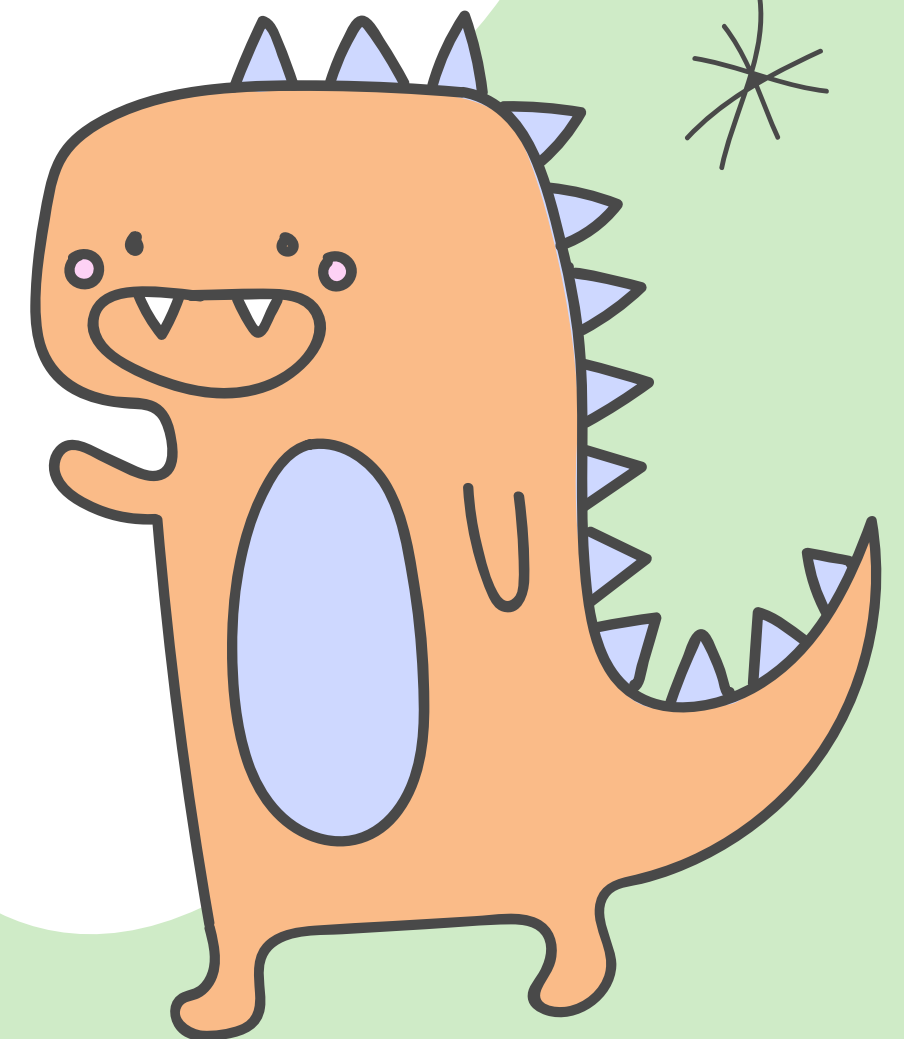
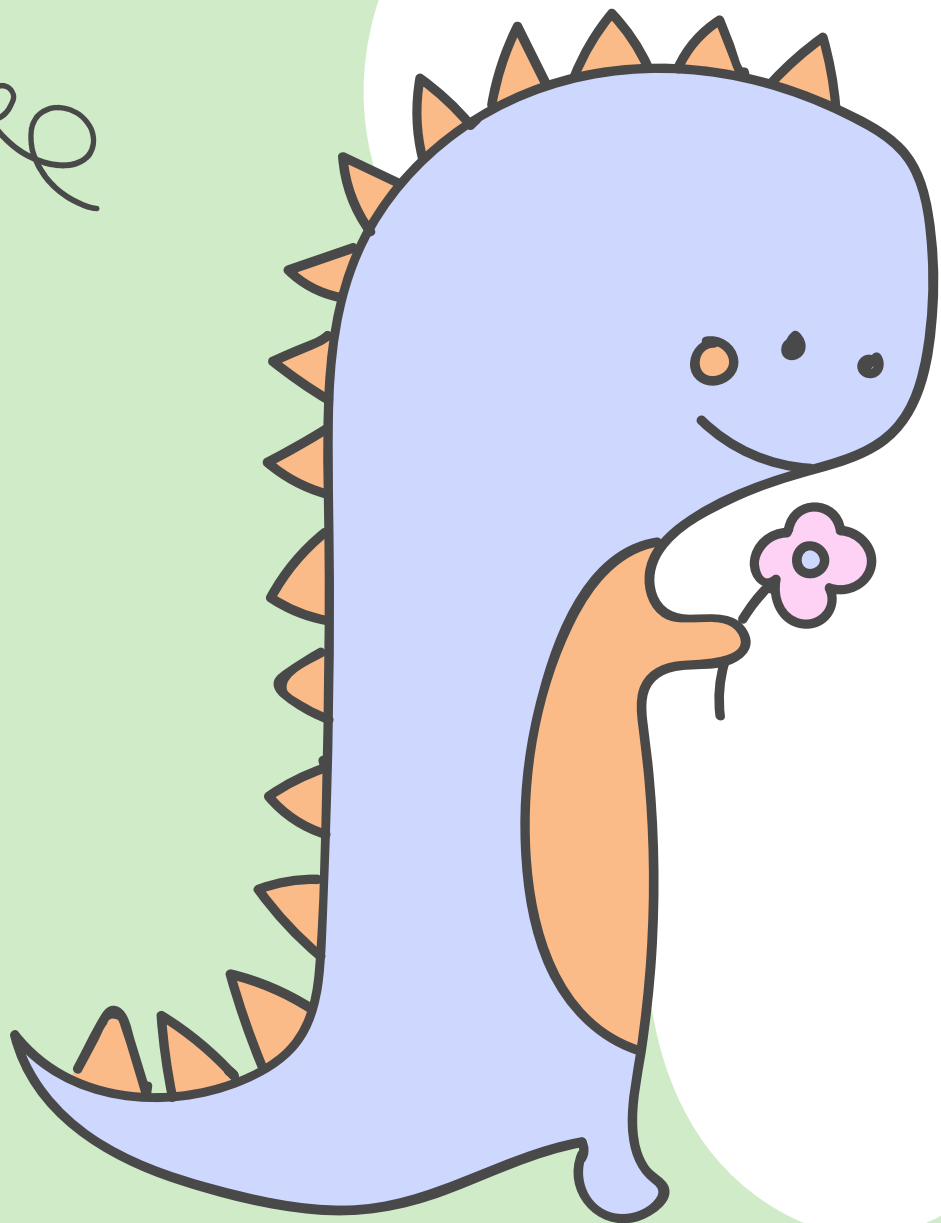
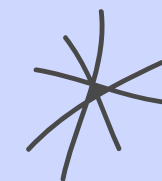


