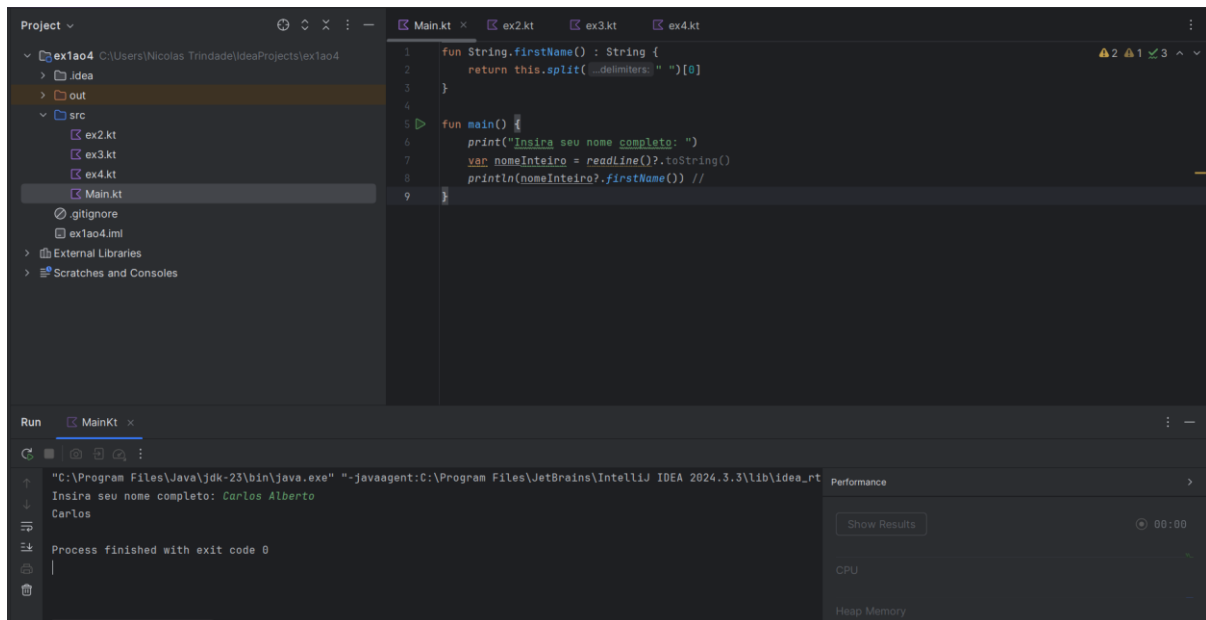


Nicolas Trindade – 2DS PAM- Exercicios Kotlin lista3/4

Exercício 01 - Criar uma função de extensão da classe String com o seguinte nome “firstName(): String” que irá retornar uma String com o primeiro nome de uma pessoa.

Resposta:



```
1 fun String.firstName() : String {
2     return this.split(" ")[0]
3 }
4
5 fun main() {
6     print("Insira seu nome completo: ")
7     var nomeInteiro = readLine()?.toString()
8     println(nomeInteiro?.firstName()) //
9 }
```

Run MainKt

"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt..."

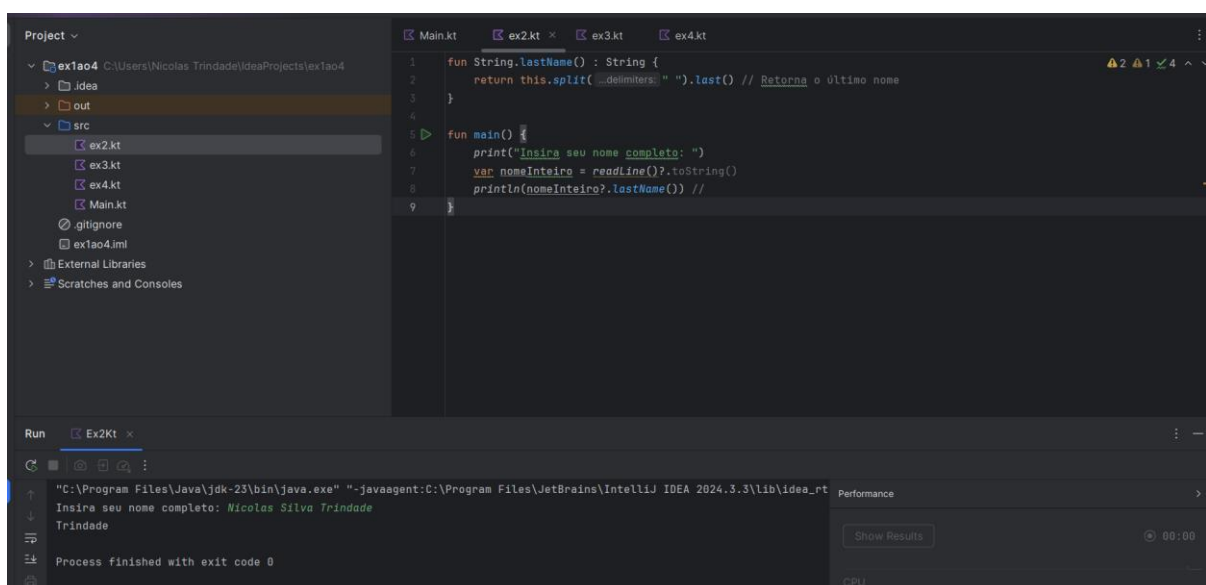
Insira seu nome completo: Carlos Alberto

Carlos

Process finished with exit code 0

Exercício 02 - Criar uma função de extensão da classe String com o seguinte nome “lastName(): String” que irá retornar uma String com o último nome de uma pessoa.

Resposta:



```
1 fun String.lastName() : String {
2     return this.split(" ").last() // Retorna o último nome
3 }
4
5 fun main() {
6     print("Insira seu nome completo: ")
7     var nomeInteiro = readLine()?.toString()
8     println(nomeInteiro?.lastName()) //
9 }
```

Run Ex2Kt

"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt..."

