Application Delivery Fundamentals: Java

accenturetechnology

Module 7: Classes e Objetos – Parte 2

Modificadores de acesso

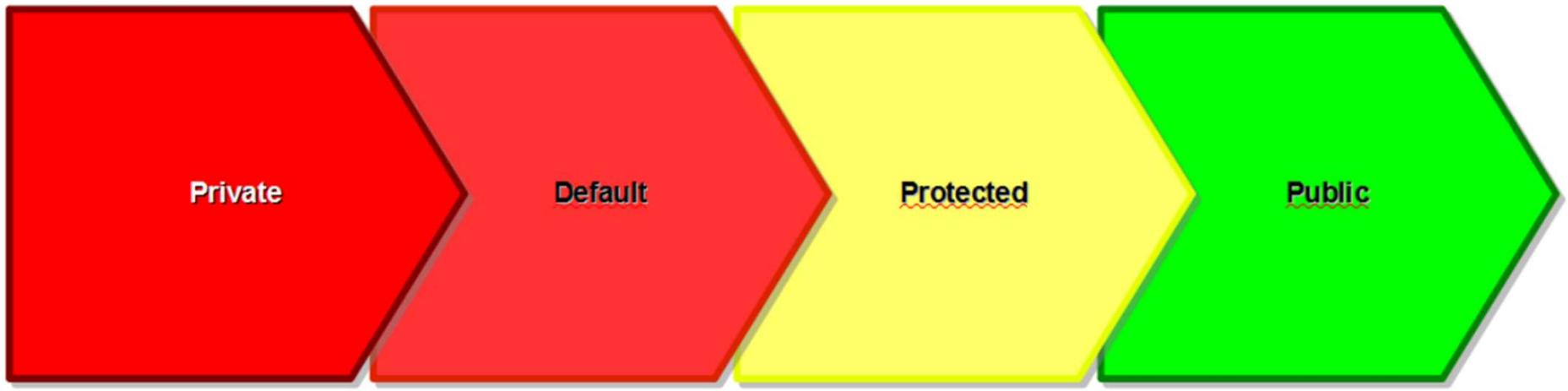
2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

“Modificadores de Acesso”

Member Modifiers

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- As classes geralmente precisam controlar como seus atributos e comportamentos são acessados.
- Modificadores permitem esse controle.



Access Modifiers

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Modificadores de acesso descrevem como uma classe pode ser acessada;
- Existem quatro tipos de modificadores de acesso. Esses são:

default / no modifier	• Uma classe pode ser acessada por classes pertencentes à mesma package
public	• Um membro da classe pode ser acessado por qualquer classe em qualquer package
protected	• Um membro da classe só pode ser acessado por si ou por suas subclasses;
private	• Um membro da classe só pode ser acessado por ele mesmo