Aula - 1

Classes e Funções no Java



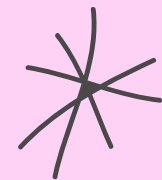
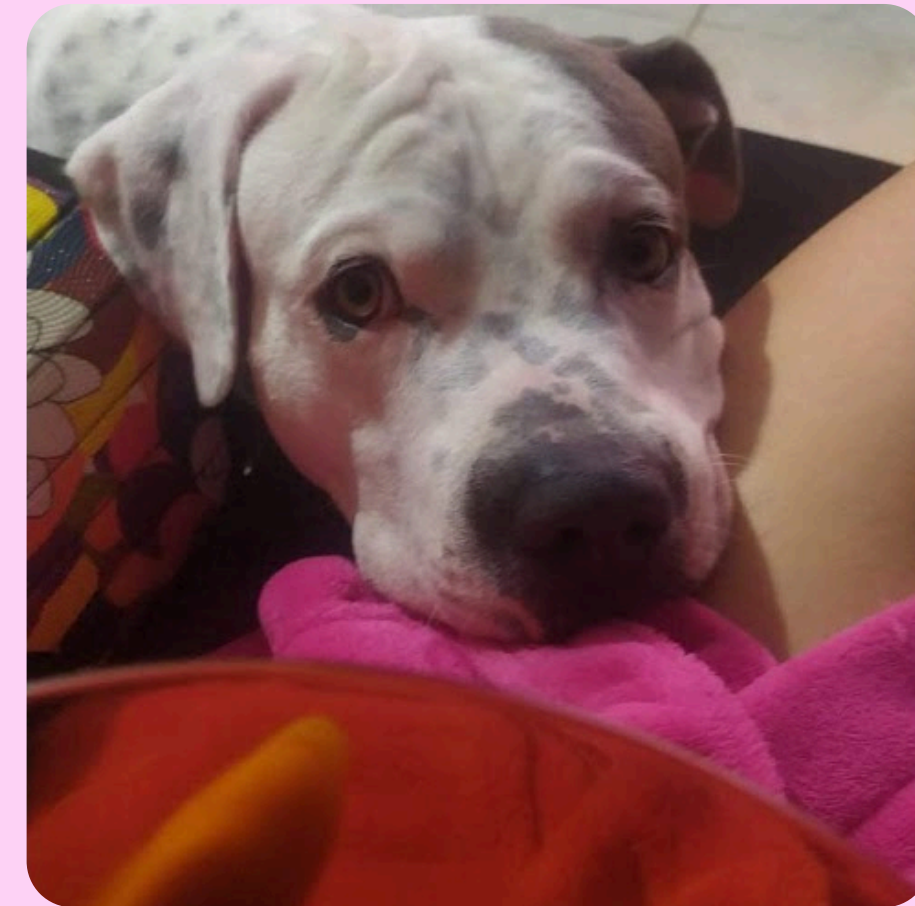
Professor
Apresentar
Namorado
Gostoso



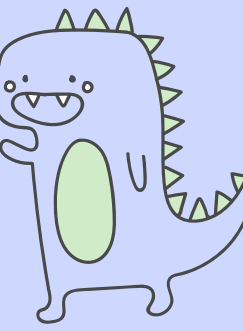
“Você está linda hoje”
Pedro Vitor Emanuel Soares Francelino
de Souza

O que é uma classe

Uma classe é uma definição ou um **molde** que descreve as propriedades e comportamentos comuns dos objetos que dela derivam. É como um plano para criar objetos, especificando quais **atributos** (variáveis) e **métodos** (funções) eles terão. Por exemplo, uma classe "Cachorro" pode ter atributos como nome, boca, olhos, e métodos como andar e comer.