Insira seu nome completo: Nicolas Silva Trindade

Trindade

Process finished with exit code 0

Exercício 03 - Crie uma classe temperatura, com base em uma temperatura em graus Celsius, a converta e exiba em Kelvin (K) e Fahrenheit (F), seguindo as fórmulas: $F = C * 1.8 + 32$; $K = C + 273,15$; Resposta:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt
Insira a temperatura em Celsius: 35.00
Valor da temperatura em F: 95.0
Valor da temperatura em K: 308.15

Process finished with exit code 0
```

```
// Classe aberta que permite herança
open class Temperatura {

    // Variável para armazenar a temperatura em Celsius
    var C: Double = 0.0

    // Função para ler a temperatura em Celsius
    fun Celsius(): Double {
        print("Insira a temperatura em Celsius: ")
        C = readLine()?.toDoubleOrNull() ?: 0.0 // Armazena o valor digitado ou 0.0 se for inválido
        return C // Corrigido: retorna o valor lido
    }

    // Função para converter Celsius para Fahrenheit
    fun tempf(): Double {
        val calcf = C * 1.8 + 32 // Corrigido: usa a variável de instância C
        println("Valor da temperatura em F: $calcf") // Corrigido: era "k", agora é "F"
        return calcf // Corrigido: retorna o valor calculado
    }
}
```

```
// Função para converter Celsius para Kelvin
fun tempk(): Double {
    val calck = C + 273.15 // Corrigido: usa a variável de instância C
    println("Valor da temperatura em K: $calck")
    return calck // Corrigido: retorna o valor calculado
}

// Função principal
fun main(args: Array<String>) {
    val temp = Temperatura()
    temp.Celsius() // Chama a função para ler a temperatura
    temp.tempf()   // Converte e imprime Fahrenheit
    temp.tempk()   // Converte e imprime Kelvin
}
```

Exercício 04 - Tem-se um conjunto de dados contendo a altura e o sexo (masculino, feminino) de 10 pessoas. Criar uma classe para calcular e escrever:

- a) a maior e a menor altura do grupo;
- b) média de altura dos homens;
- c) o número de mulheres.

Resposta:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt
Insira a altura da pessoa 1 (em metros): 1.75
Insira o sexo da pessoa 1 (M para masculino, F para feminino): m
Insira a altura da pessoa 2 (em metros): 1.70
Insira o sexo da pessoa 2 (M para masculino, F para feminino): f
Insira a altura da pessoa 3 (em metros): 2
Insira o sexo da pessoa 3 (M para masculino, F para feminino): m
Insira a altura da pessoa 4 (em metros): 1.89
Insira o sexo da pessoa 4 (M para masculino, F para feminino): f
Insira a altura da pessoa 5 (em metros): 1
Insira o sexo da pessoa 5 (M para masculino, F para feminino): m
Insira a altura da pessoa 6 (em metros): 1
Insira o sexo da pessoa 6 (M para masculino, F para feminino): m
Insira a altura da pessoa 7 (em metros): 1
Insira o sexo da pessoa 7 (M para masculino, F para feminino): m
Insira a altura da pessoa 8 (em metros): 1
Insira o sexo da pessoa 8 (M para masculino, F para feminino): m
Insira a altura da pessoa 9 (em metros): 2.02
Insira o sexo da pessoa 9 (M para masculino, F para feminino): m
Insira a altura da pessoa 10 (em metros): 1.55
Insira o sexo da pessoa 10 (M para masculino, F para feminino): f
Maior altura do grupo: 2.02 m
Menor altura do grupo: 1.0 m
Média de altura dos homens: 1.3957142857142857 m
Número de mulheres: 3

Process finished with exit code 0
```

```
// Classe aberta que permite herança
open class GrupoPessoas {

    // Lista para armazenar as alturas
    private val alturas = mutableListOf<Double>()

    // Lista para armazenar os sexos ('M' para masculino, 'F' para feminino)
    private val sexos = mutableListOf<Char>()

    // Função para coletar os dados das 10 pessoas
    fun coletarDados() {
        for (i in 1..10) {
            print("Insira a altura da pessoa $i (em metros): ")
            val altura = readLine()?.toDoubleOrNull() ?: 0.0
            alturas.add(altura)

            print("Insira o sexo da pessoa $i (M para masculino, F para feminino): ")
            val sexo = readLine()?.uppercase()?.firstOrNull() ?: 'F'
            sexos.add(sexo)
        }
    }

    // a) Função para encontrar a maior e a menor altura
    fun maiorEMenorAltura() {
        val maior = alturas.maxOrNull() ?: 0.0
        val menor = alturas.minOrNull() ?: 0.0
        println("Maior altura do grupo: $maior m")
        println("Menor altura do grupo: $menor m")
    }

    // b) Função para calcular a média de altura dos homens
```

```

fun mediaAlturaHomens(): Double {
    var soma = 0.0
    var contador = 0

    for (i in 0 until <alturas.size) {
        if (sexos[i] == 'M') {
            soma += alturas[i]
            contador++
        }
    }

    val media = if (contador > 0) soma / contador else 0.0
    println("Média de altura dos homens: $media m")
    return media
}

// c) Função para contar o número de mulheres
fun numeroMulheres(): Int {
    val qtd = sexos.count { it == 'F' }
    println("Número de mulheres: $qtd")
    return qtd
}

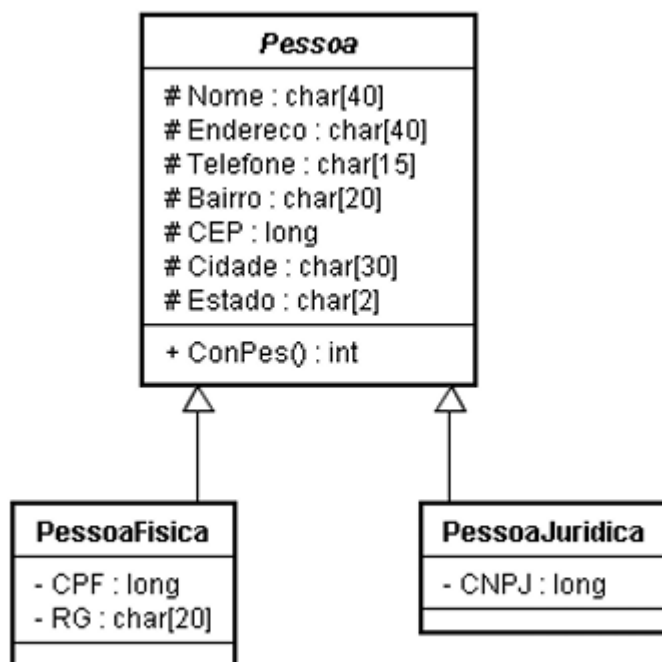
// Função principal
fun main(args: Array<String>) {
    val grupo = GrupoPessoas()

    grupo.coletarDados() // Coleta os dados das 10 pessoas
    grupo.maiorEMenorAltura() // Exibe maior e menor altura
    grupo.mediaAlturaHomens() // Exibe média de altura dos homens
    grupo.numeroMulheres() // Exibe número de mulheres
}

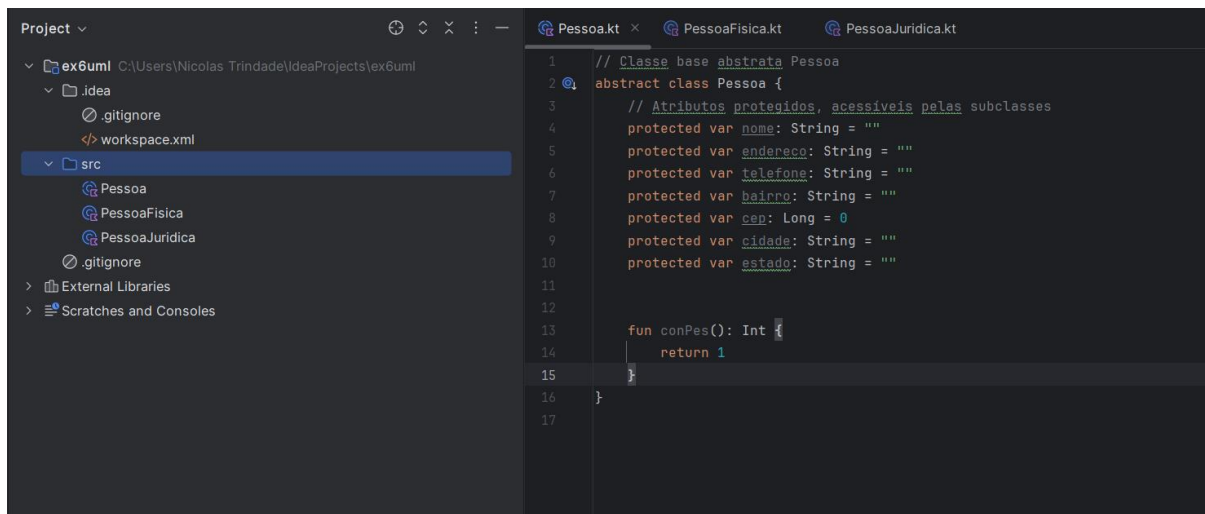
```

