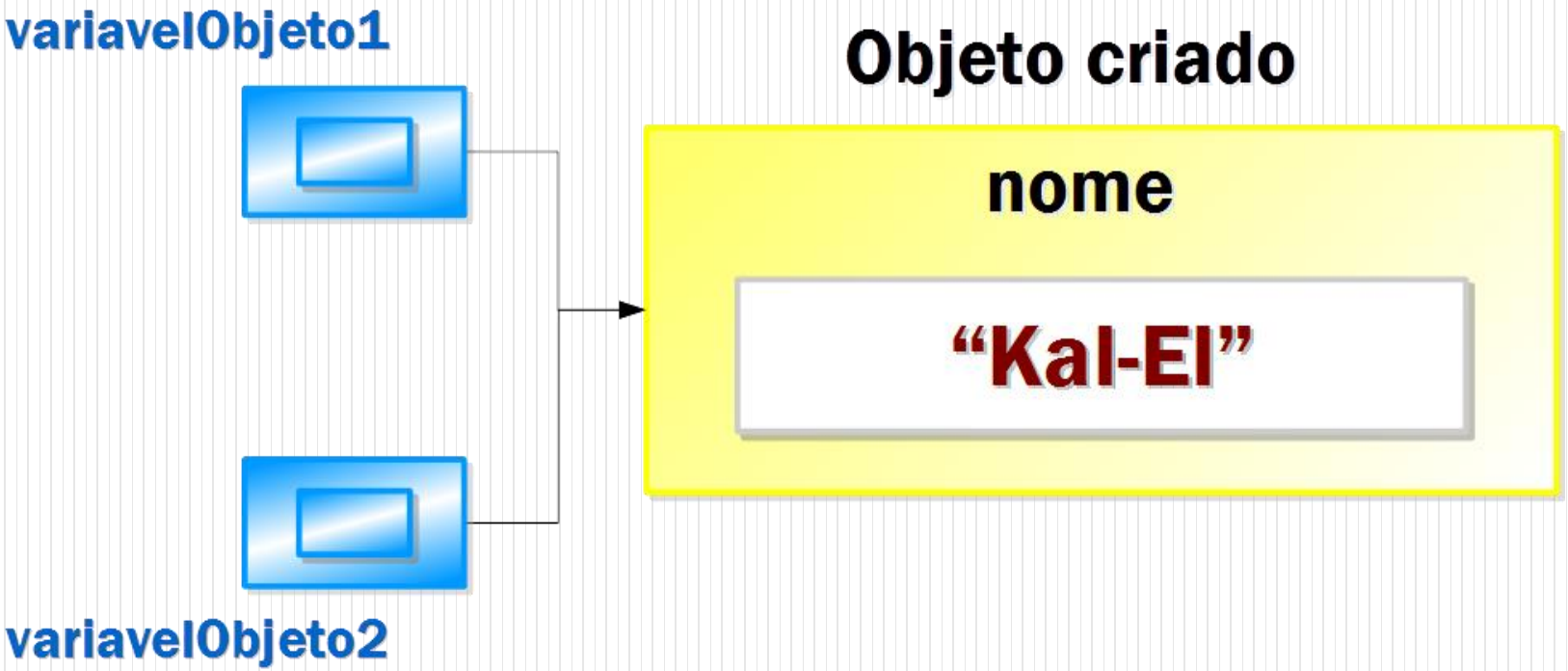


Variável-objeto

- **Não é o mesmo** que um **objeto**
- **Referencia** um **objeto**
- **Pode**:
  - **Ser inicializada** com um **objeto recém-criado**
    - Classe variavelObjeto = new Classe(“Kal-El”);
  - **Referenciar** um **objeto já criado anteriormente**
    - Classe variavelObjeto1 = new Classe(“Kal-El”);
    - Classe variavelObjeto2 = variavelObjeto1;

Variável-objeto

- Classe variavelObjeto1 = new Classe(“Kal-El”);
- Classe variavelObjeto2 = variavelObjeto1;



Exemplo

```java
package constelacao02.parte02.classes;

// Nome da classe.
public class Candidato {
    // Zero ou mais atributos.
    private String primeiroNome, segundoNome;
    private String sobrenome;
    private int idade;

    // Zero ou mais construtores.
    // Construtor 1.
    public Candidato(String primeiroNome, String sobrenome) {
        // nome é variável local.
        String nome = primeiroNome;
        this.primeiroNome = nome;
        this.sobrenome = sobrenome;
    }

    // Construtor 2.
    public Candidato(String primeiroNome, String segundoNome, String sobrenome) {
        this.primeiroNome = primeiroNome;
        this.segundoNome = segundoNome;
        this.sobrenome = sobrenome;
    }
```

Exemplo (continuação)

```
26      // Zero ou mais métodos
27      public String getPrimeiroNome() {
28          return primeiroNome;
29      }
30
31      public void setPrimeiroNome(String primeiroNome) {
32          this.primeiroNome = primeiroNome;
33      }
34
35      public String getSegundoNome() {
36          return segundoNome;
37      }
38
39      public void setSegundoNome(String segundoNome) {
40          this.segundoNome = segundoNome;
41      }
42
43      public String getSobrenome() {
44          return sobrenome;
45      }
46
47      public void setSobrenome(String sobrenome) {
48          this.sobrenome = sobrenome;
49      }
```

Exemplo (continuação)

```
51      public int getIdade() {
52          return idade;
53      }
54
55      public void setIdade(int idade) {
56          this.idade = idade;
57      }
58
59      public String nomeCompleto() {
60          return this.primeiroNome + " " + this.segundoNome + " " + this.sobrenome;
61      }
62  }
```

Mais exemplos

- Classe default:
  - class Classe {
    - …
  - }
- Classe pública que estende outra:
  - public class Classe extends SuperClasse {
    - …
  - }

Mais exemplos

- Classe default que implementa uma interface:
  - class Classe implements Interface {
    - …
  - }
- Classe pública que estende outra e implementa mais de uma interface:
  - public class Classe extends SuperClasse implements IF1, IF2 {
    - …
  - }

Mais exemplos

- Classe pública e final (não pode ser estendida):
  - public final class Classe {
    - …
  - }
- Classe pública e abstrata (não pode ser instanciada):
  - public abstract class Classe {
    - …
  - }

Mais exemplos

- Superclasse:
  - public class Superclasse {
    - public void metodo(String x) {
      - System.out.println(x);
    - }
  - }

- Subclasse:
  - public class Subclasse extends Superclasse {
    - @Override
    - public void metodo(String x) {
      - System.out.println(x);
    - }
  - }

Questões de concursos

[IBFC 2020 EBSERH – Analista de Tecnologia da Informação] Observe cuidadosamente o código Java abaixo:

```
public class OlaMundo {
    public static void main(String[] args) {
        System.println("Olá Mundo!");
    }
}
```

Questões de concursos

[IBFC 2020 EBSERH – Analista de Tecnologia da Informação] Assinale a alternativa correta.

- [A] a primeira linha do código deveria ser → private class OlaMundo {
- [B] a segunda linha do código deveria ser → public static main(String args) {
- [C] a terceira linha do código deveria ser → System.out.println("Olá Mundo!")
- [D] não deveria ter a quarta linha do código
- [E] não deveria ter a quinta linha do código

Questões de concursos

[IBFC 2020 EBSERH – Analista de Tecnologia da Informação] Assinale a alternativa correta.

- [A] a primeira linha do código deveria ser → private class OlaMundo {
- [B] a segunda linha do código deveria ser → public static main(String args) {
- **[C] a terceira linha do código deveria ser → System.out.println("Olá Mundo!")**
- [D] não deveria ter a quarta linha do código
- [E] não deveria ter a quinta linha do código

Questões de concursos

[COMPERVE 2020 TJ/RN – Analista de Sistemas Sênior] Os modificadores de acesso são palavras-chave que delimitam o nível de acesso, visibilidade e encapsulamento de elementos na linguagem Java. Sobre os modificadores de acesso, é correto afirmar:

- [A] o modificador protected permite que apenas a própria classe possa acessar o recurso.
- [B] o modificador private permite que apenas a própria classe e as classes do mesmo pacote possam acessar o recurso.

Questões de concursos

[COMPERVE 2020 TJ/RN – Analista de Sistemas Sênior] Os modificadores de acesso são palavras-chave que delimitam o nível de acesso, visibilidade e encapsulamento de elementos na linguagem Java. Sobre os modificadores de acesso, é correto afirmar:

- [A] o modificador ~~**protected**~~ **private** permite que apenas a própria classe possa acessar o recurso.
- [B] o modificador ~~**private**~~ **default** permite que apenas a própria classe e as classes do mesmo pacote possam acessar o recurso.

Questões de concursos

[COMPERVE 2020 TJ/RN – Analista de Sistemas Sênior] Os modificadores de acesso são palavras-chave que delimitam o nível de acesso, visibilidade e encapsulamento de elementos na linguagem Java. Sobre os modificadores de acesso, é correto afirmar:

- [C] o modificador public permite que apenas a própria classe e aqueles que a herdarem possam acessar o recurso.
- [D] o modificador default é atribuído quando não é informado nenhum modificador de acesso.

Questões de concursos

[COMPERVE 2020 TJ/RN – Analista de Sistemas Sênior] Os modificadores de acesso são palavras-chave que delimitam o nível de acesso, visibilidade e encapsulamento de elementos na linguagem Java. Sobre os modificadores de acesso, é correto afirmar:

- [C] o modificador ~~**public**~~ **protected** permite que apenas a própria classe e aqueles que a herdarem possam acessar o recurso.
- **[D] o modificador default é atribuído quando não é informado nenhum modificador de acesso.**

Questões de concursos

[IDECAN 2020 IFRR – Informática] A Orientação a Objetos (OO) é um paradigma de programação para o qual "tudo é um objeto", sendo Java uma das principais linguagens que implementam esse paradigma. Em relação à linguagem Java e à OO, analise as seguintes afirmativas:

- [III] Uma classe Java declarada como **final** não pode ser herdada (não pode ter subclasses Java).

Questões de concursos

[IDECAN 2020 IFRR – Informática] A Orientação a Objetos (OO) é um paradigma de programação para o qual "tudo é um objeto", sendo Java uma das principais linguagens que implementam esse paradigma. Em relação à linguagem Java e à OO, analise as seguintes afirmativas:

- [III] Uma classe Java declarada como **final** não pode ser herdada (não pode ter subclasses Java).
  - Gabarito: **CERTO**.

Questões de concursos

[CESGRANRIO 2018 Banco da Amazônia – Técnico Científico – Tecnologia da Informação] Considere o código Java listado a seguir, onde a numeração de linhas está sendo utilizada apenas como referência:

```
i.   package teste;
ii.  public class Classe {
iii. private static int a = 5;
iv.  public static void main(String[] args) {
v.       int a = 8;
vi.
vii. }
viii.}
```

Questões de concursos

[CESGRANRIO 2018 Banco da Amazônia – Técnico Científico – Tecnologia da Informação] Que comando deve ser inserido na linha vi para exibir o valor 5 na console?

- [A] System.out.println(a);
- [B] System.out.println(Classe.a);
- [C] System.out.println(Integer.parseInt(a));
- [D] System.out.println(super.a);
- [E] System.out.println(this.a);

Questões de concursos

[CESGRANRIO 2018 Banco da Amazônia – Técnico Científico – Tecnologia da Informação] Que comando deve ser inserido na linha vi para exibir o valor 5 na console?

- [A] System.out.println(a);
- **[B] System.out.println(Classe.a);**
- [C] System.out.println(Integer.parseInt(a));
- [D] System.out.println(super.a);
- [E] System.out.println(this.a);

Herança

Conceituação

Em Java, as classes podem ser derivadas de outras classes

Herdando dessas classes

Atributos

Métodos

Visão geral

Superclasse

- public class Superclasse {
  - Atributos que serão herdados
  - Métodos que serão herdados
- }

Subclasse

- public class Subclasse extends Superclasse {
  - Atributos adicionados
  - Métodos adicionados
- }

Termos na herança em Java

Subclasse

- Classe derivada
- Classe estendida
- Classe filha
- É uma classe derivada de outra

Superclasse

- Classe base
- Classe mãe
- Classe pai
- É a classe base para uma subclasse

Termos na herança em Java

Classes descendentes

São classes em um nível inferior em relação a outras em uma hierarquia de herança de classes

Onde a classe base da hierarquia é Object

Classes ancestrais

São classes em um nível superior em relação a outras em uma hierarquia de herança de classes

Onde a classe base da hierarquia é Object

Herança simples

Não há suporte à herança múltipla em Java

Cada classe em Java tem uma e apenas uma superclasse direta

Na ausência de qualquer outra superclasse explícita, cada classe é implicitamente uma subclasse de Object

Exceto a classe Object que não possui superclasse

Todas as classes na Linguagem Java são descendentes de Object

Herança de membros

- A **subclasse** **herda** **todos os membros** **acessíveis** de sua **superclasse**:
  - **Atributos**
  - **Métodos**
  - **Classes aninhadas**
- **Construtores** **não** são **membros** de uma **classe**
  - **Construtores da superclasse**:
    - **Não** são **herdados** por **subclasses**
    - **Podem ser chamados a partir** da **subclasse**

Herança de membros

- **Subclasse no mesmo pacote da sua superclasse**
  - **Herdará** desta os **membros com os níveis**:
    - **public**
    - **protected**
    - **default**
- **Subclasse em pacote diferente da sua superclasse**
  - **Herdará** desta os **membros com os níveis:**
    - **public**
    - **protected**

Herança de membros

- A **subclasse**:
  - **Não herda** os **membros privados** da sua **superclasse**
  - **Pode acessar** os **membros privados** através de **métodos públicos**, **protegidos** ou **default**
    - Disponibilizados pela superclasse

Métodos sobrescritos

- São **métodos de instância herdados** que foram **reescritos** na **subclasse**
  - **Alterando** o **comportamento deles**
  - **Substituindo eles**
- **Possuem** a **mesma assinatura** que os **métodos herdados**
- **Anotação @Override**
  - É **usada** quando houver a **sobrescrita** do **método herdado**
    - Instruindo o compilador sobre a sobrescrita

Métodos sobrescritos

- Exemplos:
  - Superclasse:
    - public class Superclasse {
      - public int teste(int x) {
        - return x * 10;
      - }
    - }
  - Subclasse:
    - public class Subclasse extends Superclasse {
      - @Override
      - public int teste(int x) {
        - return x * 10 + 15;
      - }
    - }

Métodos sobrecarregados nas subclasses

- São **métodos da subclasse** que **possuem mesmos nomes** dos **métodos da superclasse**
  - Mas **número** e **tipos de parâmetros diferentes**
- **Não escondem nem substituem** os **métodos da superclasse**
  - São **novos métodos exclusivos** para a **subclasse**

Métodos sobrecarregados nas subclasses

- Exemplos:
  - Superclasse:
    - public class Superclasse {
      - public int teste(int x) {
        - return x * 10;
      - }
    - }
  - Subclasse:
    - public class Subclasse extends Superclasse {
      - public int teste(int x, int y) {
        - return x * y * 10 + 15;
      - }
    - }

- Importante:
  - Os modificadores de níveis de acesso de métodos sobrescritos devem ser iguais ou mais liberais aos métodos herdados da superclasse

Palavra-chave super

- É **usada** para **acessar** da **superclasse**:
  - **Seus membros**
    - **Atributos ocultos** da superclasse
    - **Métodos sobrescritos** da superclasse
  - **Seus construtores**
    - **Sintaxe**:
      - **super**();
      - **super**(lista de parâmetros);

Palavra-chave super

- Se um construtor da subclasse não chamar explicitamente um construtor da superclasse, o compilador Java insere automaticamente uma chamada para o construtor sem argumento da superclasse
  - Se a superclasse não possui um construtor sem argumento, haverá um erro em tempo de compilação neste caso

Classe Object

- **Encontra-se** no **pacote java.lang**
- **Situa-se** no **topo** da **árvore de hierarquia de classes** na **linguagem Java**
- É **antecedente** de **qualquer classe**
  - **Diretamente**
    - Não é necessário escrever:
      - public class Classe extends Object
  - **Indiretamente**

Questões de concursos

[COMPERVE 2020 TJ/RN – Analista de Sistemas Sênior] Sobre a linguagem de programação Java, analise as afirmativas abaixo.

