

Polimorfismo

Jose.wellington@uniceub.br

Calendário

◀ agosto de 2013 ▶

D	S	T	Q	Q	S	S
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
-	-	-	-	-	-	-

◀ setembro de 2013 ▶

D	S	T	Q	Q	S	S
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

◀ outubro de 2013 ▶

D	S	T	Q	Q	S	S
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

◀ novembro de 2013 ▶

D	S	T	Q	Q	S	S
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

◀ dezembro de 2013 ▶

D	S	T	Q	Q	S	S
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

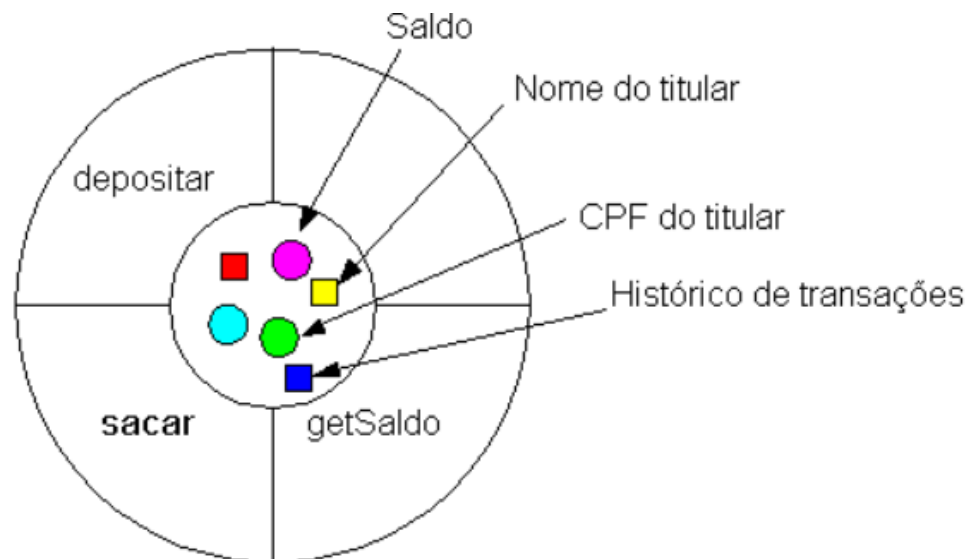
Agenda

- **Encapsulamento**
- **Método**
- **Método com parâmetros e retorno**
- **Set, Get e Is**
- **Herança**
- **Polimorfismo**
- **Exercício**

Encapsulamento

Encapsulamento

- Mecanismo utilizado visando obter segurança, modularidade e autonomia para objetos;
- Conseguido através da definição de visibilidade privada dos atributos, ganhando-se assim autonomia para definir o que o mundo externo ao objeto poderá visualizar e acessar, normalmente através de métodos públicos.



Public

Elementos **public** são utilizados em variáveis e métodos universais, onde não há problema nem necessidade de segurança daquele dado.



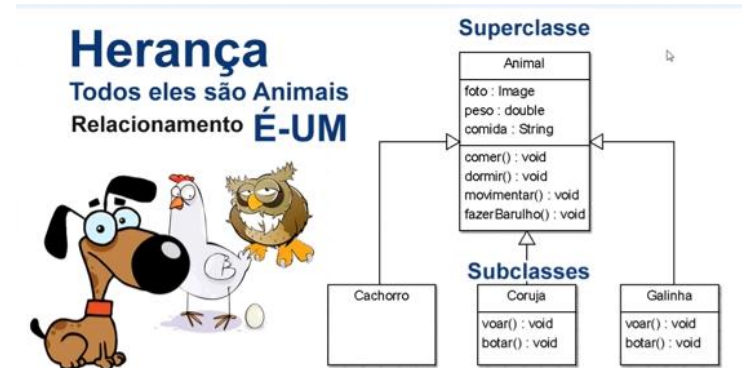
```
public class Animal {
```

```
    double peso;
    String comida;
```

```
    void dormir(){System.out.println("Dormiu");}
    void fazerBarulho(){System.out.println("Fazer Barulho");}
}
```

Private

O modificador **private** deixará visível o atributo apenas para a **classe** em que este atributo se encontra.



```
public class Animal {
```

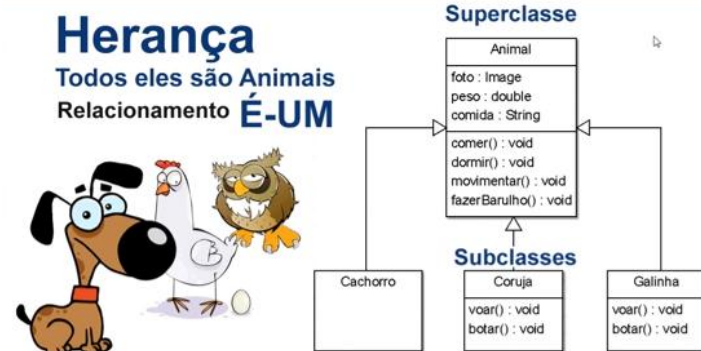
```
    private int serial;
    double peso;
    String comida;
```

```
    public Animal(double peso, String comida) {
        this.peso = peso;
        this.comida = comida;
    }
```

Protected

- O protected é pra quando você não quer deixar um atributo public, livre para todos.

Porém, você quer **compartilhar ele com as subclasses**. O protected é um intermediário entre public e private.



```

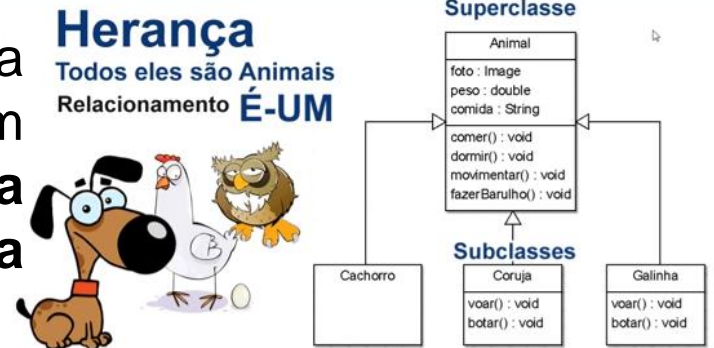
public class Animal {
    protected int serial;
    double peso;
    String comida;

    public Animal(double peso, String comida) {
        this.peso = peso;
        this.comida = comida;
    }
}