****Exercício 5 o professor falou que não era para fazer, pois ele ainda não havia explicado como seria feito o código****

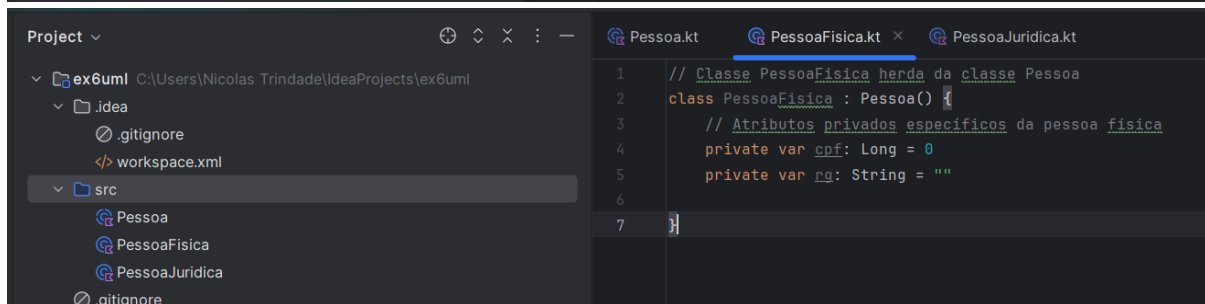
Exercício 06 - Criar as classes em kotlin de acordo com a modelagem UML abaixo. Exibir os métodos e os atributos de cada classe.



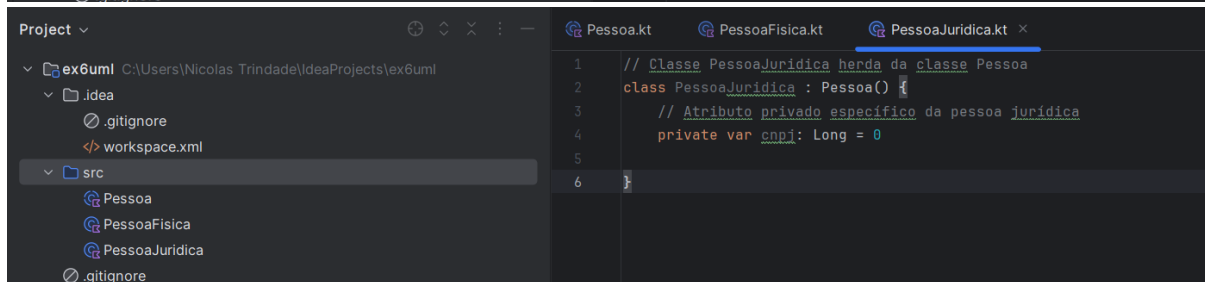
****não tem main para executar****



```
1 // Classe base abstrata Pessoa
2 abstract class Pessoa {
3     // Atributos protegidos, acessíveis pelas subclasses
4     protected var nome: String = ""
5     protected var endereco: String = ""
6     protected var telefone: String = ""
7     protected var bairro: String = ""
8     protected var cep: Long = 0
9     protected var cidade: String = ""
10    protected var estado: String = ""
11
12
13    fun conPes(): Int {
14        return 1
15    }
16 }
17
```