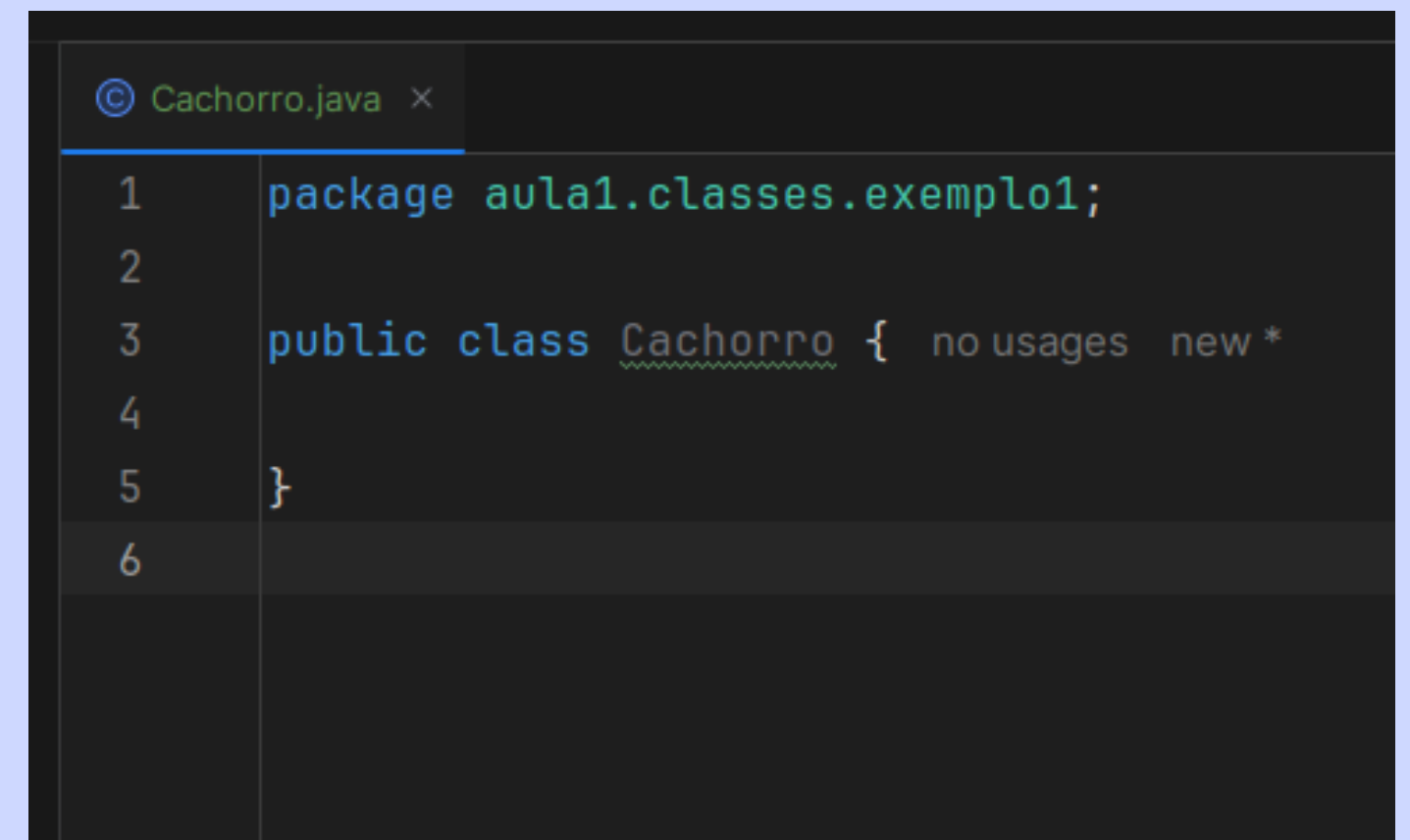
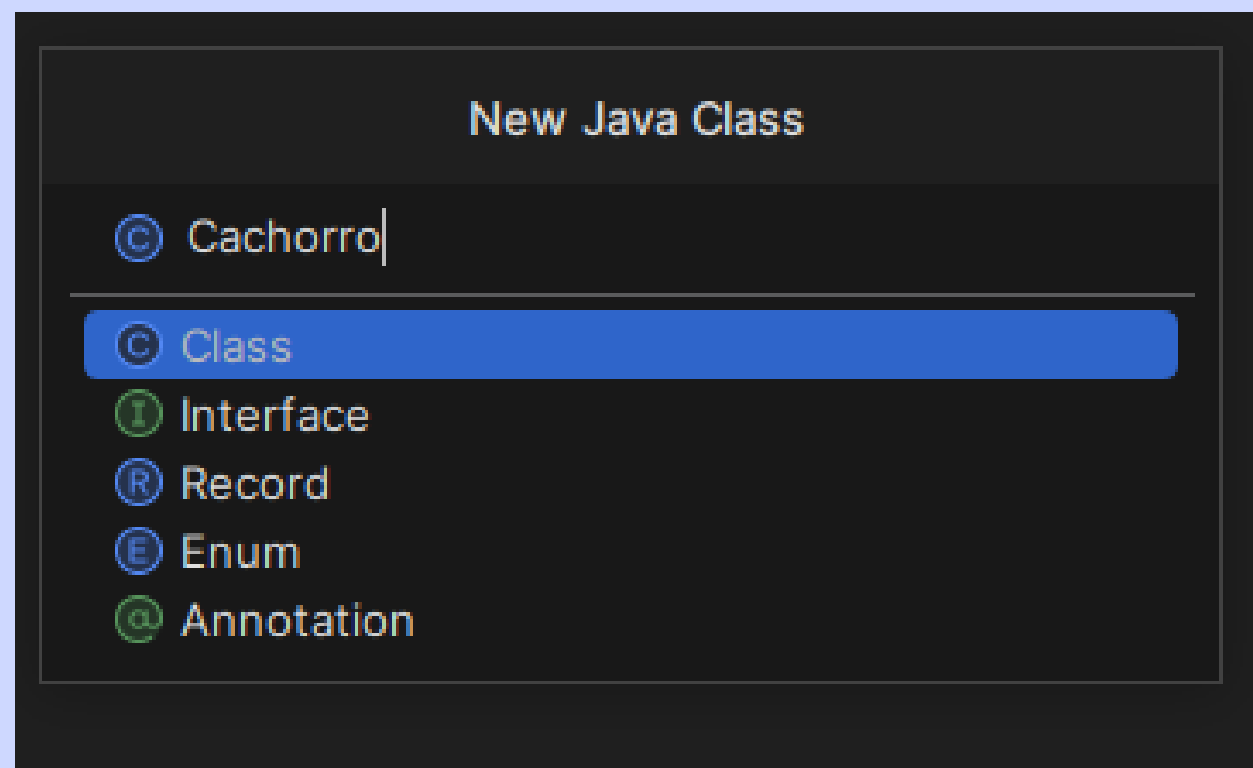