```


static

■ Quando definimos variáveis com a palavra *static* em uma classe ela terá um comportamento especial: **ela será a mesma para todos os objetos daquela classe.**



```
public static void main(String[] args) {
```

```
    Cachorro toto = new Cachorro();
```

```
    Galinha carijo = new Galinha();
```

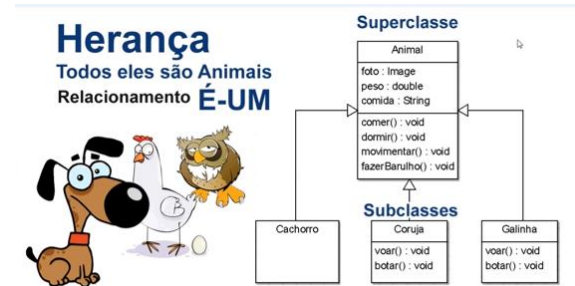
```
    System.out.println(toto instanceof Cachorro);
```

```
    System.out.println(toto instanceof Animal);
```

```
    System.out.println(toto instanceof Ave);
```

static

- Ou seja, não haverá um tipo dela em cada objeto.
- Todos os objetos, ao acessarem e modificarem essa variável, acessarão a **mesma variável**, **o mesmo espaço da memória**, e a mudança poderá ser vista em todos os objetos.



```
01. public class GlobalELocal {
02.     static int x;
03.     float y;
04.
05.     public static void main(String args[]) {
06.         char z;
07.         x = 1;
08.         z = 'a';
09.     }
10. }
```

Método

Métodos

- **Reduzem a complexidade**

- ✓ Abstração
- ✓ Encapsulam a informação
- ✓ Minimizam o tamanho do código

Aumentam a facilidade!

- **Aumentam a manutenibilidade e a correção**

- ✓ Evitam duplicação do código
- ✓ Limitam o efeito das mudanças
- ✓ Promovem a reutilização do código

Diminuem o custo!

Usando/chamando métodos

```
import java.util.Scanner;
public class menu {
    public static void menu(){
        System.out.println("Cadastro de clientes");
        System.out.println("0. Fim");
        System.out.println("1. Inclui");
        System.out.println("2. Altera");
        System.out.println("3. Exclui");
        System.out.println("4. Consulta");
        System.out.println("Opcao:");    }

    public static void inclui(){
        System.out.println("Você entrou no método Inclui.");
    }
    public static void altera(){
        System.out.println("Você entrou no método Altera.");
    }
    public static void exclui(){
        System.out.println("Você entrou no método Exclui.");
    }
    public static void consulta(){
        System.out.println("Você entrou no método Consulta.");
    }
}
```

Exemplo 01 - Método

```
public static void main(String[] args) {  
    int opcao;  
    Scanner entrada = new Scanner(System.in);  
    do{  
        menu();  
        opcao = entrada.nextInt();  
  
        switch(opcao){  
            case 1:  
                incluir();  
                break;  
            case 2:  
                altera();  
                break;  
            case 3:  
                excluir();  
                break;  
            case 4:  
                consulta();  
                break;  
            default:  
                System.out.println("Opção inválida.");  
        }  
    } while(opcao != 0);  
}
```

Método com parâmetros e retorno

```
package quadrado;
import java.util.Scanner;
public class Quadrado {
    public static int quadrado(int num){
        int quadrado;
        quadrado = num * num;
        return quadrado; }

    public static void main(String[] args) {

        int numero;
        int numero_quadrado;
        Scanner entrada = new Scanner(System.in);
        System.out.println("Entre com um inteiro: ");
        numero = entrada.nextInt();
        numero_quadrado=quadrado(numero);
        System.out.printf(numero + " elevado ao quadrado é " + numero_quadrado);
    }
}
```

Set, Get e Is

Set, Get e Is

Encapsulamento

MÉTODOS & VARIÁVEIS

1. **get** = **pegar**
2. **is** = **pegar** - **boolean**
3. **set** = **definir**

```
1 package com.exemplo.entidade;  
2  
3 public class Ponto {  
4  
5     private double x;  
6     private double y;  
7  
8 }
```

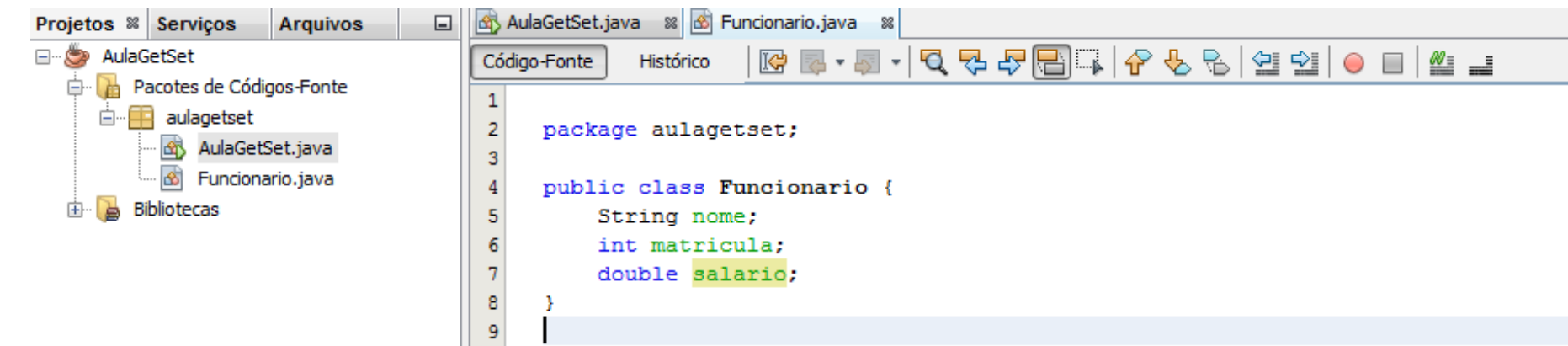
Métodos getters	Métodos setters
<pre>public String getNome() { return nome; } public double getSalario() { return salario; }</pre>	<pre>public void setNome(String nome) { this.nome = nome; } public void setSalario(double salario) { this.salario = salario; }</pre>

Set, Get e Is

Funcionario
Nome: String
Matricula: Int
Salario: double
Ativo: Boolean
getNome(): String setNome(): String
getMatricula: Int setMatricula: int
getSalario: double setSalario: double
isAtivo: Boolean getAtivo: Boolean
exibir()