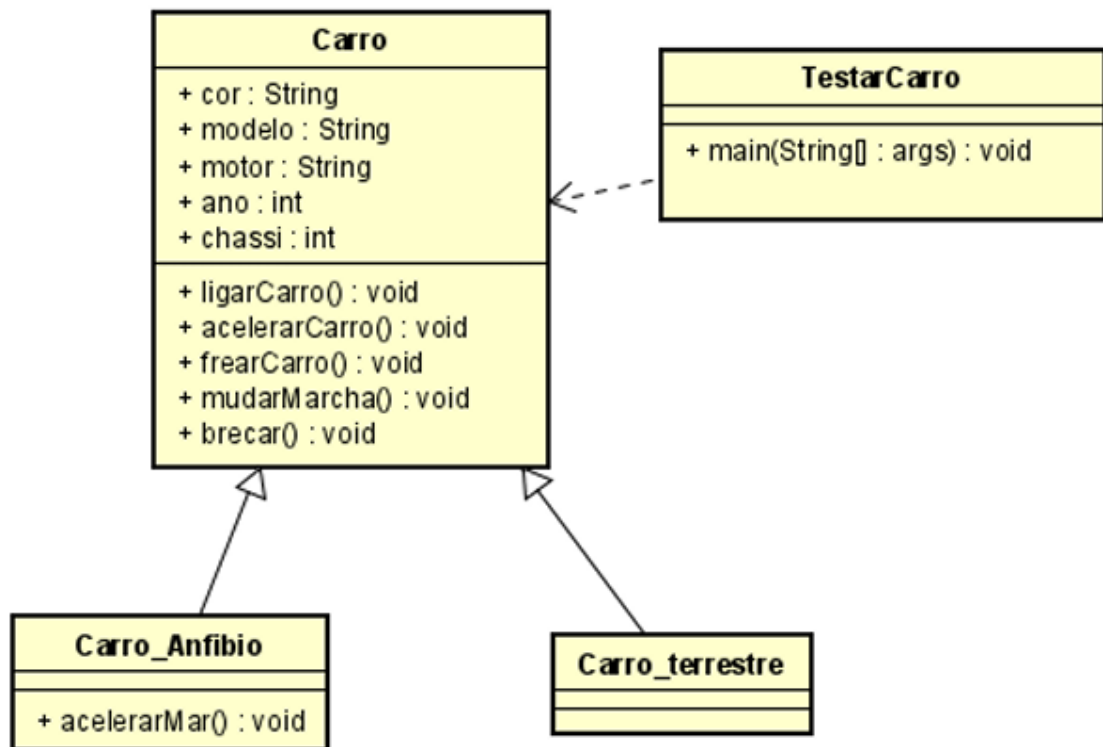


```
1 // Classe PessoaFisica herda da classe Pessoa
2 class PessoaFisica : Pessoa() {
3     // Atributos privados específicos da pessoa física
4     private var cpf: Long = 0
5     private var rg: String = ""
6
7 }
```



```
1 // Classe PessoaJuridica herda da classe Pessoa
2 class PessoaJuridica : Pessoa() {
3     // Atributo privado específico da pessoa jurídica
4     private var cnpj: Long = 0
5
6 }
```

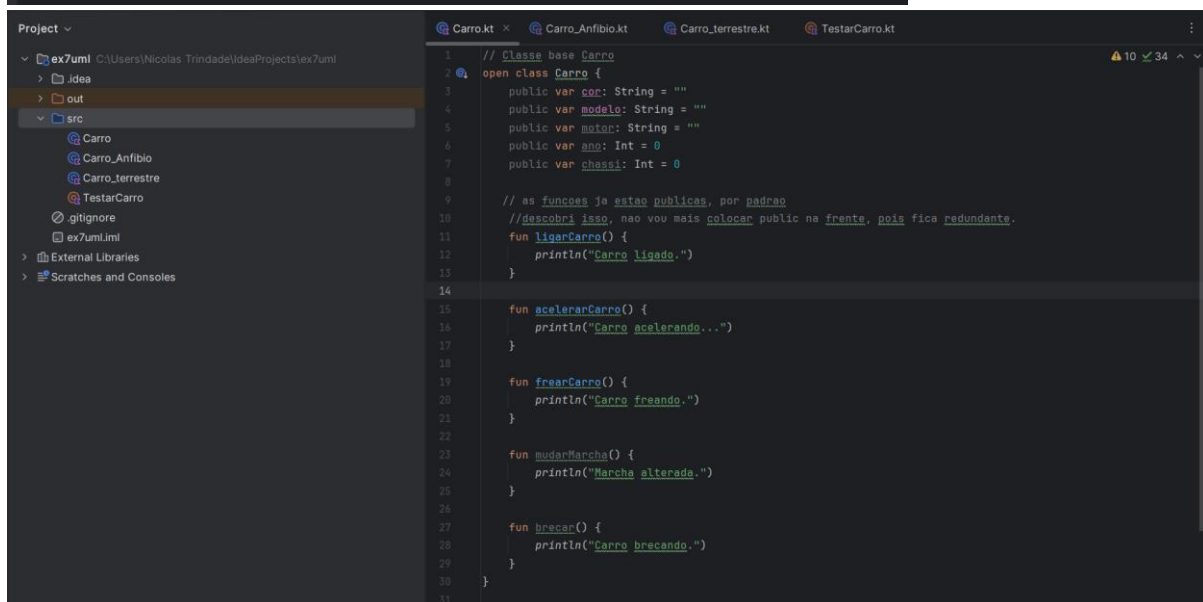
Exercício 07 - Criar as classes em kotlin de acordo com a modelagem UML abaixo.
Exibir os métodos e os atributos de cada classe.