Como criar uma classe no java?



- Pelo IntelliJ basta criar um novo projeto, e dentro dele criar uma nova classe que vamos chamá-la de “Cachorro”

- Pronto! Criamos nossa primeira classe! Ela está vazia, mas lembre que uma classe é composta por atributos e métodos



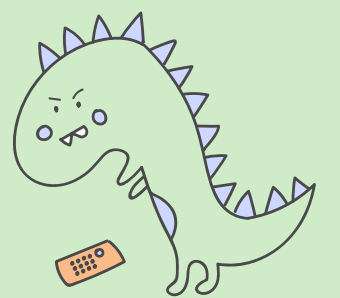
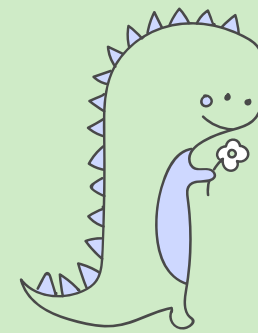
Agora vamos pensar em todo
cachorro tem...
(isso depende da abstração para cada
caso)

nome;
cor;
tamanho;
pintinhas?;
amigável;

E por que não colocar um atributo
“Dono”?
Um cachorro pode ter um dono, não é?

Agora vamos pensar em todo cachorro faz...
(agora estamos pensando em ações que
eles fazem)

latir;
andar;
comer;
brincar (com Pedro);
pegar préa;





Como passar isso para o Java?

Para criar um atributo (característica) basta apenas definir o:

- tipo de visibilidade
- tipo elementar dela (String, int, double...)
- seu nome (pode ser qualquer um que faça sentido;

```
public class Cachorro { no usages new *  
  
    private String nome; no usages  
  
}
```

Viu como é fácil?

Agora é sua vez!
Cria uma classe cachorro e
coloque alguns atributos
nela! Coloque vários tipo de
atributos!