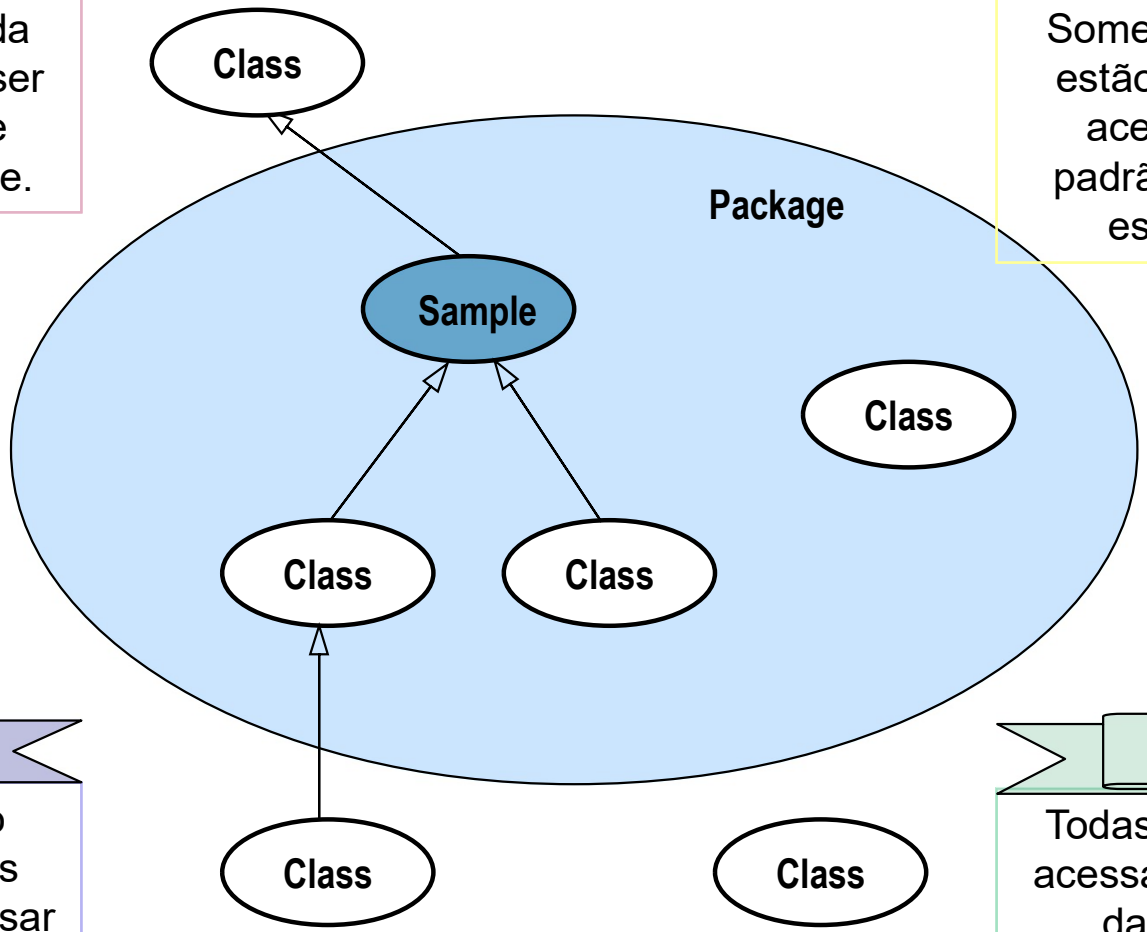
Member Access Modifiers Diagram

private

Os recursos privados da classe Sample podem ser acessados apenas de dentro da própria classe.

default

Somente as classes que estão no pacote podem acessar os recursos padrão das classes que estão no package



protected