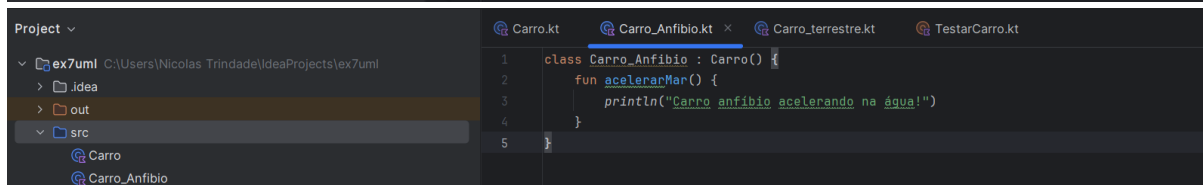
Resposta:

```
"C:\Program Files\Java\jdk-23\bin\java.exe"
Carro ligado.
Carro acelerando...
Carro freando.
-----
Carro ligado.
Carro acelerando...
Carro anfíbio acelerando na água!

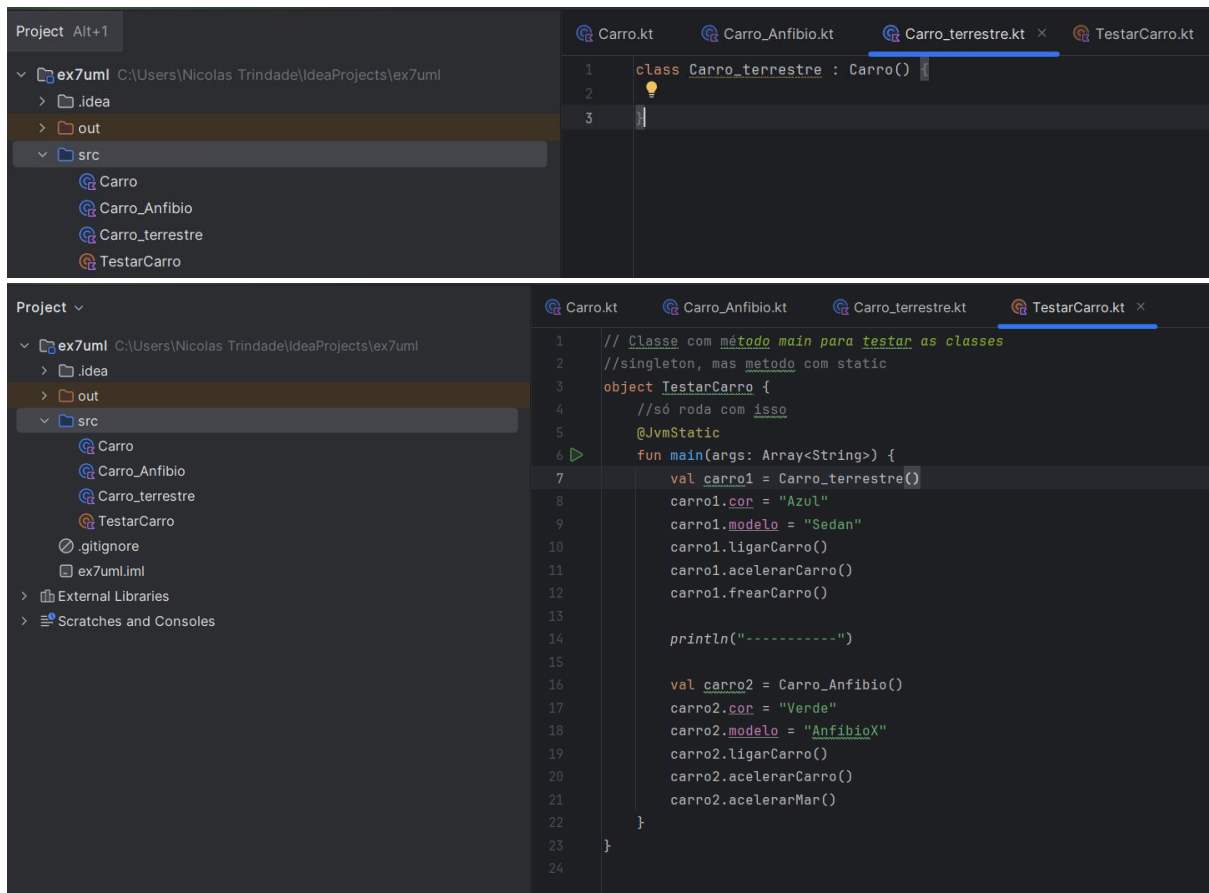
Process finished with exit code 0
```



```
1 // classe base Carro
2 open class Carro {
3     public var cor: String = ""
4     public var modelo: String = ""
5     public var motor: String = ""
6     public var ano: Int = 0
7     public var chassi: Int = 0
8
9     // as funcoes ja estao publicas, por padrao
10    // descobri isso, nao vou mais colocar public na frente, pois fica redundante.
11    fun ligarCarro() {
12        println("Carro ligado.")
13    }
14
15    fun acelerarCarro() {
16        println("Carro acelerando...")
17    }
18
19    fun frearCarro() {
20        println("Carro freando.")
21    }
22
23    fun mudarMarcha() {
24        println("Marcha alterada.")
25    }
26
27    fun breicar() {
28        println("Carro breicando.")
29    }
30 }
31
```



```
1 class Carro_Anfibio : Carro() {
2     fun acelerarMar() {
3         println("Carro anfíbio acelerando na água!")
4     }
5 }
```



Lista 04 – Kotlin _Pontuando

1) Receba um número entre 1 e 10. Calcule e mostre o resultado de uma tabuada.


```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt
Digite o número que você deseja calcular a tabuada: 6
```

```
-----
6.0 x 1 = 6.0
6.0 x 2 = 12.0
6.0 x 3 = 18.0
6.0 x 4 = 24.0
6.0 x 5 = 30.0
6.0 x 6 = 36.0
6.0 x 7 = 42.0
6.0 x 8 = 48.0
6.0 x 9 = 54.0
6.0 x 10 = 60.0
```

Process finished with exit code 0

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Fi
Digite o número que você deseja calcular a tabuada: 11
```

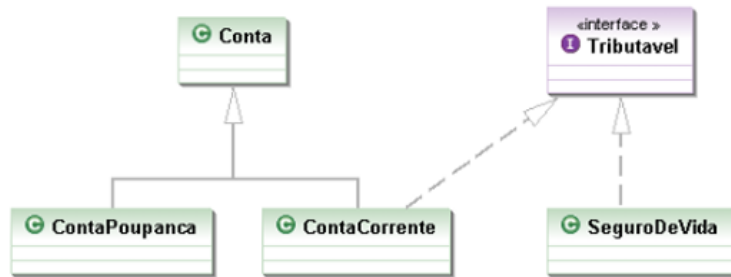
```
-----
Erro: número inválido. Digite de 1 a 10.
```

Process finished with exit code 0

ex1lista4.kt ×

```
1
2 fun main() {
3     print("Digite o número que você deseja calcular a tabuada: ")
4     val num = readLine()?.toDoubleOrNull() ?: 0.0
5     println("-----")
6
7     if (num in 1.0 ≤ .. ≤ 10.0) {
8         for (i in 1 ≤ .. ≤ 10){
9             val multi = num * i
10            println("$num x $i = $multi")
11        }
12    } else {
13        println("Erro: número inválido. Digite de 1 a 10.")
14    }
15
16 }
```

2) Nosso banco precisa tributar dinheiro de alguns bens que nossos clientes possuem. Para isso, vamos criar um sistema para isso.



- a) Crie uma interface **Tributavel** que possui o método `calculaTributos()`, que retorna um **double**.
- b) Alguns bens são tributáveis e outros não, **ContaPoupanca** não é tributável, já para **ContaCorrente** você precisa pagar 1% da conta e o **SeguroDeVida** tem uma taxa fixa de 42 reais.
- c) As classes **ContaCorrente** e **ContaPoupanca** herdam de uma classe **Conta**. Essa classe **Conta** possui um saldo e os métodos `sacar(double)`, `depositar(double)` e `obterSaldo()` que retorna o saldo da conta.
- d) Vamos criar uma classe **TestaTributavel** com um método `main()` para testar o nosso exemplo.