- [II] A palavra chave ***inherits*** define a herança de uma classe para outra.

Questões de concursos

[COMPERVE 2020 TJ/RN – Analista de Sistemas Sênior] Sobre a linguagem de programação Java, analise as afirmativas abaixo.

- [II] A palavra chave ~~***inherits***~~ ***extends*** define a herança de uma classe para outra.
  - Gabarito: **ERRADO**.

Questões de concursos

[COMPERVE 2020 TJ/RN – Analista de Sistemas Sênior] Sobre a linguagem de programação Java, analise as afirmativas abaixo.

- [III] A palavra chave ***super*** é utilizada para fazer referência à classe pai herdada.

Questões de concursos

[COMPERVE 2020 TJ/RN – Analista de Sistemas Sênior] Sobre a linguagem de programação Java, analise as afirmativas abaixo.

- [III] A palavra chave ***super*** é utilizada para fazer referência à classe pai herdada.
  - Gabarito: **CERTO**.

Questões de concursos

[CESPE 2020 TJ/PA – Analista Judiciário – Programador] Na programação orientada a objetos, a herança é uma técnica de abstração que permite categorizar as classes de objetos sob certos critérios, especificando-se as características dessas classes. As classes que são vinculadas por meio de relacionamentos de herança formam uma hierarquia de herança.

Questões de concursos

[CESPE 2020 TJ/PA – Analista Judiciário – Programador] Na linguagem de programação Java, o relacionamento de herança é definido pela palavra-chave

- [A] static.
- [B] extends.
- [C] public.
- [D] new.
- [E] this.

Questões de concursos

[CESPE 2020 TJ/PA – Analista Judiciário – Programador] Na linguagem de programação Java, o relacionamento de herança é definido pela palavra-chave

- [A] static.
- **[B] extends.**
- [C] public.
- [D] new.
- [E] this.

Polimorfismo em Java

Conceituação mais utilizada

É a propriedade de duas ou mais classes derivadas

De uma mesma superclasse

Responderem a mesma mensagem

Cada uma de uma forma diferente

Ocorre quando uma subclasse redefine um método existente na superclasse

Métodos sobrescritos

Overriding

Conceituação

- É definido como sendo um **código** que:
  - **Possui** ou **produz** "**vários comportamentos**"
  - **Pode ser aplicado** a **várias classes de objetos**
- Exemplo:
  - Uma operação mantém seu comportamento transparente para quaisquer tipos de argumentos
    - A mesma mensagem é enviada a objetos de classes distintas e eles poderão reagir de maneiras diferentes
- **Método polimórfico**
  - **Pode ser aplicado** a **várias classes de objetos**
    - Sem que haja qualquer inconveniente

Esquema


Especialização
Macaco
Mamífero
Homem
Homem X
Baleia
Generalização

Formas de invocação de métodos

Ligação ou acoplamento prematuro

Early binding

Ligação ou acoplamento tardio

Late binding

Dynamic binding

Run-time binding

Virtual Method Invocation

Formas de invocação de métodos

Ligação ou acoplamento prematuro

Acontece quando o método a ser invocado é em

Tempo de compilação

Ligação ou acoplamento tardio

Acontece quando o método a ser invocado é em

Tempo de execução

Ligação tardia em Java

- **Demonstra** um **aspecto** das **características importantes** do **polimorfismo** na **Linguagem Java**
- A **JVM**:
  - **Chama** o **método adequado** para o **objeto**
    - Que é **referenciado** pela **variável-objeto**
  - **Não chama** o **método** que é **definido** pelo **tipo** da **variável-objeto**
- Exemplo:
  - Mamifero variavelObjeto = new Macaco();
  - variavelObjeto.locomoverSe(); // Chama o método sobrescrito na subclasse Macaco.

Ligação tardia em Java

Todas as invocações de métodos ocorrem através de ligação tardia

Exceto nos casos

Métodos declarados como

Construtores

private

final

static

Superclasse abstrata Mamifero

- public abstract class Mamifero {
  - public abstract String locomoverSe();
- }

Subclasses Macaco e Baleia

- public class Macaco extends Mamifero {
  - @Override
  - public String locomoverSe() {
    - return "Pulando de galho em galho.";
  - }
- }

- public class Baleia extends Mamifero {
  - @Override
  - public String locomoverSe() {
    - return "Nadando.";
  - }
- }

Subclasse Homem e sua subclasse HomemX

- public class Homem extends Mamifero {
  - @Override
  - public String locomoverSe() {
    - return "Andando.";
  - }
- }

- public class HomemX extends Homem {
  - @Override
  - public String locomoverSe() {
    - return "Voando.";
  - }
- }

Exemplo de classe executável

- public class ExemploPolimorfismo {
  - public static void main(String[] args) {
    - Mamifero variavelObjeto;

    - System.out.println("Macaco **********");
    - variavelObjeto = new Macaco();
    - System.out.println("Locomoção: " + variavelObjeto.locomoverSe());

    - System.out.println("Baleia **********");
    - variavelObjeto = new Baleia();
    - System.out.println("Locomoção: " + variavelObjeto.locomoverSe());

Exemplo de classe executável (continuação)

```
        System.out.println("Homem **********************************************");
        variavelObjeto = new Homem();
        System.out.println("Locomoção: " + variavelObjeto.locomoverSe());

        System.out.println("Homem X ********************************************");
        variavelObjeto = new HomemX();
        System.out.println("Locomoção: " + variavelObjeto.locomoverSe());

        System.out.println("Variável-objeto de uma subclasse apontado para um objeto da
        superclasse");
        System.out.println("Erro em tempo de compilação: tipos incompatíveis");
        //HomemX variavelObjeto2 = new Homem();
    }
}
```

Resultado da execução da classe executável

- Macaco **********
- Locomoção: Pulando de galho em galho.
- Baleia **********
- Locomoção: Nadando.
- Homem **********
- Locomoção: Andando.
- Homem X **********
- Locomoção: Voando.

Resultado da execução da classe executável

- Variável-objeto de uma subclasse apontado para um objeto da superclasse
- Erro em tempo de compilação: tipos incompatíveis

Questões de concursos

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

- O polimorfismo ocorre quando a mesma operação é construída em uma mesma classe ou quando o método da subclasse sobrepõe-se ao método da superclasse.

Questões de concursos

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

- O polimorfismo ocorre quando a mesma operação é construída em uma mesma classe ou quando o método da subclasse sobrepõe-se ao método da superclasse.
  - Gabarito: **CERTO**.

Questões de concursos

[IBADE 2020 Prefeitura de Santa Luzia D`Oeste/RO – Analista de Sistemas]
Qual saída é esperada para o código Java a seguir?

```
class Animal{
   public void andar(){
      System.out.print("Animal anda. ");
   }
   public void pensar(){
      System.out.println("Animal pensa. ");
   }
}

class Humano extends Animal{
   public void pensar(){
      System.out.print("Humano pensa. ");
   }
}

public class Main {
   public static void main(String[] args) {
      Animal animal = new Humano();
      animal.andar();
      animal.pensar();
   }
}
```

Comentários

- Código:
  - class Animal {
    - public void andar() {
      - System.out.print("Animal anda. ");
    - }
    - public void pensar() {
      - System.out.println("Animal pensa. ");
    - }
  - }

- Código (continuação):
  - class Humano extends Animal {
    - public void pensar() {
      - System.out.print("Humano pensa. ");
    - }
  - }

Comentários

- Código (continuação):
  - public class Main {
    - public static void main(String[] args) {
      - Animal animal = new Humano();
      - animal.andar();
      - animal.pensar();
    - }
  - }

- Resultado da execução:
  - Animal anda. Humano pensa.

Questões de concursos

[IBADE 2020 Prefeitura de Santa Luzia D`Oeste/RO – Analista de Sistemas] Qual saída é esperada para o código Java a seguir?

- [A] Animal anda. Animal pensa..
- [B] Animal anda. Animal pensa. Humano pensa..
- [C] Humano anda. Humano pensa..
- [D] Humano anda. Animal pensa..
- [E] Animal anda. Humano pensa..

Questões de concursos

[IBADE 2020 Prefeitura de Santa Luzia D`Oeste/RO – Analista de Sistemas] Qual saída é esperada para o código Java a seguir?

- [A] Animal anda. Animal pensa..
- [B] Animal anda. Animal pensa. Humano pensa..
- [C] Humano anda. Humano pensa..
- [D] Humano anda. Animal pensa..
- **[E] Animal anda. Humano pensa..**

Questões de concursos

[CESGRANRIO 2014 Banco da Amazônia – Técnico Científico – Analise de Sistemas] Sejam as seguintes classes Java, que ocupam arquivos distintos:

---------- arquivo V1.java ----------

```
public class V1 {
        int v[]={2,3,1,4,2,5,3,8,2,3};

        int mv1() {
                int s=0;
                for(int i=0;i<v.length;i+=2)
                        s+=v[i];
                return s;
        }

        float mv2(int x) {
                return x+mv1();
        }

        float mv2(float x) {
                return 3+x*mv1();
        }
}
```

Questões de concursos

[CESGRANRIO 2014 Banco da Amazônia – Técnico Científico – Analise de Sistemas] Sejam as seguintes classes Java, que ocupam arquivos distintos:

```
---------- arquivo V2.java ----------

public class V2 extends V1 {
        int mv1() {
                int s=0;
                for(int i=1;i<v.length;i+=2)
                        s+=v[i];
                return s;
        }

        float mv2(float x) {
                return x*mv1();
        }
}
```

Questões de concursos

[CESGRANRIO 2014 Banco da Amazônia – Técnico Científico – Analise de Sistemas] Sejam as seguintes classes Java, que ocupam arquivos distintos:

```
---------- arquivo QX.java ----------

public class QX {

        public static void main(String[] args) {
                V1 a=new V2();

                System.out.printf("%.1f\n",a.mv2(2));
        }
}
```

Comentários

- Código:
  - public class V1 {
    - int v[] = {2, 3, 1, 4, 2, 5, 3, 8, 2, 3};
    - int mv1() {
      - int s = 0;
      - for(int i = 0; i < v.length; i += 2)
        - s += v[i];
      - return s;
    - }
    - float mv2(int x) {
      - return x + mv1();
    - }
    - float mv2(float x) {
      - return 3 + x * mv1();
    - }
  - }

Comentários

- Código:
  - public class QX {
    - public static void main(String []args) {
      - V1 a = new V1();
      - System.out.println("mv1: " + a.mv1());
      - System.out.println("mv2 int: " + a.mv2(2));
      - System.out.println("mv2 float: " + a.mv2(2.0f));
    - }
  - }

- Resultado da execução:
  - mv1: 10
  - mv2 int: 12.0
  - mv2 float: 23.0

Comentários

- Código:
  - public class V2 extends V1 {
    - int mv1() {
      - int s = 0;
      - for(int i = 1; i < v.length; i += 2)
        - s += v[i];
      - return s;
    - }
    - float mv2(float x) {
      - return x * mv1();
    - }
  - }

Comentários

- Código:
  - public class QX2 {
    - public static void main(String []args) {
      - V1 a = new V2();
      - System.out.println("mv1: " + a.mv1());
      - System.out.println("mv2 int: " + a.mv2(2));
      - System.out.println("mv2 float: " + a.mv2(2.0f));
    - }
  - }

- Resultado da execução:
  - mv1: 23
  - mv2 int: 25.0
  - mv2 float: 46.0

Questões de concursos

[CESGRANRIO 2014 Banco da Amazônia – Técnico Científico – Analise de Sistemas] O que será exibido no console quando o método main() for executado?

- [A] 12,0
- [B] 20,0
- [C] 23,0
- [D] 25,0
- [E] 46,0

Questões de concursos

[CESGRANRIO 2014 Banco da Amazônia – Técnico Científico – Analise de Sistemas] O que será exibido no console quando o método main() for executado?