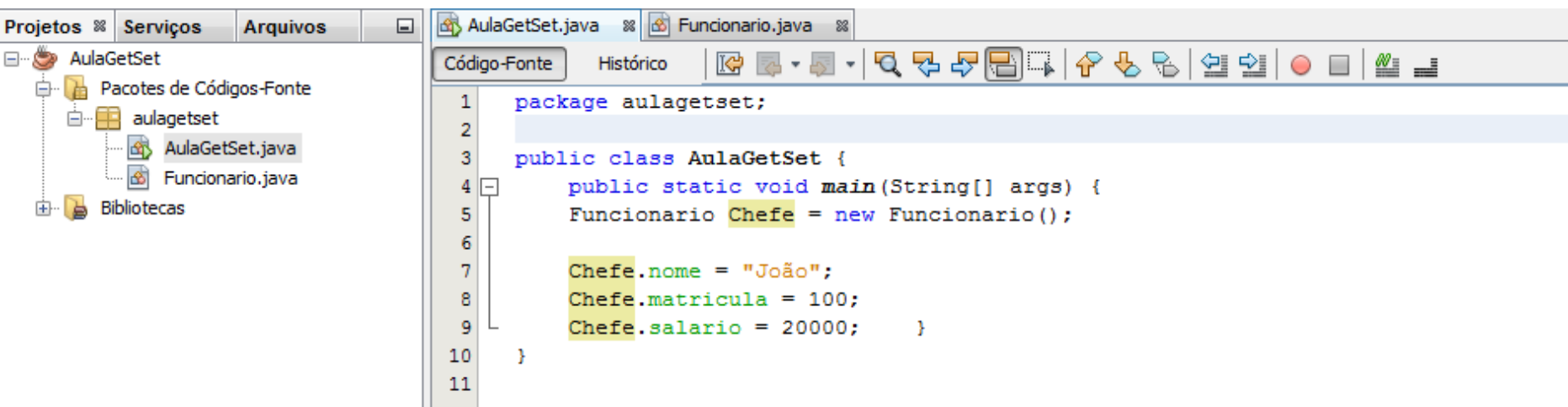


Set, Get e Is



The screenshot shows an IDE with a project named 'AulaGetSet'. The 'Arquivos' (Files) view on the left shows the project structure: 'Pacotes de Códigos-Fonte' (Source Packages) containing 'aulagetset', which contains 'AulaGetSet.java' and 'Funcionario.java'. The 'Código-Fonte' (Source Code) editor is open, showing the 'Funcionario.java' file. The code defines a package 'aulagetset' and a public class 'Funcionario' with three attributes: 'nome' (String), 'matricula' (int), and 'salario' (double).

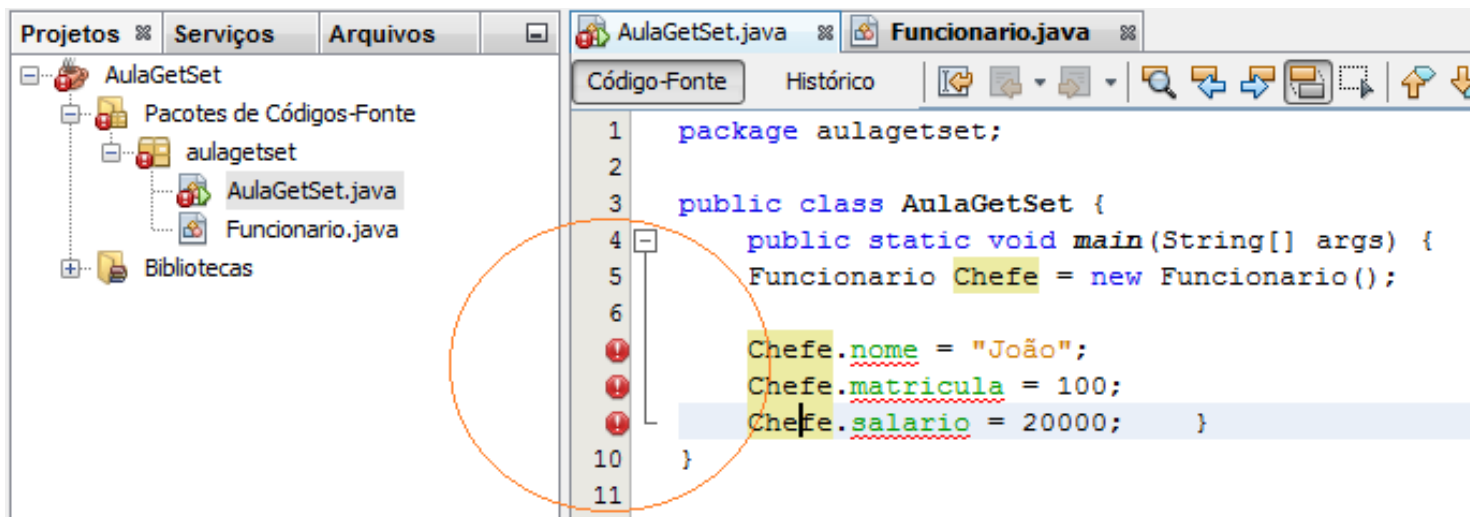
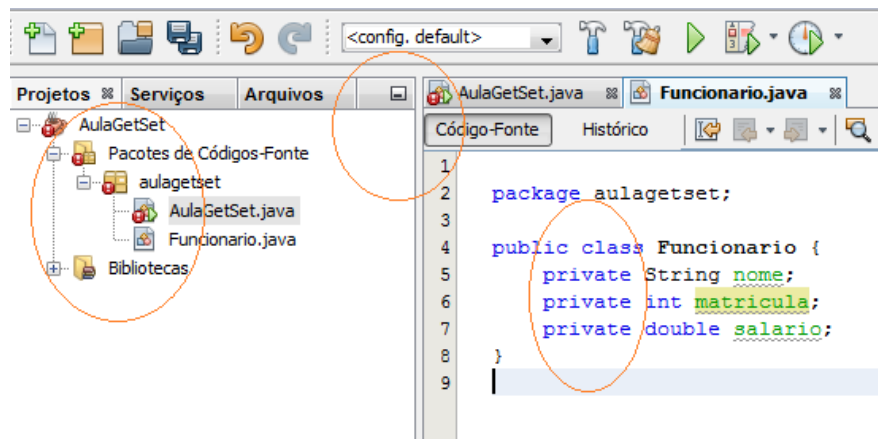
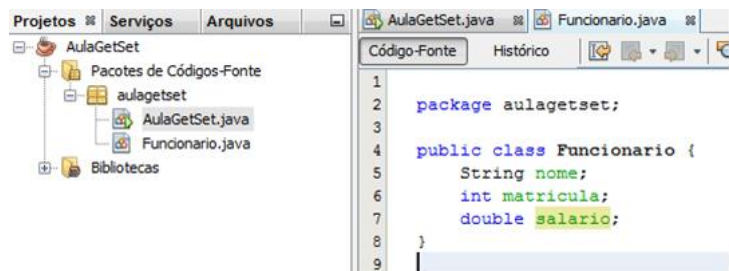
```
1 package aulagetset;
2
3
4 public class Funcionario {
5     String nome;
6     int matricula;
7     double salario;
8 }
9
```