```
"C:\Program Files\Java\jdk-23\bin\java.exe
Saldo Conta Corrente: 1000.0
Tributo Conta Corrente: 10.0
Saldo Conta Poupança: 500.0
Tributo Seguro de Vida: 42.0

Process finished with exit code 0
```

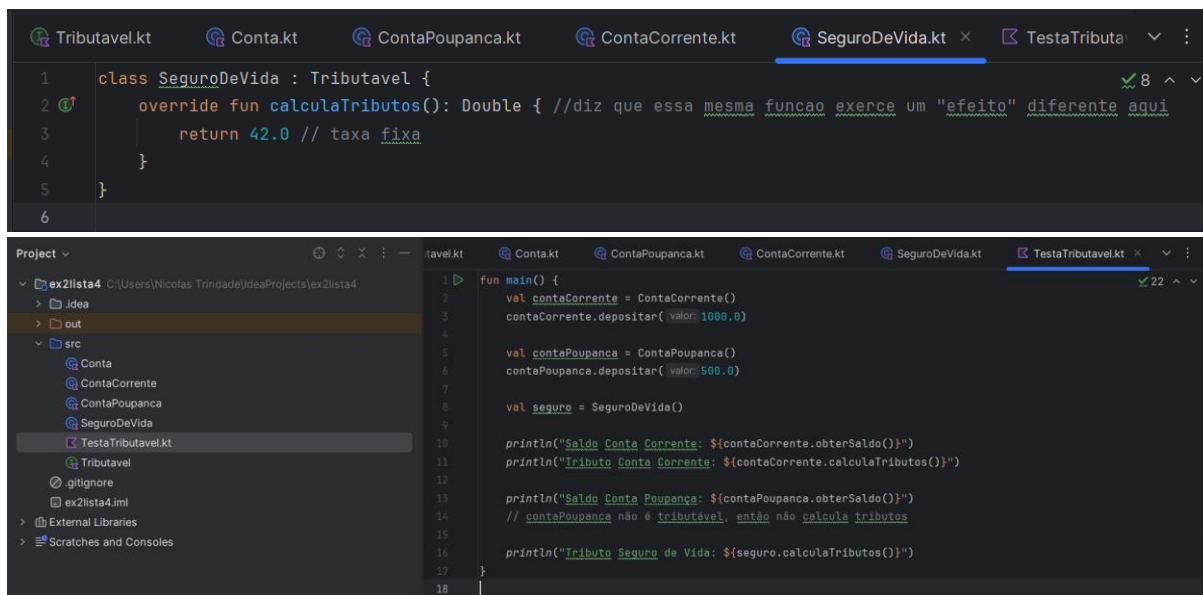
The screenshot shows an IDE with three tabs: **Tributavel.kt**, **Conta.kt**, and **ContaPou**. The **Tributavel.kt** tab is active, displaying the following Kotlin code:

```
1 interface Tributavel {
2     fun calculaTributos(): Double
3 }
4
```

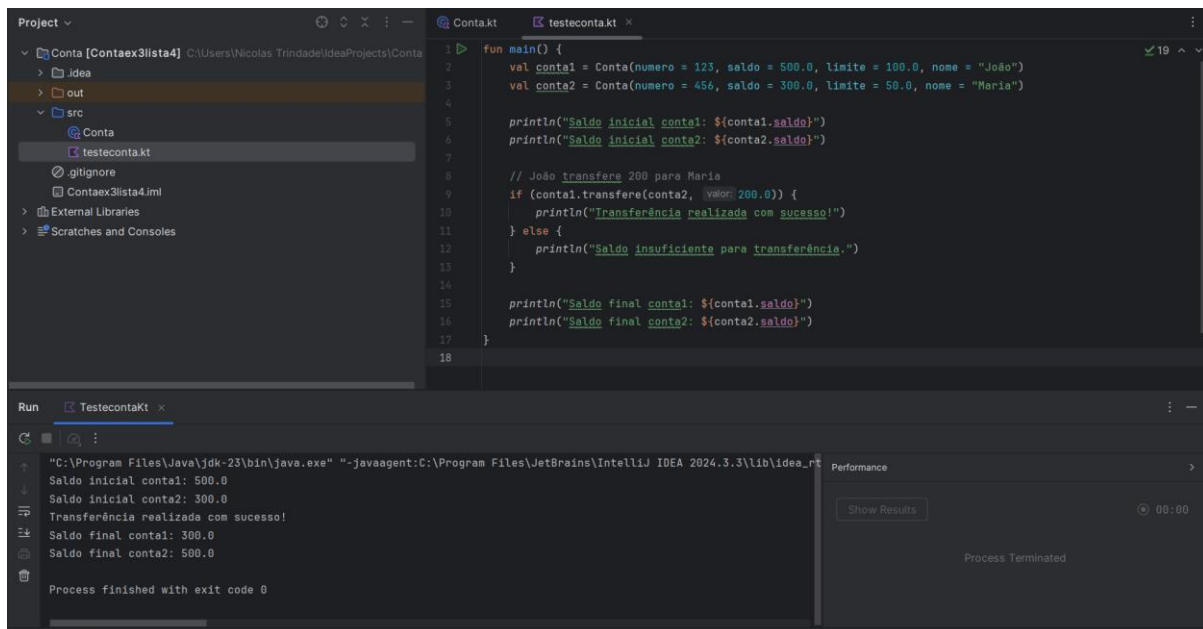
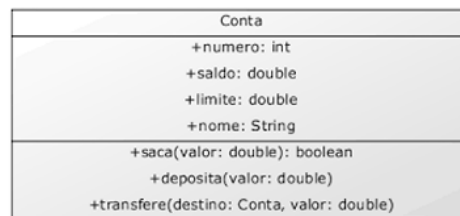
```
Tributavel.kt x Conta.kt x ContaPoupanca.kt
1  @↓ open class Conta {
2      protected var saldo: Double = 0.0
3
4      fun depositar(valor: Double) {
5          saldo += valor
6      }
7
8      fun sacar(valor: Double) {
9          if (valor <= saldo) {
10             saldo -= valor
11         } else {
12             println("Saldo insuficiente.")
13         }
14     }
15
16     fun obterSaldo(): Double {
17         return saldo
18     }
19 }
20
```