Classes que estão no pacote e todas as suas subclasses podem acessar recursos protegidos da classe Sample.

public

Todas as classes podem acessar recursos públicos da classe Sample.

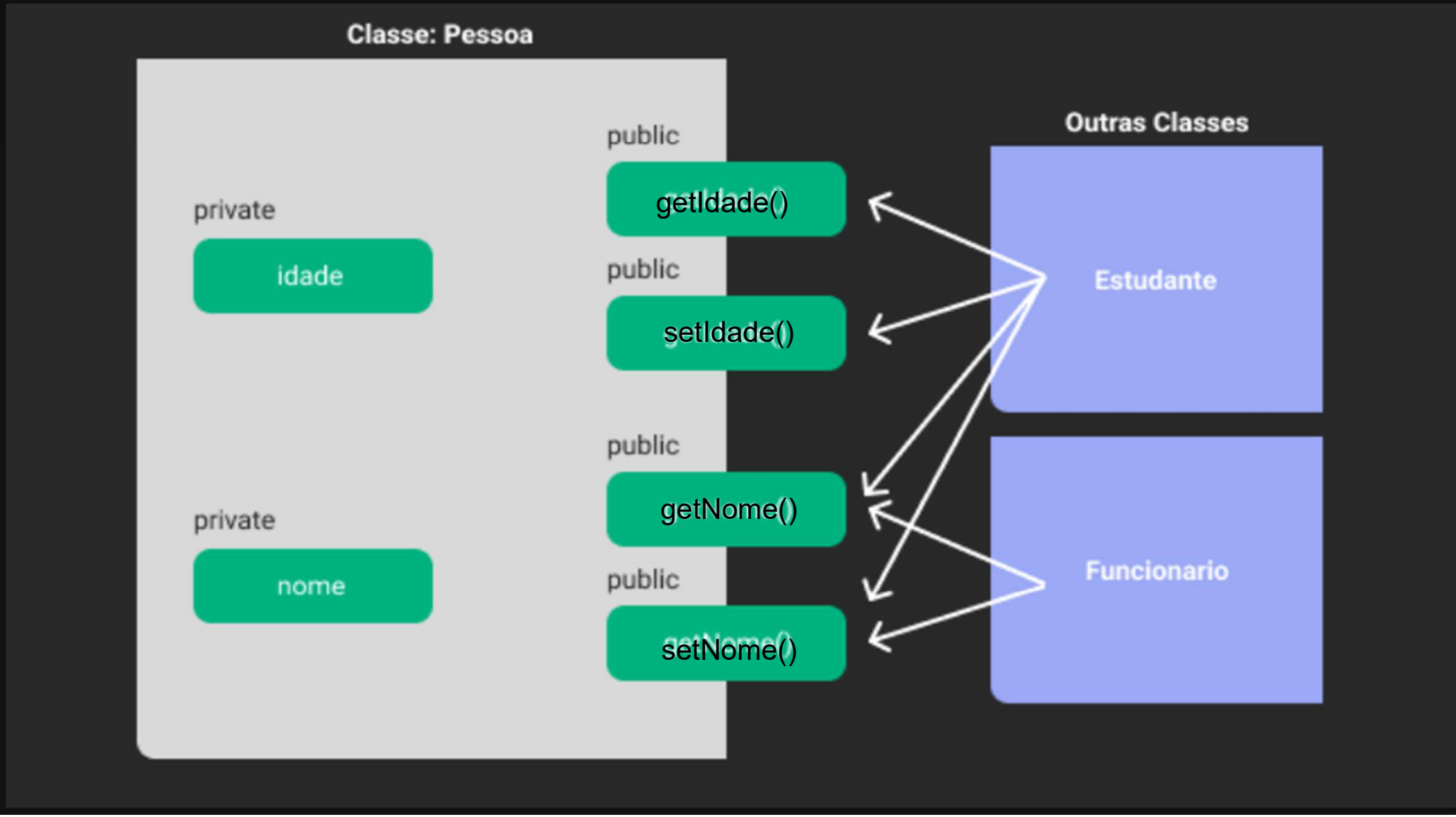
* *Default* is not a modifier; it is just the name of the access level if no access modifier is specified.

