



High performance. Delivered.



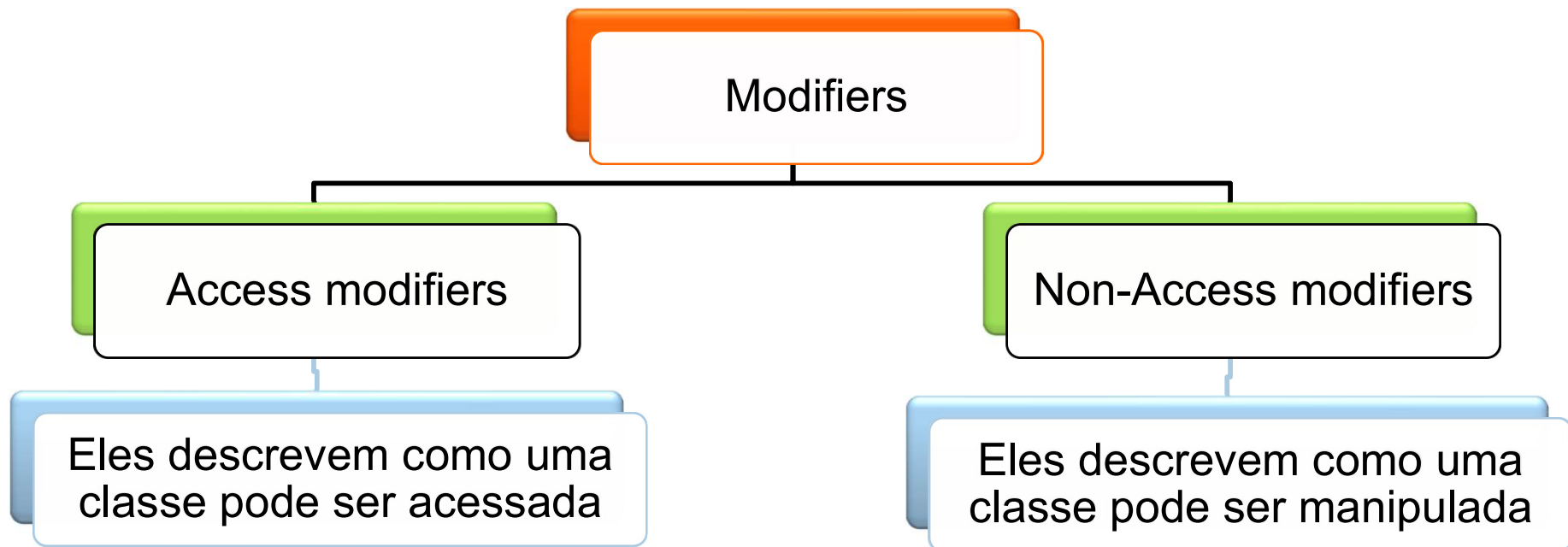
Application Delivery Fundamentals: Java

Module 4: Classes and Objects

[illegible]