```
Tributavel.kt Conta.kt ContaPoupanca.kt x
1  class ContaPoupanca : Conta() {
2      // Nenhuma mudança, não tributavel
3  }
4
```

```
Tributavel.kt Conta.kt ContaPoupanca.kt ContaCorrente.kt x SeguroDeVida.kt TestaTributa
1  class ContaCorrente : Conta(), Tributavel {
2      override fun calculaTributos(): Double { //diz que essa mesma funcao exerce um "efeito" diferente aqui
3          return saldo * 0.01 // 1% do saldo
4      }
5  }
6
```



3) Conforme o diagrama de classe abaixo desenvolva o programa utilizando as técnicas da linguagem kotlin trabalhadas. O programa deve representa a classe como um todo.

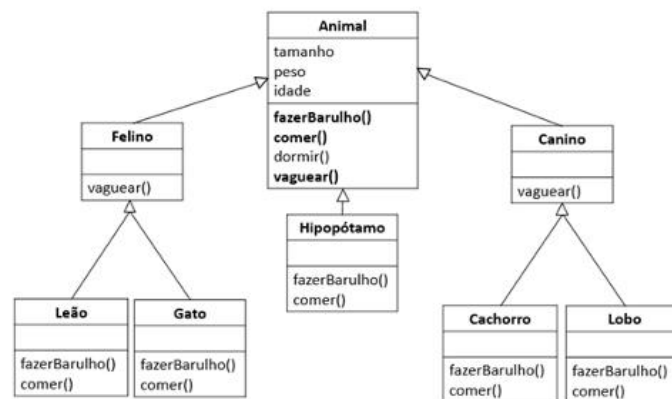


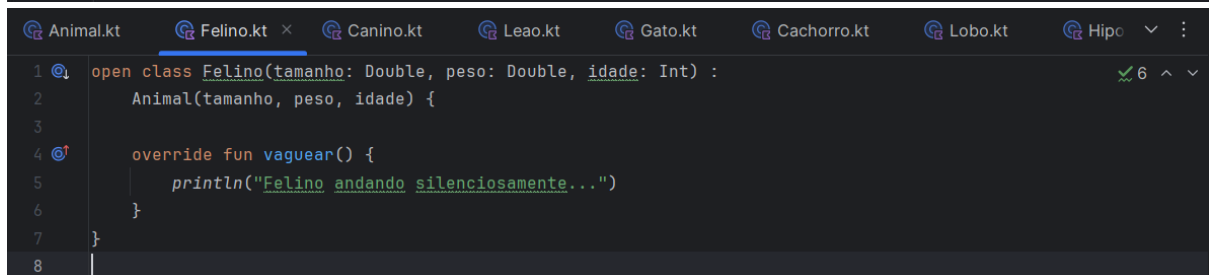
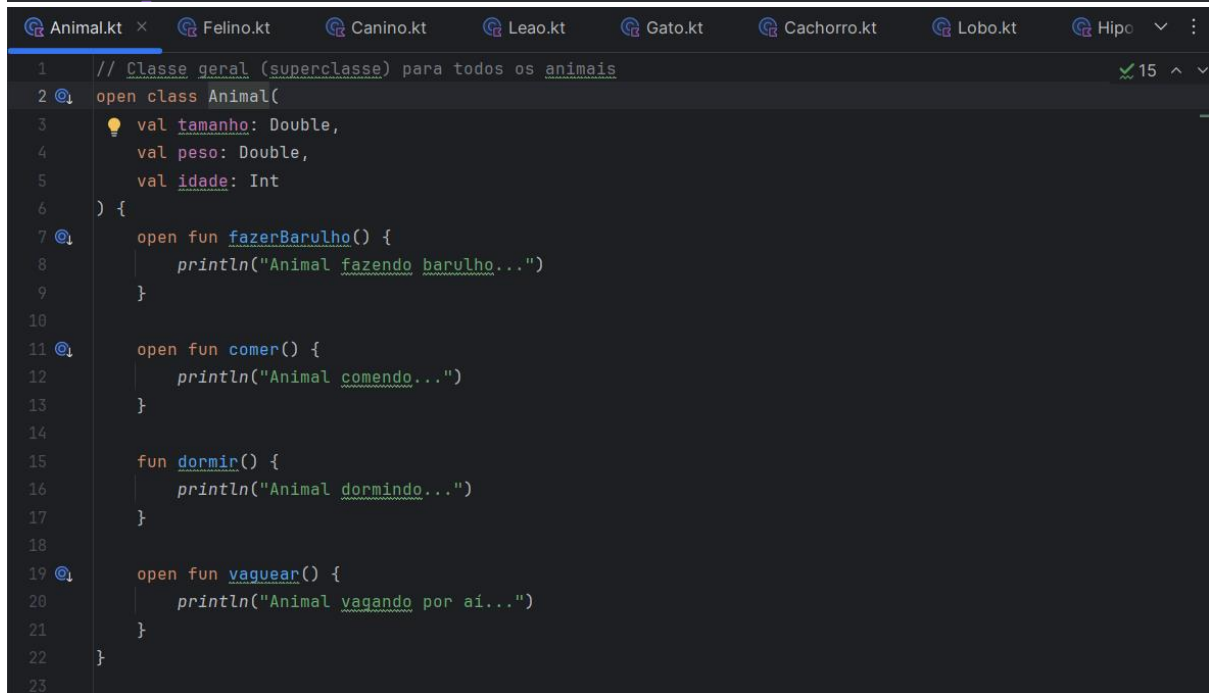
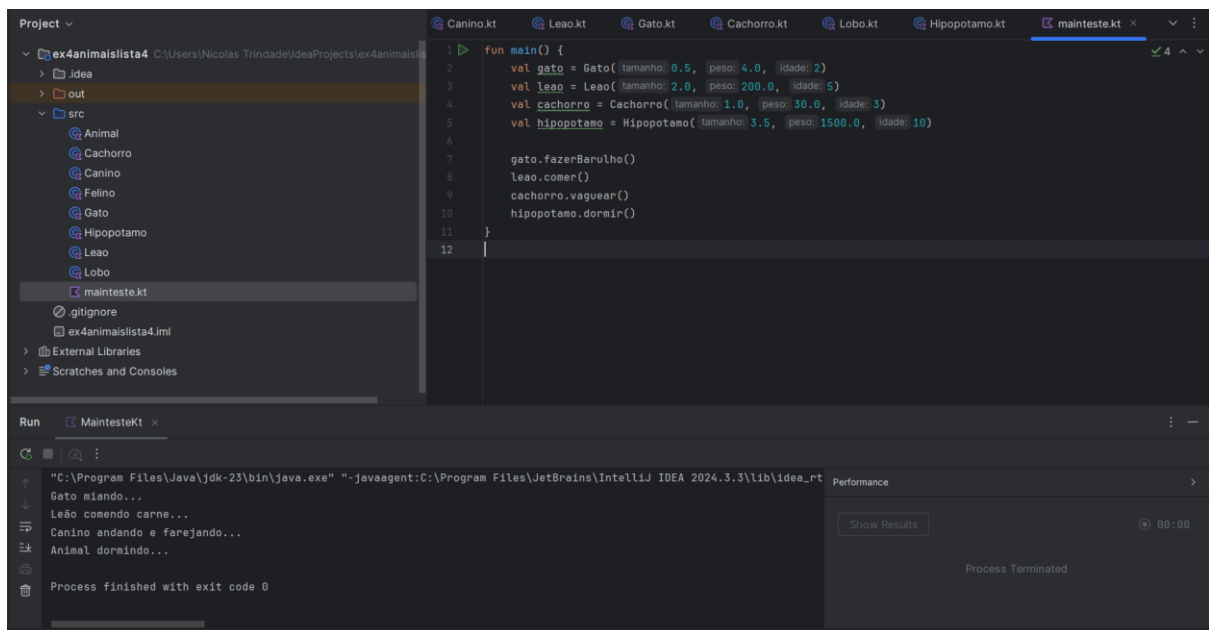
```