The screenshot shows the same IDE with the 'AulaGetSet.java' file open in the 'Código-Fonte' editor. The code defines a package 'aulagetset' and a public class 'AulaGetSet' with a static 'main' method. Inside the 'main' method, a 'Funcionario' object named 'Chefe' is instantiated and its attributes are set: 'nome' to 'João', 'matricula' to 100, and 'salario' to 20000.

```
1 package aulagetset;
2
3 public class AulaGetSet {
4     public static void main(String[] args) {
5         Funcionario Chefe = new Funcionario();
6
7         Chefe.nome = "João";
8         Chefe.matricula = 100;
9         Chefe.salario = 20000;
10    }
11 }
```

Set, Get e Is



Set, Get e Is

Encapsulamento

MÉTODOS & VARIÁVEIS

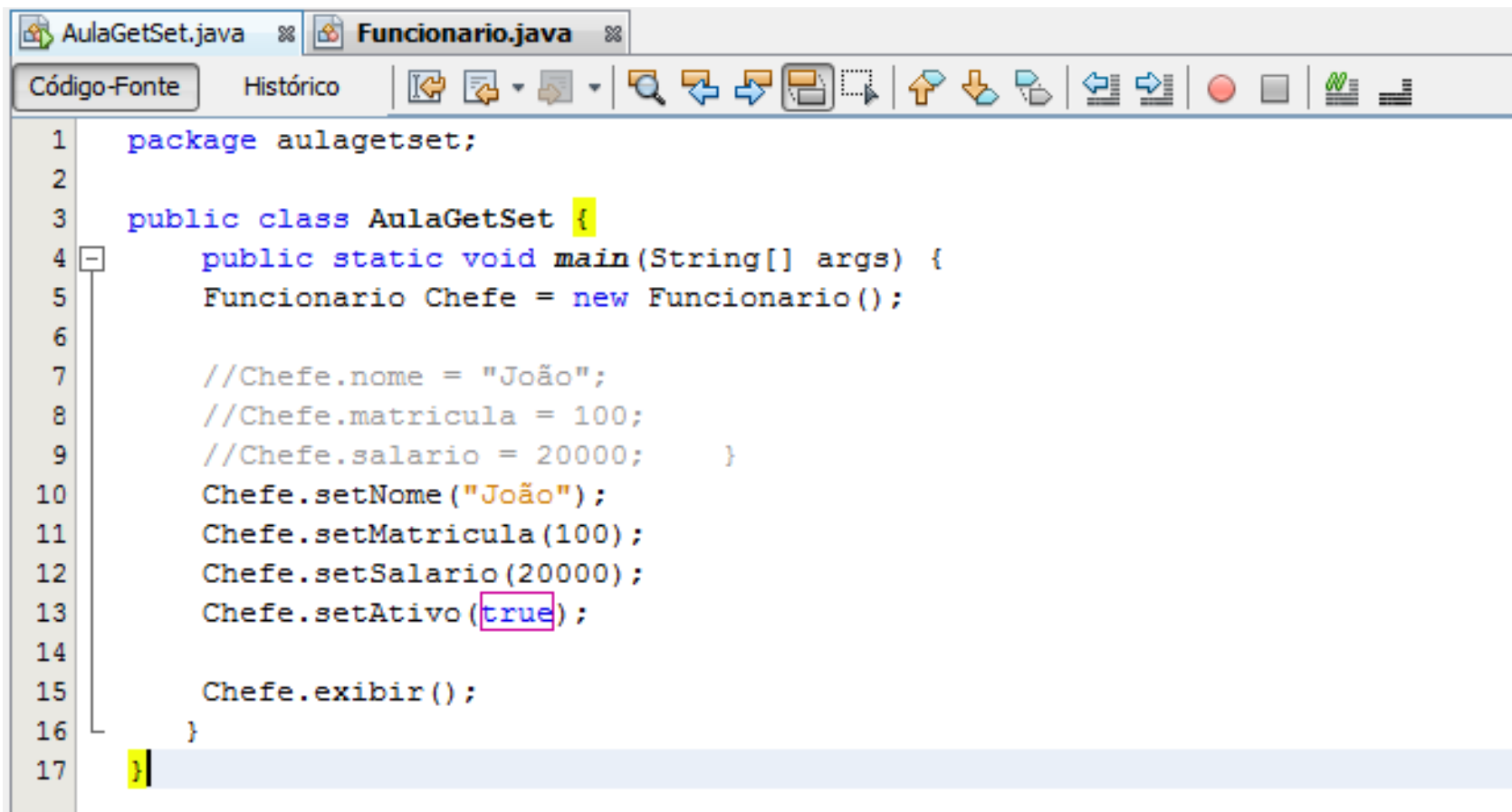
1. **get** = **pegar**
2. **is** = **pegar** - **boolean**
3. **set** = **definir**

Funcionario
Nome: String
Matricula: Int
Salario: double
Ativo: Boolean
getNome(): String setNome(): String
getMatricula: Int setMatricula: int
getSalario: double setSalario: double
isAtivo: Boolean getAtivo: Boolean
exibir()



```
2
3 public class Funcionario {
4     private String nome;
5     private int matricula;
6     private double salario;
7     private boolean ativo;
8
9     public void exibir() {
10         System.out.println("Matricula    = " + getMatricula());
11         System.out.println("Funcionario = " + getNome());
12         System.out.println("Salario     = " + getSalario() );
13         System.out.println("Ativo      = " + isAtivo() );    }
14     public void setNome( String nome ){
15         this.nome = nome;    }
16     public String getNome(){
17         return this.nome;    }
18
19     public void setMatricula( int Mat ){
20         this.matricula = Mat;    }
21     public int getMatricula(){
22         return this.matricula;    }
23
24     public double getSalario(){
25         return this.salario;    }
26     public void setSalario( double sal ){
27         this.salario = sal;    }
28
29     public boolean isAtivo() {
30         return ativo; }
31     public void setAtivo (boolean ativ) {
32         this.ativo = ativ; } }
```

