



High performance. Delivered.

# Application Delivery Fundamentals: Java

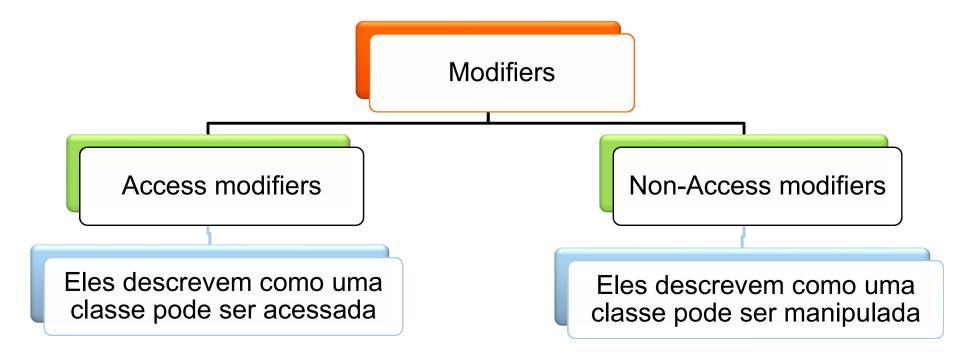
Module 4: Classes and Objects

#### Modificadores de acesso

"Modificadores de Acesso"

#### **Member Modifiers**

- As classes geralmente precisam controlar como seus atributos e comportamentos são acessados.
- Modificadores permitem esse controle.
- Existem dois tipos de modificadores.



#### **Access Modifiers**

- 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

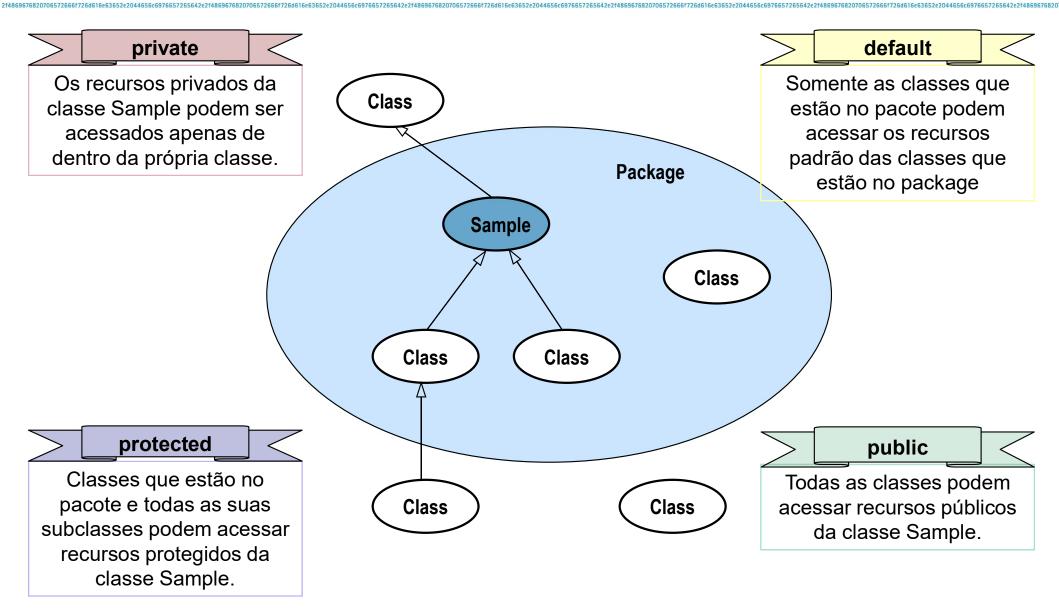
protected

 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



<sup>\*</sup> Default is not a modifier; it is just the name of the access level if no access modifier is specified.

#### **Non-access Modifiers**

Non-access modifiers - descreve como uma classe pode ser manipulada.

static

 Membro pertencente a uma classe; compartilhado por todas as instâncias da classe;

final

 Membro declarado como constante. Não pode ser modificado uma vez declarado;

abstract

 O método é declarado, mas nunca é instanciado. Este modificador pode ser aplicado a métodos para estender uma classe;

strictfp

 O método implementa aritmética estrita de ponto flutuante

# Non-access Modifiers (cont.)

Non-access Modifiers - describes how a class can be manipulated.

synchronized

 O método é executado por apenas um thread por vez. Pode ser aplicado a blocos de código e métodos;

native

 A implementação do método é escrita em outro idioma. Só pode ser aplicado a métodos;

transient

 Úma variável de instância não é salva quando seu objeto é persistido ou serializado. Só pode ser aplicado a variáveis;

volatile

 A variável é modificada de forma assíncrona executando segmentos simultaneamente. Pode ser aplicado apenas a variáveis

#### **Métodos**

#### "Métodos Getters e Setters"

#### Métodos Acessores e Modificadores

- 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



#### 'Setter' Methods

- 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).

#### 'Getter' Methods

- 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.