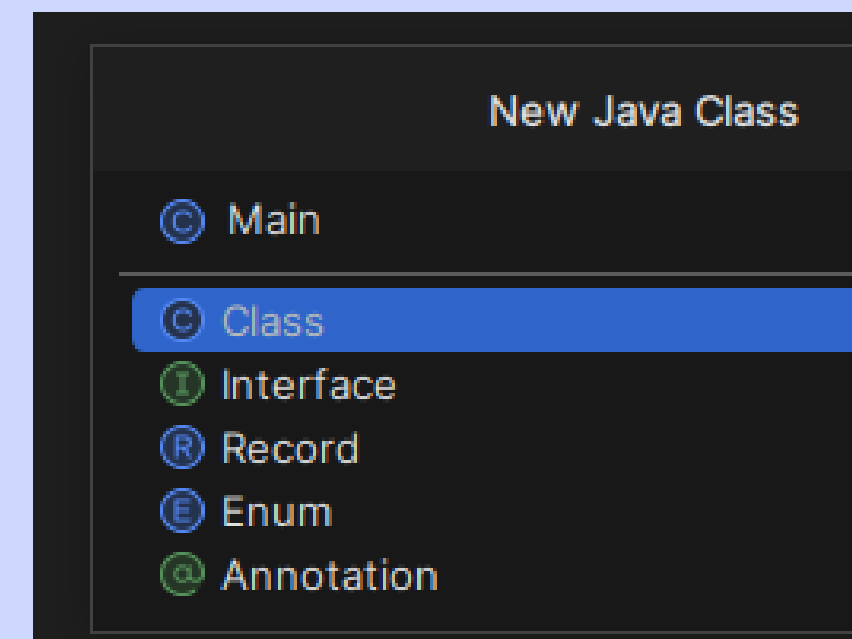


Como instanciar uma classe?



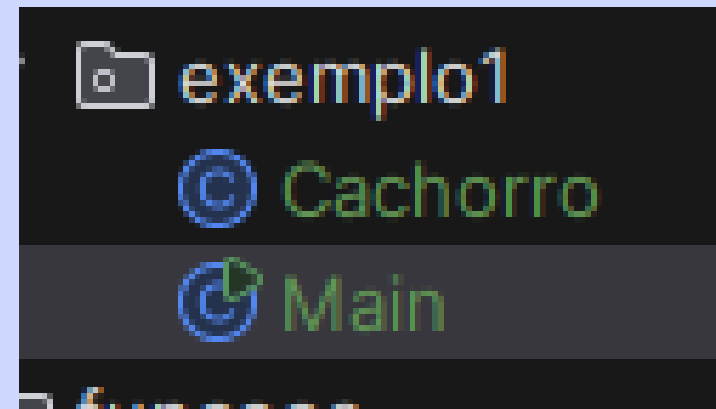
Quanto estamos instanciando uma classe quer dizer que o modelo passa a virar um objeto real. Cachorro é um modelo, existem vários, mas Apolo é único. No Java Apolo só vai passar a existir se ele for instanciado.

Lembra do método main? Para uma melhor organização, cada classe terá seu arquivo próprio, e temos uma classe principal para executar TUDO que queremos! (mais para frente você verá que podemos instanciar classes dentro de outras classes).



```
1 package aula1.classes.exemplo1;
2
3 public class Main {
4     public static void main(String[] args) {
5
6     }
7 }
8
```

Agora temos isso aqui:



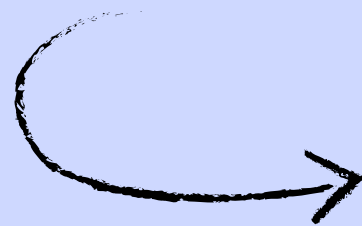
Uma classe de modelo, e uma classe que é a nossa principal. Todo começa e termina nela!
VAMOS CRIAR APOLO dentro da classe principal (Main)!

```
public class Main {  
    new *  
    public static void main(String[] args) {  
        Cachorro apolo = new Cachorro();  
    }  
}
```

Pronto, criamos um objeto cachorro. Agora ele é real, e podemos executar todos seus métodos, verificar seus atributos e tudo mais!

Se adicionarmos um novo atributo na classe cachorro do tipo “public” podemos verificar o que o objeto “apolo” possui:

```
public class Cachorro { 2 usages new *  
  
    private String nome; no usages  
    public String dono = "Laura"; no usages  
  
}
```



```
public class Main { new *  
    public static void main(String[] args) { n  
  
        Cachorro apolo = new Cachorro();  
        System.out.println(apolo.dono);  
  
    }  
}
```



Laura

Construtores!

Da forma que está, TODOS os cachorros que forem criados terão o mesmo dono. Como podemos mudar isso? Quero falar explicitamente quem é o dono de cada cachorro que for criado.



Construtores!