Set, Get e Is



The screenshot shows an IDE window with two tabs: 'AulaGetSet.java' and 'Funcionario.java'. The 'Código-Fonte' (Source Code) tab is active. The code in 'AulaGetSet.java' is as follows:

```
1 package aulagetset;
2
3 public class AulaGetSet {
4     public static void main(String[] args) {
5         Funcionario Chefe = new Funcionario();
6
7         //Chefe.nome = "João";
8         //Chefe.matricula = 100;
9         //Chefe.salario = 20000;    }
10        Chefe.setNome("João");
11        Chefe.setMatricula(100);
12        Chefe.setSalario(20000);
13        Chefe.setAtivo(true);
14
15        Chefe.exibir();
16    }
17 }
```

The code defines a package 'aulagetset' and a class 'AulaGetSet'. The 'main' method creates a 'Funcionario' object named 'Chefe' and sets its attributes using setter methods: 'setNome', 'setMatricula', 'setSalario', and 'setAtivo'. The 'setAtivo' method is called with the argument 'true', which is highlighted with a red box. The 'exibir' method is also called. The code is enclosed in a 'package' statement and a class declaration.

Herança

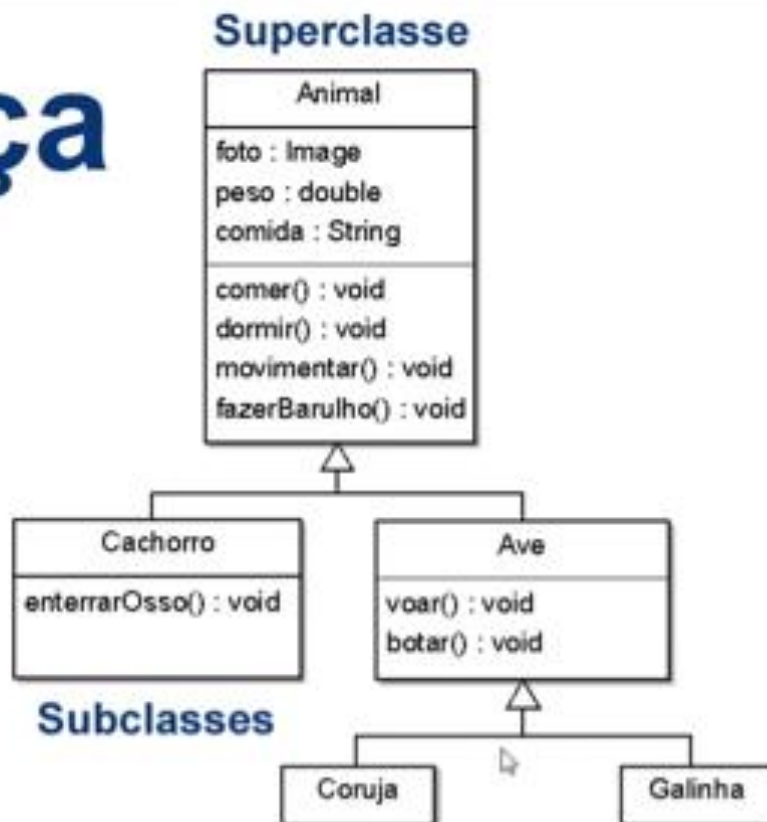
Herança

O que você ganha com herança

1. **Evita código duplicado**
2. **Você generaliza o comportamento comum a um grupo de classe**
3. **Quando vc precisar alterar um grupo de comportamento em todas as classes só altera uma vez único local**
4. **Todas alterações elas serão aplicadas em todas subclasses**