Métodos

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

“Métodos Getters e Setters”

Getters e setters são métodos que permitem o acesso indireto a atributos privados de uma classe. Em um contexto ideal, todos os atributos de uma classe são mantidos privados para protegê-los de acessos e modificações não autorizadas. Para acessar ou modificar esses atributos, são utilizados métodos públicos conhecidos como getters e setters.



Métodos Acessores e Modificadores

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820

- Métodos que dão acesso ao dados de um objeto;
- Métodos **GET** são métodos assessores;
- Permitem que você recupere o valor dos atributos;
- Métodos **GET** não recebem parâmetros;

- Métodos **SET** são métodos modificadores;
- Permitem que você defina o valor dos atributos
- Métodos **SET** recebem parâmetros;

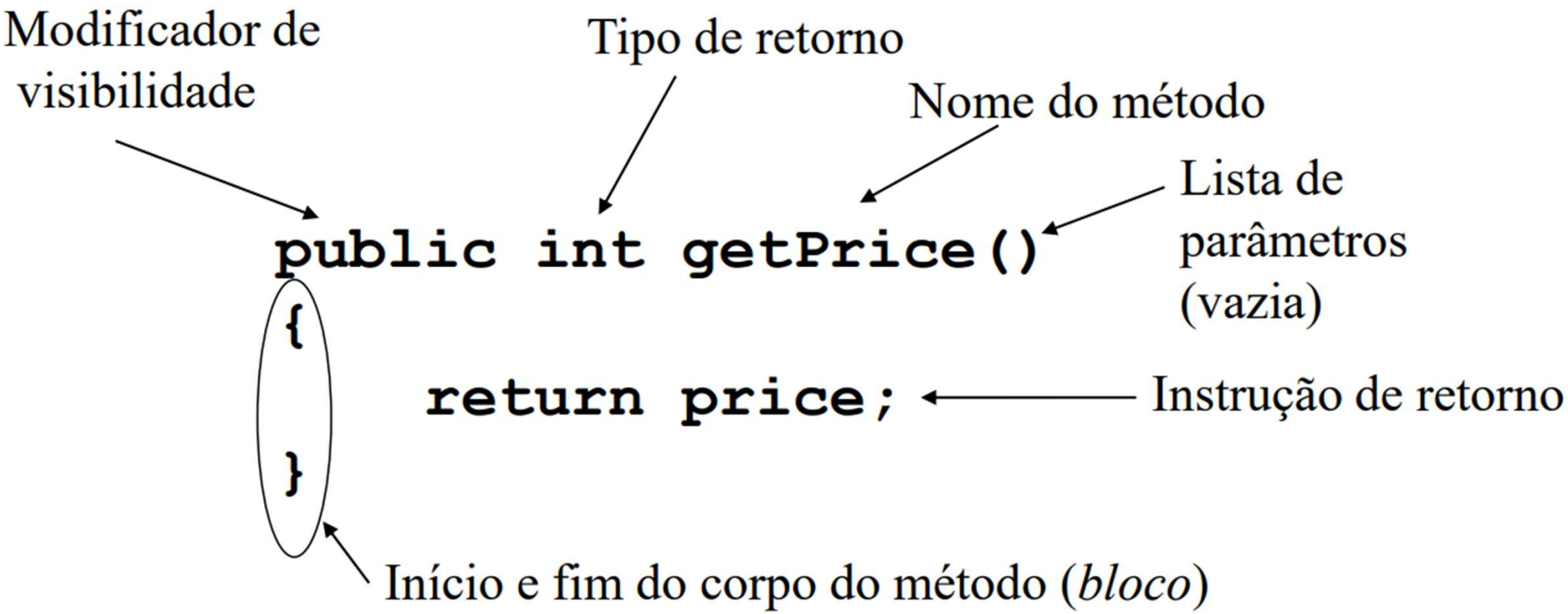
- São declarados como **Public**

Refer to the Caneta.java sample code.

Métodos Acessores e Modificadores

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

O Método Get



Métodos Acessores e Modificadores

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

O Método Set

```
public void setPrice(int price)
{
        this.price = price;
}
```

Métodos Acessores e Modificadores

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

‘Getter’ Methods

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Os métodos "**Getter**" ou "accessor" permitem que objetos retornem os valores de seus atributos particulares.
- Nem todos os atributos privados precisam ter métodos getter; isso depende do design.
- Os métodos **Getter** sempre devem retornar apenas uma cópia dos valores dos atributos, e não os próprios atributos.

‘Setter’ Methods

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Os campos ou o estado de um objeto geralmente são implementados por atributos privados;
- Para modificar atributos privados, os objetos apresentam interfaces públicas chamadas métodos "**setter**" ou "mutator“;
- Nem todos os atributos privados exigem métodos de configuração; isso deve depender do design;
- Esses métodos devem controlar estritamente como os campos são modificados e executar validações apropriadas nos parâmetros passados;
- Sempre que os atributos de um objeto são modificados, o objeto deve validar a correção de seu estado (invariantes).

Perguntas:

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

O que é modificador de acesso?

São palavras-chave que garantem níveis de acesso aos atributos, métodos e classes

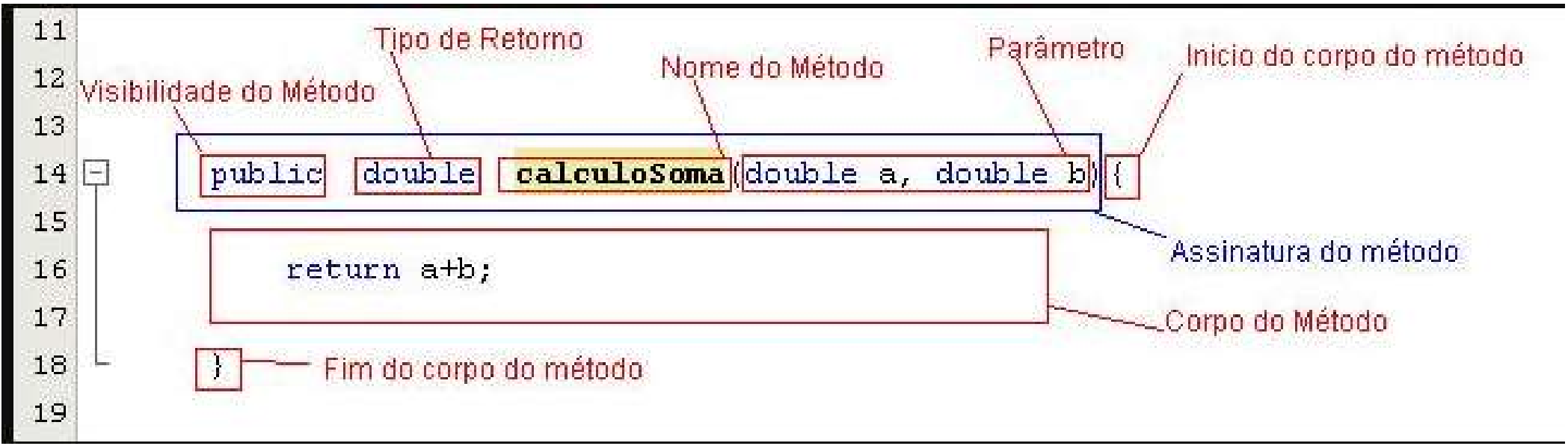
Para que servem os metodos Get e Set?