1 class Conta(
2     val numero: Int,
3     var saldo: Double,
4     val limite: Double,
5     val nome: String
6 ) {
7
8     // Método para sacar dinheiro, só permite se saldo+limite for suficiente
9     fun saca(valor: Double): Boolean {
10         return if (valor <= saldo + limite) {
11             saldo -= valor
12             true
13         } else {
14             false
15         }
16     }
17
18     // Método para depositar dinheiro
19     fun deposita(valor: Double) {
20         saldo += valor
21     }
22
23     // Método para transferir dinheiro para outra conta
24     fun transfere(destino: Conta, valor: Double): Boolean {
25         if (saca(valor)) {
26             destino.deposita(valor)
27             return true
28         }
29         return false
30     }
31 }

```

4) Conforme o diagrama de classe abaixo desenvolva o programa utilizando as técnicas da linguagem kotlin trabalhadas. O programa deve representa a classe com um todo.





```
Animal.kt Felino.kt Canino.kt × Leao.kt Gato.kt Cachorro.kt Lobo.kt Hipo...
1 open class Canino(tamanho: Double, peso: Double, idade: Int) :
2     Animal(tamanho, peso, idade) {
3
4     override fun vaguear() {
5         println("Canino andando e farejando...")
6     }
7 }
8
```

```
Animal.kt Felino.kt Canino.kt Leao.kt × Gato.kt Cachorro.kt Lobo.kt Hipopotamo.kt mainteste.kt
1 class Leao(tamanho: Double, peso: Double, idade: Int) :
2     Felino(tamanho, peso, idade) {
3
4     override fun fazerBarulho() {
5         println("Leão rugindo!")
6     }
7
8     override fun comer() {
9         println("Leão comendo carne...")
10    }
11 }
12
```

```
Animal.kt Felino.kt Canino.kt Leao.kt Gato.kt × Cachorro.kt Lobo.kt Hipopotamo.kt mainteste.kt
1 class Gato(tamanho: Double, peso: Double, idade: Int) :
2     Felino(tamanho, peso, idade) {
3
4     override fun fazerBarulho() {
5         println("Gato miando...")
6     }
7
8     override fun comer() {
9         println("Gato comendo ração...")
10    }
11 }
```

```
Animal.kt Felino.kt Canino.kt Leao.kt Gato.kt Cachorro.kt × Lobo.kt Hipopotamo.kt mainteste.kt
1 class Cachorro(tamanho: Double, peso: Double, idade: Int) :
2     Canino(tamanho, peso, idade) {
3
4     override fun fazerBarulho() {
5         println("Cachorro latindo!")
6     }
7
8     override fun comer() {
9         println("Cachorro comendo ração...")
10    }
11 }
```

```
Animal.kt Felino.kt Canino.kt Leao.kt Gato.kt Cachorro.kt Lobo.kt × Hipopotamo.kt mainteste.kt
1 class Lobo(tamanho: Double, peso: Double, idade: Int) :
2     Canino(tamanho, peso, idade) {
3
4     override fun fazerBarulho() {
5         println("Lobo vivando!")
6     }
7
8     override fun comer() {
9         println("Lobo comendo carne...")
10    }
11 }
```

```
Animal.kt  Felino.kt  Canino.kt  Leao.kt  Gato.kt  Cachorro.kt  Lobo.kt  Hipopotamo.kt  mainteste.kt
1  class Hipopotamo(tamanho: Double, peso: Double, idade: Int) :
2      Animal(tamanho, peso, idade) {
3
4      override fun fazerBarulho() {
5          println("Hipopótamo roncando...")
6      }
7
8      override fun comer() {
9          println("Hipopótamo comendo plantas...")
10     }
11 }
12
```

```
Animal.kt  Felino.kt  Canino.kt  Leao.kt  Gato.kt  Cachorro.kt  Lobo.kt  Hipopotamo.kt  mainteste.kt
1  fun main() {
2      val gato = Gato( tamanho: 0.5, peso: 4.0, idade: 2)
3      val leao = Leao( tamanho: 2.0, peso: 200.0, idade: 5)
4      val cachorro = Cachorro( tamanho: 1.0, peso: 30.0, idade: 3)
5      val hipopotamo = Hipopotamo( tamanho: 3.5, peso: 1500.0, idade: 10)
6
7      gato.fazerBarulho()
8      leao.comer()
9      cachorro.vaguear()
10     hipopotamo.dormir()
11 }
12
```