



Alunas:

Flávia Avelino dos Santos e Priscila Maria Franca Da Silva

Um pouco da história

- ❖ Criada em 2003 a partir da publicação de um artigo de James Strachan publicou em seu blog
- ❖ Em janeiro de 2007 houve sua primeira versão 1.0. Em 2 de julho de 2012, o Groovy 2.0 foi lançado apresentando novos recursos tais como a adição de compilação estática e inferência de tipos
- ❖ Até hoje tem a versão 2.4 como o principal lançamento.

Principais características

- Linguagem Orientada a Objeto
- Suporte para tipagem estática e dinâmica
- Possui recursos inspirados em Linguagens como Ruby, SmallTalk e Python.
- Sintaxe similar ao do Java
- Interpretada ou compilada para bytecode.
- Fornece várias simplificações comparadas ao padrão da linguagem Java, além de recursos avançados como *closures* e suporte nativo a *listas* e *mapas*.
- MetaProgramação

Groovy vs Java

- Os tipos primitivos do Java são convertidos para suas respectivas classes encapsuladoras de forma transparente para o programador.
- Simplicidade nas expressões

```
if (stringExemplo != null && !stringExemplo.isEmpty()) {...} //código Java
if (stringExemplo()) {...} //código Groovy
```
- Importações default. Os seguintes pacotes e classes são importados por default, sendo desnecessário usar *import* explícito para utilizá-los:
- Na linguagem Java você é obrigado a digitar *private* para declarar atributos privados. No Groovy todos os atributos de uma classe são por padrão *private*.
- Na linguagem Java você é obrigado a digitar *public* para declarar a classe pública. No Groovy não, pois todas as classes são por padrão pública
- Na linguagem Java você é obrigado a digitar os *get's* e *set's* para expor os atributos. No Groovy não, eles serão automaticamente e dinamicamente gerados para você.

O Groovy integrado ao Java

- Groovy é complementar ao Java
- Sua integração ocorre dentro da Virtual Machine
- Faz uso das bibliotecas Java



JAVA

Arquivo Editar Localizar Visualizar Formatar Linguagem Configurações
Janela ?



conhecer.java x

```
1 public class Greeter {  
2     private String owner;  
3  
4     public String getOwner(){  
5         return owner;  
6     }  
7  
8  
9     public void setOwner(String owner){  
10        this.owner = owner;  
11    }  
12  
13    public String greet(String name){  
14        return "Ola" +name+ ", Eu sou" +owner;  
15    }  
16 }  
17 Greeter greeter = new Greeter();  
18 greeter.setOwner("Juliano");  
19  
20 System.out.println(greeter.greet("Paulo"));
```

length: 370 line Ln: 20 Col: 44 Sel: 0 | 0

Dos\Windows

UTF-8

Groovy

G GroovyConsole

File Edit View History Script Help



```
1 class Greeter{  
2     String owner  
3     String greet(String name){  
4         "Ola ${name}, Me chamo ${owner}"  
5     }  
6 }  
7  
8 def greeter = new Greeter(owner: "Maria")  
9  
10 println greeter.greet("Pedro")
```

```
groovy> class Greeter{  
groovy>     String owner  
groovy>     String greet(String name){  
groovy>         "Ola ${name}, Me chamo ${owner}"  
groovy>     }  
groovy> }  
groovy> def greeter = new Greeter(owner: "Maria")  
groovy> println greeter.greet("Pedro")
```

Ola Pedro, Me chamo Maria

Execution complete. Result was null.