Eles servem para pegarmos informações de variáveis da classe que são definidas como 'private',

Assinatura de um método

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

```
public double calculoSoma(double a, double b) {
    return a+b;
}
```



Métodos

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

“OverLoad”

Overload

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696

```java
public class Classe {

    private int numa;
    private int numb;

    public Classe(){
        this.numa = 2;
        this.numb = 2;
    }

    public Classe(int numa, int numb){
        this.numa = numa;
        this.numb = numb;
    }

    public int somaValores(){
        return numa+numb;
    }

    public void somaValores(int a, int b){
        this.numa = a;
        this.numb = b;
        int total = a+b;
    }

    public int somaValores(double b, double a){
        double total = a+b;
        int contotal = (int)total;
        return contotal;
    }
```

Sobrecarga (Overload) de Construtores - O primeiro não possui parâmetro, já o segundo possui.

Sobrecarga de Métodos, todos eles tem a mesma função, mais recebem parametros diferentes, e tem retornos diferentes, ou nenhum retorno

Restrições ao uso de modificadores

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820

- Nem todas as combinações de variáveis e métodos de instância e classe são permitidas:
  - Os métodos de instância podem acessar variáveis de instância e métodos de instância diretamente.
  - Os métodos de instância podem acessar variáveis de classe e métodos de classe diretamente.
  - Os métodos de classe podem acessar variáveis e métodos de classe diretamente.
  - Os métodos de classe não podem acessar variáveis de instância ou métodos de instância diretamente. Os métodos de classe também não podem usar a palavra-chave **this**, pois não há instância para isso se referir.

Static Attributes and Methods

- **`Static`** atributos e métodos estão associados à classe e são compartilhados por todas as instâncias da classe
- Atributos e métodos estáticos são definidos usando a palavra-chave **`static`**
- Somente os métodos static podem acessar diretamente atributos e chamar métodos que também são static
- Os membros **`Static`** de uma classe podem ser acessados ou mencionados pelo nome da classe

```
ClassName.static member

Calendar.getInstance();
```

Static Code Blocks

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Os blocos de código podem ser marcados como estáticos
- Blocos de código estático são executados apenas uma vez quando a classe é carregada pela JVM pela primeira vez
- Somente blocos de código estático podem acessar diretamente atributos e métodos estáticos

```
public class SampleClass{

        static{
                //your code here
        }

   }
```

Refer to the StaticSample2.java sample code.

Activity

2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207

- **Atividade 8**
  - Crie uma classe java ContaCorrente.java ;
  - Com os atributos: **numero, nome, saldo e data;**
  - Com os métodos: Depositar, Sacar, ExibirExtrato e Tranferir;
  - No método constructor inicialize os atributos;

Activity

2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207

- **Atividade - 8**
  - Crie uma classe java Cliente.java
  - Com os atributos:**nome, cpf e sobrenome**
  - Alterar a classe ContaCorrente para incorporar os dados do cliente;

Activity

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

- **Atividade 8**
  - Crie uma classe principal java PrincipalContaCorrente.java com o método main
  - Instancie a classe ContaCorrente;
  - Execute os métodos **Depositar, Sacar e Transferir**;
  - Exibir o saldo de cada **transação**.
  - Instanciar a conta1;
  - Exibir Saldo e fazer um **depósito**;
  - Instanciar a conta2;
  - Exiba o nome do cliente da conta1;
  - Caso a conta origem fique negativa cancelar a transferência.

Perguntas:

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

O que é um método?

É o que define o comportamento da Classe.

O que é assinatura do método?

É a identificação do método.

Wrappers

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

“Wrappers”

Wrappers

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Classes Wrappers

- Uma classe wrapper é uma representação de um tipo primitivo como um objeto
- Cada tipo primitivo possui seu wrapper
- Wrapper são úteis se você trabalha com classes e métodos que só aceitam objetos como argumentos

Classes Wrappers

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Wrap = embrulhar

Ele serve para pôr uma “roupagem” em coisas para que elas se adaptem ao que você precisa.

Exemplo : Wrappers de tipos primitivos

Você tem um tipo primitivo (**long**) mas precisa de um objeto que tenha a mesma significação.