- [A] 12,0
- [B] 20,0
- [C] 23,0
- **[D] 25,0**
- [E] 46,0

Classes e métodos abstratos

Classes abstratas

- Podem ou não incluir métodos abstratos
- Não podem ser instanciadas
  - Mas podem ter subclasses
- Podem ter:
  - Atributos estáticos
  - Métodos estáticos
- Na linguagem Java:
  - São classes declaradas com o modificador abstract

Classes que herdam de classe abstrata

Classes descendentes concretas

Implementam todos os métodos abstratos de uma classe abstrata

Classes descendentes abstratas

Podem não implementar todos os métodos abstratos de uma classe abstrata

As suas classes descendentes devem implementar os métodos

Métodos abstratos

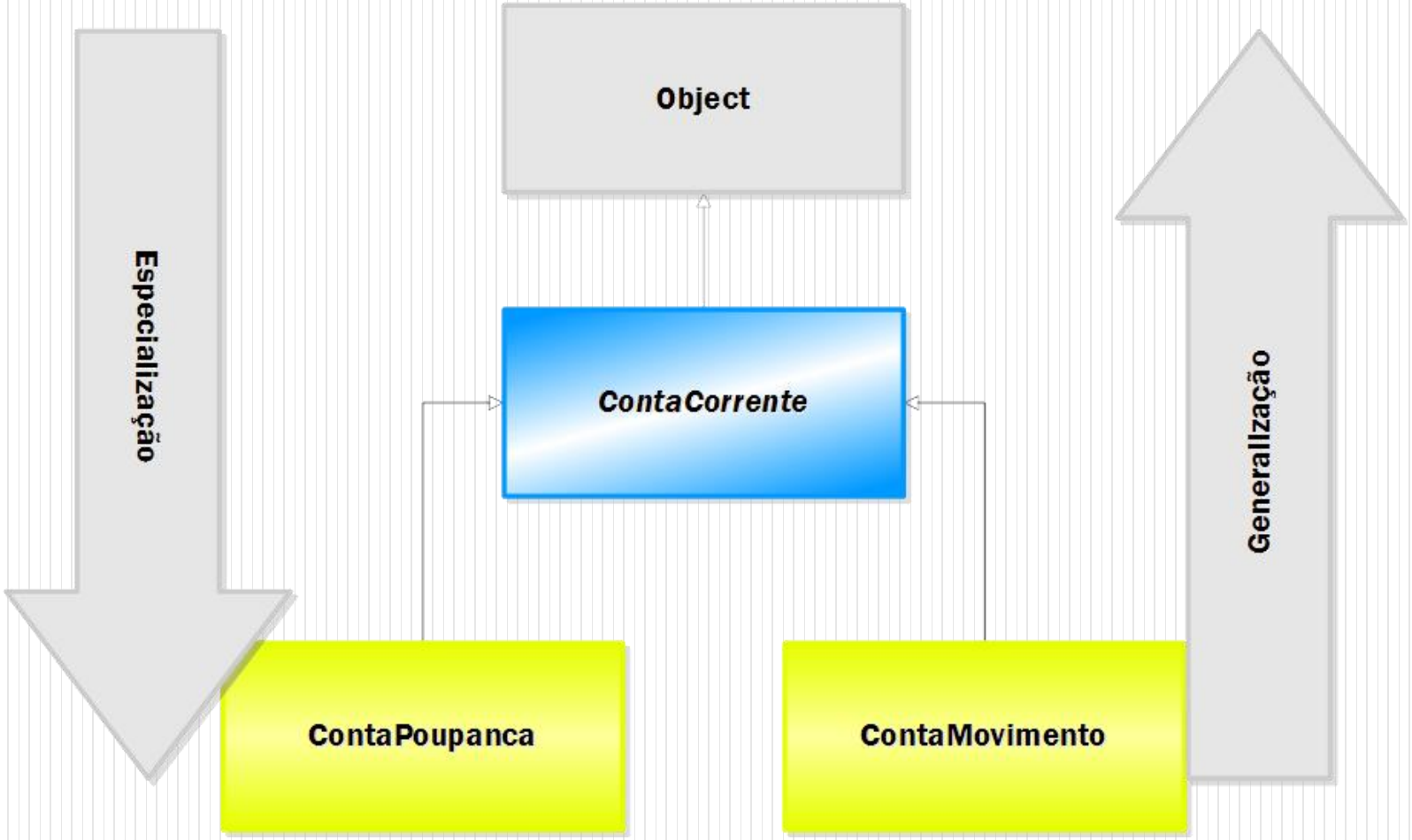
- São **métodos declarados sem implementação**
- Na **linguagem Java**:
  - Cão **métodos declarados**:
    - Com o **modificador abstract**
    - **Sem chaves**
    - **Seguidos** por um **ponto e vírgula**
- São **implementados** por **alguma subclasse** da **classe abstrata** que **contém esses métodos**
  - Caso uma subclasse não implemente o método abstrato:
    - Ela será também abstrata

1

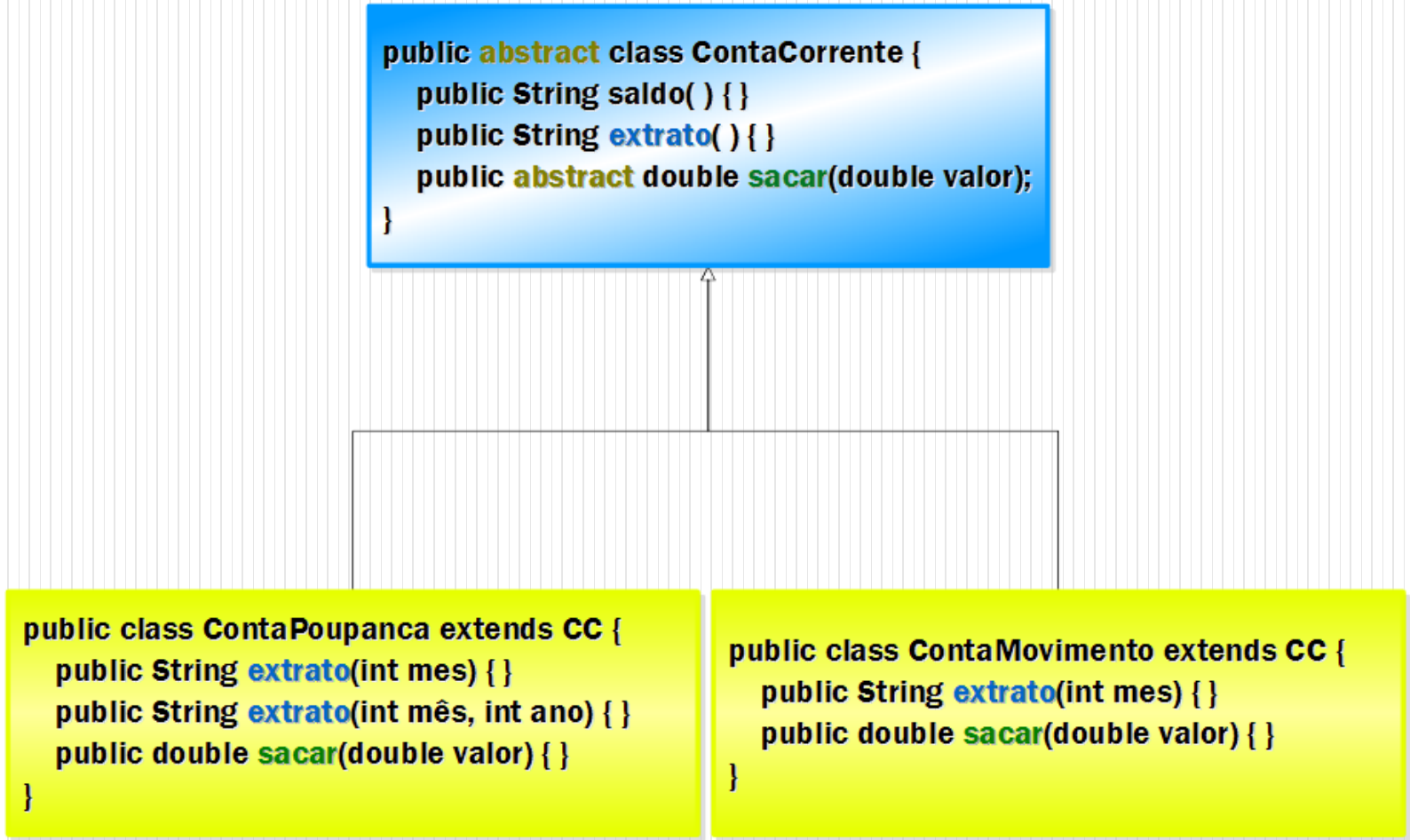
Exemplo

```
public abstract class ContaCorrente {
    public abstract double sacar(double valor);
}
public class ContaPoupanca extends ContaCorrente {
    @Override
    public double sacar(double valor) {
        // Implementação do método.
    }
}
```

Esquema


Object
ContaCorrente
ContaPoupanca
ContaMovimento
Especialização
Generalização

Esquema (continuação)



Exemplo

- ContaCorrente variavelObjetoConta;
- // Polimorfismo universal de inclusão.
- variavelObjetoConta = new **ContaPoupanca**();
- // Polimorfismo dinâmico, universal ou verdadeiro.
- // Método sobrescrito sacar de ContaPoupanca.
- System.out.println(variavelObjetoConta.**sacar**(2500));
- // Polimorfismo ad–hoc ou estático.
- // Método sobrecarregado extrato de ContaPoupanca.
- System.out.println(variavelObjetoConta.extrato(3));

Questões de concursos

[IDECAN 2020 IFRR – Informática] A Orientação a Objetos (OO) é um paradigma de programação para o qual "tudo é um objeto", sendo Java uma das principais linguagens que implementam esse paradigma. Em relação à linguagem Java e à OO, analise as seguintes afirmativas:

- [II] Uma classe Java abstrata obrigatoriamente deve possuir um ou mais métodos abstratos.

Questões de concursos

[IDECAN 2020 IFRR – Informática] A Orientação a Objetos (OO) é um paradigma de programação para o qual "tudo é um objeto", sendo Java uma das principais linguagens que implementam esse paradigma. Em relação à linguagem Java e à OO, analise as seguintes afirmativas:

- [II] Uma classe Java abstrata ~~**obrigatoriamente deve**~~ **não precisa** possuir um ou mais métodos abstratos.
  - Gabarito: **ERRADO**.

Questões de concursos

[COMPERVE 2020 TJ/RN – Analista de Sistemas Sênior] Sobre a linguagem de programação Java, analise as afirmativas abaixo.

- [I] É possível instanciar uma classe abstrata.

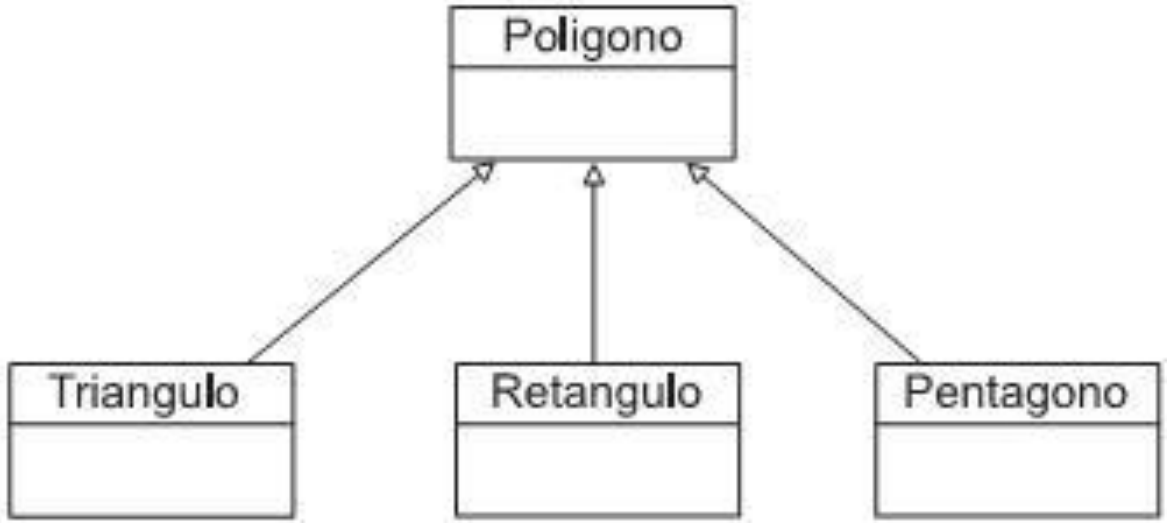
Questões de concursos

[COMPERVE 2020 TJ/RN – Analista de Sistemas Sênior] Sobre a linguagem de programação Java, analise as afirmativas abaixo.

- [I] **Não** É possível instanciar uma classe abstrata.
  - Gabarito: **ERRADO**.

Questões de concursos

[CESGRANRIO 2011 FINEP – Analista – Desenvolvimento de Sistemas] O diagrama de classes UML 2.3 abaixo contém parte das classes de uma aplicação usada no ensino de geometria.



Questões de concursos

[CESGRANRIO 2011 FINEP – Analista – Desenvolvimento de Sistemas] Caso essa aplicação permita instanciar apenas triângulos, retângulos e pentágonos, como deveria ser definida a classe Polígono em um programa Java?

- [A] public final class Poligono { }
- [B] public interface class Poligono { }
- [C] public static class Poligono { }
- [D] public abstract class Poligono { }
- [E] public class Poligono { }

Questões de concursos

[CESGRANRIO 2011 FINEP – Analista – Desenvolvimento de Sistemas] Caso essa aplicação permita instanciar apenas triângulos, retângulos e pentágonos, como deveria ser definida a classe Polígono em um programa Java?

- [A] public final class Poligono { }
- [B] public interface class Poligono { }
- [C] public static class Poligono { }
- **[D] public abstract class Poligono { }**
- [E] public class Poligono { }

Interfaces

Conceituação

- **Não são classes**
  - São um **conjunto de requisitos** para que **classes possam se adequar a ela**
    - "Se a classe estiver em conformidade com uma interface, então um determinado serviço será realizado"
- São como um **contrato** ou **padrão**
  - **Descrevem o que** as **classes devem fazer**
    - **Sem especificar como** devem fazer

Conceituação

- São também um **tipo de referência como** uma **classe**
  - Mas podem conter apenas:
    - Atributos estáticos
    - Assinaturas de métodos
    - Tipos aninhados
- **Não possuem**:
  - **Atributos de instância**
  - **Métodos implementados**

Conceituação

- **Não podem ser instanciadas**
  - Apenas podem ser:
    - **Implementadas** por **classes**
    - **Estendidas** por **outras insterfaces**
- Na **linguagem Java**, **todos os métodos** de uma **interface** são **automaticamente public**
  - Não é necessário usar esse modificador

Interfaces e hierarquia de classes

Não fazem parte de nenhuma hierarquia de classes

Embora as interfaces trabalhem em combinação com elas

Interfaces e herança múltipla

Fornecem uma alternativa para herança múltipla na linguagem Java

Uma classe pode herdar de apenas uma classe

Mas pode implementar mais de uma interface

Uma interface pode herdar várias outras interfaces

Interfaces e tipos dos objetos

Os objetos podem ter vários tipos

- Tipo de sua própria classe
- Tipos das suas classes ancestrais
- Tipos de todas as interfaces que elas implementam

Declaração de interfaces na Linguagem Java

- **Modificadores de acesso** de **nível superior**:
  - **public**
  - **default**
- Palavra-chave **interface**
- **Nome da interface**
  - **Começa** com **letra maiúscula** por **convenção**
  - A **nomeação segue** as **regras de nomeação de variáveis**

Declaração de interfaces na Linguagem Java

- **Lista de nomes das interfaces** a **serem** **estendidas**
  - Se houver
  - A lista é **precedida** pela palavra-chave **implements**
  - Os nomes são separados por vírgulas
  - Uma **interface** pode **estender** **mais de uma** **interface**

Declaração de interfaces na Linguagem Java

- **Corpo da interface**
  - É **envolvido** por **chaves {}**
  - **Contém declarações de métodos**
  - **Pode conter declarações de variáveis finais**
    - **Todas as variáveis** em uma **interface** são **implicitamente**:
      - **public**
      - **static**
      - **final**
    - Esses modificadores podem ser omitidos

Declaração de interfaces na Linguagem Java

- **Métodos**
  - A **declaração seguida** por um **ponto e vírgula**
  - **Não há chaves {}**
    - Não há implementação dos métodos de interface
  - **Todos os métodos declarados em uma interface** são **implicitamente public**
    - Esse modificador pode ser omitido

