INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARAÍBA
Campus João Pessoa

Programação Orientada a Objetos

Fausto Maranhão Ayres

8
Listas polimórficas

Listas polimórficas

- Em Java, todas as coleções permitem armazenar objetos de diferentes tipos de uma hierarquia

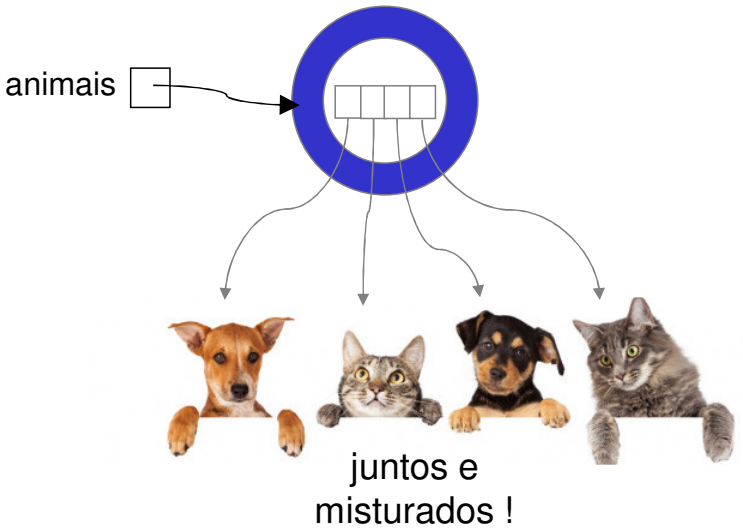
Exemplos:

```
//uma lista com diferentes tipos de animal
ArrayList<Animal> animais = new ArrayList<>();


//um array com diferentes tipos de animal
Animal[] animais = Animal[100];
```

Teste

```
ArrayList<Animal> animais = new ArrayList<>();
animais.add( new Cachorro("rex", 7)        );
animais.add( new Gato("fifi",3, 5)         );
animais.add( new Cachorro("lessy", 4)      );
animais.add( new Gato("nino", 6, 2)        );
```

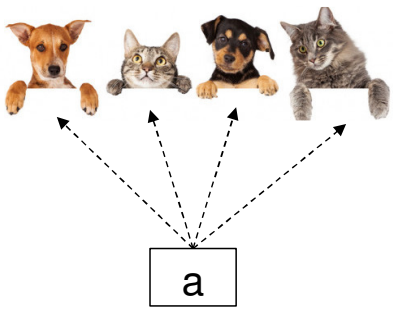


Polimorfismo de método e de variável

```
//exibir todos os animais da lista

for (Animal a : animais){
  System.out.println( a );
}
```

```
nome=rex, peso=7.0, som= au au
nome=fifi, peso=3.0, som= miau!
nome=lessy, peso=4.0, som= au au
nome=nino, peso=6.0, som= miau!
```



Variável polimórfica→ a

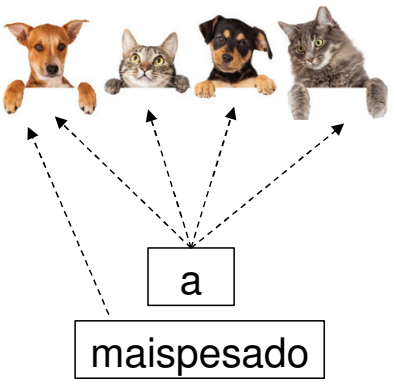
Método polimórfico→ emitirSom()

Observe que toString() de Animal é chamado e que, por sua vez, chama o emitirSom() de cada tipo de objeto

Polimorfismo de variável

```
//exibir o animal mais pesado

double maxpeso = 0;
Animal maispesado = null;
for (Animal a : animais)
  if(a.getPeso() >= maxpeso){
    maxpeso = a.getPeso();
    maispesado = a;
  }
System.out.println( maispesado );
nome=rex, peso=7.0, som= au au
```

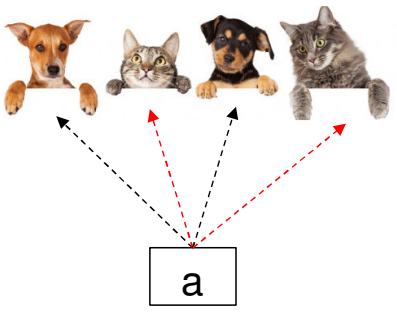


A variável “Animal maispesado” é polimórfica, pois armazena um objeto *Gato* ou *Cachorro*

Operador instanceof

```
//Contar os gatos

int cont=0;
for (Animal a : animais){
  if(a instanceof Gato)
      cont++;
}
System.out.println( cont );   //2
```



O operador *instanceof* compara o **tipo do objeto** contido em “a“ com o **tipo Gato**

Casting de objeto

(Casting) de objeto

- A operação de casting é usada para substituir o tipo da variável pelo tipo de objeto que ela irá conter

Teste:

```
Animal a1 = new Cachorro("rex", 5);
Cachorro c1 = (Cachorro) a1;      //ok
```

o tipo da variável (**Animal)** é substituído pelo tipo do objeto (**Cachorro)** para validar a atribuição

Exceção de casting

- O casting produz uma exceção se o tipo do objeto encontrado na **execução** for diferente do tipo do objeto especificado na **compilação**

```
Animal a1 = new Cachorro("rex", 5);
Gato g1 = (Gato) a1;    //compila
```

o tipo da variável (**Animal)** é substituído pelo tipo do objeto (**Gato)** para validar a atribuição

Mas lança **ClassCastException**, pois o tipo do objeto encontrado na execução foi Cachorro e não o tipo Gato aceito na compilação

Casting explícito x implícito

Ex: criar uma lista de gatos

```
ArrayList<Gato> gatos = new ArrayList<>();
for (Animal a : animais){
  if(a instanceof Gato)
      gatos.add( (Gato)a );    //casting explicito
}
System.out.println( gatos );  // [fifi,nino]
```

```
ArrayList<Gato> gatos = new ArrayList<>();
for (Animal a : animais){
  if(a instanceof Gato g)      //casting implicito
      gatos.add(g);
}
System.out.println( gatos );  // [fifi,nino]
```

O *instanceof* permite o **casting implícito** para uma variável temporária associada a ele (java16)

Casting explícito x implícito

Ex: criar uma lista de gatos saltadores

```
Arraylist<Gato> saltadores  new ArrayList<>();
for (Animal a : animais){
  if(a instanceof Gato && ((Gato)a).getSalto()>= 5)
      saltadores.add( (Gato)a);
}
System.out.println( saltadores );  // [fifi]
```

```
Arraylist<Gato> saltadores  new ArrayList<>();
for (Animal a : animais){
  if(a instanceof Gato g  && g.getSalto()>= 5)
      saltadores.add(g);
}
System.out.println( saltadores );  // [fifi]
```