#### **Perguntas:**

# O que é modificador de acesso?

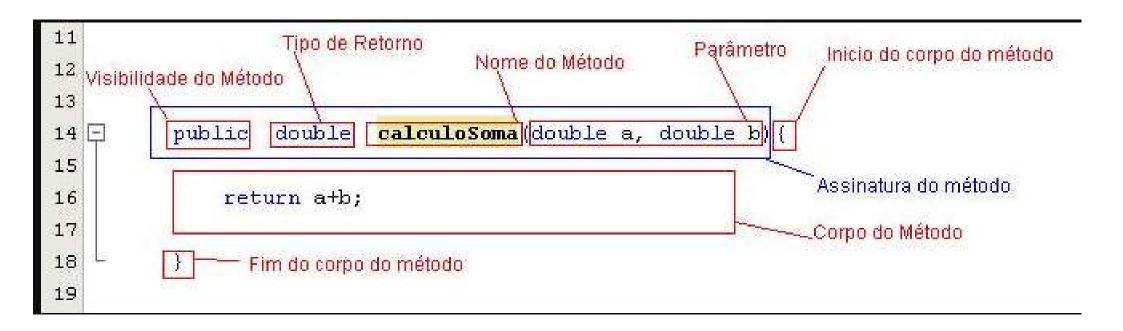
São palavras-chave que garantem níveis de acesso aos atributos, métodos e classes

# 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

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



#### **Overload**

```
public class Classe (
10
                                    Sobrecarga (Overload) de Construtores - O primeiro
                                    não possul parâmetro, já o segundo possul.
11
          private int numa;
          private int numb;
15 -
          public Classe() (
               this.numm = 2;
16
               this.numb = 2;
17
18
19
                                                             Sobrecarga de Métodos, todos eles tem a
          public Classe (int numa, int numb) (
20 -
                                                             mesma função, mais recebem
21
               this.numa = numa;
                                                            parametros diferentes, e tem retornos
               this.numb = numb;
22
                                                             diferentes, ou nenhum retorno.
23
24
25 -
          public int somaValores() {
26
               return numa+numb;
27
28
29 -
          public void somaValores (int a, int b) (
               this.numa = a;
30
               this.numb = b;
31
               int total = a+b;
32
33
34
35 -
          public int somaValores (double b, double a) {
               double total = a+b;
36
37
               int contotal = (int) total;
               return contotal;
38
39
```

#### **Restrictions on Use of Modifiers**

- 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 palavrachave 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

<Class Name>.<static member>
Calendar.getInstance();

#### **Static Code Blocks**

- 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

i

Refer to the StaticSample2.java sample code.





#### Atividade 1

- Crie uma classe java ContaCorrente.java dentro do package sef.module4.activity;
- Com os atributos: numero, nome, saldo e data;
- Com os métodos: Depositar, Sacar,
   ExibirExtrato e Tranferir;
- No método constructor inicialize os atributos;





# Activity – 1

#### Atividade 1

- Crie uma classe java Cliente.java dentro do package sef.module4.activity
- Com os atributos:nome, cpf e sobrenome
- Alterar a classe ContaCorrente para incorporar os dados do cliente;



# Activity - 1

# 3

#### Atividade 1

- Crie uma classe principal java
   PrincipalContaCorrente.java dentro do package sef.module4.activity 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.



# **Activity – Create a Calculator**

- In this activity, you will:
  - Abra o arquivo 'Calculator.java' no pacote sef.module4.activity
  - Leias as instruções e crie o Código para completer o programa.



#### **Perguntas:**

#### O que é um método?

É o que define o comportamento da Classe.

# O que é assinatura do método?

É à identificação do método.

# Wrappers

#### "Wrappers"



# **Wrappers**

# 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

#### 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





#### 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

# Wrappers

2468696768207065726661726661665765365262044656c697665726564262748696768707065726661726661666365262044656c697665726564262748696768207065726661726661666365262044656c697665726564262748696768207065726661726677266617266617266617266617266772666172666172666772666172667726667726667726667726667726667726667726667726667726667726667726667726772677267726772677267726772667726677267726

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

#### 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.

#### 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

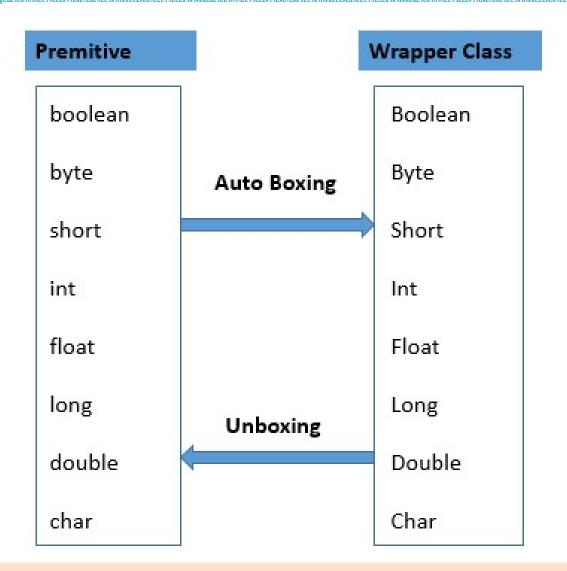
Refer to the WrappingUnwrapping.java sample code.