Classes que implementam uma interface

Classes concretas

Devem implementar todos os métodos de uma interface

Classes abstratas

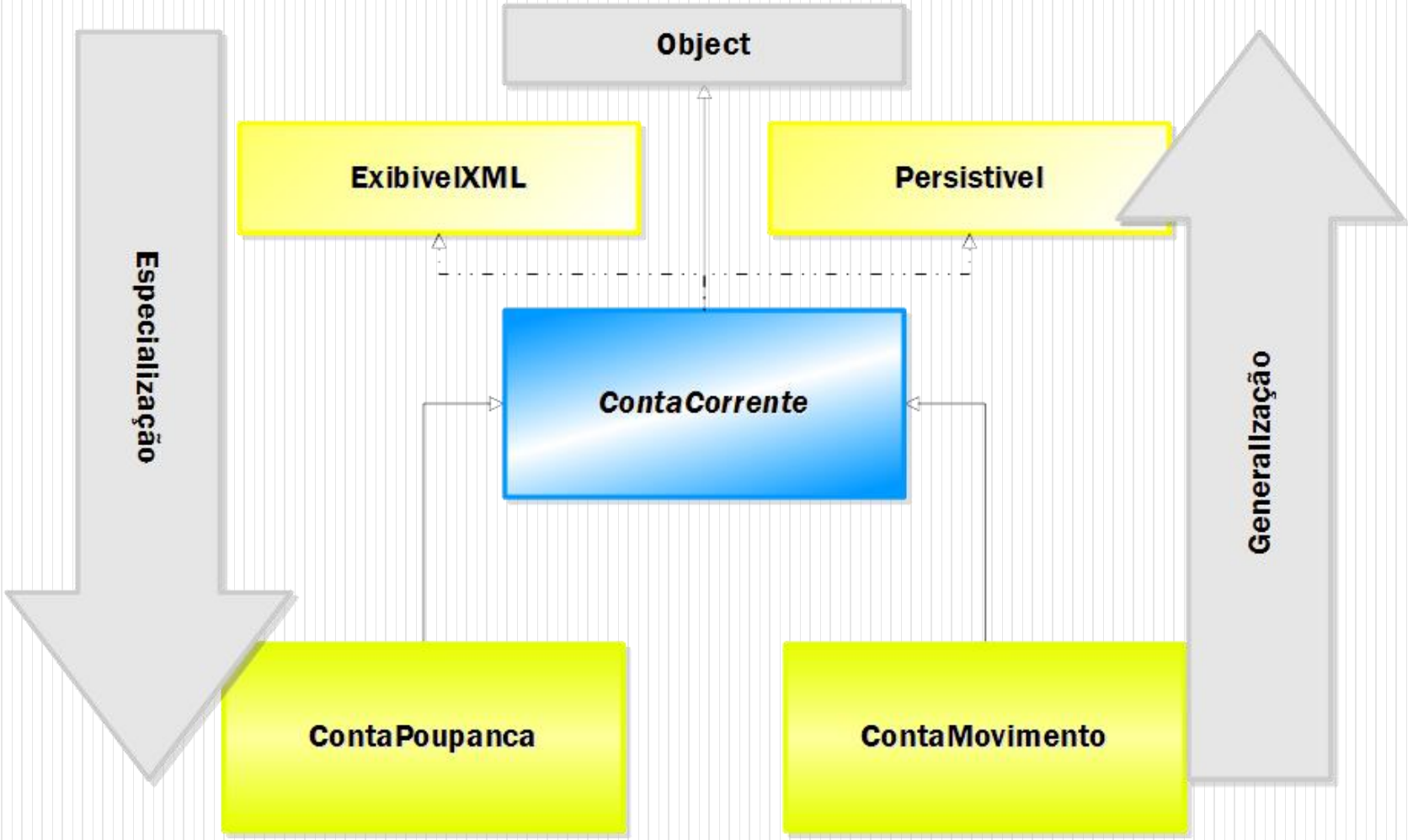
Podem não implementar todos os métodos de uma interface

As suas classes descendentes devem implementar os métodos

Exemplo

- Interface:
  - public **interface Persistivel** {
    - String **insere**();
  - }
- Classe que implementa a interface:
  - public class **Classe implements Persistivel** {
    - @Override
    - public String **insere**() {
      - return "INSERT INTO Tabela VALUES (Valor1, Valor2);";
    - }
  - }

Exemplo (continuação)



Exemplo (continuação)

- ContaCorrente variavelObjetoConta;
- // Polimorfismo universal de inclusão.
- variavelObjetoConta = new **ContaPoupanca**();
- // Polimorfismo dinâmico, universal ou verdadeiro.
- // Método sobrescrito sacar de ContaPoupanca.
- System.out.println(variavelObjetoConta.**sacar**(2500));
- // Polimorfismo ad-hoc ou estático.
- // Método sobrecarregado extrato de ContaPoupanca.
- System.out.println(variavelObjetoConta.extrato(3));

Exemplo (continuação)

- Persistivel variavelObjetoPersistivel;
- // Polimorfismo universal de inclusão.
- variavelObjetoPersistivel = new **ContaMovimento**();
- // Polimorfismo dinâmico, universal ou verdadeiro.
- // Método sobrescrito insere de Persistivel.
- System.out.println(variavelObjetoPersistivel.**insere**());

Questões de concursos

[IDECAN 2020 IFRR – Informática] A Orientação a Objetos (OO) é um paradigma de programação para o qual "tudo é um objeto", sendo Java uma das principais linguagens que implementam esse paradigma. Em relação à linguagem Java e à OO, analise as seguintes afirmativas:

- [I] Uma classe Java pode implementar mais de uma interface Java.

Questões de concursos

[IDECAN 2020 IFRR – Informática] A Orientação a Objetos (OO) é um paradigma de programação para o qual "tudo é um objeto", sendo Java uma das principais linguagens que implementam esse paradigma. Em relação à linguagem Java e à OO, analise as seguintes afirmativas:

- [I] Uma classe Java pode implementar mais de uma interface Java.
  - Gabarito: **CERTO**.

Questões de concursos

[COMPERVE 2020 TJ/RN – Analista de Sistemas Sênior] S Uma interface na linguagem Java é apenas um contrato que a classe deve cumprir com a interface que a implementa. Sobre interfaces na linguagem Java, é correto afirmar:

- [A] as variáveis são implicitamente public static final.
- [B] elas fazem parte de herança.
- [C] uma interface pode implementar uma class.
- [D] as variáveis e métodos podem ter qualquer modificador de acesso.

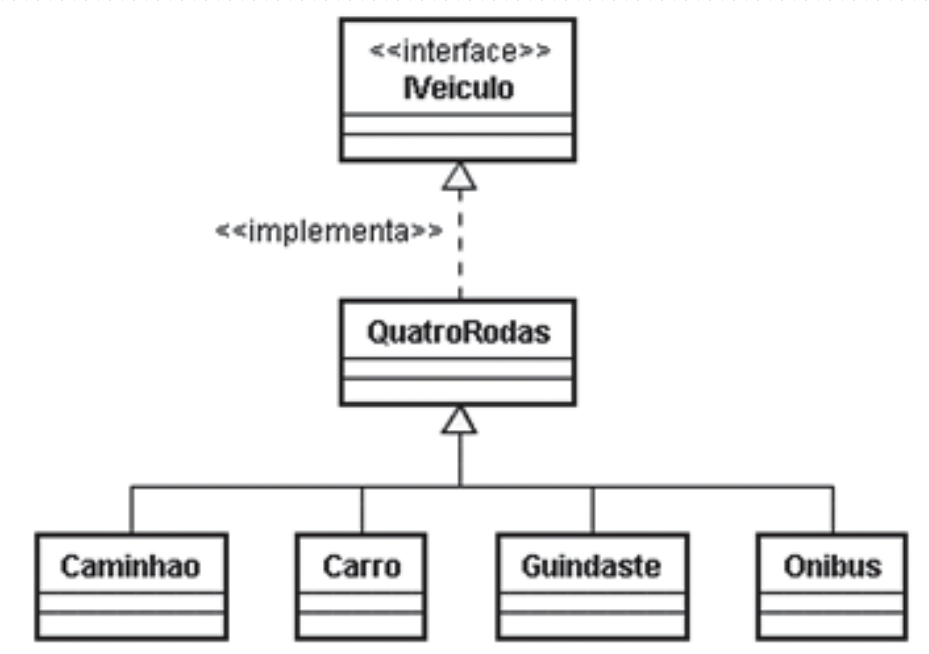
Questões de concursos

[COMPERVE 2020 TJ/RN – Analista de Sistemas Sênior] S Uma interface na linguagem Java é apenas um contrato que a classe deve cumprir com a interface que a implementa. Sobre interfaces na linguagem Java, é correto afirmar:

- **[A] as variáveis são implicitamente public static final.**
- [B] elas **não** fazem parte de herança.
- [C] uma interface **não** pode implementar uma class.
- [D] as variáveis e métodos **não** podem ter qualquer modificador de acesso.

Questões de concursos

[CESGRANRIO 2018 Banco da Amazônia – Técnico Científico – Tecnologia da Informação] Considere a hierarquia de classes mostrada na Figura a seguir:



Questões de concursos

[CESGRANRIO 2018 Banco da Amazônia – Técnico Científico – Tecnologia da Informação] Considere agora o seguinte fragmento de código Java baseado na hierarquia anterior:

```
public class Main {
    public static void main(String[] args) {
        Caminhao caminhao = new Caminhao();      // linha 1
        IVeiculo veiculo = caminhao;             // linha 2
        QuatroRodas fw = new Guindaste();        // linha 3
        fw = veiculo;                            // linha 4
        veiculo = fw;                            // linha 5
    }
}
```

Questões de concursos

[CESGRANRIO 2018 Banco da Amazônia – Técnico Científico – Tecnologia da Informação] A linha marcada por um comentário que causará um erro em tempo de compilação é a linha

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

Questões de concursos

[CESGRANRIO 2018 Banco da Amazônia – Técnico Científico – Tecnologia da Informação] A linha marcada por um comentário que causará um erro em tempo de compilação é a linha

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

Mais da Linguagem Java

Tratamento de exceções

Visão geral

Tratamento

- try { ... }
- catch (TipoExcecao e) { ... }
- finally { ... }

Propagação

- public int metodo() throws TipoExcecao { ... }

Lançamento

- if (condição)
  - throw new TipoExcecao(“Alguma coisa não deu certo”);

Criação de exceções

- public class MinhaExcecao extends Throwable { ... }

Cláusula try

Envolve e tenta executar um código que

Pode causar exceção

Lança uma exceção

Deve ser seguida por

Um ou mais blocos catch

Um bloco finally

Cláusulas catch

Recebem um objeto de exceção como parâmetro de exceção

Apenas uma das cláusulas catch é executada

Cláusula finally

Contém instruções que devem ser executadas

Independentemente da ocorrência ou não de exceções

Sintaxe simples

- **try** {
  - // Executa até linha onde ocorrer exceção.
- } **catch** (**TipoExcecao** **e**) {
  - // Executa somente se ocorrer TipoExcecao.
- }
- // Executa se exceção for capturada ou se não ocorrer.

Sintaxe com mais de um bloco catch

- **try** {
  - // Executa até linha onde ocorrer exceção.
- } **catch** (**TipoExcecao1 e**) {
  - // Executa somente se ocorrer TipoExcecao1.
- } **catch** (**TipoExcecao2 e**) {
  - // Executa somente se ocorrer TipoExcecao2.
- }
- // Executa se exceção for capturada ou se não ocorrer.

Sintaxe com finally

- **try** {
  - // Executa até linha onde ocorrer exceção.
- } **catch** (**TipoExcecao1 e**) {
  - // Executa somente se ocorrer TipoExcecao1.
- } **catch** (**TipoExcecao2 e**) {
  - // Executa somente se ocorrer TipoExcecao2.
- } **finally** {
  - // Executa sempre.
- }
- // Executa se exceção for capturada ou se não ocorrer.

Fluxo do tratamento de exceções

Se o código da cláusula try lançar uma exceção de uma classe especificada em um das cláusulas catch

O programa

Pula o restante do código da cláusula try

Executa o código da cláusula catch correspondente à exceção

Se o código da cláusula try não lançar nenhuma exceção

O programa

Executa normalmente o código da cláusula try

Pula a(s) cláusula(s) catch

Se houver a cláusula finally em ambos os casos

O código dela será executado

Questões de concursos

[Instituto AOCP 2021 ITEP/RN – Perito Criminal – Computação] Assinale a alternativa correta sobre as características da linguagem de programação Java. (Marque CERTO ou ERRADO o texto da letra)

- [B] Não possui tratamento de exceções.

Questões de concursos

[Instituto AOCP 2021 ITEP/RN – Perito Criminal – Computação] Assinale a alternativa correta sobre as características da linguagem de programação Java. (Marque CERTO ou ERRADO o texto da letra)

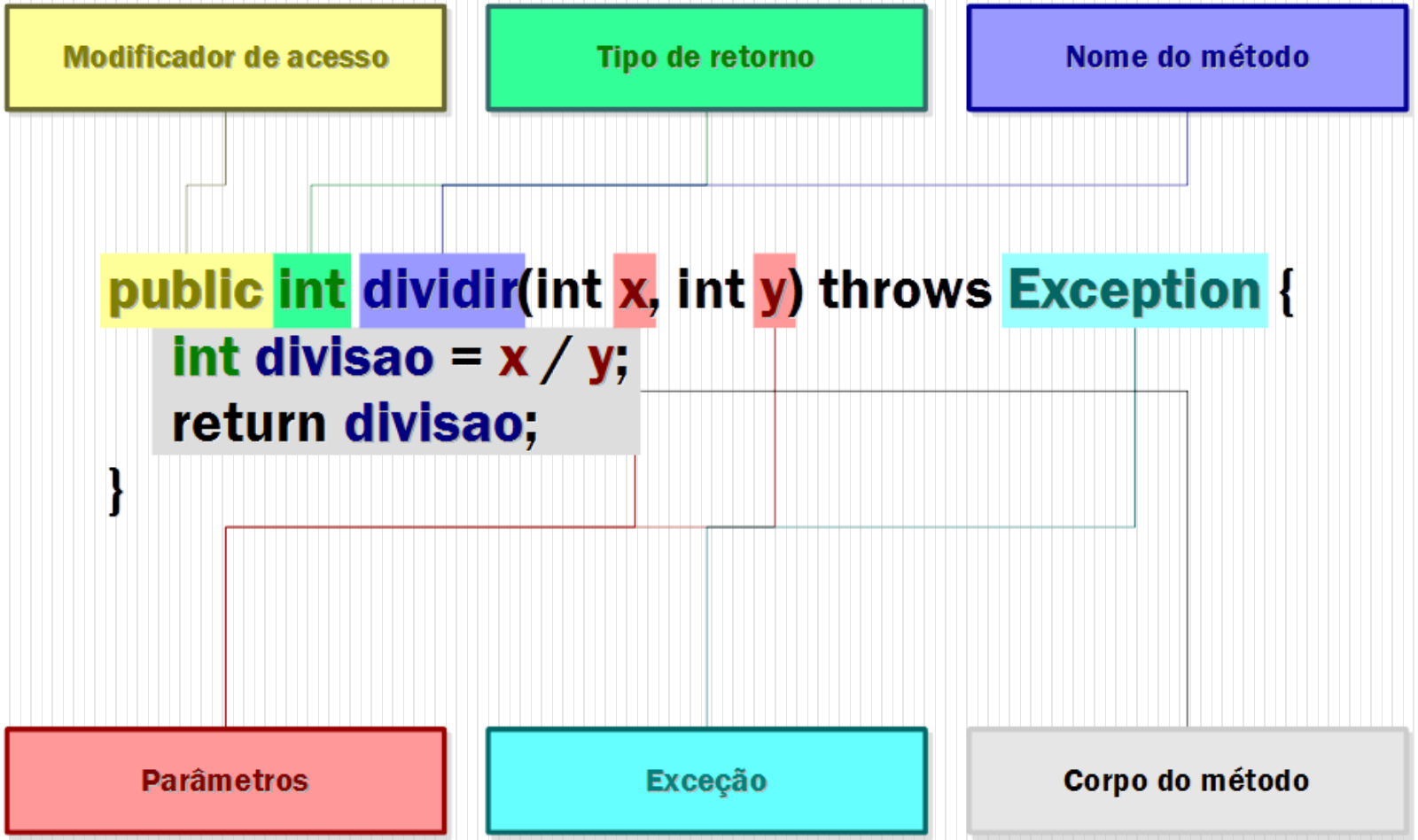
- [B] ~~**Não**~~ possui tratamento de exceções.
  - Gabarito: **ERRADO**.

