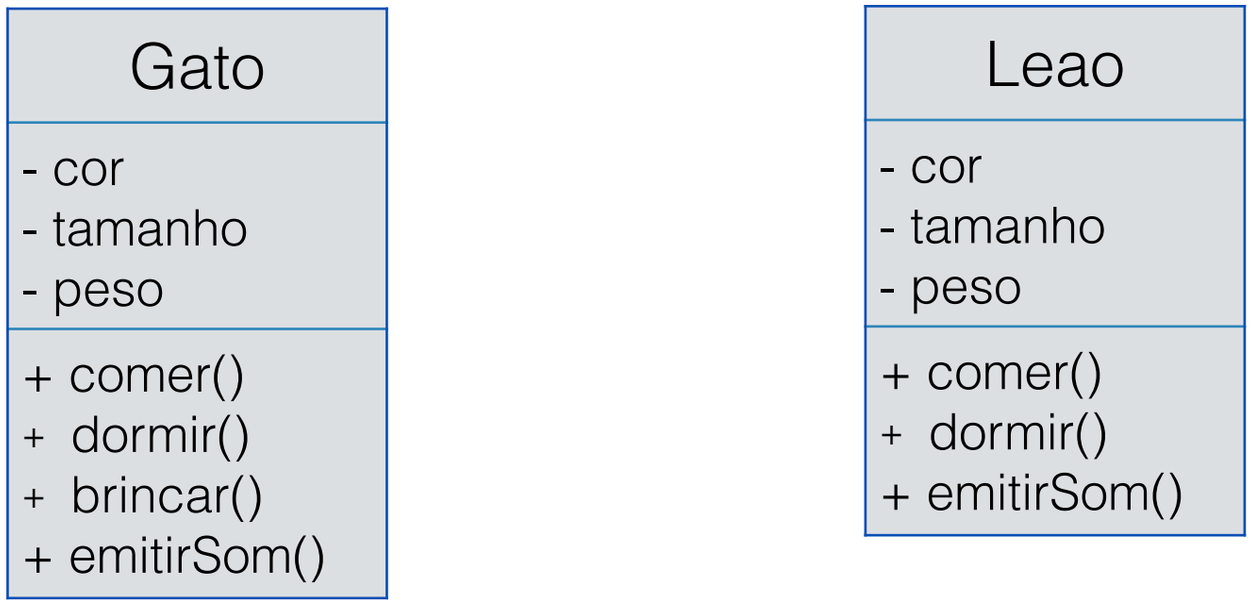

JAVA™

Encapsulamento
Encapsulando a classe Cachorro

```
class Cachorro{
    private int tamanho;

    public int getTamanho(){
        return tamanho;
    }

    public void setTamanho(int s){
        tamanho = s;
    }

    void latir(){
        if (tamanho > 60){
            System.out.println("Woof! Woof!");
        } else if (tamanho > 14){
            System.out.println("Ruff! Ruff!");
        } else {
            System.out.println("Yip! Yip!");
        }
    }
}
```

```
class TesteCachorro{
    public static void main(String[] args){
        Cachorro um = new Cachorro();
        um.setTamanho(70);
        Cachorro dois = new Cachorro();
        dois.setTamanho(8);
        System.out.println("Cachorro um: " + um.getTamanho());
        System.out.println("Cachorro dois: " + dois.getTamanho();
        um.latir();
        dois.latir();
    }
}
```

Exercício/Atividade



Expressões booleanas

Operadores "E" e "OU" (&&, ||)

```
if (preco >= 300 && preco < 400){
    camera = "x";
}
```

```
if (marca.equals("A") || marca.equals("B")){
    // executa algo somente para a marca A ou a marca B
}
```

```
if ((tipoZoom.equals("otico") &&
    (zoom >= 3 && zoom <= 8)) ||
    (tipoZoom.equals("digital") &&
    (zoom >= 5 && zoom <= 12))){
    // executa uma operação relacionada ao zoom
}
```

Expressões booleanas

Operador de exceção (!= e !)

```
if (modelo != 2000){
    // executa algo que não é aplicável ao modelo 2000
}
```

```
if (!marca.equals("X")){
    // executa algo que não é aplicável a marca X
}
```

As matrizes também são objetos

Quando você tiver declarado uma matriz, não poderá inserir nada que não seja do tipo declarado para ela.

Todo elemento de uma matriz é apenas uma variável. Em outras palavras, um dos oito tipos primitivos de variável ou uma variável de referência. Portanto, em uma matriz de tipo int (int[]), cada elemento pode conter um inteiro. Em uma matriz do tipo Cachorro (Cachorro[]) cada elemento pode conter... Um objeto Cachorro? Não, lembre-se de que uma variável de referência só armazena um referência e não o próprio objeto. E é claro que ainda teremos de criar os objetos Cachorro.