Copyright © 2023 Accenture All Rights Reserved. 2

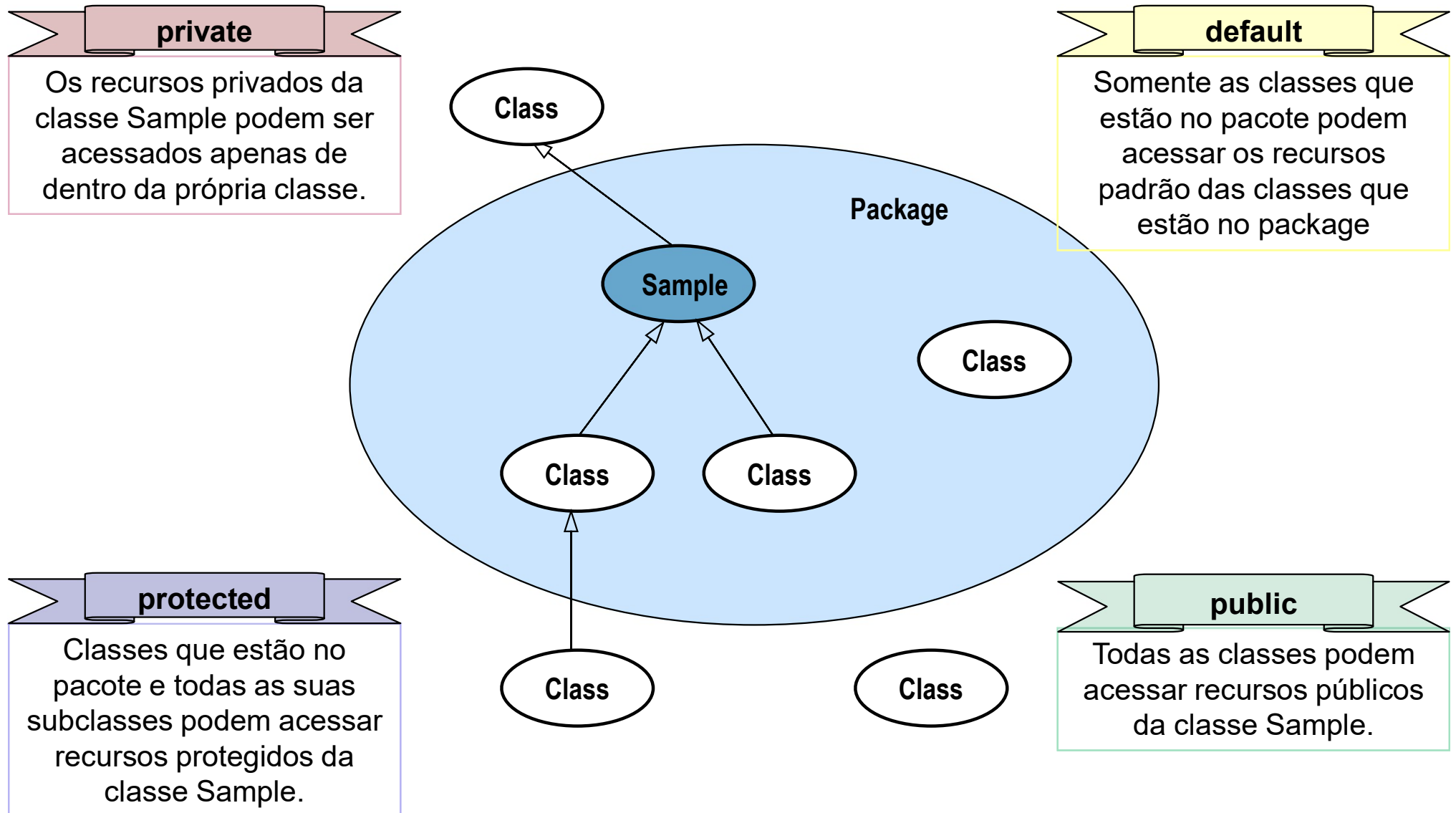
- As classes geralmente precisam controlar como seus atributos e comportamentos são acessados.
- Modificadores permitem esse controle.
- Existem dois tipos de modificadores.



214869676820706572666f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820

- | | |
|----------------------------------|---|
| default /
no modifier | <ul style="list-style-type: none">• Uma classe pode ser acessada por classes pertencentes à mesma package |
| public | <ul style="list-style-type: none">• Um membro da classe pode ser acessado por qualquer classe em qualquer package |
| protected | <ul style="list-style-type: none">• Um membro da classe só pode ser acessado por si ou por suas subclasses; |
| private | <ul style="list-style-type: none">• Um membro da classe só pode ser acessado por ele mesmo |

Member Access Modifiers Diagram



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

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

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.

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

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

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

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

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

[illegible]

Copyright © 2023 Accenture All Rights Reserved. 8

- The diagram shows a blue pen on the left. To its right is a box representing the **Caneta** class. The box is divided into three sections: a header section for the class name, an attributes section, and a methods section.

Caneta
+ modelo - ponta
+ getModelo() + setModelo(m) + getPonta() + setPonta(p)





- Copyright © 2023 Accenture All Rights Reserved.

- Copyright © 2023 Accenture All Rights Reserved.

[illegible]

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',

The diagram shows a Java method definition with labels pointing to its parts:

- Visibilidade do Método:** public
- Tipo de Retorno:** double
- Nome do Método:** calculoSoma
- Parâmetro:** (double a, double b)
- Início do corpo do método:** {
- Assinatura do método:** public double calculoSoma(double a, double b)
- Corpo do Método:** return a+b;
- Fim do corpo do método:** }

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

Overload

```
10 public class Classe {  
11  
12     private int numa;  
13     private int numb;
```

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

```
15 public Classe() {  
16     this.numa = 2;  
17     this.numb = 2;  
18 }
```

```
20 public Classe(int numa, int numb) {  
21     this.numa = numa;  
22     this.numb = numb;  
23 }
```

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

```
25 public int somaValores() {  
26     return numa+numb;  
27 }
```

```
29 public void somaValores(int a, int b) {  
30     this.numa = a;  
31     this.numb = b;  
32     int total = a+b;  
33 }
```

```
35 public int somaValores(double b, double a) {  
36     double total = a+b;  
37     int contotal = (int)total;  
38     return contotal;  
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 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

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