Questões de concursos

[Instituto AOCP 2019 IBGE – Analista Censitário – Análise de Sistemas – Desenvolvimento de Aplicações] Na linguagem Java, o tratamento de exceções ajuda o usuário a entender tanto o tipo de dado esperado quanto um erro informado pelo programa. Sabendo disso, assinale a alternativa que apresenta corretamente a cláusula que especifica as exceções que um método pode apresentar se ocorrerem problemas, devendo essa cláusula aparecer após a lista de parâmetros e antes do corpo do método.

- [A] Exception.
- [B] Treatment.
- [C] Error.
- [D] Catch.
- [E] Throws.

Comentários



Questões de concursos

[Instituto AOCP 2019 IBGE – Analista Censitário – Análise de Sistemas – Desenvolvimento de Aplicações] Na linguagem Java, o tratamento de exceções ajuda o usuário a entender tanto o tipo de dado esperado quanto um erro informado pelo programa. Sabendo disso, assinale a alternativa que apresenta corretamente a cláusula que especifica as exceções que um método pode apresentar se ocorrerem problemas, devendo essa cláusula aparecer após a lista de parâmetros e antes do corpo do método.

- [A] Exception.
- [B] Treatment.
- [C] Error.
- [D] Catch.
- **[E] Throws.**

Questões de concursos

[VUNESP 2019 Câmara de Sertãozinho/SP – Auxiliar Legislativo – Informática] Na linguagem Java, a sintaxe correta de um bloco de controle de exceção é:

- [A] try { // código a ser executado } catch (TipoExcecao nomeExcecao) { // tratamento da exceção }
- [B] try { // código a ser executado } catch (nomeExcecao: TipoExcecao) { // tratamento da exceção }
- [C] try { // código a ser executado } except { // tratamento da exceção }
- [D] if (TipoExcecao) { // código a ser executado } else { // tratamento da exceção }
- [E] switch (nomeExcecao){ default: // código a ser executado break; case TipoExcecao: // tratamento da exceção break; }

Questões de concursos

[VUNESP 2019 Câmara de Sertãozinho/SP – Auxiliar Legislativo – Informática] Na linguagem Java, a sintaxe correta de um bloco de controle de exceção é:

- **[A] try { // código a ser executado } catch (TipoExcecao nomeExcecao) { // tratamento da exceção }**
- [B] try { // código a ser executado } catch (nomeExcecao: TipoExcecao) { // tratamento da exceção }
- [C] try { // código a ser executado } except { // tratamento da exceção }
- [D] if (TipoExcecao) { // código a ser executado } else { // tratamento da exceção }
- [E] switch (nomeExcecao){ default: // código a ser executado break; case TipoExcecao: // tratamento da exceção break; }

Números e Strings

Strings

O que é uma string

- É uma **sequência de caracteres** que **provêm** de **algum alfabeto** (conjunto de todos os caracteres possíveis)
  - **Cada caractere que compõe uma string pode ser referenciado** por um **índice**
    - O índice de um caractere é número de caracteres antes daquele
    - O primeiro caractere possui o índice 0
  - **Unicode** é o **alfabeto** usado para **definir strings em Java**
- É um **objeto** em **Java**
- A **plataforma Java provê classe java.lang.String** para **manipular strings**

Literais de string

- **Consistem** em **zero** ou **mais caracteres**
- **Devem vir** entre **aspas duplas**
  - Diferente dos literais de caracteres que devem vir entre aspas simples
- **“a”** é **diferente** de **'a'**
  - O **primeiro representa** um **conjunto unitário de caractere**
  - O **segundo representa** apenas um **elemento de caractere**

Literais de string e literais de caractere

Não podemos atribuir

Um literal de String para uma variável do tipo char	Um literal de caractere para uma variável do tipo String
char aChar = “a”;	String aString = 'a';
O compilador acusará um erro	O compilador acusará um erro

Literais de string e literais de caractere

String aString = a

char aChar = a

Objeto String

- É **imutável**
  - Ele não poderá ser alterado ao ser criado
- **Formas de criação de um objeto String**:
  - **Forma mais direta**:
    - String nome = “Kal-El Gildo Araújo”;
    - “Kal-El Gildo Araújo” é um literal de string
  - **Operador new**:
    - String nome = new String(“Rogério Gildo Araújo”);

Questões de concursos

[Instituto Consulplan 2021 Prefeitura de Colômbia/SP – Técnico em Tecnologia da Informação e Sistemas] Sobre a linguagem Java, marque V para as afirmativas verdadeiras e F para as falsas.

- A classe String é um exemplo de uma classe final.

Questões de concursos

[Instituto Consulplan 2021 Prefeitura de Colômbia/SP – Técnico em Tecnologia da Informação e Sistemas] Sobre a linguagem Java, marque V para as afirmativas verdadeiras e F para as falsas.

- A classe String é um exemplo de uma classe final.
  - Gabarito: **CERTO**.
  - A classe String não pode se estendida

Questões de concursos

[VUNESP 2020 Prefeitura de Ilhabela/SP – Analista – Tecnologia da Informação e Comunicação] Considere o programa Java a seguir:

```
public class Classe {
    public static void main(String[] args) {
        String string = 'xyzk';
        System.out.println(string);
    }
}
```

Questões de concursos

[VUNESP 2020 Prefeitura de Ilhabela/SP – Analista – Tecnologia da Informação e Comunicação] Esse programa não pode ser compilado, pois

- [A] variáveis não podem ser nomeadas com a palavra “string”.
- [B] o método main não pode ser declarado como “static”.
- [C] cadeias de caracteres devem ser delimitadas por aspas duplas.
- [D] não foi especificado um valor de retorno para o método.
- [E] o método main não pode ser declarado como “public”.

Questões de concursos

[VUNESP 2020 Prefeitura de Ilhabela/SP – Analista – Tecnologia da Informação e Comunicação] Esse programa não pode ser compilado, pois

- [A] variáveis ~~não~~ podem ser nomeadas com a palavra “string”.
- [B] o método main ~~não pode~~ **deve** ser declarado como “static”.
- **[C] cadeias de caracteres devem ser delimitadas por aspas duplas.**
- [D] não foi especificado um valor de retorno para o método.
- [E] o método main ~~não pode~~ **deve** ser declarado como “public”.

Expressões Lambda

Conceituação

Foram adicionadas no Java 8

São pequenos blocos de código que

- Recebem parâmetros
- Retorna um valor

São semelhantes aos métodos, mas

- Não precisam de um nome
- Podem ser implementadas diretamente no corpo de um método

Sintaxe

(parâmetro1, parâmetro2, parâmetroN) -> expressão

Componentes

Lista de parâmetro	Token de seta	Corpo
Pode estar vazia	É usado para vincular	Contém expressões e declarações para a expressão lambda
	A lista de parâmetros	O corpo de expressão

Exemplo 1

- Código:
  - import java.util.ArrayList;
  - public class ExemploLambda1 {
    - public static void main(String[] args) {
      - ArrayList<Integer> nums = new ArrayList<Integer>();
      - nums.add(5);
      - nums.add(9);
      - nums.add(8);
      - nums.add(1);
      - nums.forEach((n) -> { System.out.println(n); });
    - }
  - }

- Resultado da execução:
  - 5
  - 9
  - 8
  - 1

Interfaces funcionais

Functional Interfaces

São todas as interfaces que possuem um método abstrato

Seguem o conceito de Single Abstract Method

Um método à ser implementado

SAM

Lambda e interfaces funcionais

Expressões lambda

Podem ser armazenadas em variáveis

Se o tipo da variável for uma interface funcional

Devem ter o mesmo número de parâmetros e o mesmo tipo de retorno do método da interface funcional

Lambda e interfaces funcionais

Java possui várias interfaces funcionais incorporadas

Por exemplo, java.util.function.Consumer

Usada por listas

Exemplo 2

- Código:
  - import java.util.ArrayList;
  - import java.util.function.Consumer;
  - public class ExemploLambda2 {
    - public static void main(String[] args) {
      - ArrayList<Integer> nums = new ArrayList<Integer>();
      - nums.add(5);
      - nums.add(9);
      - nums.add(8);
      - nums.add(1);
      - Consumer<Integer> metodo = (n) -> { System.out.println(n); };
      - nums.forEach(metodo);
    - }
  - }

- Resultado da execução:
  - 5
  - 9
  - 8
  - 1

Exemplo 3

- Código:
  - interface Formatar {
    - String run(String str);
  - }

Exemplo 3 (continuação)

- Código:
  - public class ExemploLambda3 {
    - public static void imprimir(String str, Formatar formato) {
      - System.out.println(formato.run(str));
    - }
    - ...
  - }

Exemplo 3 (continuação)

- Código:
  - public class ExemploLambda3 {
    - ...
    - public static void main(String[] args) {
      - Formatar tudoMaiusculo = (s) -> s.toUpperCase();
      - Formatar tudoMinusculo = (s) -> s.toLowerCase();
      - imprimir("Rogerão", tudoMaiusculo);
      - imprimir ("Rogerão", tudoMinusculo);
    - }
  - }

- Resultado da execução:
  - ROGERÃO
  - rogerão

Questões de concursos

[IDIB 2021 CRF/MS – Analista de Informática] Sobre as funções lambda em Java, assinale a alternativa correta.

(variável) -> (método)

- [A] Ela foi inserida no Java 7 e é utilizada para imprimir variáveis na tela.
- [B] Na versão do Java 8, ela não pode ser utilizada com Threads e Collections.
- [C] Foi criada no Java 8 para simplificar a criação de métodos sem declaração.
- [D] Foi definida para declarações de variáveis em tempo de execução.

Questões de concursos

[IDIB 2021 CRF/MS – Analista de Informática] Sobre as funções lambda em Java, assinale a alternativa correta.

(variável) -> (método)

- [A] Ela foi inserida no Java 7 e é utilizada para imprimir variáveis na tela.
- [B] Na versão do Java 8, ela não pode ser utilizada com Threads e Collections.
- **[C] Foi criada no Java 8 para simplificar a criação de métodos sem declaração.**
- [D] Foi definida para declarações de variáveis em tempo de execução.

Questões de concursos

[CESPE/CEBRASPE 2020 Ministério da Economia – Tecnologia da Informação – Desenvolvimento de Software] Considerando as linguagens de programação Java (versão 8 ou superior) e PHP (versão 7 ou superior), julgue o próximo item.

- Uma expressão lambda é usada principalmente para definir a implementação procedural de uma interface associativa.

Questões de concursos

[CESPE/CEBRASPE 2020 Ministério da Economia – Tecnologia da Informação – Desenvolvimento de Software] Considerando as linguagens de programação Java (versão 8 ou superior) e PHP (versão 7 ou superior), julgue o próximo item.

- Uma expressão lambda é usada principalmente para definir a implementação ~~**procedural**~~ de uma interface ~~**associativa**~~ funcional.
  - Gabarito: **ERRADO**.

Algoritmos em geral

Filas

Princípios

FIFO

First In First Out

Primeiro a entrar é o primeiro a sair

LILO

Last In Last Out

Último a entrar é o último a sair

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] A classe Queue a seguir é uma implementação parcial do tipo abstrato de dados Fila.

```
import java.util.ArrayList;

public class Queue<ELM> {

   private ArrayList<ELM> lst=new ArrayList<ELM>();

   public boolean isEmpty() {
      return lst.isEmpty();
   }

   public void enqueue(ELM s) {
   }

   public ELM dequeue() {
   }
}
```

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Nesse contexto, qual implementação dos métodos enqueue() e dequeue() completa a classe Queue, de modo que todos os elementos inseridos em uma fila possam ser recuperados de acordo com a propriedade FIFO?

- [A]

```
public void enqueue(ELM s) {
  lst.add(s);
}

public ELM dequeue() {
  if(!lst.isEmpty())
    return lst.get(0);
  else
    return null;
}
```

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Nesse contexto, qual implementação dos métodos enqueue() e dequeue() completa a classe Queue, de modo que todos os elementos inseridos em uma fila possam ser recuperados de acordo com a propriedade FIFO?

- [A]

```
public void enqueue(ELM s) {
  lst.add(s);
}

public ELM dequeue() {
  if(!lst.isEmpty())
    return lst.get(0);
  else
    return null;
}
```

- O método enqueue adiciona um elemento no final da lista
- O método dequeue:
  - Não retira o primeiro elemento
  - Apenas retorna o elemento

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Nesse contexto, qual implementação dos métodos enqueue() e dequeue() completa a classe Queue, de modo que todos os elementos inseridos em uma fila possam ser recuperados de acordo com a propriedade FIFO?

- [B]

```
public void enqueue(ELM s) {
  lst.add(0,s);
}

public ELM dequeue() {
  if(!lst.isEmpty())
    return lst.remove(lst.size()-1);
  else
    return null;
}
```

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Nesse contexto, qual implementação dos métodos enqueue() e dequeue() completa a classe Queue, de modo que todos os elementos inseridos em uma fila possam ser recuperados de acordo com a propriedade FIFO?

- **[B]**

```
public void enqueue(ELM s) {
  lst.add(0,s);
}

public ELM dequeue() {
  if(!lst.isEmpty())
    return lst.remove(lst.size()-1);
  else
    return null;
}
```

- O método enqueue adiciona um elemento no início da lista
- O método dequeue retira o último elemento da lista

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Nesse contexto, qual implementação dos métodos enqueue() e dequeue() completa a classe Queue, de modo que todos os elementos inseridos em uma fila possam ser recuperados de acordo com a propriedade FIFO?