Herança

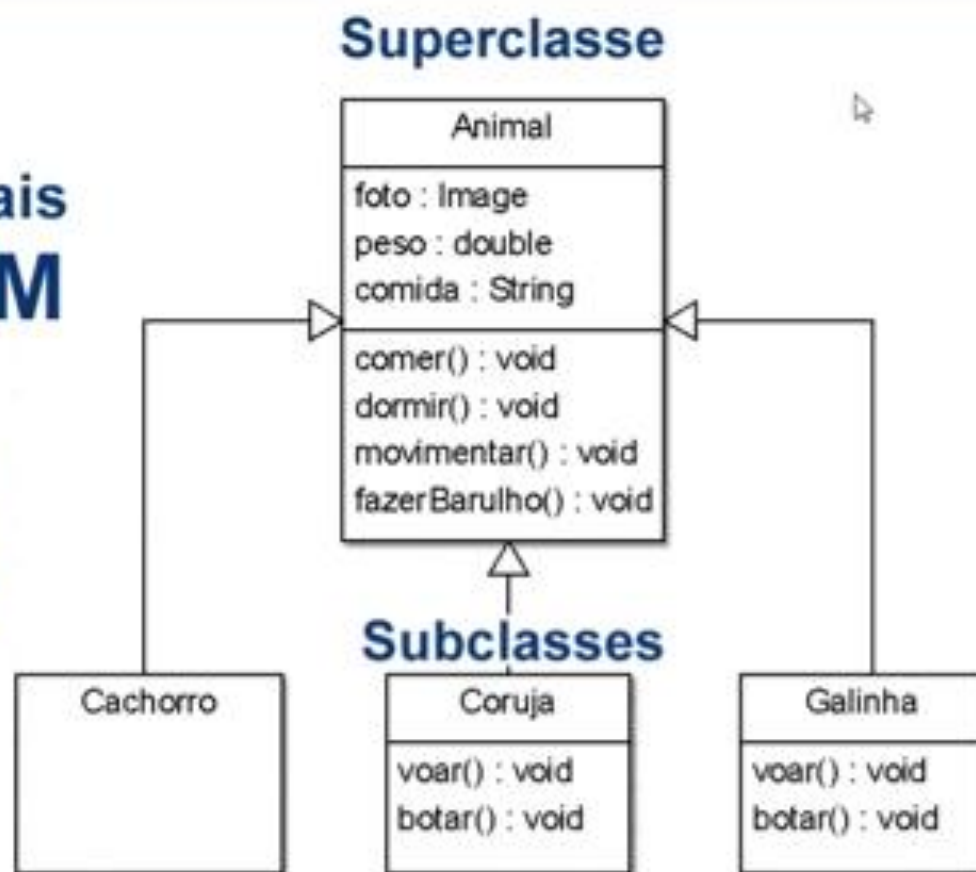
Herança



Herança

Herança

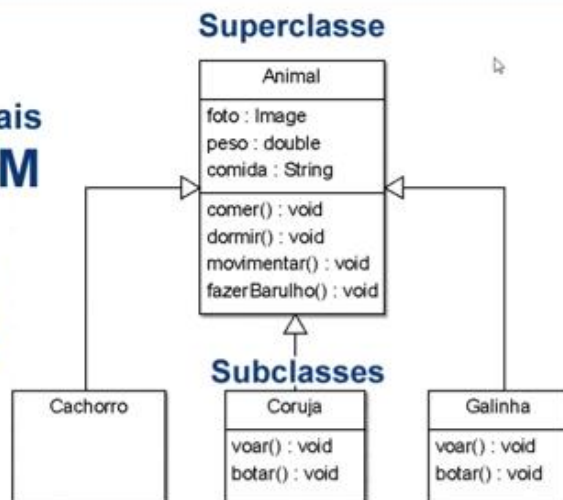
Todos eles são Animais
Relacionamento **É-UM**



Herança

Herança

Todos eles são Animais
Relacionamento É-UM



```
public class Animal {
```

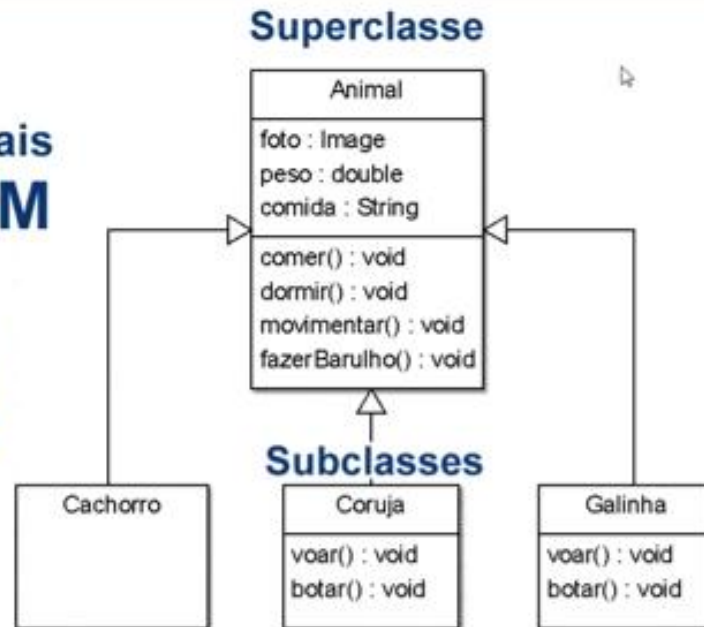
```
    double peso;  
    String comida;
```

```
    void dormir(){System.out.println("Dormiu");}  
    void fazerBarulho(){System.out.println("Fazer Barulho")  
}
```

Herança

Herança

Todos eles são Animais
Relacionamento **É-UM**



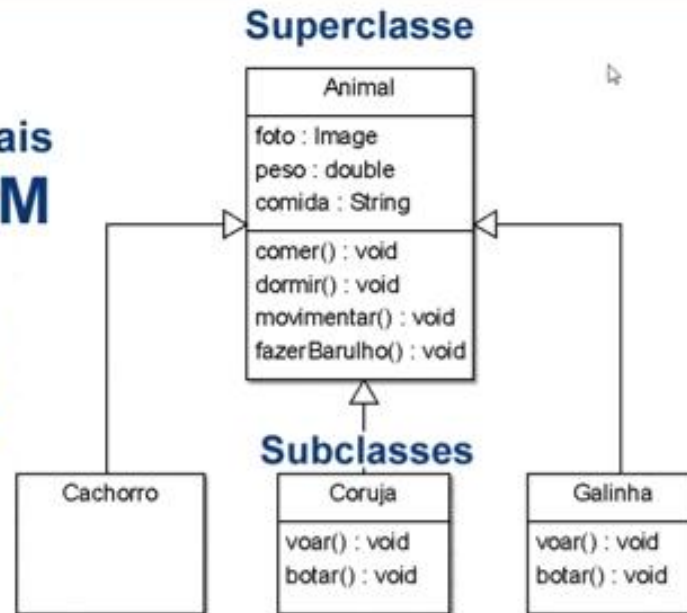
```
public class Cachorro extends Animal {
}

```

Herança

Herança

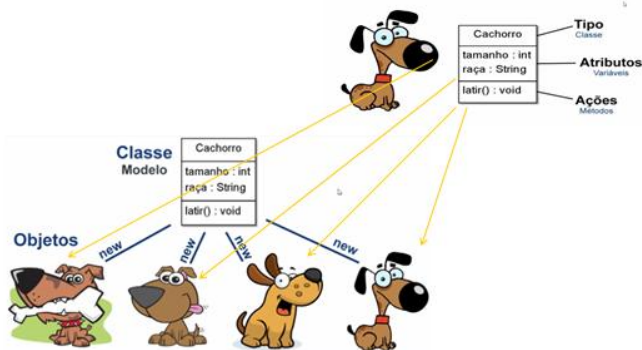
Todos eles são Animais
Relacionamento **É-UM**



```
public class Galinha extends Animal {

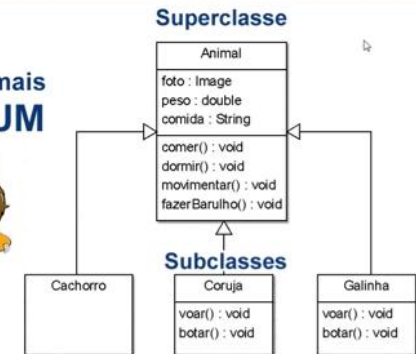
}
```


Herança



Herança

Todos eles são Animais
Relacionamento É-UM



```
public static void main(String[] args) {
```

```
    Cachorro toto = new Cachorro();  
    toto.comida = "Carne";  
    toto.dormir();
```

```
    Galinha cariyo = new Galinha();  
    cariyo.dormir();
```