4:13

Groovy Protótipos

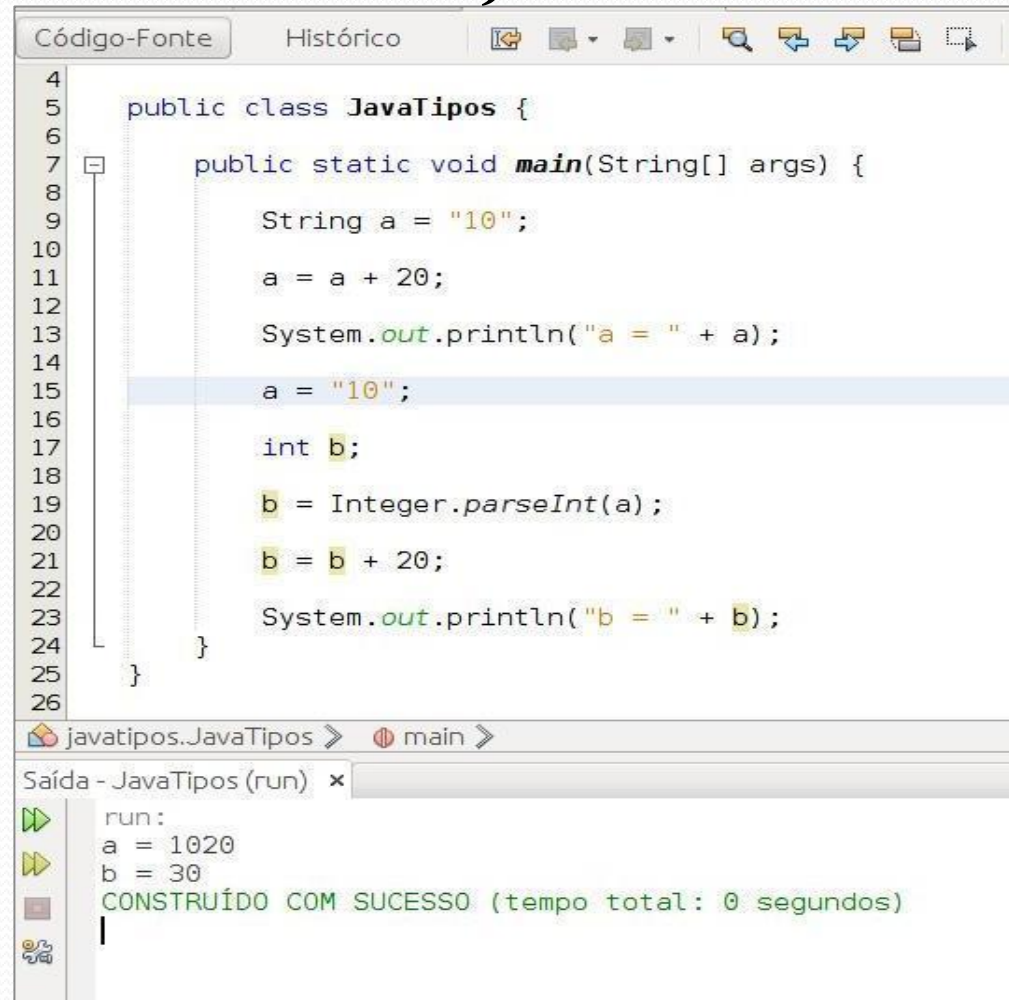
- É uma forma de programação orientada a objeto onde novos objetos são criados a partir de um protótipo, e eles podem ter sua estrutura alterada em tempo de execução.
- Forma de programação em que o comportamento de um novo objeto é realizado através de um processo de expansão do comportamento de objetos já existentes

Conversão de Tipo

Groovy

```
linux@Computer: ~  
linux@Computer:~$ groovysh  
Groovy Shell (2.4.12, JVM: 1.8.0_131)  
Type ':help' or ':h' for help.  
-----  
groovy:000> a = "10"  
==> 10  
groovy:000> a + 20  
==> 1020  
groovy:000> a = 10  
==> 10  
groovy:000> a + 20  
==> 30  
groovy:000> 
```

Java



```
Código-Fonte  Histórico  
4  
5 public class JavaTipos {  
6  
7     public static void main(String[] args) {  
8  
9         String a = "10";  
10  
11         a = a + 20;  
12  
13         System.out.println("a = " + a);  
14  
15         a = "10";  
16  
17         int b;  
18  
19         b = Integer.parseInt(a);  
20  
21         b = b + 20;  
22  
23         System.out.println("b = " + b);  
24     }  
25 }  
26  
javatipos.JavaTipos > main >  
Saída - JavaTipos (run) x  
run:  
a = 1020  
b = 30  
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```


Conversão de Tipo

Groovy

```
linux@Computer: ~  
linux@Computer:~$ groovysh  
Groovy Shell (2.4.12, JVM: 1.8.0_131)  
Type ':help' or ':h' for help.  
-----  
groovy:000> a = 10  
==> 10  
groovy:000> a  
==> 10  
groovy:000> a = "Jane"  
==> Jane  
groovy:000> a  
==> Jane  
groovy:000> a = 20  
==> 20  
groovy:000> a  
==> 20  
groovy:000> 
```

Java