- [C]

```
public void enqueue(ELM s) {
  lst.add(0,s);
}

public ELM dequeue() {
  if(!lst.isEmpty())
    return lst.remove(0);
  else
    return null;
}
```

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Nesse contexto, qual implementação dos métodos enqueue() e dequeue() completa a classe Queue, de modo que todos os elementos inseridos em uma fila possam ser recuperados de acordo com a propriedade FIFO?

- [C]

```
public void enqueue(ELM s) {
  lst.add(0,s);
}

public ELM dequeue() {
  if(!lst.isEmpty())
    return lst.remove(0);
  else
    return null;
}
```

- O método enqueue adiciona um elemento no início da lista
- O método dequeue retira o primeiro elemento da lista

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Nesse contexto, qual implementação dos métodos enqueue() e dequeue() completa a classe Queue, de modo que todos os elementos inseridos em uma fila possam ser recuperados de acordo com a propriedade FIFO?

- [D]

```
public void enqueue(ELM s) {
  lst.add(s);
}

public ELM dequeue() {
  if(!lst.isEmpty())
    return lst.remove(lst.size()-1);
  else
    return null;
}
```

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Nesse contexto, qual implementação dos métodos enqueue() e dequeue() completa a classe Queue, de modo que todos os elementos inseridos em uma fila possam ser recuperados de acordo com a propriedade FIFO?

- [D]

```
public void enqueue(ELM s) {
  lst.add(s);
}

public ELM dequeue() {
  if(!lst.isEmpty())
    return lst.remove(lst.size()-1);
  else
    return null;
}
```

- O método enqueue adiciona um elemento no final da lista
- O método dequeue retira o último elemento da lista

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Nesse contexto, qual implementação dos métodos enqueue() e dequeue() completa a classe Queue, de modo que todos os elementos inseridos em uma fila possam ser recuperados de acordo com a propriedade FIFO?

- [E]

```
public void enqueue(ELM s) {
  lst.add(lst.size(),s);
}

public ELM dequeue() {
  if(!lst.isEmpty())
    return lst.remove(lst.size()-1);
  else
    return null;
}
```

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Nesse contexto, qual implementação dos métodos enqueue() e dequeue() completa a classe Queue, de modo que todos os elementos inseridos em uma fila possam ser recuperados de acordo com a propriedade FIFO?

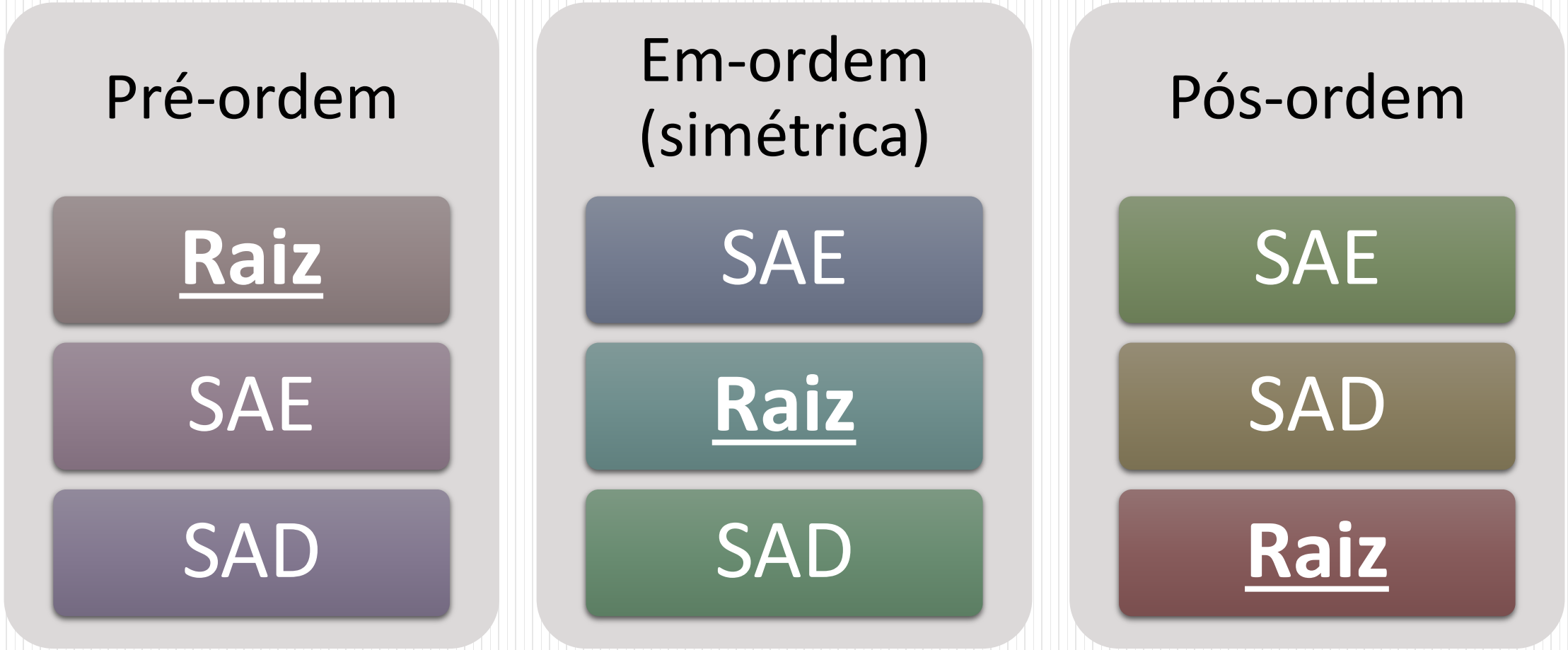
- [E]

```
public void enqueue(ELM s) {
  lst.add(lst.size(),s);
}

public ELM dequeue() {
  if(!lst.isEmpty())
    return lst.remove(lst.size()-1);
  else
    return null;
}
```

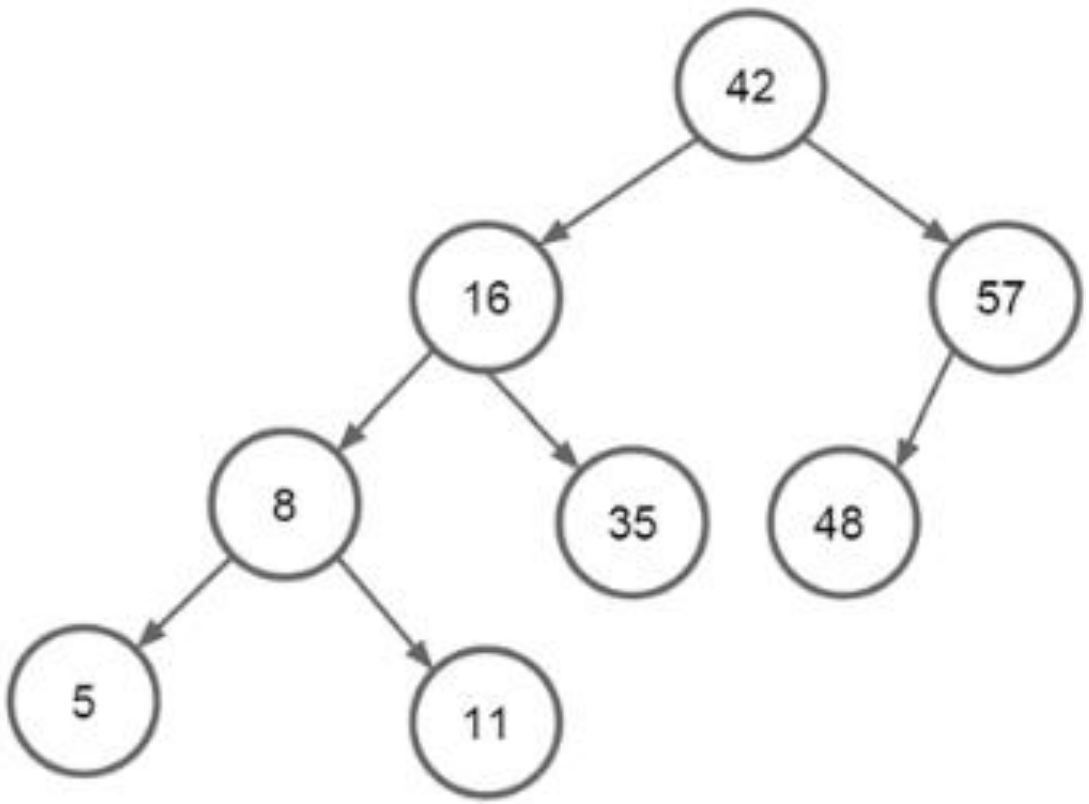
- O método enqueue adiciona um elemento no final da lista
- O método dequeue retira o último elemento da lista

Árvores binárias

Ordens de percurso em árvore binária



Ordens de percurso em árvore binária



Pré-ordem

- 42 16 8 5 11 35 57 48

Em-ordem

- 5 8 11 16 35 42 48 57

Pós-ordem

- 5 11 8 35 16 48 57 42

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] A classe Java ArvNo, exibida abaixo, é usada para representar os nós de uma árvore binária.

```
package estruturas;

class ArvNo {
	int info;
	ArvNo esq=null,dir=null;
}
```

Questões de concursos

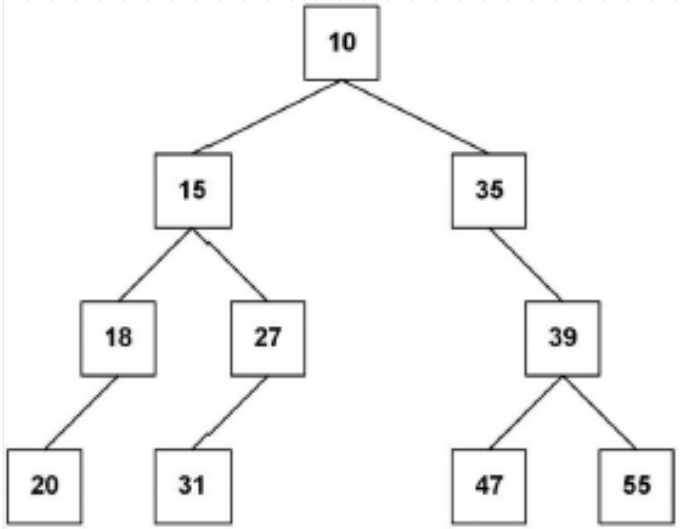
[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Ela é usada na implementação de uma árvore binária pela classe Arv, exibida a seguir.

```
package estruturas;

public class Arv {
	private ArvNo raiz;

	public Arv(){
	}

	public void exibe(){
		percorre(raiz);
		System.out.println("\n");
	}

	private void percorre(ArvNo r) {
		if(r==null)
			return;

		percorre(r.dir);
		percorre(r.esq);
		System.out.print(r.info+" ");

	}
}
```

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Que árvore terá os valores de seus nós exibidos em ordem descendente quando for percorrida pelo método percorre(), definido na classe Arv?

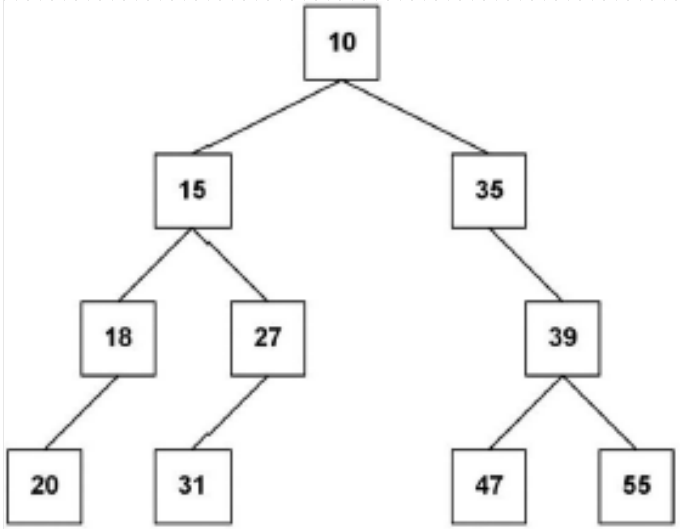
- [A]



Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Que árvore terá os valores de seus nós exibidos em ordem descendente quando for percorrida pelo método percorre(), definido na classe Arv?

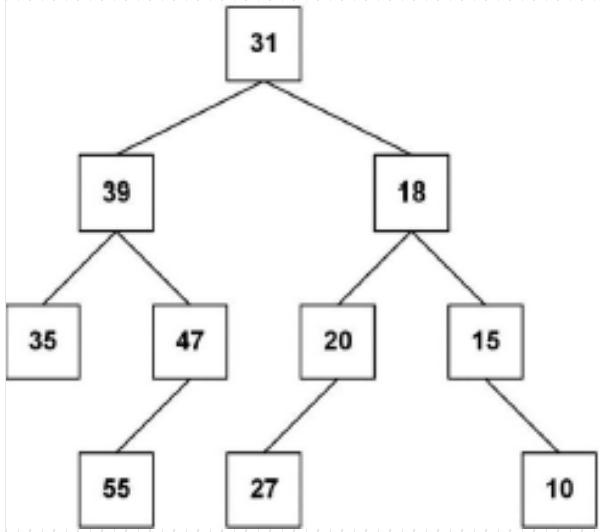
- **[A]** **55 47 39 35 31 27 20 18 15 10**



Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Que árvore terá os valores de seus nós exibidos em ordem descendente quando for percorrida pelo método percorre(), definido na classe Arv?

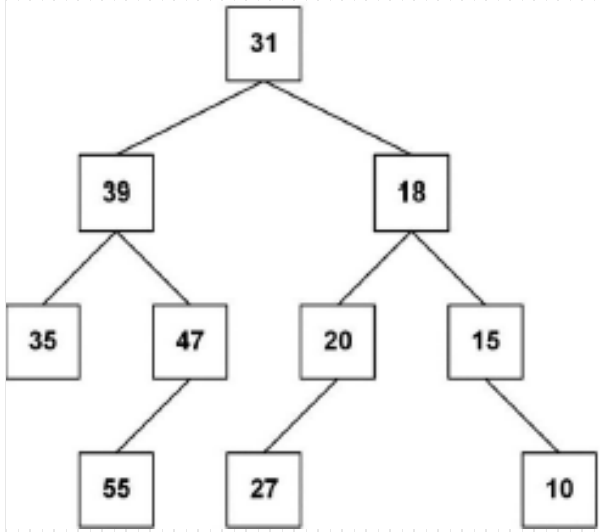
- [B]



Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Que árvore terá os valores de seus nós exibidos em ordem descendente quando for percorrida pelo método percorre(), definido na classe Arv?