# **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.



i

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

"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;

```
import javax.swing.JOptionPane;

public class HellowordBox {

public static void main(String[] args) {
    JOptionPane.showMessageDialog(null, "Hello \nword ");
}

}

}

}

10 }

11 }

12 }
```

i

Refer to the HellowwordBox.java sample code.

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

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

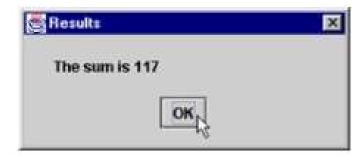
Demonstra uso de wrappers e entrada de dados

i

Refer to the Somatorio.java sample code.







### Variáveis

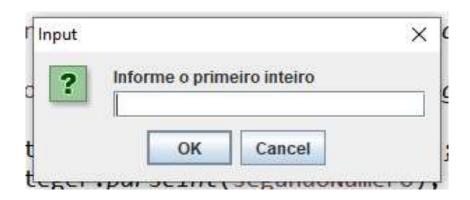
- firstNumber e secondNumber são variáveis do tipo String (package java.lang)
  - strings encapsuladas (wrapper)

```
10
11
12
```

```
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.

```
numero1 = Integer.parseInt(primeiroNumero);
numero2 = Integer.parseInt(segundoNumero);
numero1 = Integer.parseInt(segundoNumero);
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

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







### Atividade 2

- Alterar o Xadrez e do Jogo da Velha, para utilizar o jOptionPane.
- Implementar as validações necessárias.

### **Perguntas:**

## 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;





### jOptionPane

Crie uma classe Retangulo que obedeça à descrição abaixo:

# Retangulo - lado1: double - lado2: double - area: double - perimetro: double + Retangulo() + Retangulo(lado1: double, lado2: double) + calcularArea(): void + calcularPerimetro(): void

- A classe possui os atributos lado1, lado2, area e perimetro, todos do tipo float.
- O método calcularArea deve realizar o cálculo da área do retângulo (area = lado1 \* lado2). Em seguida, deve escreve o valor da área na tela.
- O método calcularPerimetro faz o cálculo do perimetro (perimetro = 2\*lado1 + 2\*lado2). Em seguida, deve escreve o valor do perímetro na tela.

# Atividade - 3



## jOptionPane

Crie o método main e instancie a classe Retangulo, criando um objeto novoRetangulo do tipo Retangulo.

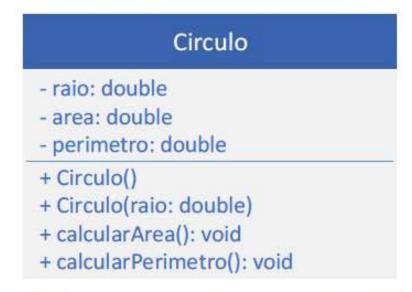
- Atribua o valor 10 ao atributo lado1.
- Atribua o valor 5 ao atributo lado2.
- Chame o método calcularArea.
- Chame o método calcularPerimetro.
- Atribua o valor 7 ao atributo lado2.
- Chame o método calcularArea.
- Chame o método calcularPerimetro.
- Crie outras 5 instancias de retângulos, conforme as instruções anteriores.





### jOptionPane

2) Crie uma classe Circulo que obedeça à descrição abaixo:



- A classe possui os atributos raio, area e perímetro, todos do tipo float.
- O método calcularArea deve realizar o cálculo da área do retângulo (area = raio \* raio \* 3.14). Em seguida, deve escreve o valor da area na tela.
- O método calcularPerimetro faz o cálculo do perimetro (perimetro = 2 \* 3.14 \* raio). Em seguida, deve escreve o valor do perímetro na tela.

### Atividade - 4



## jOptionPane

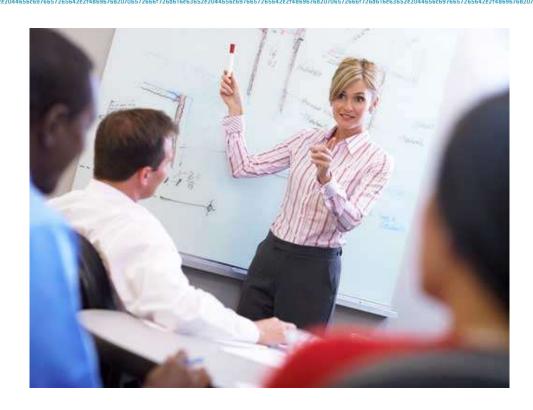
Crie o método main e instancie a classe Circulo, criando um objeto novoCirculo do tipo Circulo.

- Atribua o valor 10 ao atributo raio.
- Chame o método calcularArea.
- Escreva na tela o valor da área.
- Chame o método calcularPerimetro.
- Escreva na tela o valor do perímetro.
- Atribua o valor 4 ao atributo raio.
- Chame o método calcularArea.
- Escreva na tela o valor da área.
- Chame o método calcularPerimetro.
- Escreva na tela o valor do perímetro.
- Instancie outros 5 círculos conforme instruções anteriores.

### **Questions and Comments**

 What questions or comments do you have?





# **Checkpoint Question**

### **KAHOOT!**