```
1  
2  
3 package javatipos;  
4  
5 public class JavaTipos {  
6  
7     public static void main(String[] args) {  
8  
9         int a = 10;  
10  
11         System.out.println("a = " + a );  
12  
13         String b;  
14  
15         b = "Jane";  
16  
17         System.out.println("b = " + b );  
18     }  
19 }  
20
```

Saída - JavaTipos (run) x

run:
a = 10
b = Jane
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

Metaclasse

- MOP - Meta-Object Protocol
- “Mágica da programação”;
- Adiciona comportamento às classes em tempo de execução;
- Adicionar métodos às classes ou a apenas 1 objeto específico;
- Adiciona propriedades aos objetos já existentes;
- Todas as classes escritas em Groovy herdam de Object e implementam a interface GroovyObject;

Groovy

Metaclass

Resultado

```
GroovyConsole
File Edit View History Script Help

1 class Pessoa {
2   String nome
3   int idade
4
5   Pessoa(String nome, int idade){
6     this.nome = nome;
7     this.idade = idade;
8   }
9 }
10
11 Pessoa.metaClass.telefone = 00000000
12
13 def p1 = new Pessoa("Ana", 22)
14
15 assert p1.telefone == 00000000
16
17 Pessoa.metaClass.Mudar_Telefone(){int num -> delegate.telefone = num}
18
19 p1.Mudar_Telefone(25709494)
20
21
22 def p2 = new Pessoa("Paulo", 30)
23
24 assert p2.telefone == 00000000
25
26 p1.metaClass.pais = "Brasil"
27 p1.metaClass.Meu_Pais(){-> println "Eu sou do pais: $delegate.pais"}
28 p1.metaClass.idioma = "Portugues"
29
30
31 println "Objeto p1 Caracteristicas"
32 println "Nome = " + p1.nome
33 println "Idade = " + p1.idade
34 println "Telefone = " + p1.telefone
35 println "Idioma = " + p1.idioma
36 p1.Meu_Pais();
37 println " "
38
39 println "Objeto p2 Caracteristicas"
40 println "Nome = " + p2.nome
41 println "Idade = " + p2.idade
42 println "p2 telefone = " + p2.telefone
43 println "Idioma = " + p2.idioma
44
```

```
Objeto p1 Caracteristicas
Nome = Ana
Idade = 22
Telefone = 25709494
Idioma = Portugues
Eu sou do pais: Brasil
```

```
Objeto p2 Caracteristicas
Nome = Paulo
Idade = 30
p2 telefone = 0
Exception thrown
```