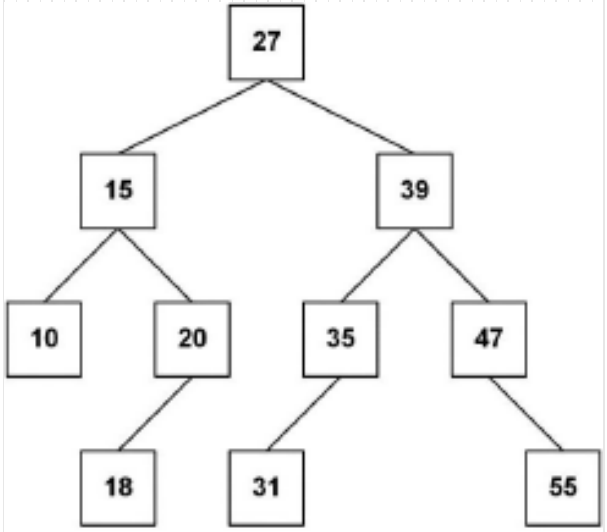
- [B] 10 15 27 20 18 55 47 35 39 31



Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Que árvore terá os valores de seus nós exibidos em ordem descendente quando for percorrida pelo método percorre(), definido na classe Arv?

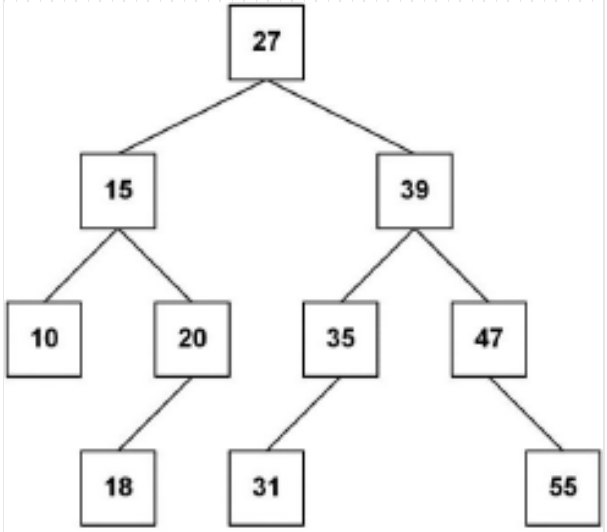
- [C]



Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Que árvore terá os valores de seus nós exibidos em ordem descendente quando for percorrida pelo método percorre(), definido na classe Arv?

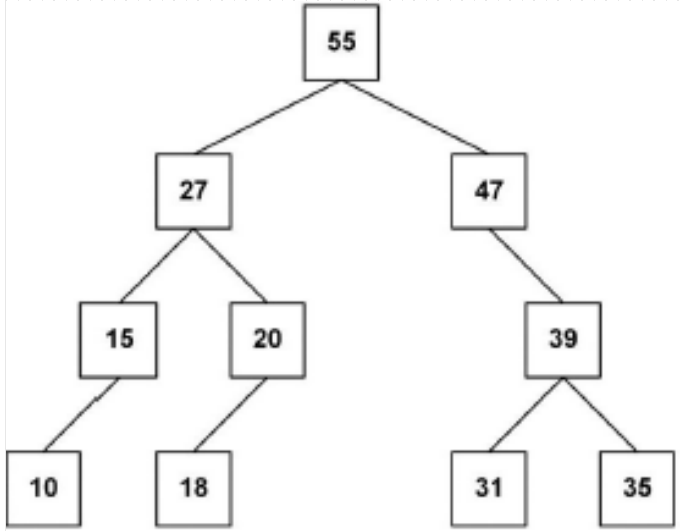
- [C] 55 47 31 35 39 18 20 10 15 27



Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Que árvore terá os valores de seus nós exibidos em ordem descendente quando for percorrida pelo método percorre(), definido na classe Arv?

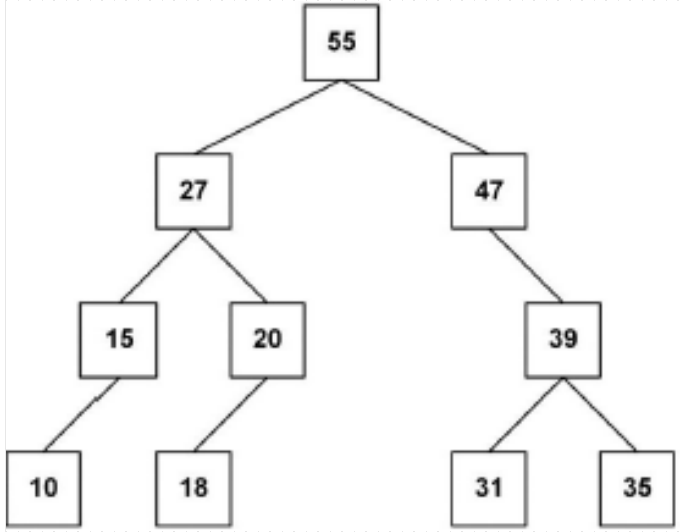
- [D]



Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Que árvore terá os valores de seus nós exibidos em ordem descendente quando for percorrida pelo método percorre(), definido na classe Arv?

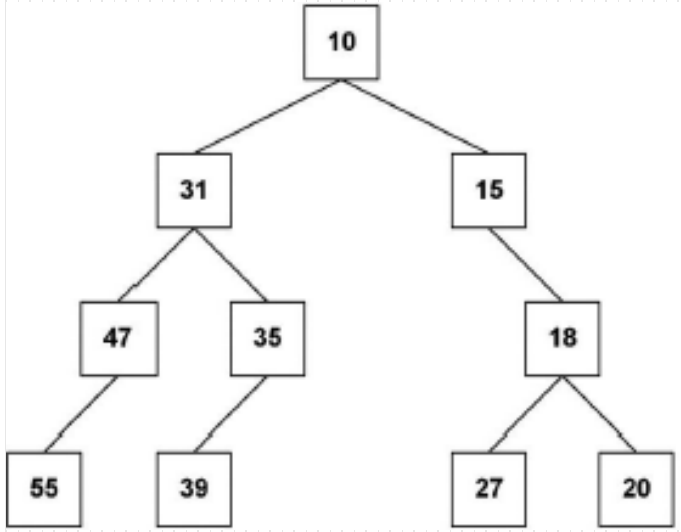
- [D] 35 31 39 47 18 20 10 15 27 55



Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Que árvore terá os valores de seus nós exibidos em ordem descendente quando for percorrida pelo método percorre(), definido na classe Arv?

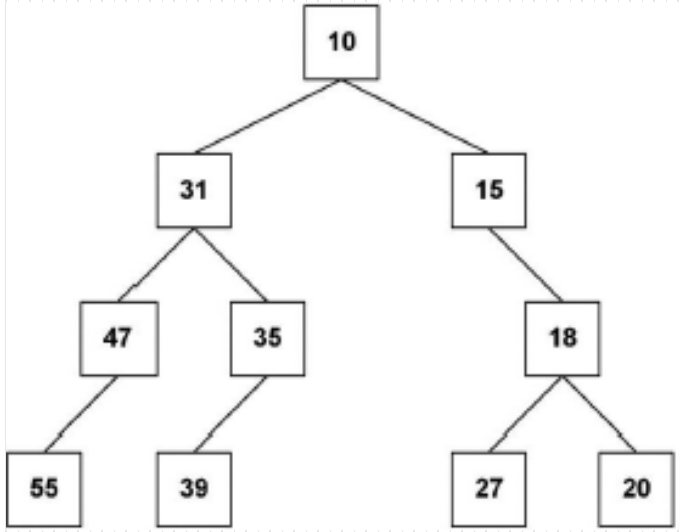
- [E]



Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] Que árvore terá os valores de seus nós exibidos em ordem descendente quando for percorrida pelo método percorre(), definido na classe Arv?

- [E] 20 27 18 15 39 35 55 47 31 10



Questões de concursos

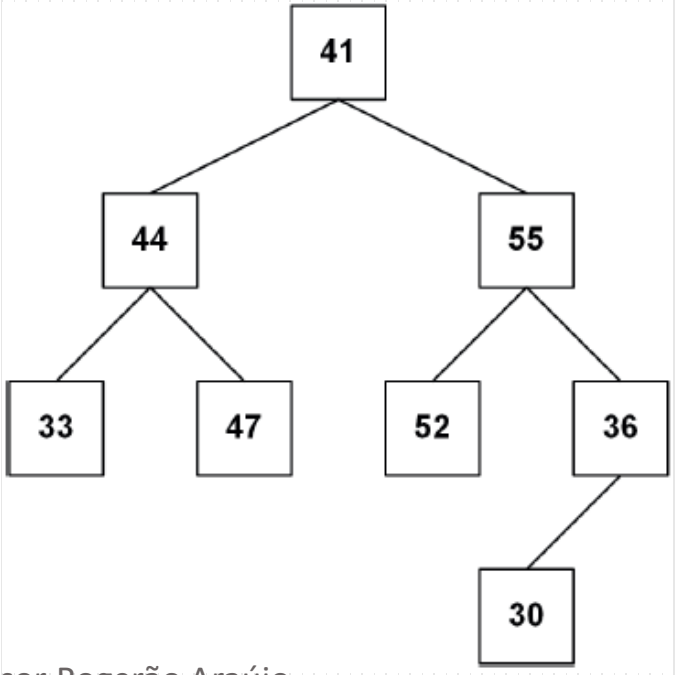
[CESGRANRIO 2021 Banco do Brasil – Escriturário: Agente de Tecnologia – Prova 1 – Questão 51] Um programador escreveu uma função para percorrer, em pós-ordem, uma árvore binária e exibir, no console, os valores referentes aos nós dessa árvore. Após essa função ter sido executada, foi exibido o seguinte resultado:

41 44 33 47 55 52 36 30

Questões de concursos

[CESGRANRIO 2021 Banco do Brasil – Escriturário: Agente de Tecnologia – Prova 1 – Questão 51] Que árvore essa função percorreu para exibir o resultado acima?

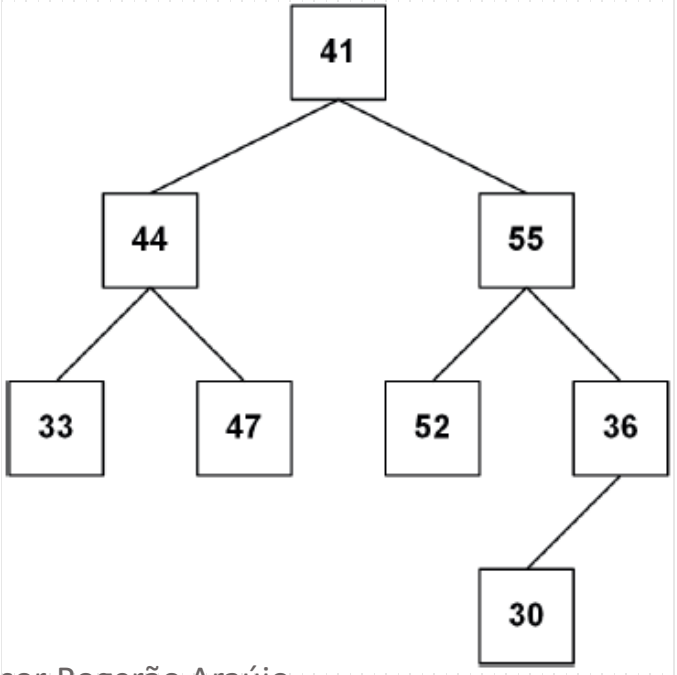
- [A]



Questões de concursos

[CESGRANRIO 2021 Banco do Brasil – Escriturário: Agente de Tecnologia – Prova 1 – Questão 51] Que árvore essa função percorreu para exibir o resultado acima?

- [A]

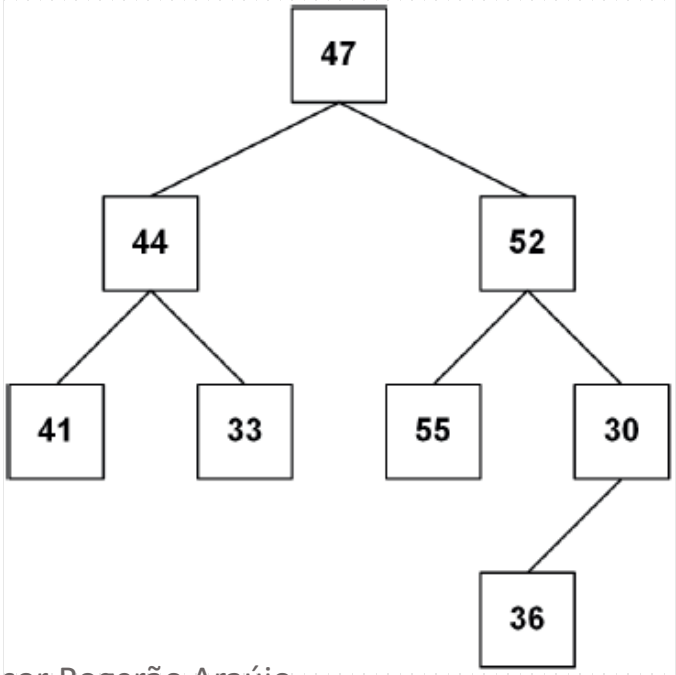


- Pós-ordem do enunciado:
  - 41 44 33 47 55 52 36 30
- Pós-ordem da letra:
  - 33 47 44 52 30 36 55 41

Questões de concursos

[CESGRANRIO 2021 Banco do Brasil – Escriturário: Agente de Tecnologia – Prova 1 – Questão 51] Que árvore essa função percorreu para exibir o resultado acima?

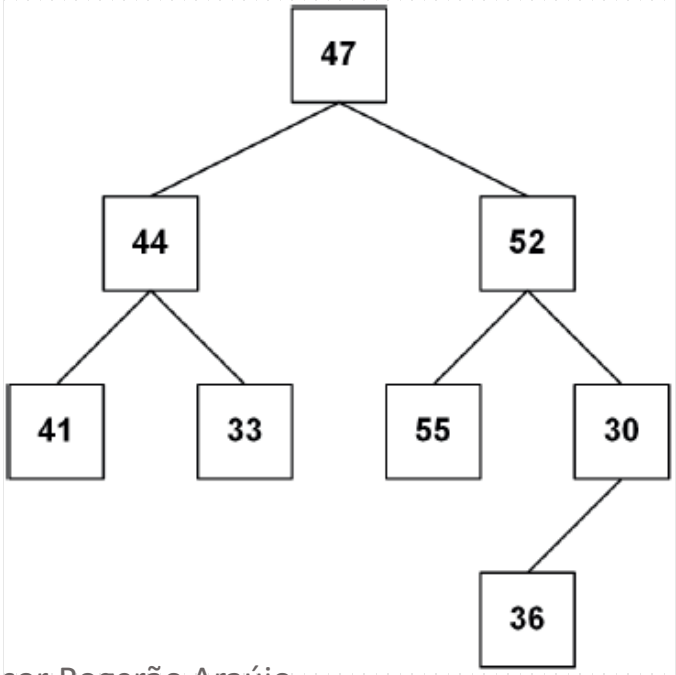
- [B]



Questões de concursos

[CESGRANRIO 2021 Banco do Brasil – Escriturário: Agente de Tecnologia – Prova 1 – Questão 51] Que árvore essa função percorreu para exibir o resultado acima?