Nesse caso, você pega o valor e o “**embrulha**” em um objeto da classe java.lang.Long

Classes Wrappers

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Wrap = embrulhar

- Para cada um dos tipos de dados básicos existe um tipo "empacotador" correspondente. São eles:

Tipo primitivo	Classe correspondente
boolean	Boolean
char	Character
int	Integer
long	Long
float	Float
double	Double

Classes Wrappers

2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207

Wrappers

```
Int pInt = 500;
Integer wInt = new Integer(pInt);
Int p2 = wInt.IntValue();
```

Classes Wrappers

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Wrap = embrulhar

- Exemplo:

```
Integer n1 = new Integer(10);
Integer n2 = new Integer(20);
if (n1.equals(n2) == true)
    System.out.println("Valores iguais!");
else
    System.out.println("Valores diferentes!");
```

Obs: o método *equals* deve ser usado no lugar do operador "== " porque o teste "if (n1 == n2) ... " vai comparar as referências e não os valores. No caso, as referências nunca serão iguais.

Classes Wrappers

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Wrap = embrulhar

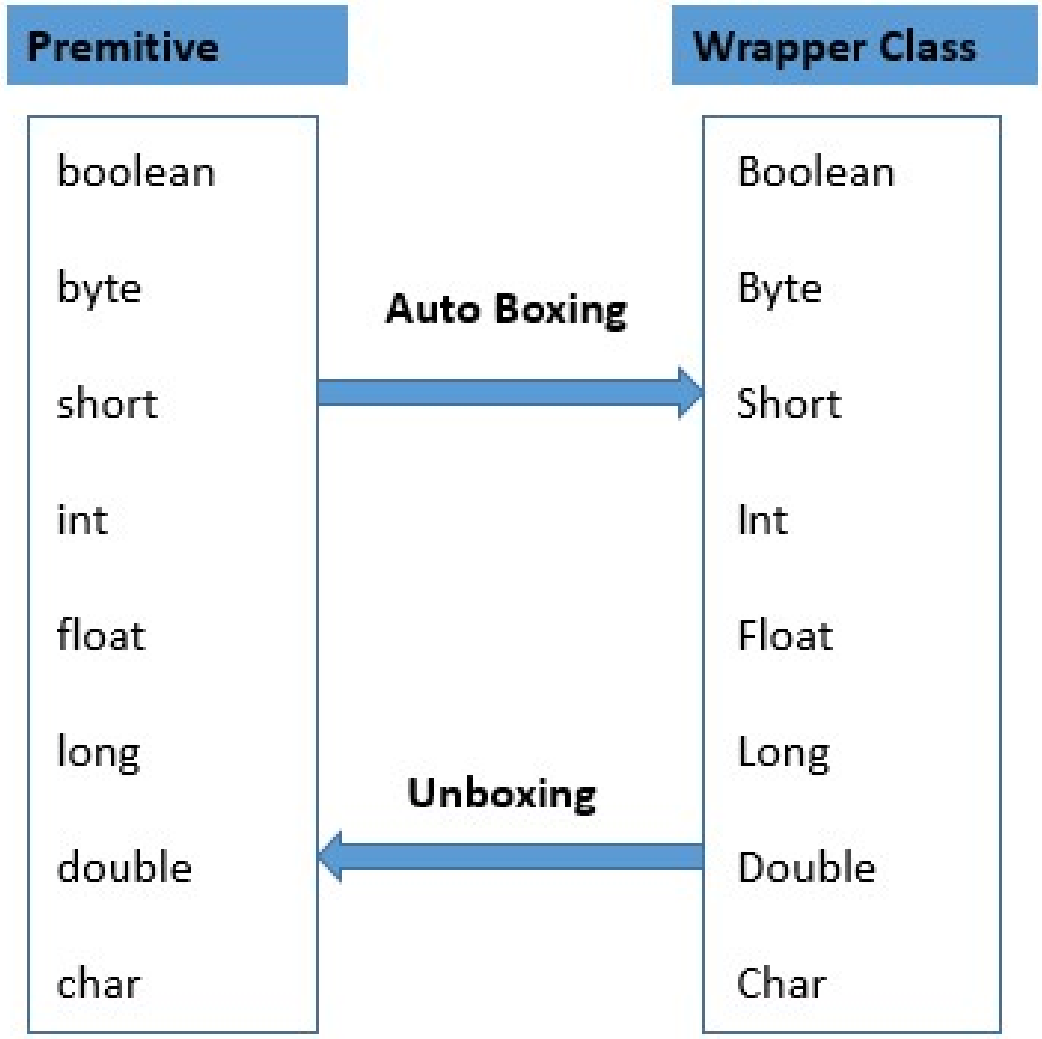
- Exemplos de métodos de conversão de *strings*:

  Um método parse para cada tipo de dado.

```
String texto = "12345";
int inteiro = Integer.parseInt(texto);
byte - Byte.parseByte(aString)
short - Short.parseShort(aString)
int - Integer.parseInt(aString)
long - Long.parseLong(aString)
float - Float.parseFloat(aString)
double - Double.parseDouble(aString)
boolean - Boolean.valueOf(aString).booleanValue();
```

i Refer to the WrappingUnwrapping.java sample code.