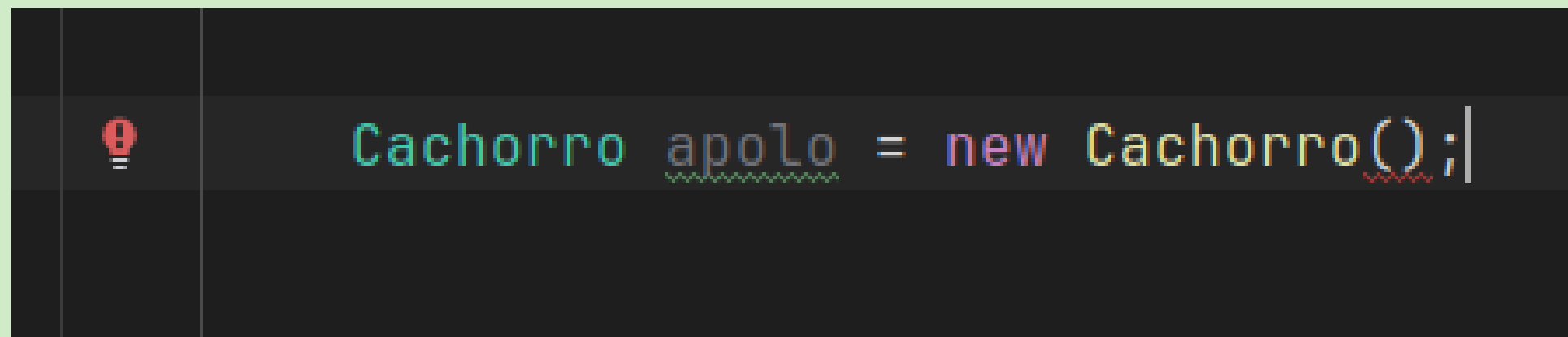
O construtor é um método especial, que tem o mesmo nome da classe. Se colocarmos um construtor. A objeto será iniciado com que passarmos para ele!

```
public class Cachorro { 2 usages new *  
  
    private String nome; 1 usage  
    public String dono; 1 usage  
  
    public Cachorro(String nome, String dono){  
        this.nome = nome;  
        this.dono = dono;  
    }  
  
}
```

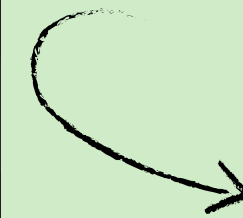
```
public class Main { new *  
    public static void main(String[] args) { new *  
  
        Cachorro apolo = new Cachorro(nome: "Apolo", dono: "Laura");  
  
    }  
}
```

Construtores!

Quando colocamos um construtor PRECISAMOS adicionar o que ele pede, no caso um “nome” e um “dono”. Caso não, o programa não funcionará.



```
Cachorro apolo = new Cachorro();|
```



```
/home/pedrofrs/IdeaProjects/aulas-java/codigos/src/aula1/classes/exem  
java: constructor Cachorro in class aula1.classes.exemplo1.Cachorro c  
required: java.lang.String,java.lang.String  
found:    no arguments  
reason: actual and formal argument lists differ in length
```

O construtor precisa de alguma coisa dentro dos parênteses, mas está vazio :(

E as funções/métodos?



Como sabemos funções são ações que podem ser utilizadas quantas vezes for necessário.

Ela segue o mesmo padrão dos atributos, porém o TIPO dela precisa ser definido (assim como os atributos) mas as funções podem retornar alguma coisa.



O que isso quer dizer?

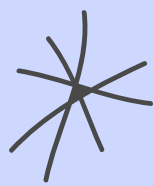
Funções que não retornam nada

```
public void latir(){ no usages new *  
    System.out.println("AUAUAU");  
}
```

Funções que retornam algo

```
private double pesoCachorro(){  
    return this.peso;  
}
```





Vamos adicionar métodos!
AGORA É COM VOCÊ



Link dos códigos:

<https://github.com/pedrofrs/aulas-java>

Página de recursos

Use estes elementos e ilustrações na sua apresentação do Canva. Bom design!