Polimorfismo

Polimorfismo

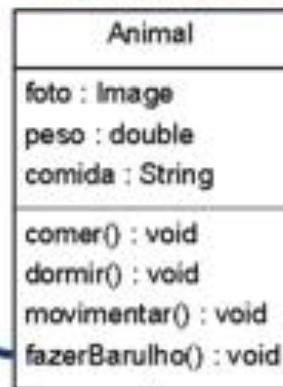
Polimorfismo

Comportamento Diferente

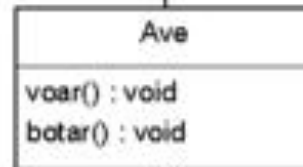
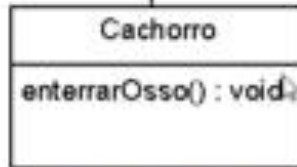
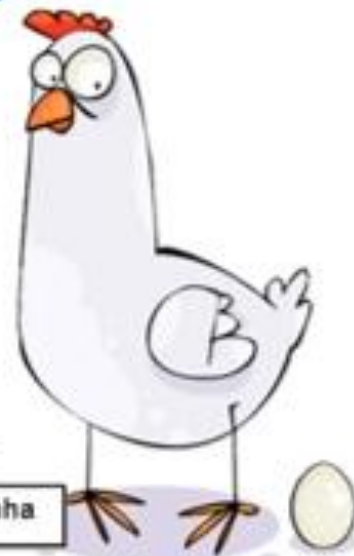
Au, Au !



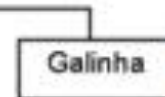
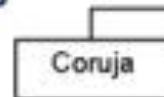
Superclasse



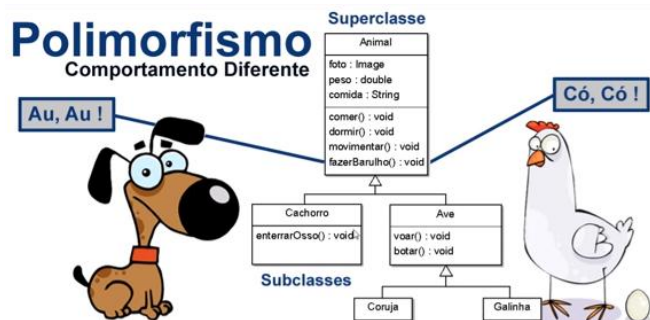
Có, Có !



Subclasses

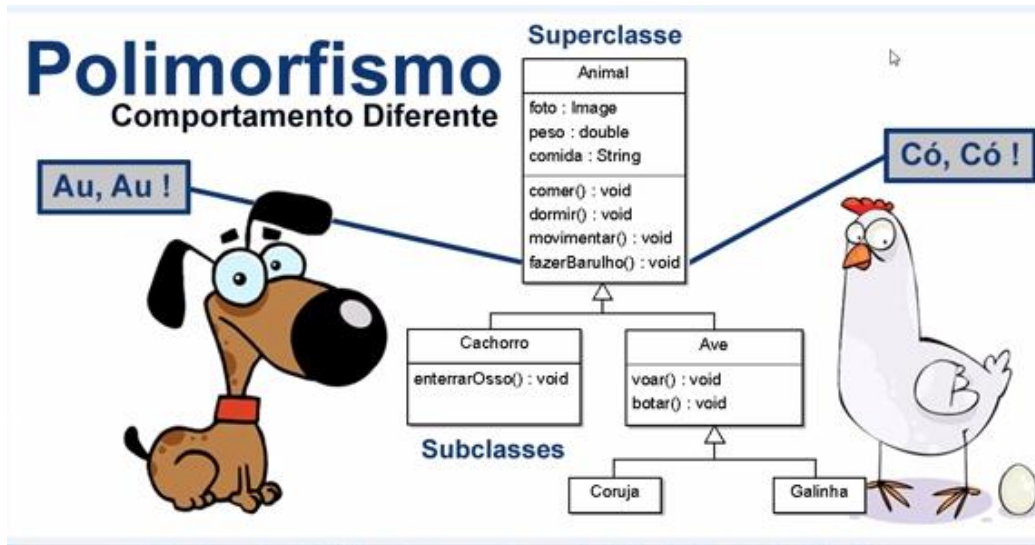


Polimorfismo



- ✓ **Polimorfismo**, que vem do grego "*muitas formas*".
- ✓ É o termo definido em linguagens orientadas a objeto - como o Java - para a possibilidade de se usar o mesmo elemento de forma diferente.
- ✓ Especificamente em Java, polimorfismo se encontra no fato de podemos modificar totalmente o código de um método herdado de uma classe diferente, ou seja, sobrescrevemos o método da classe pai.
- ✓ Portanto, polimorfismo está intimamente ligado a herança de classes.

Polimorfismo

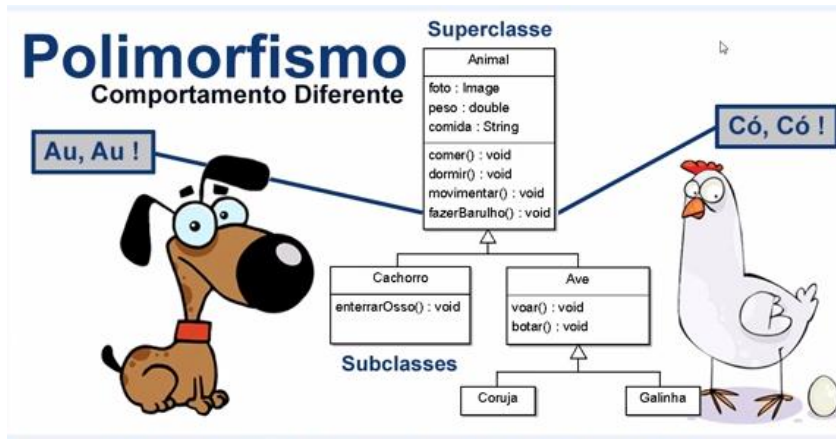


```
public class Animal {
```

```
    double peso;  
    String comida;
```