Classes Wrappers



Refer to the Autoboxing.java e unboxing.java sample code.

Classes Wrappers

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Autoboxing

- A conversão automática de tipos primitivos para o objeto de suas classes de wrapper correspondentes é conhecida como autoboxing.
- Por exemplo - conversão de int em Integer, long em Long, double em Double etc

Unboxing

- É apenas o processo reverso do autoboxing. A conversão automática de um objeto de uma classe de wrapper em seu tipo primitivo correspondente é conhecida como unboxing.
- Por exemplo - conversão de inteiro em inteiro, longo em longo, duplo em dobro, etc.

Classe JOptionPane

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

“Classe JOptionPane”

Classe JOptionPane

JOptionPane é uma classe que possibilita a criação de uma caixa de dialogo padrão que solicita um valor para o usuário ou retorna uma informação;

```
3  import javax.swing.JOptionPane;
4
5  public class HellowordBox {
6
7      public static void main(String[] args) {
8          JOptionPane.showMessageDialog(null, "Hello \nword ");
9
10     }
11
12 }
13
```

Refer to the HellowwordBox.java sample code.

Classe JOptionPane

2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207

Chama método `showMessageDialog` da classe `JOptionPane`

- Requer dois argumentos
  - Por enquanto, o primeiro argumento será sempre `null`
  - O segundo argumento é o string a apresentar

`showMessageDialog` é um método `static` da classe `JOptionPane`

- métodos `static` são chamados usando o nome da classe, ponto (`.`) e o nome do método.

Classe JOptionPane

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Próximo programa: soma dois valores

  Usa *input dialogs* para entrada de 2 valores pelo usuário

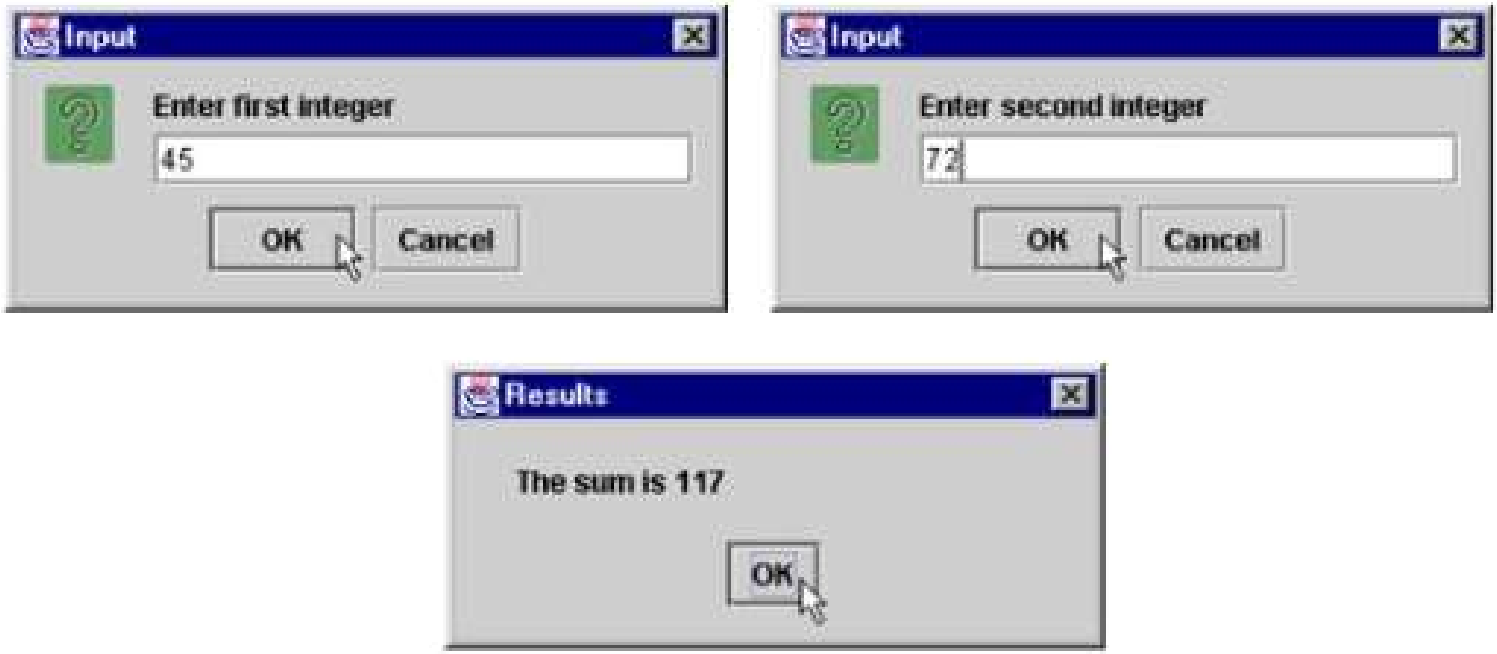
  Usa *message dialog* para exibir resultado da soma dos 2 valores fornecidos