```
groovy.lang.MissingPropertyException: No such property: idioma for class: Pessoa
    at ConsoleScript38.run(ConsoleScript38:46)
```

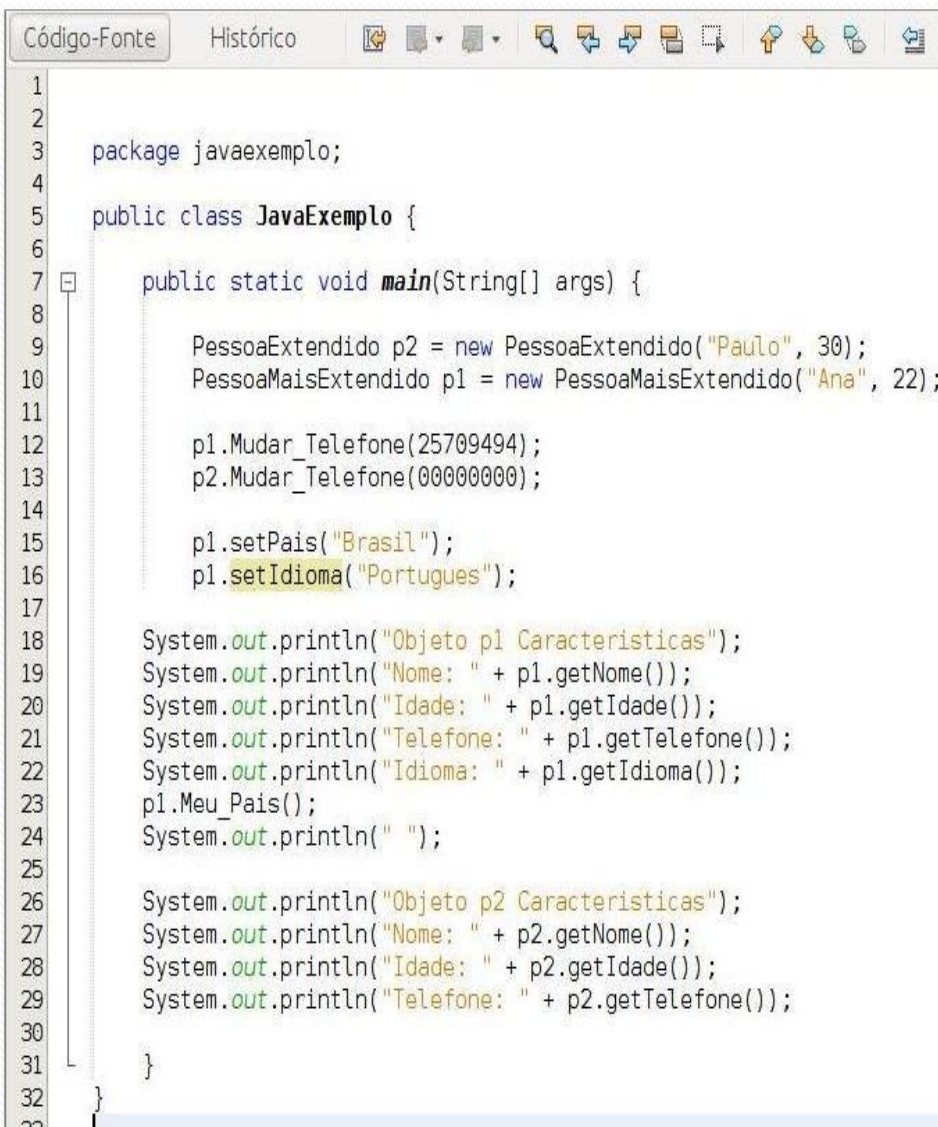
Execution terminated with exception.

Java

```
Código-Fonte  Histórico
1
2 package javaexemplo;
3
4 public class Pessoa {
5
6     private String nome;
7     private int idade;
8
9
10 Pessoa(String nome, int idade){
11     this.nome = nome;
12     this.idade = idade;
13 }
14
15 public String getNome() { return this.nome;}
16
17 public int getIdade() { return this.idade;}
18
19 }
20
21
```

```
Código-Fonte  Histórico
1
2 package javaexemplo;
3
4 public class PessoaExtendido extends Pessoa{
5
6     private int telefone;
7
8     PessoaExtendido(String nome, int idade){
9         super (nome, idade);
10    }
11
12    void Mudar_Telefone(int num ){
13        this.telefone = num;
14    }
15
16    public int getTelefone() {return this.telefone;}
17
18    @Override
19    public String getNome() {
20        return super.getNome();
21    }
22
23    @Override
24    public int getIdade() {
25        return super.getIdade();
26    }
27
28 }
```

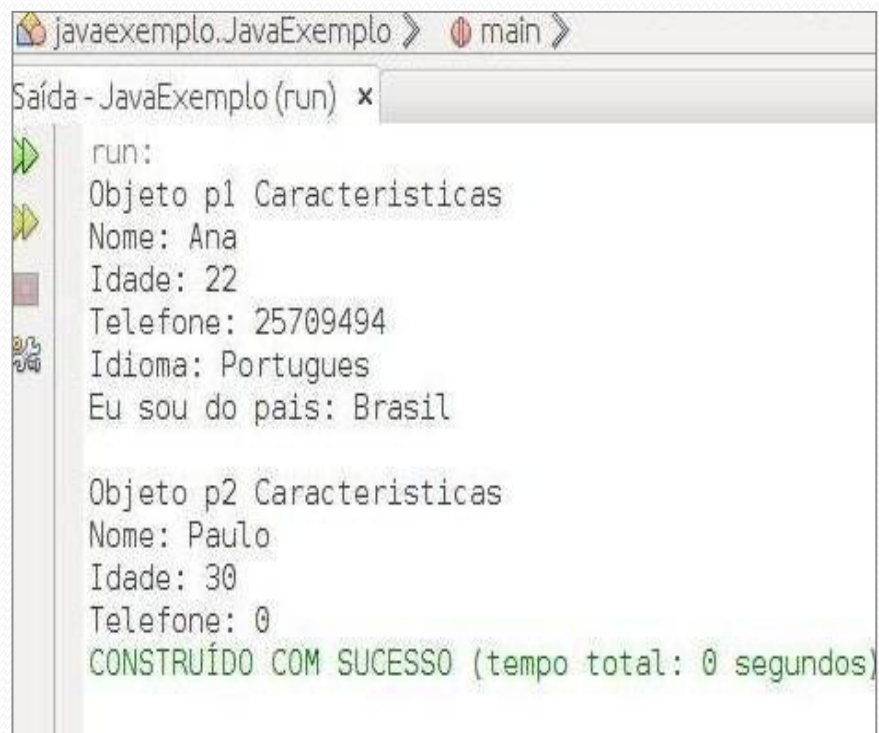
```
Código-Fonte  Histórico
5 public class PessoaMaisExtendido extends PessoaExtendido{
6
7     private String pais, idioma;
8
9     PessoaMaisExtendido(String nome, int idade){
10         super (nome, idade);
11     }
12
13     public void setPais(String pais) { this.pais = pais;}
14
15     void Meu_Pais(){System.out.println("Eu sou do pais: " + this.pais);}
16
17     public void setIdioma(String idioma) { this.idioma = idioma;}
18
19     public String getIdioma() {return this.idioma;}
20
21     @Override
22     public String getNome() {
23         return super.getNome();
24     }
25
26     @Override
27     public int getIdade() {
28         return super.getIdade();
29     }
30
31     @Override
32     public int getTelefone() {
33         return super.getTelefone();
34     }
35
36     void Mudar_Telefone(int num ){
37         super.Mudar_Telefone(num);
38     }
39 }
```

The screenshot shows an IDE window titled 'Código-Fonte' with a toolbar. The code is as follows:

```
1  
2  
3 package javaexemplo;  
4  
5 public class JavaExemplo {  
6  
7     public static void main(String[] args) {  
8  
9         PessoaExtendido p2 = new PessoaExtendido("Paulo", 30);  
10        PessoaMaisExtendido p1 = new PessoaMaisExtendido("Ana", 22);  
11  
12        p1.Mudar_Telefone(25709494);  
13        p2.Mudar_Telefone(00000000);  
14  
15        p1.setPais("Brasil");  
16        p1.setIdioma("Portugues");  
17  
18        System.out.println("Objeto p1 Caracteristicas");  
19        System.out.println("Nome: " + p1.getNome());  
20        System.out.println("Idade: " + p1.getIdade());  
21        System.out.println("Telefone: " + p1.getTelefone());  
22        System.out.println("Idioma: " + p1.getIdioma());  
23        p1.Meu_Pais();  
24        System.out.println(" ");  
25  
26        System.out.println("Objeto p2 Caracteristicas");  
27        System.out.println("Nome: " + p2.getNome());  
28        System.out.println("Idade: " + p2.getIdade());  
29        System.out.println("Telefone: " + p2.getTelefone());  
30  
31    }  
32 }
```

Resultado



The screenshot shows an IDE window titled 'javaexemplo.JavaExemplo' with a toolbar. The output is as follows:

```
run:  
Objeto p1 Caracteristicas  
Nome: Ana  
Idade: 22  
Telefone: 25709494  
Idioma: Portugues  
Eu sou do pais: Brasil  
  
Objeto p2 Caracteristicas  
Nome: Paulo  
Idade: 30  
Telefone: 0  
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Metaclasses