As matrizes são sempre objetos, não importando se foram declaradas para conter tipos primitivos ou referencias de objetos.

Matrizes

1 - Declare uma variável do tipo matriz int

```
int[] nums;
```

2 - Crie uma nova matriz int de tamanho 7 e a atribua a variável int[] nums já declarada

```
nums = new int[7];
```

3 - Forneça para cada elemento da matriz um valor int.

```
nums[0] = 6;
nums[1] = 19;
nums[2] = 44;
nums[3] = 42;
nums[4] = 10;
nums[5] = 20;
nums[6] = 1;
```

Criando uma matriz de objetos

Declare uma variável de matriz Cachorro

Cachorro[] filhotes;

Crie uma nova matriz Cachorro com tamanho igual a 7 e a atribua a variável Cachorro[] filhotes já declarada

filhotes = new Cachorro[7];

O que está faltando?

Objetos Cachorro! Temos uma matriz de referencias Cachorro, mas nenhum objeto Cachorro real.

Criando uma matriz de objetos

Crie novos objetos Cachorro e atribua-os aos elementos da matriz

```
filhotes[0] = new Cachorro();
filhotes[1] = new Cachorro();
```

Criando uma matriz de objetos

E como acessar os objetos Cachorro dentro da matriz?

Usaremos a notação da matriz:

```
filhotes[1].nome = "bidu";
filhotes[1].latir();
```

Loops ***for***
loop for aperfeiçoado

A partir do Java 5.0 a linguagem passou a ter um segundo tipo de loop for chamado for aperfeiçoado, que torna mais fácil a iteração por todos os elementos de uma matriz.

Declara uma variável de iteração que armazenará apenas um elemento da matriz

O sinal de dois-pontos (:) significa "DE"

O código a ser repetido entra aqui (o corpo)

for (String nome: nomeMatriz){ }

Os elementos da matriz DEVEM ser compatíveis com o tipo declarado para a variável

A cada iteração um elemento diferente da matriz será atribuído atribuído a variável "nome"

O conjunto de elementos que você deseja percorrer.

Exercícios

Cada um dos arquivos Java abaixo representa um arquivo-fonte completo. Sua tarefa é personificar o compilador e determinar se cada um deles pode ser compilado. Se não puderem ser compilados, como você os corrigiria?

```
class Livros{
    String titulo;
    String autor;
}

class LivrosTeste{
    public static void main(String[] args){
        Livros[] meusLivros = new Livros[3];

        int x = 0;

        meusLivros[0].titulo = "Java forever";
        meusLivros[1].titulo = "Java em orbita";
        meusLivros[2].titulo = "Cozinhe com Java";
        meusLivros[0].autor = "bob";
        meusLivros[1].autor = "sue";
        meusLivros[2].autor = "ian";

        while (x < 3){
            System.out.print(meusLivros[x].titulo);
            System.out.print(" de ");
            System.out.println(meusLivros[x].autor);
            x = x +1;
        }
    }
}
```

```
class Hobbits{
    String nome;

    public static void main(String[] args){
        Hobbits[] h = new Hobbits[3];

        int z = 0;

        while (z < 4){
            z = z +1;

            h[z] = new Hobbits();
            h[z].nome = "Bilbo";

            if (z == 1){
                h[z].nome = "Frodo";
            }
            if (z == 2){
                h[z].nome = "Sam";
            }

            System.out.print(h[z].nome + " é um ");
            System.out.println("bom nome para um Hobbit");
        }
    }
}
```

Exercícios

Um grupo de componentes Java está participando do jogo "Quem sou eu?" Eles lhe darão uma pista e você tentara adivinhar quem são, baseado no que disserem. Suponha que eles sempre digam a verdade quando falam de si mesmos. Se por acaso disserem algo que possa ser verdadeiro para mais de um deles, anote todos aos quais a frase possa ser aplicada. Preencha as linhas em branco próximas a frase com os nomes de um ou mais candidatos.

Candidatos: variável de instância, argumento, retorno, método de captura, método de configuração, encapsulamento, public,
private, passar por valor, método

Uma classe pode ter quantos quiser. _____
O método só pode ter um. _____
Prefiro minhas variáveis de instancia privadas. _____
Só os métodos de configuração devem atualizá-los. _____
Um método pode ter muitos deles. _____
Retorno algo por definição. _____
Não devo ser usado com variáveis de instancia. _____
Posso ter muitos argumentos. _____
Por definição, uso um argumento. _____
Ajudam a criar o encapsulamento. _____
Estou sempre sozinho. _____