- ```
public class SampleClass{

 static{
 //your code here
 }

}
```





## • Atividade 1

-



## • Atividade 1

- 
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a marker in her right hand and a folder in her left. Two men in business suits are seated at a glass table in the foreground, looking towards the whiteboard. The man on the left is holding a laptop, and the man on the right has his hand on his chin, appearing to be in deep thought. The room has large windows in the background, letting in bright light.

## • Atividade 1

-

[illegible]

- 
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a marker in her right hand and a folder in her left. Two men in business suits are seated at a glass table in the foreground, looking towards the whiteboard. The man on the left is holding a laptop, and the man on the right has his hand on his chin, appearing to be in deep thought. The room has large windows in the background, letting in bright light.

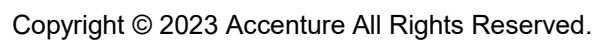
[illegible]

## É à identificação do método.



[illegible]

# Java Wrap



# Wrappers

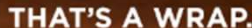
## Classes Wrappers

- 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

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

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

A large burrito filled with chicken, vegetables, and cheese. The burrito is wrapped in a light-colored tortilla and is shown from a side-on perspective, revealing the filling. The filling includes chunks of white chicken, shredded purple cabbage, green bell peppers, orange carrots, and melted white cheese. The burrito is set against a plain white background.



| Tipo primitivo | Classe corrispondente |
|----------------|-----------------------|
| boolean        | Boolean               |
| char           | Character             |
| int            | Integer               |
| long           | Long                  |
| float          | Float                 |
| double         | Double                |

## 28





```
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!");
```

Copyright © 2023 Accenture All Rights Reserved.

# 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)
```

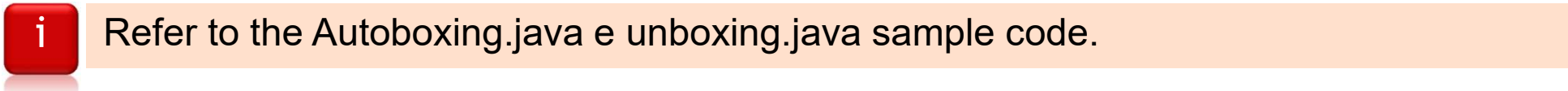
```
boolean = Boolean.valueOf(aString).booleanValue();
```



Refer to the WrappingUnwrapping.java sample code.

- 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

- 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.



[illegible]

Copyright © 2023 Accenture All Rights Reserved. 33



2f4869676820706572666f72d6d16e36352e2044656c6976657265642ef4869676820706572666f72d6d16e36352e2044656c6976657265642ef4869676820706572666f72d6d16e36352e2044656c6976657265642ef4869676820706572666f72d6d16e36352e2044656c6976657265642ef4869676820706572666f72d6d16e36352e2044656c6976657265642ef48696768207

### § Requer dois argumentos

- § O segundo argumento é o string a apresentar

- 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



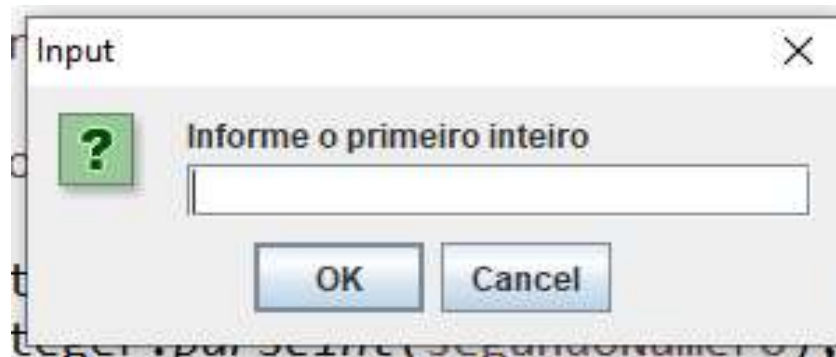
Refer to the Somatorio.java sample code.

## Variáveis

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

10  
11  
12  
13

- Método `JOptionPane.showInputDialog` apresenta:







§ Note que o argumento da função aparece como texto.

```
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`



- § 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

[illegible]

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



# Atividade - 3

- **jOptionPane**

1) 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 ( $\text{area} = \text{lado1} * \text{lado2}$ ). Em seguida, deve escrever o valor da área na tela.
- O método calcularPerimetro faz o cálculo do perímetro ( $\text{perimetro} = 2 * \text{lado1} + 2 * \text{lado2}$ ). Em seguida, deve escrever o valor do perímetro na tela.

- **jOptionPane**

- 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.



# Atividade - 4

- **jOptionPane**

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

## Circulo

- raio: double
- area: double
- perimetro: double

---

- + Circulo()
- + Circulo(raio: double)
- + calcularArea(): void
- + calcularPerimetro(): void

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





- **jOptionPane**

- 



[illegible]

# Kahoot!