Resultado

```
GroovyConsole
File Edit View History Script Help

1 class Animal {
2     String nome
3     String tipo
4     int idade
5
6     Animal(String nome, int idade, String tipo){
7         this.nome = nome;
8         this.idade = idade;
9         this.tipo = tipo;
10    }
11 }
12
13 Animal.metaClass.raca = "";
14
15 def cat = new Animal("Felix", 2, "Gato")
16 def dog = new Animal("Bob", 1, "Cachorro")
17
18 cat.raca = "siames"
19 dog.raca = "vira-lata"
20
21 cat.metaClass.miar(){-> println "miau miau"}
22
23 dog.metaClass.latir(){-> println "au au"}
24
25 println "Caracteristicas Gato"
26 println "Nome: " + cat.nome
27 println "Idade: " + cat.idade
28 println "Tipo: " + cat.tipo
29 println "Raca: " + cat.raca
30 cat.miar()
31
32 println " "
33 println "Caracteristicas Cachorro"
34 println "Nome: " + dog.nome
35 println "Idade: " + dog.idade
36 println "Tipo: " + dog.tipo
37 println "Raca: " + dog.raca
38 dog.latir()
39
40 println " "
41 assert cat.miar() == "au au"
42
```

```
Caracteristicas Gato
Nome: Felix
Idade: 2
Tipo: Gato
Raca: siames
miau miau
```

```
Caracteristicas Cachorro
Nome: Bob
Idade: 1
Tipo: Cachorro
Raca: vira-lata
au au
```

```
miau miau
Exception thrown
```

Assertion failed:

```
assert cat.miar() == "au au"
      |   |       |
      |   null    false
      Animal@7cf9bf0c
```

at ConsoleScript0.run(ConsoleScript0:42)

Execution terminated with exception.

Java

```
1
2 package javaclasses;
3
4 public class Animal {
5
6     private String nome;
7     private String tipo;
8     private int idade;
9
10
11     Animal (String nome, int idade, String tipo){
12         this.nome = nome;
13         this.idade = idade;
14         this.tipo = tipo;
15     }
16
17     public String getNome() { return this.nome;}
18
19     public int getIdade() { return this.idade;}
20
21     public String getTipo() { return this.tipo;}
22
23 }
24
25
```

```
2 package javaclasses;
3
4 public class Gato extends Animal{
5
6     private String raca;
7
8     Gato (String nome, int idade, String tipo){
9         super(nome, idade, tipo);
10     }
11
12     void miar(){ System.out.println("miau miau");
13     }
14
15     @Override
16     public String getNome() {
17         return super.getNome();
18     }
19
20     @Override
21     public int getIdade() {
22         return super.getIdade();
23     }
24
25     @Override
26     public String getTipo() {
27         return super.getTipo();
28     }
29
30     public String getRaca() { return this.raca; }
31
32     public void setRaca(String raca) { this.raca = raca;}
33 }
```

```
2 package javaclasses;
3
4 public class Cachorro extends Animal{
5
6     private String raca;
7
8     Cachorro (String nome, int idade, String tipo){
9         super(nome, idade, tipo);
10     }
11
12     void latir(){System.out.println("au au");}
13
14     @Override
15     public String getNome() {
16         return super.getNome();
17     }
18
19     @Override
20     public int getIdade() {
21         return super.getIdade();
22     }
23
24     @Override
25     public String getTipo() {
26         return super.getTipo();
27     }
28
29     public String getRaca() { return this.raca;}
30
31     public void setRaca(String raca) { this.raca = raca;}
32
33 }
```


Java

```
Código-Fonte  Histórico  [ícone] [ícone] [ícone] [ícone] [ícone] [ícone] [ícone] [ícone] [ícone] [ícone]

1
2 package javaclasses;
3
4 public class JavaClasses {
5
6     public static void main(String[] args) {
7
8         Gato cat = new Gato("Felix", 2, "Gato");
9         Cachorro dog = new Cachorro("Bob", 1, "Cachorro");
10
11         cat.setRaca("siames" );
12         dog.setRaca("vira-lata");
13
14
15         System.out.println("Características Gato");
16         System.out.println("Nome: " + cat.getNome());
17         System.out.println("Idade: " + cat.getIdade());
18         System.out.println("Tipo: " + cat.getTipo());
19         System.out.println("Raca: " + cat.getRaca());
20         cat.miar();
21
22         System.out.println(" ");
23
24         System.out.println("Características Cachorro");
25         System.out.println("Nome: " + dog.getNome());
26         System.out.println("Idade: " + dog.getIdade());
27         System.out.println("Tipo: " + dog.getTipo());
28         System.out.println("Raca: " + dog.getRaca());
29         dog.latir();
30     }
31 }
32
```

Resultado

```
javaclasses.JavaClasses > main >

Saída - JavaClasses(run) x

run:
Características Gato
Nome: Felix
Idade: 2
Tipo: Gato
Raca: siames
miau miau

Características Cachorro
Nome: Bob
Idade: 1
Tipo: Cachorro
Raca: vira-lata
au au
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```



Obrigada!