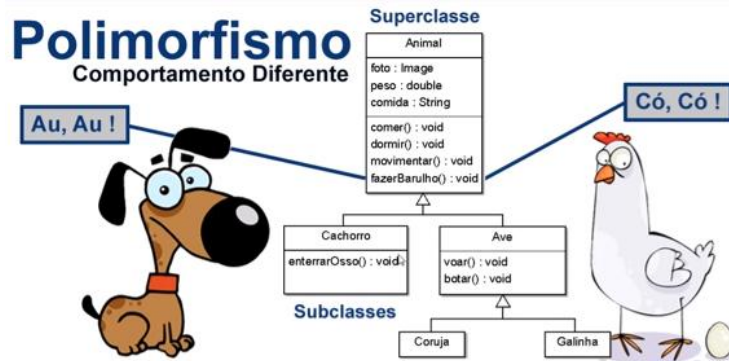
```
    void dormir(){System.out.println("Dormiu");}  
    void fazerBarulho(){System.out.println("Fazer Barulho")  
}
```

Polimorfismo



```
public class Animal {  
  
    double peso;  
    String comida;  
  
    public Animal(double peso, String comida) {  
        this.peso = peso;  
        this.comida = comida;  
    }  
  
    void dormir(){System.out.println("Dormiu");}  
    void fazerBarulho(){System.out.println("Fazer Barulho")  
}
```

Polimorfismo



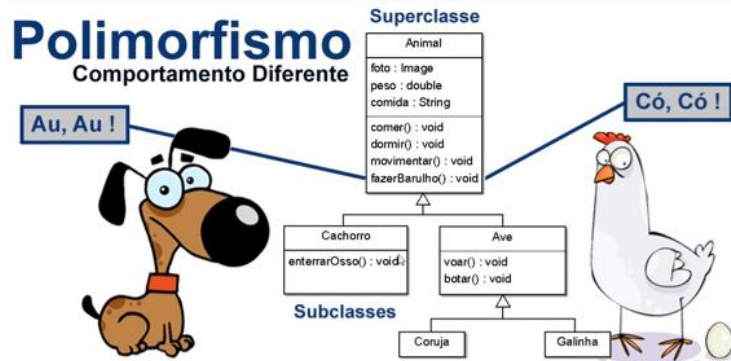
```
public class Cachorro extends Animal {
```

```
    public Cachorro() {
        super(30, "Carne");
    }
```



```
        void fazerBarulho() {
            System.out.println("Au, Au !");
        }
    }
```

Polimorfismo



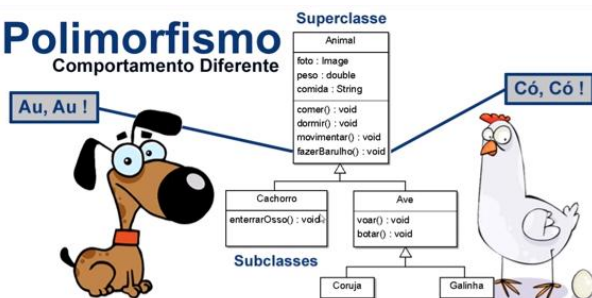
```
public class Galinha extends Animal {

    public Galinha() {
        super(2, "Milho");
    }

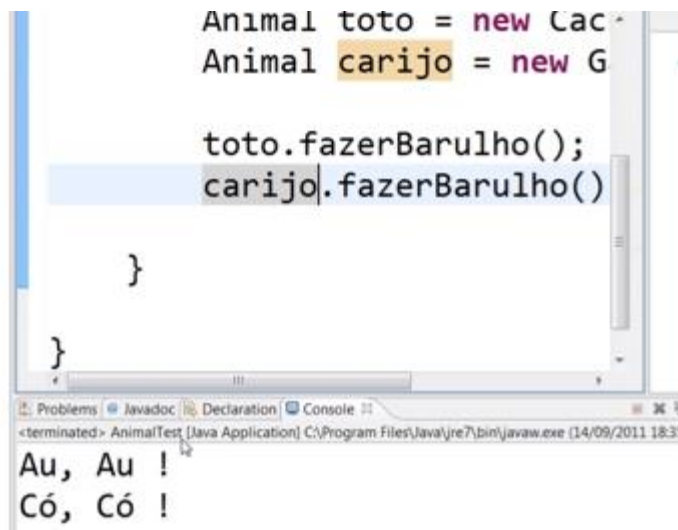
    void fazerBarulho() {
        System.out.println("Có, Có !");
    }
}
```


Polimorfismo

Polimorfismo Comportamento Diferente

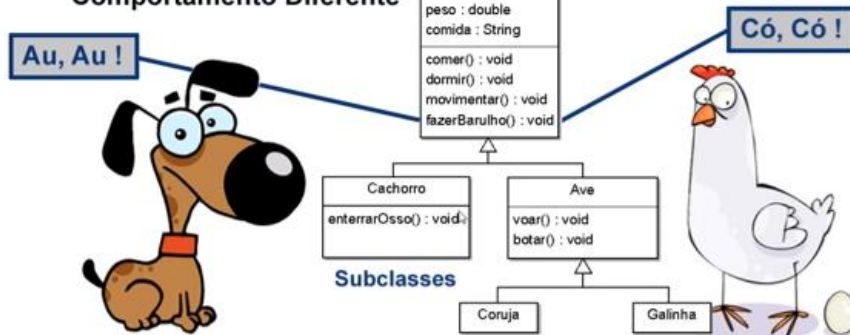


```
public class AnimalTest {  
  
    public static void main(String[] args) {  
  
        Animal toto = new Cachorro();  
        Animal carijo = new Galinha();  
  
        toto.fazerBarulho();  
        carijo.fazerBarulho()  
    }  
}
```



Outra Forma (Mais complexa) Sem Polimorfismo

Polimorfismo Comportamento Diferente



```
public static void barulho(String animal) {  
    if(animal.equals("Cachorro")) {  
        System.out.println("Au, Au !");  
    } else if (animal.equals("Galinha")) {  
        System.out.println("Có, Có !");  
    }  
}
```


Exercício

Exercício 20

Entre com a altura e peso e calcule o IMC através do método com retorno

Sabendo que:

1) Entre com seu peso, em kilos:

2) Entre com sua altura, em metros:

3) Método $IMC = \text{Peso} / \text{Altura} * \text{Altura}$

4) Método Mostrar a classificação

IMC - Classificação

< 19	Muito Magro
< 25	Normal
< 30	Sobrepeso
< 35	Obeso grau 1
< 40	Obeso grau 2
> 40	Obeso grau 3

Agenda

- **Encapsulamento**
- **Método**
- **Método com parâmetros e retorno**
- **Set, Get e Is**
- **Herança**
- **Polimorfismo**
- **Exercício**

Jose.wellington@uniceub.br