  Demonstra uso de *wrappers* e entrada de dados

i Refer to the Somatorio.java sample code.

Classe JOptionPane

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207



Variáveis

- `firstNumber` e `secondNumber` são variáveis do tipo `String` (package `java.lang`)
  - strings encapsuladas (*wrapper*)

Classe JOptionPane

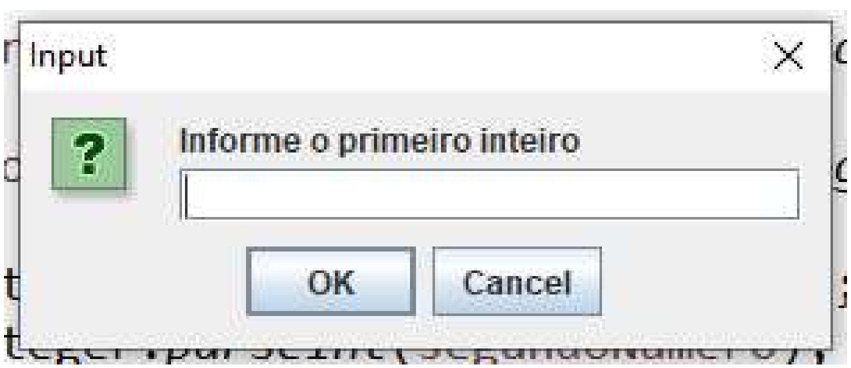
2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

```
        primeiroNumero = JOptionPane.showInputDialog("Informe o primeiro inteiro");

        segundoNumero = JOptionPane.showInputDialog("Informe o segundo inteiro");
```

Lê `String` do usuário, representando o primeiro número a ser adicionado

- Método `JOptionPane.showInputDialog` apresenta:



§ Note que o argumento da função aparece como texto.

Classe JOptionPane

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

```
14          numero1 = Integer.parseInt(primeiroNumero);
15          numero2 = Integer.parseInt(segundoNumero);
16
17          soma = numero1 + numero2;
```

Método `Integer.parseInt`

- Converte um argumento `String` em um inteiro (tipo `int`)
  - Classe `Integer` em `java.lang`
- O inteiro retornado por `Integer.parseInt` é atribuído a variável `number1`
  - Lembre que `number1` foi declarada como sendo do tipo `int`

Classe JOptionPane

```
19 JOptionPane.showMessageDialog(null,  "A soma foi = "+ soma, "Resultado", JOptionPane.PLAIN_MESSAG
```

Outras versões de `showMessageDialog`

- Requerem 4 argumentos
  - Primeiro argumento continua `null` por enquanto…
- Segundo: string a apresentar
- Terceiro: string para a barra de título
- Quarto: tipo de mensagem no diálogo
  - `JOptionPane.PLAIN_MESSAGE` - sem ícone
  - `JOptionPane.ERROR_MESSAGE`
  - `JOptionPane.INFORMATION_MESSAGE`
  - `JOptionPane.WARNING_MESSAGE`
  - `JOptionPane.QUESTION_MESSAGE`

Perguntas:

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f4869676820

O que é um wrapper?

É a utilização de um tipo primitivo como se fosse um objeto;

Para que serve o JOptionPane?

É uma classe que possibilita a criação de uma caixa de dialogo padrão que solicita um valor para o usuário ou retorna uma informação;

Questions and Comments

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

- What questions or comments do you have?