- [B]

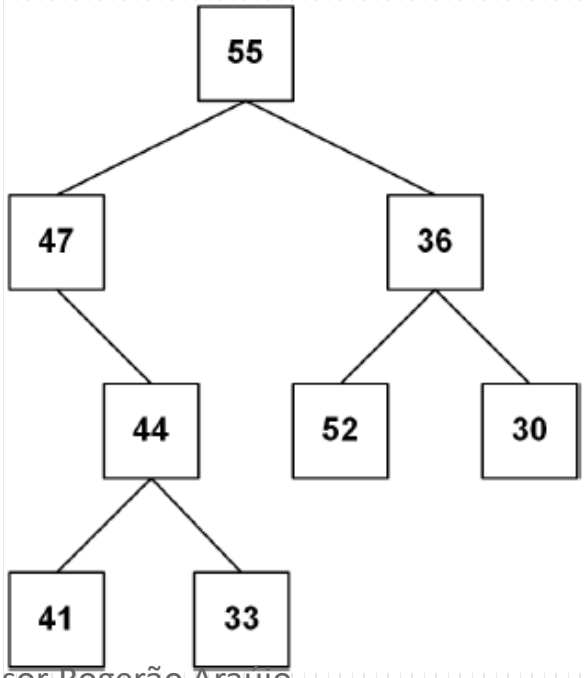


- Pós-ordem do enunciado:
  - 41 44 33 47 55 52 36 30
- Pós-ordem da letra:
  - 41 33 44 55 36 30 52 47

Questões de concursos

[CESGRANRIO 2021 Banco do Brasil – Escriturário: Agente de Tecnologia – Prova 1 – Questão 51] Que árvore essa função percorreu para exibir o resultado acima?

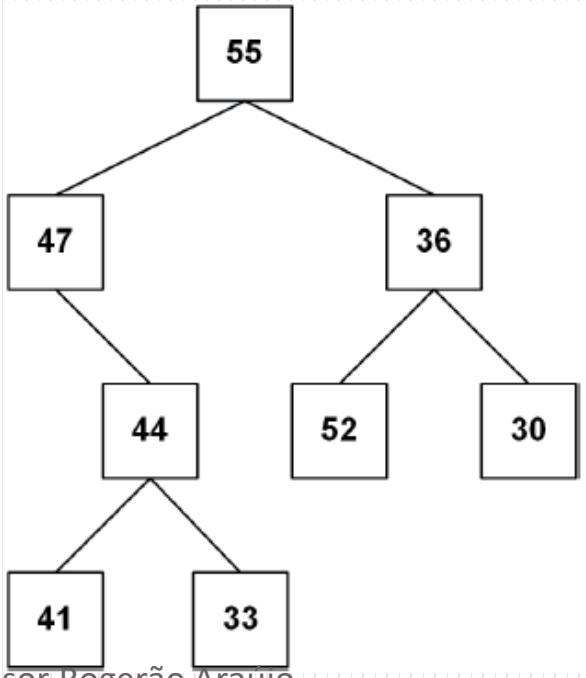
- [C]



Questões de concursos

[CESGRANRIO 2021 Banco do Brasil – Escriturário: Agente de Tecnologia – Prova 1 – Questão 51] Que árvore essa função percorreu para exibir o resultado acima?

- [C]

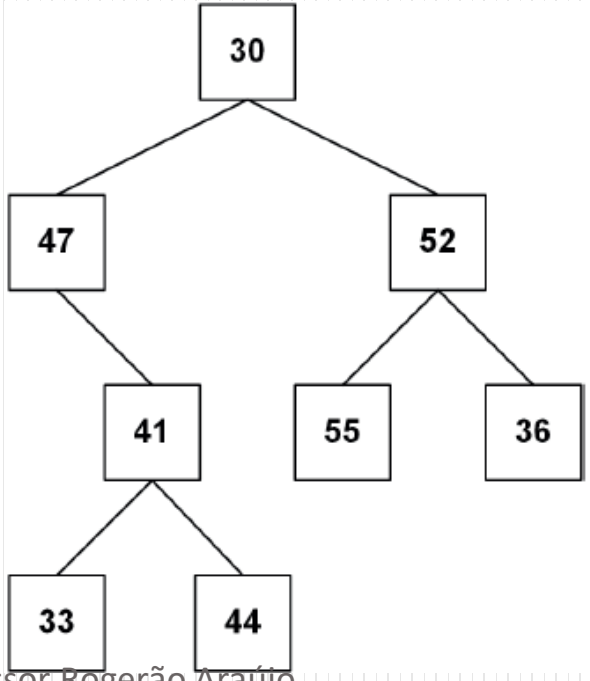


- Pós-ordem do enunciado:
  - 41 44 33 47 55 52 36 30
- Pós-ordem da letra:
  - 41 33 44 47 52 30 36 55

Questões de concursos

[CESGRANRIO 2021 Banco do Brasil – Escriturário: Agente de Tecnologia – Prova 1 – Questão 51] Que árvore essa função percorreu para exibir o resultado acima?

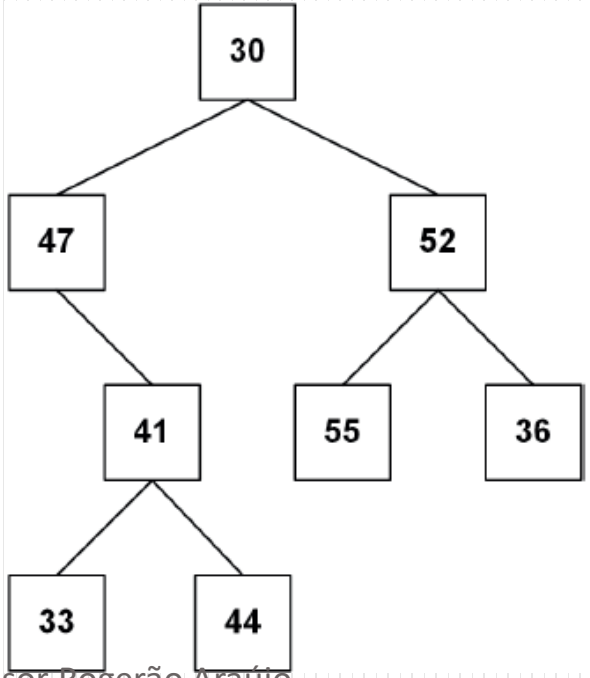
- [D]



Questões de concursos

[CESGRANRIO 2021 Banco do Brasil – Escriturário: Agente de Tecnologia – Prova 1 – Questão 51] Que árvore essa função percorreu para exibir o resultado acima?

- [D]

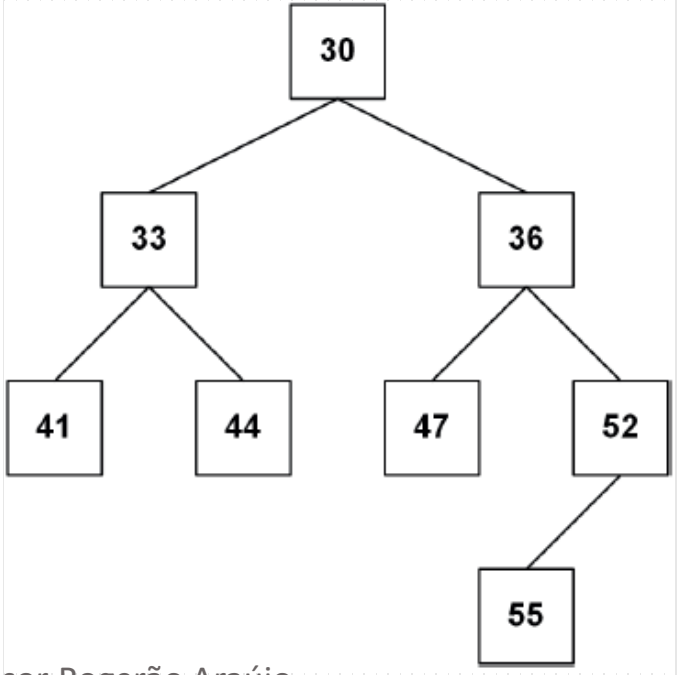


- Pós-ordem do enunciado:
  - 41 44 33 47 55 52 36 30
- Pós-ordem da letra:
  - 33 44 41 47 55 36 52 30

Questões de concursos

[CESGRANRIO 2021 Banco do Brasil – Escriturário: Agente de Tecnologia – Prova 1 – Questão 51] Que árvore essa função percorreu para exibir o resultado acima?

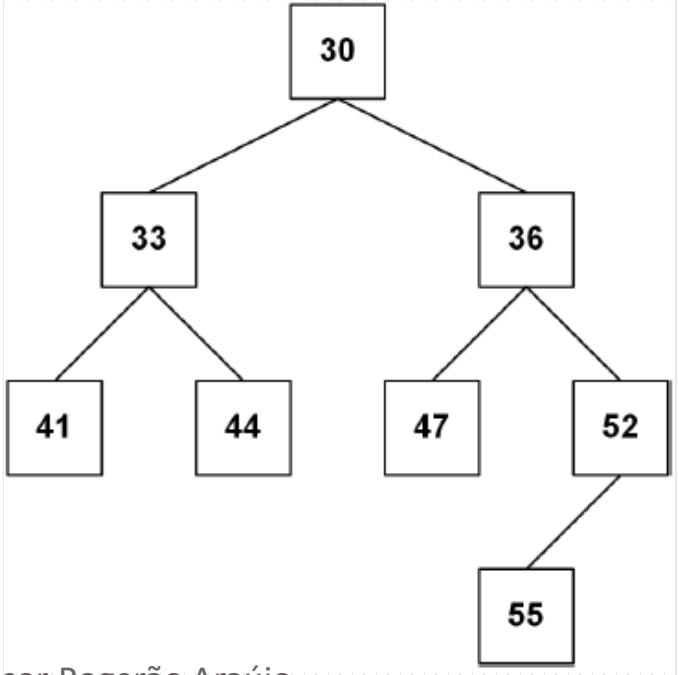
- [E]



Questões de concursos

[CESGRANRIO 2021 Banco do Brasil – Escriturário: Agente de Tecnologia – Prova 1 – Questão 51] Que árvore essa função percorreu para exibir o resultado acima?

- **[E]**



- Pós-ordem do enunciado:
  - 41 44 33 47 55 52 36 30
- Pós-ordem da letra:
  - 41 44 33 47 55 52 36 30

Busca sobre arrays

Busca sequencial sobre arrays

Conceituação

É o método de busca mais simples

A partir do primeiro registro, pesquisa-se sequencialmente até encontrar a chave procurada

É aplicável a uma tabela organizada como

- Vetor
- Lista encadeada
- Array

Encontrando a chave no array

- A busca é encerrada
- É retornado o índice do registro que contém a chave x

Caso não esteja presente

- O valor retornado é -1

Algoritmo

- int buscaSequencial(int vetor[], int chave) {
  - int limiteInferior = 0;
  - int limiteSuperior = vetor.length - 1;
  - for (int i = limiteInferior; i <= limiteSuperior; i++) {
    - if (vetor[i] == chave)
      - return i; // Chave encontrada
  - }
  - return -1; // Chave não encontrada
- }

Busca binária sobre arrays

Algoritmo

O elemento buscado é comparado ao elemento do meio do arranjo

Se elemento buscado é igual ao elemento do meio	Se elemento buscado é menor que elemento do meio	Se elemento buscado é maior que elemento do meio
Busca bem-sucedida	Busca-se na metade inferior do arranjo	Busca-se na metade superior do arranjo

Algoritmo iterativo

- int buscaBinaria (int vetor[], int chave) {
  - int limiteInferior = 0;
  - int limiteSuperior = vetor.length - 1;
  - int meio;
  - while (limiteInferior <= limiteSuperior) {
    - meio = (limiteInferior + limiteSuperior) / 2;
    - if (chave == vetor[meio])
      - return meio; // Chave encontrada
    - else
      - if (chave < vetor[meio])
        - limiteSuperior = meio - 1;
      - else
        - limiteInferior = meio + 1;
  - }
  - return -1; // Chave não encontrada
- }

Algoritmo recursivo

- int buscaBinaria (int vetor[], int chave, int limiteInferior, int limiteSuperior) {
  - int meio = (limiteInferior + limiteSuperior) / 2;
  - if (chave == vetor[meio])
    - return meio; // Chave encontrada
  - else
    - if (chave < vetor[meio])
      - return buscaBinaria(chave, vetor, limiteInferior, meio - 1);
    - else
      - return buscaBinaria(chave, vetor, meio + 1, limiteSuperior);
  - if (limiteInferior >= limiteSuperior)
    - return -1; // Chave não encontrada
- }

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] A classe Java a seguir contém dois métodos (busca e buscaBin) que implementam um algoritmo de busca binária sobre um array de inteiros.

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico]

```
public class Main {

   public static void main(String[] args) {
      int arry[]= {220,158,133,100,98,96,80,60,55,22,8};

      busca(arry,61);
   }

   public static int busca(int vet[], int elem) {
      return buscaBin(vet,elem,0,vet.length-1);
   }

   private static int buscaBin(int vet[], int elem, int ini, int fin) {
      if(ini > fin)
            return -1;

      int m=(ini+fin)/2;

      System.out.printf("%d " ,vet[m]);

      if(vet[m]==elem)
            return m;
      else
            if(vet[m]>elem)
                  return buscaBin(vet,elem,m+1,fin);
            else
                  return buscaBin(vet,elem,ini,m-1);
   }
}
```

Comentários

Índice	Valor
0	**220**
1	158
2	133
3	100
4	98
5	**96**
6	80
7	60
8	55
9	22
10	**8**

Chave	61
ini	0
	220
fin	10
	8
m	5
	96
96	

Comentários

Índice	Valor
0	220
1	158
2	133
3	100
4	98
5	96
6	**80**
7	60
8	**55**
9	22
10	**8**

Chave	61
ini	6
	80
fin	10
	8
m	8
	55
96	**55**

Comentários

Índice	Valor
0	220
1	158
2	133
3	100
4	98
5	96
6	**80**
7	**60**
8	55
9	22
10	8

Chave	61
ini	6
	80
fin	7
	60
m	6
	80
96 55 80	

Comentários

Índice	Valor
0	220
1	158
2	133
3	100
4	98
5	96
6	80
7	**60**
8	55
9	22
10	8

Chave	61
ini	7
	60
fin	7
	60
m	7
	60
96 55 80 60	

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] O que será exibido no console quando o método main() for executado?

[A] 96 80 60

[B] 96 133 220

[C] 96 55 60 80

[D] 96 55 80 60

[E] 96 133 158 220

Questões de concursos

[CESGRANRIO 2021 Banco da Amazônia – Técnico Científico] O que será exibido no console quando o método main() for executado?

[A] 96 80 60

[B] 96 133 220

[C] 96 55 60 80

[D] 96 55 80 60

[E] 96 133 158 220

Dúvidas?

Prof. Rogerão Araújo

www.instagram.com/profRogeraoAraujo